

# FPGA-Accelerated Handwritten Digit Recognition Using an INT8 LeNet CNN on the DE1-SoC

Yizheng Tang (yt2992)  
Weiwei Wu (ww2766)  
Sirui Chen (sc5746)  
Chenxi Shen(cs4634)  
Tian Li (tl3468)

May 2026

## Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Project Background . . . . .	6
1.2	Motivation for FPGA-Based CNN Acceleration . . . . .	6
1.3	Project Objective . . . . .	6
1.4	Target Application: Handwritten Digit Recognition . . . . .	6
1.5	Main Contributions . . . . .	7
1.6	Report Organization . . . . .	7
<b>2</b>	<b>System Block Diagram</b>	<b>7</b>
2.1	Overall System Architecture . . . . .	7
2.2	Software-Hardware Co-Design Flow . . . . .	8
2.3	Browser / Image Input Interface . . . . .	9
2.4	Flask Server and Preprocessing Module . . . . .	10
2.5	Ethernet TCP Communication Path . . . . .	11
2.6	HPS Linux Inference Server . . . . .	11
2.7	Lightweight HPS-to-FPGA AXI Bridge . . . . .	11
2.8	FPGA CNN Accelerator Core . . . . .	12
2.9	End-to-End Inference Dataflow . . . . .	12
<b>3</b>	<b>CNN Algorithm and Model Design</b>	<b>13</b>
3.1	LeNet-5 Inspired CNN Structure . . . . .	13
3.2	Input Image Format: 32×32 Grayscale Pixels . . . . .	13
3.3	Feature Extraction: Convolution and Pooling . . . . .	14
3.4	Classification: Flattening and Fully Connected Layers . . . . .	14
3.5	Output Logits and Digit Prediction . . . . .	14
3.6	Hardware-Aware Architecture Design . . . . .	14
<b>4</b>	<b>Quantization and Parameter Preparation</b>	<b>14</b>
4.1	Need for Integer Quantization . . . . .	14

4.2	Training and Quantization Strategy	15
4.3	Implementation of Quantization Logic	15
4.4	Parameter Files Used by Software and Hardware	15
4.5	Alternative Quantization Methods	16
4.6	Static BRAM Configuration	16
<b>5</b>	<b>Software Input Processing</b>	<b>16</b>
5.1	Supported Input Sources	16
5.2	Browser Canvas Input	17
5.3	Uploaded JPG / PNG Image Input	17
5.4	Flask Backend Image Reception	17
5.5	Grayscale Conversion	17
5.6	Foreground Cropping and Background Removal	17
5.7	Resizing to 32×32 Pixels	18
5.8	Conversion to 8-Bit Pixel Data	18
5.9	Generating FPGA-Compatible Input Format	18
5.10	Preprocessing Limitations and Error Sources	18
<b>6</b>	<b>Software-Hardware Communication Protocol</b>	<b>18</b>
6.1	Ethernet Architecture and Protocol Selection	18
6.2	TCP Implementation and JSON Payload Handling	19
6.3	Inference Result Reporting and Transmission	19
6.4	Latency Measurement	19
6.5	HPS-Side JSON Parsing and Input Validation	19
6.6	One-Connection-Per-Inference Design	20
6.7	Error Handling and Response Format	20
6.8	Network and Hardware Protocols	20
6.9	Protocol Limitations	20
6.10	Code Mapping	20
<b>7</b>	<b>HPS Software and MMIO Interface</b>	<b>21</b>
7.1	HPS Linux Runtime Environment	21
7.2	C-Based Inference Server	21
7.3	Receiving Pixel Data from TCP Socket	21
7.4	Memory-Mapped I/O Overview	21
7.5	/dev/mem, mmap(), and MMIO	22
7.6	Lightweight HPS-to-FPGA AXI Bridge	22
7.7	Physical Address to Virtual Address Mapping	23
7.8	Register Read Function	23
7.9	Register Write Function	23
7.10	Pixel Upload Sequence	23
7.11	Accelerator Start Sequence	23
7.12	Done Polling Sequence	24
7.13	Reading Prediction and Logits	24
7.14	Returning Results to the Software Client	24
<b>8</b>	<b>Register Map Design</b>	<b>24</b>
8.1	Register Map Overview	24

8.2	Register Address Space Allocation . . . . .	24
8.3	Address Alignment Requirement . . . . .	25
8.4	Reason for Using 32-Bit Registers . . . . .	25
8.5	Register Region Separation . . . . .	25
8.6	Base Address Selection . . . . .	25
8.7	Offset Address Definition . . . . .	26
8.8	Control Register . . . . .	26
8.9	Status Register . . . . .	26
8.10	Result Register . . . . .	26
8.11	LOGIT[0:9] Registers . . . . .	27
8.12	PIXEL_INPUT[0:1023] Register Window . . . . .	27
8.13	Pixel Register Address Calculation . . . . .	28
8.14	Register Reset Values . . . . .	28
8.15	Register Read / Write Permissions . . . . .	28
8.16	Start / Done Handshake Register Protocol . . . . .	28
8.17	Complete Register Address Table . . . . .	29
8.18	Register Map Verification . . . . .	29
<b>9</b>	<b>FPGA Hardware Architecture</b>	<b>29</b>
9.1	Top-Level FPGA Design . . . . .	29
9.2	soc_system_top Module . . . . .	30
9.3	HPS Subsystem Integration . . . . .	30
9.4	hps_lenet_regs.sv Interface . . . . .	30
9.5	lenet_int8_top.sv Core . . . . .	31
9.6	Clock and Reset Design . . . . .	31
9.7	Input Pixel Buffer . . . . .	31
9.8	Weight and Bias Memory . . . . .	31
9.9	Intermediate Feature Map Storage . . . . .	31
9.10	Output Logit Storage . . . . .	31
9.11	Top-Level Signal Connections . . . . .	32
9.12	FPGA Hardware Block Diagram . . . . .	32
<b>10</b>	<b>CNN Accelerator Hardware Design</b>	<b>33</b>
10.1	Execution Flow and Global FSM . . . . .	33
10.2	Convolution Datapath . . . . .	33
10.3	PE and MAC Design . . . . .	34
10.4	Bias Addition and Quantization . . . . .	34
10.5	Max Pooling Datapath . . . . .	34
10.6	Flatten / Fully Connected Datapath . . . . .	35
10.7	Output and Completion Signaling . . . . .	35
10.8	Hardware Latency Analysis . . . . .	35
<b>11</b>	<b>Memory Architecture</b>	<b>36</b>
11.1	Memory Requirement Analysis . . . . .	36
11.2	Input Pixel Storage . . . . .	36
11.3	Kernel / Bias Parameter Memory . . . . .	36
11.4	Feature Map Memory Organization . . . . .	36

11.5	BRAM / M10K Usage . . . . .	36
11.6	\$readmemh Initialization . . . . .	37
11.7	Addressing and Access Scheduling . . . . .	37
11.8	Reuse Strategy and Bandwidth Limitations . . . . .	37
<b>12</b>	<b>Verification and Testing</b>	<b>37</b>
12.1	Verification Overview . . . . .	37
12.2	Python Golden Model . . . . .	37
12.3	Layer-by-Layer Output Comparison . . . . .	38
12.4	Quantized Model Verification . . . . .	38
12.5	Register Interface Test . . . . .	38
12.6	HPS Register Readback Test . . . . .	38
12.7	Pixel Register Write Test . . . . .	38
12.8	Start / Done Handshake Test . . . . .	39
12.9	ModelSim RTL Simulation . . . . .	39
12.10	FPGA On-Board Testing . . . . .	39
12.11	Browser to FPGA Test . . . . .	40
12.12	Test Cases for Digits 0–9 . . . . .	40
12.13	Misclassification Case Analysis . . . . .	41
<b>13</b>	<b>Results and Performance Evaluation</b>	<b>41</b>
13.1	Functional Correctness Results . . . . .	41
13.2	Recognition Accuracy . . . . .	41
13.3	Example Correct Predictions . . . . .	42
13.4	Example Incorrect Predictions . . . . .	42
13.5	FPGA Core Runtime . . . . .	42
13.6	HPS Register Access Time . . . . .	42
13.7	Ethernet Transmission Time . . . . .	43
13.8	End-to-End Inference Latency . . . . .	43
13.9	Runtime Variation Analysis . . . . .	43
13.10	FPGA Resource Utilization . . . . .	43
13.11	Timing Analysis and Fmax . . . . .	44
13.12	Comparison with Software-Only Inference . . . . .	45
13.13	Performance Bottleneck Analysis . . . . .	46
<b>14</b>	<b>Failure Modes and Limitations</b>	<b>46</b>
14.1	Image Preprocessing Errors . . . . .	46
14.2	Stroke Thickness and Digit Shape Sensitivity . . . . .	46
14.3	Similar Digit Misclassification . . . . .	46
14.4	Quantization Error . . . . .	46
14.5	Overflow Risk . . . . .	46
14.6	TCP Transmission Delay Variation . . . . .	47
14.7	Register Access Overhead . . . . .	47
14.8	Polling-Based Control Limitation . . . . .	47
14.9	FPGA Resource Limitation . . . . .	47
14.10	Timing Closure Limitation . . . . .	47
14.11	Scalability Limitation . . . . .	47

<b>15</b>	<b>Future Work</b>	<b>47</b>
<b>16</b>	<b>Division of Work</b>	<b>48</b>
16.1	Yizheng Tang . . . . .	48
16.2	Chenxi Shen . . . . .	48
16.3	Sirui Chen . . . . .	49
16.4	Weiwei Wu . . . . .	49
16.5	Tian Li . . . . .	49
<b>17</b>	<b>Conclusion</b>	<b>49</b>
17.1	Project Summary . . . . .	49
17.2	Main Technical Achievements . . . . .	50
17.3	Main Challenges . . . . .	50
17.4	Final Remarks . . . . .	50
<b>18</b>	<b>References</b>	<b>50</b>
<b>A</b>	<b>Final Source Code</b>	<b>50</b>
A.1	Web Demo and Software Reference Code . . . . .	50
A.2	HPS Software Code . . . . .	86
A.3	FPGA RTL Code . . . . .	96
A.4	Testbench Code . . . . .	132
<b>B</b>	<b>Final FPGA Parameter Files</b>	<b>135</b>

# 1 Introduction

## 1.1 Project Background

Convolutional neural networks (CNNs) have become one of the most widely used neural network architectures for image classification tasks. Compared with traditional image processing algorithms, CNNs can automatically extract spatial features from input images and perform classification based on learned parameters. However, CNN inference requires a large number of multiply-accumulate operations, especially in convolutional and fully connected layers. When inference is executed only on a general-purpose processor, the computation can become inefficient in terms of latency and energy usage.

This project focuses on accelerating CNN inference on a DE1-SoC embedded platform. The target application is handwritten digit recognition, where a user draws or uploads a digit image and the system predicts one of ten digit classes, from 0 to 9. In the final version, the work was not just the CNN datapath itself; it also included the browser interface, preprocessing, Ethernet transfer, HPS Linux server, `/dev/mem` register access, and the FPGA accelerator that had to agree with the quantized software reference.

## 1.2 Motivation for FPGA-Based CNN Acceleration

FPGAs are well suited for CNN acceleration because they allow customized parallel datapaths, on-chip memory reuse, and fixed-point arithmetic. Instead of executing CNN operations sequentially on a general-purpose CPU, an FPGA can implement multiply-accumulate units, local buffers, and control logic directly in hardware. This makes it possible to reduce computation overhead and improve system efficiency.

For this project, fixed-point integer arithmetic is used instead of floating-point operations. Model parameters and intermediate activations are quantized to integer format, which avoids the cost of floating-point units and keeps the design small enough for the Cyclone V device. This choice also made debugging easier because the same integer operations could be reproduced in the Python reference model.

## 1.3 Project Objective

The objective of this project is to implement a complete FPGA-based CNN accelerator system for handwritten digit recognition. A working classifier alone would not have been enough for the demo; the system also needed a reliable path from browser input to HPS software, from HPS software to FPGA registers, and then back to the browser with prediction, logits, and timing information.

The complete system follows a software-hardware co-design approach. On the software side, a browser-based interface or uploaded image is processed by a Flask backend. The image is converted into a  $32 \times 32$  grayscale pixel array and then sent to the DE1-SoC board through Ethernet using a TCP-based communication protocol. On the HPS side, a C program runs under Linux and receives the input data from the TCP socket. The HPS then accesses the FPGA accelerator through memory-mapped I/O using `/dev/mem` and the Lightweight HPS-to-FPGA AXI bridge.

## 1.4 Target Application: Handwritten Digit Recognition

The target application is to classify a  $32 \times 32$  grayscale input image into one of ten digit classes, from 0 to 9. The CNN model is inspired by the LeNet-5 architecture, which is a classic convolutional neural network structure for digit recognition.

On the hardware side, the FPGA fabric contains a custom CNN inference core and a memory-mapped register interface. The register interface provides control, status, result, logit, and pixel input registers for communication between the HPS and FPGA. The input pixels are written into a memory-mapped register window, and the accelerator is started through a control register. After computation is complete, the HPS reads back the predicted digit and output logits from the FPGA registers and sends the result back to the software client.

The CNN accelerator performs the main inference operations, including convolution, pooling, flattening, fully connected computation, quantization, and output comparison.

## 1.5 Main Contributions

The major contributions of this project are as follows:

- A LeNet-style CNN model was trained and quantized for handwritten digit recognition.
- A software preprocessing pipeline was developed to convert browser input or uploaded images into FPGA-compatible  $32 \times 32$  pixel data.
- A TCP-based communication path was implemented between the Flask software backend and the HPS running on the DE1-SoC board.
- A C-based HPS inference server was developed to receive image data, access FPGA registers, start the accelerator, and return results.
- A memory-mapped register interface was designed for HPS-FPGA communication through the Lightweight AXI bridge.
- A custom INT8 CNN accelerator was implemented in FPGA fabric.
- The complete system was verified through software reference comparison, register readback tests, RTL simulation, and end-to-end hardware testing.

## 1.6 Report Organization

The remainder of this report is organized as follows. Section 2 introduces the overall system block diagram and end-to-end dataflow. Section 3 describes the CNN algorithm and model structure. Section 4 explains the quantization method and parameter preparation flow. Section 5 presents the software input processing pipeline. Section 6 discusses the Ethernet TCP communication protocol. Section 7 describes the HPS software and MMIO interface. Section 8 provides the detailed register map design. Sections 9 and 10 describe the FPGA top-level architecture and CNN accelerator hardware design. Section 11 discusses memory organization. Section 12 presents the verification methodology. Section 13 evaluates the system results and performance. Finally, Sections 14 to 17 discuss limitations, future work, division of work, lessons learned, and conclusion.

# 2 System Block Diagram

## 2.1 Overall System Architecture

The complete system is an end-to-end software-hardware co-design for handwritten digit recognition. It connects a browser-based user interface, a Flask software backend, Ethernet TCP communication, an HPS Linux inference server, and a custom FPGA CNN accelerator.

The high-level system flow is shown in Figure 1. The path begins with browser-side image input, moves through software preprocessing and Ethernet transfer, and then enters the HPS-to-FPGA memory-mapped control path.

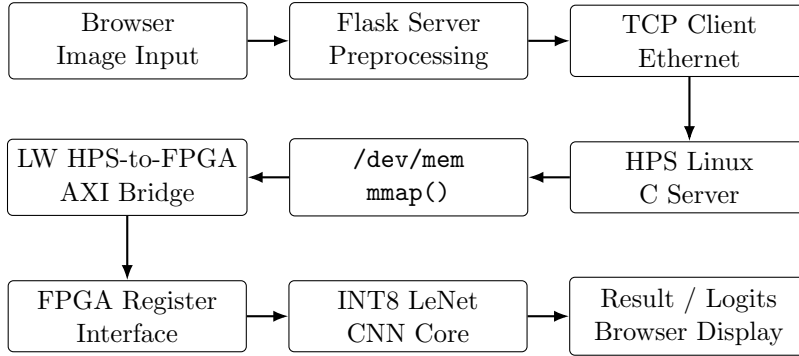


Figure 1: End-to-end software and hardware inference flow.

The browser provides the user input. The Flask server converts the input image into a  $32 \times 32$  grayscale pixel array. The processed pixels are sent to the HPS over Ethernet using TCP. The HPS C server writes the pixels into FPGA registers through memory-mapped I/O and starts the CNN accelerator. After the FPGA finishes computation, the HPS reads back the predicted digit and logits and returns them to the web application.

## 2.2 Software-Hardware Co-Design Flow

The project uses a software-hardware co-design structure. Software handles the parts that changed often during debugging, such as image input, preprocessing, network communication, and result visualization. Hardware is reserved for the repeated CNN computation where fixed control flow and local memory reuse are useful.

The software side includes the browser drawing pad and image upload interface, Flask backend server, image preprocessing pipeline, Python TCP client, and software reference inference paths for comparison. The hardware and HPS side includes the HPS Linux C inference server, Lightweight HPS-to-FPGA AXI bridge, memory-mapped register interface, FPGA CNN accelerator core, and on-chip memories for pixels, feature maps, and parameters.

This separation also made debugging more manageable. When the browser preview looked wrong, the issue was usually in preprocessing; when the quantized software and FPGA outputs disagreed, the issue was more likely in parameter packing, register transfer, or RTL arithmetic.

Figure 2 provides an implementation-oriented view of this boundary. It highlights how the browser, Flask server, TCP client, HPS server, `/dev/mem` mapping, register interface, and RTL control logic are connected in the working system.

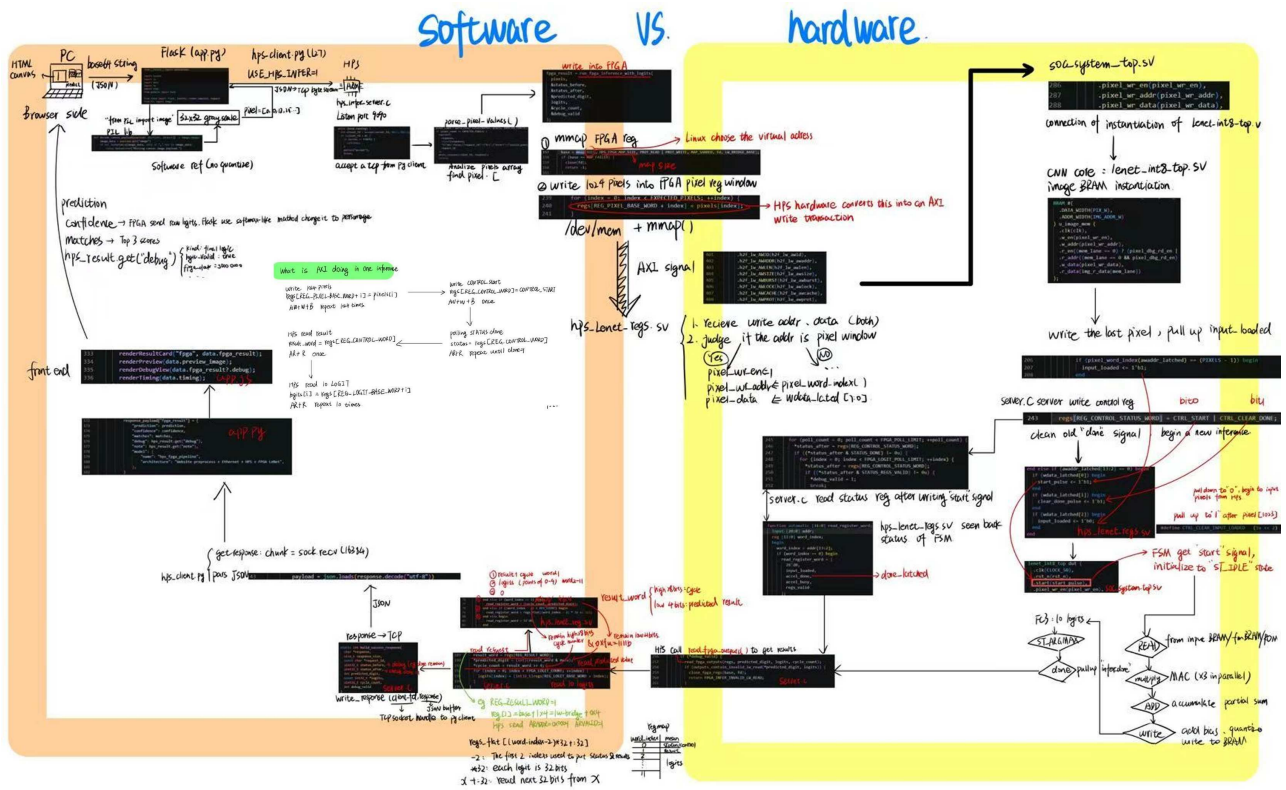


Figure 2: Annotated software-to-hardware integration flow used during system debugging.

### 2.3 Browser / Image Input Interface

The browser interface allows the user to either draw a digit on a canvas or upload an image. The drawing pad provides a black background and white stroke, which matches the expected digit format used by the preprocessing pipeline.

The user interface also includes controls such as stroke width, image inversion, prediction, upload, clear, and label saving. The browser does not directly communicate with the FPGA. Instead, it sends the canvas image to the Flask backend through an HTTP request.

The browser input stage is important because the quality of the handwritten digit directly affects recognition accuracy. A poorly centered or unclear drawing may lead to incorrect classification even if the FPGA hardware is functioning correctly.

Figure 3 shows the final web interface. The input canvas, preprocessing preview, FPGA prediction, software reference outputs, final logits, and speed comparison are displayed together. This layout was useful during testing because a single run could show whether the drawn digit, processed  $32 \times 32$  input, quantized reference, and FPGA result were all consistent.

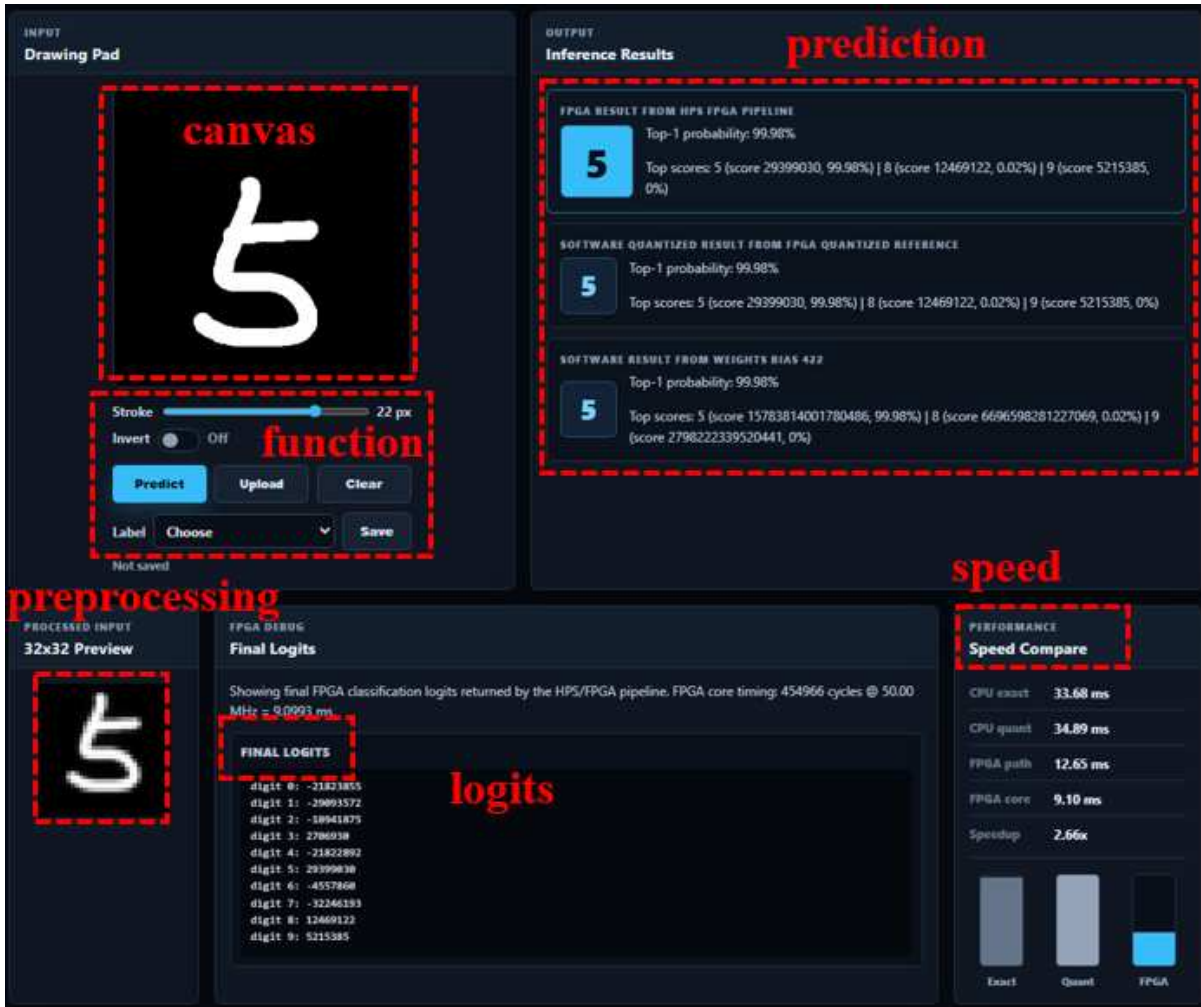


Figure 3: Annotated browser interface for drawing input, checking preprocessing, comparing predictions, and inspecting latency.

## 2.4 Flask Server and Preprocessing Module

The Flask server receives the image from the browser and performs preprocessing before sending data to the HPS. The preprocessing module converts the image into a format compatible with the FPGA accelerator.

The preprocessing pipeline is summarized in Table 1.

Table 1: Image Preprocessing Pipeline

Step	Operation	Output Purpose
1	RGBA/RGB to grayscale conversion	Single-channel image
2	Foreground detection and bounding-box crop	Remove unused background
3	Resize and center the digit	Match the LeNet input geometry
4	Optional smoothing and contrast adjustment	Improve robustness for drawings
5	Flatten to $32 \times 32$ unsigned 8-bit vector	FPGA input pixel window

The final output of preprocessing is a list of 1024 pixel values. Each pixel is an unsigned 8-bit grayscale value. This pixel vector is used both by the software reference model and by the FPGA

inference path. The final web interface, preprocessing code, software predictor, TCP client, and frontend files are included in Appendix [A.1](#).

## 2.5 Ethernet TCP Communication Path

After preprocessing, the Flask backend sends the 1024-pixel input vector to the HPS board through Ethernet using TCP. TCP was selected because it provides reliable, ordered delivery. This is useful for this project because the complete image must arrive correctly and in order before inference starts.

The TCP client sends a request containing the image data. The HPS server receives the request, validates the input length, performs FPGA register access, and sends back a response containing the predicted digit and logits. Although TCP adds some communication overhead, it simplifies the software protocol and reduces the risk of corrupted or missing pixel data.

## 2.6 HPS Linux Inference Server

The HPS Linux inference server is implemented in C and runs on the ARM processor system of the DE1-SoC board. It listens for TCP connections from the Flask backend.

For each inference request, the server performs the following operations:

1. Receive 1024 input pixels through the TCP socket.
2. Open `/dev/mem`.
3. Map the Lightweight HPS-to-FPGA bridge address using `mmap()`.
4. Clear old FPGA status flags.
5. Write all input pixels into the FPGA pixel register window.
6. Write the control register to start inference.
7. Poll the status register until the FPGA asserts done.
8. Read the predicted digit and output logits.
9. Send the result back to the Flask client.

This C server acts as the bridge between network-level software communication and low-level FPGA register access.

## 2.7 Lightweight HPS-to-FPGA AXI Bridge

The Lightweight HPS-to-FPGA AXI bridge provides the memory-mapped communication path between the HPS and the FPGA fabric. From the HPS software perspective, FPGA registers appear as physical memory addresses starting at the lightweight bridge base address:

```
1 #define LW_BRIDGE_BASE 0xFF200000u
```

The HPS maps this region into user space through `/dev/mem` and `mmap()`. Once mapped, the C program accesses FPGA registers using normal pointer reads and writes. For example:

```
1 regs[REG_PIXEL_BASE_WORD + index] = pixels[index];
```

This C statement becomes a memory-mapped write. The HPS hardware converts it into an AXI write transaction across the lightweight bridge. On the FPGA side, the register interface receives the AXI signals, decodes the address, and stores the data.

## 2.8 FPGA CNN Accelerator Core

The FPGA fabric contains two major hardware blocks:

1. HPS-FPGA register interface
2. CNN accelerator core

The register interface is implemented in `hps_lenet_regs.sv`. It receives AXI read/write transactions from the HPS and exposes a simple control/status/data interface to the CNN core.

The CNN accelerator core is implemented in `lenet_int8_top.sv`. It performs the LeNet-style inference process using integer arithmetic. The main operations are:

Conv1 -> Pool1 -> Conv2 -> Pool2 -> FC1 -> FC2 -> FC3 -> Argmax

The accelerator uses on-chip BRAM/M10K memories for input pixels, intermediate feature maps, and packed parameter ROMs. Multiply-accumulate operations are reused across layers to reduce FPGA resource usage. Quantization is performed using shift-based rescaling and saturation logic.

## 2.9 End-to-End Inference Dataflow

The complete inference dataflow begins when the user draws or uploads a digit and ends when the predicted result is displayed in the browser.

1. The user draws a digit or uploads an image in the browser.
2. The browser sends the image to the Flask server.
3. Flask preprocesses the image into a  $32 \times 32$  grayscale pixel vector.
4. The Python client sends the 1024 pixels to the HPS C server over TCP.
5. The HPS C server receives and validates the pixel data.
6. The HPS maps the lightweight bridge using `/dev/mem` and `mmap()`.
7. The HPS writes pixels into the FPGA `PIXEL_INPUT` register window.
8. The HPS writes the control register to start the accelerator.
9. The FPGA CNN core performs inference.
10. The FPGA asserts done and makes the result/logits available.
11. The HPS polls status, then reads `RESULT` and `LOGIT[0:9]`.

12. The HPS sends the prediction response back to Flask.
13. Flask returns the result to the browser.
14. The browser displays the FPGA result, software reference result, logits, and timing information.

This end-to-end structure demonstrates the complete integration of web software, network communication, embedded Linux, memory-mapped HPS-FPGA access, and FPGA-based neural network acceleration.

### 3 CNN Algorithm and Model Design

#### 3.1 LeNet-5 Inspired CNN Structure

The hardware accelerator utilizes a convolutional neural network architecture based on LeNet-5, a pioneering structure developed for handwritten digit recognition. This implementation is tailored to classify MNIST-style digit images.

The design follows a sequential hierarchy consisting of two convolutional layers for feature extraction, two max-pooling stages for spatial downsampling, and three fully connected layers for classification. This structure provides a balance between accuracy and hardware cost. The model contains approximately 61,000 parameters, which is small enough to fit within the FPGA’s on-chip memory while still providing strong classification accuracy for handwritten digits.

Figure 4 shows the layer dimensions used by the LeNet-style classifier. The input is expanded into convolutional feature maps, reduced by subsampling, and finally passed through fully connected layers to generate ten digit scores.

Layer Name	Input Dim	Operation	Output Dim	Weights (Stored as int8)
Input	32×32×1	Raw Pixel Data	32×32×1	0
Conv1	32×32×1	5×5 Kernel, 6 Filters	28×28×6	150 weights + 6 biases
Pool1	28×28×6	2×2 Max Pooling	14×14×6	0
Conv2	14×14×6	5×5 Kernel, 16 Filters	10×10×16	2,400 weights + 16 biases
Pool2	10×10×16	2×2 Max Pooling	5×5×16	0
Flatten	5×5×16	Unspool to 1D Vector	400×1	0
FC1	400×1	Matrix Mult (120 nodes)	120×1	48,000 weights + 120 biases
FC2	120×1	Matrix Mult (84 nodes)	84×1	10,080 weights + 84 biases
FC3 (Out)	84×1	Matrix Mult (10 nodes)	10×1	840 weights + 10 biases

Figure 4: LeNet-style layer dimensions for 32 × 32 handwritten digit classification.

#### 3.2 Input Image Format: 32×32 Grayscale Pixels

The network consumes 32 × 32 grayscale images. Each input contains 1024 pixels. In the software preprocessing stage, pixels are represented as unsigned 8-bit values in the range 0 to 255. Inside the inference model, these values are interpreted in an integer-friendly format suitable for fixed-point computation.

Using a fixed 32 × 32 input size simplifies the hardware design because all layer dimensions and memory addresses can be statically determined. This also matches the LeNet-style structure, where the input image is progressively reduced by convolution and pooling operations.

### 3.3 Feature Extraction: Convolution and Pooling

Feature extraction is performed through two convolutional layers. The first convolutional layer uses six  $5 \times 5$  filters to detect simple low-level features such as edges and stroke directions. The second convolutional layer uses sixteen  $5 \times 5$  filters to capture more complex combinations of these features.

Between the convolutional layers,  $2 \times 2$  max-pooling is applied to downsample the feature maps. Pooling reduces computational cost, lowers memory usage, and provides some tolerance to small input shifts. After the convolution and pooling stages, the spatial resolution is reduced to a final  $5 \times 5$  feature map with 16 channels.

### 3.4 Classification: Flattening and Fully Connected Layers

After feature extraction, the final  $5 \times 5 \times 16$  feature map is flattened into a 400-element vector. This vector is then passed through three fully connected layers. The first two fully connected layers learn higher-level digit representations, while the final layer maps these learned features to the ten possible digit classes.

To improve hardware efficiency, the model uses 8-bit signed weights and ReLU activation functions in the hidden layers. ReLU is especially suitable for FPGA implementation because it only requires a sign check and zero replacement for negative values.

### 3.5 Output Logits and Digit Prediction

The final fully connected layer produces ten output logits, with each logit corresponding to one digit class from 0 to 9. These logits are kept at 32-bit precision to preserve accuracy during the final decision stage. An argmax operation is then applied to select the index with the highest logit value. This index becomes the predicted digit and is transmitted back to the HPS as the final classification result.

### 3.6 Hardware-Aware Architecture Design

The CNN architecture was selected because it efficiently extracts spatial features using shared weights and local kernels. Compared with a standard multilayer perceptron, a CNN greatly reduces memory usage while still detecting important patterns such as edges, curves, and strokes.

Several design choices were made to simplify FPGA implementation. Traditional sigmoid activations and average pooling were replaced with ReLU and max pooling. ReLU only requires simple comparison logic, while max pooling can be implemented with a small set of comparators. Finally, the model was quantized from 32-bit floating-point arithmetic to 8-bit signed integer arithmetic. This reduces the size and resource cost of the MAC units while maintaining strong recognition accuracy.

## 4 Quantization and Parameter Preparation

### 4.1 Need for Integer Quantization

The primary motivation for moving away from standard 32-bit floating-point representation is the resource limitation of FPGA hardware. Storing a model in FP32 requires significant memory, and implementing floating-point multiply-accumulate units consumes many logic resources and DSP blocks.

To improve efficiency and maximize inference speed, this project uses INT8 quantization. This provides a four-times reduction in model parameter storage compared with FP32. By using 8-bit signed integers, the design can use simpler integer MAC datapaths and store all model parameters in on-chip memory. This keeps power consumption, area usage, and memory bandwidth requirements manageable.

## 4.2 Training and Quantization Strategy

The training and quantization strategy followed a two-stage pipeline: high-precision training followed by fixed-point integer conversion. The model was first trained in PyTorch using 32-bit floating-point precision to establish a baseline that was easy to evaluate. Only after the floating-point model behaved well were the weights converted into the integer files used by the software reference and FPGA ROMs.

After training, the learned kernels and fully connected weights were frozen and converted for FPGA inference using post-training quantization. The network stores all convolutional and fully connected parameters as signed 8-bit integers. Instead of using complex affine scaling, a shift-based quantization approach was used. For each layer, a scaling factor was selected so that the FP32 values could be mapped into the INT8 range while preserving classification accuracy.

## 4.3 Implementation of Quantization Logic

Quantization is required on the FPGA because arithmetic operations increase bit width. Multiplying two 8-bit signed values produces a 16-bit result, and accumulating many products can expand intermediate values to 32 bits. To pass data into the next layer, these 32-bit accumulations must be scaled back to a smaller activation range.

The FPGA implementation uses shift-based rescaling instead of division. The main rounding operation is:

$$y = \frac{x + 2^{n-1}}{2^n}$$

which is implemented as:

$$y = (x + (1 \ll (n - 1))) \gg n$$

This approach performs round-to-nearest behavior before truncation. Dedicated clamping logic prevents wrap-around errors by saturating values that exceed the supported activation range.

The selected shift values are:

Table 2: Selected Layer Shift Values

Conv1 Shift	Conv2 Shift	FC1 Shift	FC2 Shift	Accuracy
6	8	8	7	98.20%

The output layer uses a shift value of zero so that the final logits remain in full 32-bit precision. This gives the argmax operation the highest possible resolution when selecting the predicted digit.

## 4.4 Parameter Files Used by Software and Hardware

The final project keeps trained parameters in `parameter/weights.bias_422`. The software reference path reads the unpacked INT8 hex files, while the FPGA accelerator reads lane-packed parameter files matched to `MAC_LANES = 3`. The relevant files are summarized in Table 3; the FPGA-ready packed files are included in Appendix B.

Table 3: Final Parameter Files Used by the Project

File Group	Files	Use
Software reference	<code>conv1.hex</code> , <code>conv2.hex</code> , <code>fc1.hex</code> <code>fc2.hex</code> , <code>fc3.hex</code>	Parsed by <code>predictor.py</code>
FPGA packed ROMs	<code>conv1_pack3.hex</code> , <code>conv2_pack3.hex</code> <code>fc1_pack3.hex</code> , <code>fc2_pack3.hex</code> <code>fc3_pack3.hex</code>	Loaded by <code>\$readmemh</code>
Layer diagram	<code>layer_dimension.png</code>	Report figure for model dimensions

## 4.5 Failure Analysis of Alternative Quantization Methodologies

Alternative methods such as affine post-training quantization and quantization-aware training were considered. Affine quantization can preserve numerical information through zero-point offsets and non-power-of-two scaling factors, but it increases hardware complexity because it requires additional offset correction and multiplication by scale factors.

Quantization-aware training was also explored, but it did not become the final path. In our experiments, the INT8 QAT version was harder to stabilize, and the extra training complexity was not worth the small benefit for this class project. An INT8-FP32 hybrid path was also tested, but the mixed precision made the software and hardware behavior harder to compare. For the final implementation, the simpler shift-based post-training quantization method was easier to verify layer by layer and matched the FPGA datapath more directly.

## 4.6 Static BRAM Configuration

To integrate the trained model into the hardware fabric, the finalized 8-bit signed weights and biases are exported from the Python environment into layer-specific hex memory initialization files. During Quartus compilation, these files are referenced by parameterized ROM modules through Verilog `INIT_FILE` parameters and loaded using `$readmemh`.

This static initialization makes the accelerator ready for inference immediately after FPGA configuration. By storing all parameters in on-chip BRAM or ROM-style memories, the CNN core can access weights and biases with deterministic low latency and avoid runtime parameter loading from the HPS.

# 5 Software Input Processing

## 5.1 Supported Input Sources

The handwritten digit recognition system supports two primary input modalities through its web-based interface. The first modality uses an interactive HTML5 canvas that allows the user to draw digits directly in the browser. The second modality allows the user to upload pre-existing image files in standard formats such as JPEG and PNG.

Both input paths are handled by the Flask backend. Regardless of input source, the image is converted into a common preprocessing pipeline. This ensures that both browser-drawn digits and uploaded images are normalized into the same  $32 \times 32$  grayscale format expected by the CNN accelerator.

## 5.2 Browser Canvas Input

The browser-based input mechanism uses an HTML5 canvas element as the drawing surface. The canvas provides a high-resolution input area for users to sketch handwritten digits. The frontend JavaScript code supports configurable brush width, clearing, prediction, upload, and optional color inversion.

When the user draws on the canvas, JavaScript event handlers capture mouse or touch coordinates, convert them into canvas-relative coordinates, and render continuous line segments. The stroke rendering uses round line caps and joins to create smooth handwritten strokes. The final canvas image is serialized as a PNG-format data URL and sent to the Flask backend through an HTTP POST request.

## 5.3 Uploaded JPG / PNG Image Input

The image upload path supports common image formats, especially JPEG and PNG. When a user selects a file, the browser reads the file and converts it to a data URL representation. This data URL is transmitted to the backend in base64-encoded form.

Uploaded images can have arbitrary dimensions and aspect ratios. Therefore, the preprocessing pipeline must crop, resize, center, and normalize them before inference. This makes it possible to use the same CNN accelerator for both drawn and uploaded digit images.

## 5.4 Flask Backend Image Reception

The Flask backend implements a prediction endpoint that receives JSON-formatted requests from the frontend. Each request contains an image field with a base64-encoded image string. The backend decodes the data URL, converts the raw bytes into a PIL image object, and passes the image to the preprocessing pipeline.

The backend also performs error handling. If the image data is missing, malformed, or cannot be decoded, the server returns a JSON error response to the browser. This prevents invalid image payloads from being forwarded to the HPS or FPGA.

## 5.5 Grayscale Conversion

The first preprocessing step is grayscale conversion. The PIL `convert("L")` method converts the input image into an 8-bit grayscale representation. If the input image contains an alpha channel, it is first composited onto a solid background to avoid transparency artifacts.

This step ensures that the subsequent CNN input contains one intensity value per pixel rather than RGB color channels. Since handwritten digit recognition primarily depends on shape rather than color, grayscale input is sufficient.

## 5.6 Foreground Cropping and Background Removal

After grayscale conversion, the preprocessing pipeline detects the foreground digit by applying an intensity threshold. Pixels above the threshold are treated as foreground. The system then computes a bounding box around all foreground pixels and crops the image to this region.

A small margin is added around the detected bounding box to preserve antialiased stroke edges and avoid excessive cropping. If no foreground pixels are detected, the backend returns an error indicating that no digit was found. This prevents blank inputs from being sent to the FPGA.

## 5.7 Resizing to $32 \times 32$ Pixels

The cropped digit may have arbitrary dimensions. Directly resizing it to  $32 \times 32$  could distort the aspect ratio, so the preprocessing pipeline uses aspect-ratio-preserving resizing. The digit is first scaled to fit inside a  $24 \times 24$  interior region, then placed on a  $32 \times 32$  black canvas.

This creates a border around the digit and makes the image more consistent with the training distribution. Centering and padding are important because the CNN model expects digit strokes to appear near the center of the image.

## 5.8 Conversion to 8-Bit Pixel Data

After spatial normalization, the image is converted into 8-bit pixel values. Optional smoothing and contrast adjustment are applied to improve robustness. The final image contains 1024 unsigned 8-bit values in row-major order.

These pixel values are directly used by the software reference model and also serialized for FPGA inference. Each pixel is represented as an integer between 0 and 255.

## 5.9 Generating FPGA-Compatible Input Format

The final  $32 \times 32$  grayscale image is converted into a Python list of 1024 integers using row-major order. For FPGA inference, the list is serialized as JSON and sent to the HPS C server through the TCP client.

The request includes the command type, request ID, image width, image height, and pixel list. This format is easy to debug and makes it possible to inspect the full input payload during development.

## 5.10 Preprocessing Limitations and Error Sources

The preprocessing pipeline has several limitations. A fixed foreground threshold may fail for unusual contrast conditions. Very thin strokes may disappear during resizing, while very thick strokes may merge and change the apparent digit shape. Center-of-mass adjustment can also shift noisy or poorly cropped inputs incorrectly.

These limitations affect recognition accuracy even when the FPGA hardware is correct. If the processed  $32 \times 32$  preview is distorted, the accelerator will classify that distorted input. Therefore, input quality and preprocessing robustness remain important parts of the overall system.

# 6 Software-Hardware Communication Protocol

## 6.1 Ethernet Architecture and Protocol Selection

The communication backbone of the system uses Ethernet to exchange data between the Flask application and the HPS backend. Ethernet was selected over lower-bandwidth serial protocols such as UART because each inference requires transmitting a 1024-pixel image payload. Ethernet provides sufficient bandwidth for interactive inference.

The system uses TCP/IP rather than UDP. TCP provides reliable, ordered delivery, which ensures that every pixel arrives correctly and in the proper sequence. This is important because the FPGA cannot begin inference until the full image has been received and validated.

## 6.2 TCP Implementation and JSON Payload Handling

The system uses a TCP client-server model. The HPS runs a C-based server on port 9090. The Flask backend acts as the client and sends a JSON request for each inference.

A typical request has the following form:

```
1 {  
2   "command": "infer",  
3   "request_id": "...",  
4   "width": 32,  
5   "height": 32,  
6   "pixels": [0, 0, ..., 255]  
7 }
```

The request is sent as a newline-terminated UTF-8 JSON string:

```
1 sock.sendall((json.dumps(request) + "\n").encode("utf-8"))
```

The newline termination makes the protocol simple because the receiver can treat one line as one complete request.

## 6.3 Inference Result Reporting and Transmission

Once the FPGA asserts the done and valid status bits, the HPS reads the inference outputs from hardware registers. These outputs include the predicted digit, the ten output logits, and the execution cycle count. The server packages these values into a structured JSON response and sends the response back to the Flask client over the same TCP connection.

A successful response contains the prediction, source information, debug status values, FPGA cycle count, compute time, and logits. This allows the browser interface to display both the classification result and useful diagnostic information.

## 6.4 Latency Measurement and Fair Performance Comparison

The Flask client can measure the full round-trip latency of each inference request, but this value includes TCP communication, JSON serialization, HPS control, register access, FPGA execution, and response handling. Therefore, it represents end-to-end system latency rather than pure CNN execution time.

For fair comparison, the FPGA core compute time is calculated using the cycle count returned by the accelerator:

$$T_{\text{FPGA}} = \frac{\text{cycle count}}{50 \text{ MHz}}$$

This cycle-based timing isolates the CNN computation from communication and software overhead.

## 6.5 HPS-Side JSON Parsing and Input Validation

On the HPS side, the C server receives the TCP payload using `recv()` and stores it in a request buffer. The server checks that the request contains the `infer` command and extracts the request ID.

The pixel array is parsed into a `uint8_t pixels[1024]` buffer. The server validates that the request contains exactly 1024 pixel values and that each value is between 0 and 255. If validation fails, the server returns an error response instead of writing invalid data into FPGA registers.

## 6.6 One-Connection-Per-Inference Design

The current implementation uses one TCP connection for one inference request. The Python client opens a socket, sends one JSON request, waits for one response, and then closes the connection.

This design was simple and reliable for an interactive lab demo. Each inference transaction is independent, so a failed request does not corrupt future requests or leave the HPS server in a half-finished state. The tradeoff is that connection setup and teardown add extra latency. A persistent TCP connection could reduce overhead in a higher-throughput implementation.

## 6.7 Error Handling and Response Format

The HPS server returns structured JSON responses. Successful responses contain `"ok": true`, while failures contain `"ok": false` and an error string. Implemented error cases include unsupported commands, invalid pixel payloads, `/dev/mem` access failure, invalid lightweight bridge readback, FPGA timeout, and response buffer overflow.

This structured format makes debugging easier because the frontend can identify whether the failure happened during network communication, input parsing, MMIO access, or FPGA execution.

## 6.8 Separation Between Network Protocol and Hardware Protocol

The project uses two different protocols at different system layers. At the software network layer, Flask and HPS communicate using TCP and JSON. At the HPS-to-FPGA hardware layer, the HPS communicates with FPGA registers using MMIO through `/dev/mem` and the Lightweight AXI bridge.

This separation keeps the FPGA logic simple. The FPGA does not parse JSON and does not implement a network stack. The HPS C server acts as a translator between the network protocol and the hardware register protocol.

## 6.9 Protocol Limitations

Although TCP and JSON are easy to debug, they are not the most efficient possible format. JSON is human-readable but larger than binary data because pixel values are transmitted as text. The current design is acceptable because the image is small, but for larger images or real-time video, a compact binary protocol would be more efficient.

The main limitations are JSON parsing overhead, one connection per inference, variable TCP latency, sequential HPS request handling, and the need to receive and write the entire pixel array before starting the FPGA. These limitations are acceptable for a classroom demo, but they would matter more for batched inputs or real-time use.

## 6.10 Code Mapping

The software-hardware communication path is distributed across several files.

Table 4: Communication Path Code Mapping

Function	File
Flask receives browser request	<code>web_demo/app.py</code>
Python TCP client builds JSON request	<code>web_demo/hps_client.py</code>
HPS C server listens on port 9090	<code>hps_infer_server.c</code>
HPS parses pixel payload	<code>hps_infer_server.c</code>
HPS writes pixels to FPGA registers	<code>hps_infer_server.c</code>
FPGA register constants	<code>hps_fpga_regs.h</code>
FPGA AXI register interface	<code>hps_lenet_regs.sv</code>
CNN accelerator core	<code>lenet_int8_top.sv</code>

The final HPS inference server and register header are included in Appendix [A.2](#).

## 7 HPS Software and MMIO Interface

### 7.1 HPS Linux Runtime Environment

The HPS side of the DE1-SoC runs an embedded Linux environment on the ARM processor system. This environment provides standard POSIX APIs, including file I/O, sockets, memory mapping, and signal handling. The inference server is implemented as a C program running in user space.

The HPS Linux environment is responsible for receiving input data from the network, accessing FPGA registers through memory-mapped I/O, and returning inference results to the client. The server also installs signal handlers so that it can shut down cleanly when interrupted.

### 7.2 C-Based Inference Server

The inference server is a single-threaded TCP server written in C. It follows the standard socket programming sequence: `socket()`, `bind()`, `listen()`, and `accept()`. The server listens on port 9090 and handles one client request at a time.

For each connection, the server reads the JSON payload, parses the input pixels, runs FPGA inference, builds a JSON response, sends it back to the client, and closes the socket. This single-threaded design is simple and sufficient for an interactive demonstration.

### 7.3 Receiving Pixel Data from TCP Socket

The server receives the request using `recv()` and stores it in a fixed-size buffer. It then checks for the expected `infer` command and parses the `pixels` array. Pixel values are converted from decimal strings into integers and stored in a 1024-element buffer.

The server verifies that exactly 1024 values are present. It also checks that every value is in the valid 8-bit range. These checks prevent incomplete or corrupted inputs from being written into the FPGA register window.

### 7.4 Memory-Mapped I/O Overview

Memory-mapped I/O allows software to access hardware registers as if they were normal memory locations. The FPGA accelerator exposes its control, status, result, logit, and pixel input registers

through the HPS-to-FPGA lightweight bridge. The HPS maps the physical bridge address into user space and then accesses the registers through a volatile pointer.

This avoids the need for a custom kernel driver, which simplifies development. However, using `/dev/mem` requires appropriate permissions and must be handled carefully because it gives access to physical memory.

## 7.5 Relationship Between `/dev/mem`, `mmap()`, and MMIO

The `/dev/mem` device file represents the physical memory address space. The HPS server opens this file and uses `mmap()` to map the lightweight bridge physical address into the process virtual address space.

The mapping uses the following key parameters:

- Physical base address: `0xFF200000`
- Mapping size: `0x1030` bytes
- Protection: read and write
- Mapping type: shared

The user-space mapping size is chosen to cover every accelerator register from the first control/status word through the final pixel input word. The last pixel register is `PIXEL_INPUT[1023]`, so the final used byte offset is:

$$\text{last\_offset} = 0x0030 + 1023 \times 4 = 0x102C$$

Because `PIXEL_INPUT[1023]` is a 32-bit register, the mapped user-space region must include four bytes starting at `0x102C`. Therefore, the required mapping length is:

$$\text{user\_space\_size} = \text{last\_offset} + 4 = 0x102C + 4 = 0x1030 \text{ bytes}$$

After `mmap()` succeeds, the physical FPGA register range:

$$0xFF200000 \leq \text{physical address} < 0xFF201030$$

is visible through a user-space virtual base pointer. A byte offset  $\Delta$  from the lightweight bridge base is accessed as:

$$\text{user\_addr}(\Delta) = \text{user\_base} + \Delta$$

Equivalently, after casting the mapping to `volatile uint32_t *`, software accesses a register by word index:

$$\text{regs}[\text{word\_index}], \quad \text{word\_index} = \frac{\Delta}{4}$$

After mapping, the returned address is cast to `volatile uint32_t *`. The `volatile` keyword is essential because it prevents the compiler from optimizing away reads and writes to hardware registers.

## 7.6 Lightweight HPS-to-FPGA AXI Bridge

The lightweight bridge connects the HPS processor to custom FPGA logic. Each pointer read or write in the C program becomes an AXI transaction across this bridge. On the FPGA side, the register interface decodes the address and performs the corresponding register operation.

The bridge is used for pixel writes, control writes, status polling, result reads, and logit reads. This provides a compact and deterministic interface between software and hardware.

## 7.7 Physical Address to Virtual Address Mapping

The `open_fpga_regs()` function encapsulates the mapping process. It opens `/dev/mem`, maps the FPGA register region, and returns a pointer that can be used for register access.

For example, `regs[0]` corresponds to the control/status register at physical address `0xFF200000`. Similarly, `regs[12]` corresponds to the first pixel input register at byte offset `0x0030`.

## 7.8 Register Read Function

The HPS reads the result register to obtain the predicted digit and cycle count. The lower four bits contain the predicted digit, and the upper bits contain the FPGA cycle count.

```
1 uint32_t result_word = regs[REG_RESULT_WORD];
2 *predicted_digit = (int)(result_word & 0xfu);
3 *cycle_count = result_word >> 4;
```

The server also reads ten signed 32-bit logits from consecutive logit registers. These logits are returned to the browser for debugging and confidence analysis.

## 7.9 Register Write Function

Register writes are used to upload pixels and issue control commands. The HPS writes each pixel to the pixel input window:

```
1 for (index = 0; index < EXPECTED_PIXELS; ++index) {
2     regs[REG_PIXEL_BASE_WORD + index] = pixels[index];
3 }
```

Only the lower 8 bits of each 32-bit word are used by the FPGA. Control commands are issued by writing to the control/status register.

## 7.10 Pixel Upload Sequence

The pixel upload sequence begins by clearing stale status flags. Then the HPS writes 1024 pixel values into the pixel window. The register interface tracks the number of pixels written and asserts the input-loaded status bit when the full image has been received.

This transfer method is simple and reliable. Its main limitation is that it requires 1024 separate MMIO writes for each image.

## 7.11 Accelerator Start Sequence

After pixel upload, the server starts the accelerator by writing:

```
1 regs[REG_CONTROL_STATUS_WORD] = CTRL_START | CTRL_CLEAR_DONE;
```

The FPGA register interface converts this write into a one-cycle `start_pulse`. The CNN core then begins executing the fixed inference sequence.

## 7.12 Done Polling Sequence

After issuing the start command, the server polls the status register until the done bit becomes high. The polling loop includes a timeout to avoid waiting forever if the hardware enters an invalid state.

Once the done bit is set, the server waits for the result-valid bit before reading the result and logits. This ensures that output registers are stable before readback.

## 7.13 Reading Prediction and Logits

After inference completes, the server reads the packed result word and the ten logit registers. The predicted digit is the argmax result computed by hardware. The logits are signed 32-bit scores corresponding to the ten digit classes.

Returning logits is useful because it allows the browser interface to display more than just the final predicted digit. If the top two logits are close, the model is less confident.

## 7.14 Returning Results to the Software Client

The server builds a JSON response containing the predicted digit, logits, FPGA cycle count, compute time, status values, and request ID. The response is sent back through the TCP connection.

This completes one inference transaction from browser input to FPGA computation and back to browser display.

# 8 Register Map Design

## 8.1 Register Map Overview

The HPS communicates with the FPGA accelerator through a memory-mapped register interface exposed on the Lightweight HPS-to-FPGA bridge. The register map is implemented in `hps_lenet_regs.sv` and accessed by the HPS C inference server through `/dev/mem` and `mmap()`.

The base physical address of the lightweight bridge is `0xFF200000`. All accelerator registers are placed as offsets from this base address.

Table 5: HPS-FPGA Register Map Overview

Start-End Offset	Name	Access	Comment
0x0000-0x0004	CONTROL_STATUS	RW	Write control commands; read accelerator status
0x0004-0x0008	RESULT	RO	Predicted digit and FPGA cycle count
0x0008-0x0030	LOGIT_0-LOGIT_9	RO	Ten signed 32-bit final logit values
0x0030-0x1030	PIXEL_INPUT[0:1023]	RW	Runtime input image window

## 8.2 Register Address Space Allocation

The total mapped register space is `0x1030` bytes. The address space is allocated as shown in Table 6.

Table 6: Register Address Space Allocation

Offset Range	Register Region
0x0000	CONTROL_STATUS
0x0004	RESULT
0x0008--0x002C	LOGIT_0 to LOGIT_9
0x0030--0x102C	PIXEL_INPUT[0] to PIXEL_INPUT[1023]

The final used register is `PIXEL_INPUT[1023]` at offset:

$$0x0030 + 1023 \times 4 = 0x102C$$

Since each register is 4 bytes, the mapped region must extend to:

$$0x102C + 4 = 0x1030$$

### 8.3 Address Alignment Requirement

All registers are 32-bit aligned. This means every register offset is a multiple of 4 bytes. In hardware, the byte address is converted to a word index by dropping the lower two address bits:

$$\text{word\_index} = \text{addr}[13 : 2]$$

Using aligned 32-bit registers simplifies both HPS software pointer arithmetic and FPGA address decoding logic.

### 8.4 Reason for Using 32-Bit Registers

The register interface uses 32-bit registers because the HPS lightweight bridge and CPU MMIO accesses naturally operate on 32-bit words. Although each input pixel only needs 8 bits, using one 32-bit register per pixel keeps the interface simple and reliable.

For pixel writes, only the lower 8 bits are used. For logits and result values, the full 32-bit register is used.

### 8.5 Reason for Separating Control, Status, Result, Logit, and Pixel Regions

The register map separates different functions into different regions to make the hardware/software protocol clear. The control/status register is accessed frequently, so it is placed at the beginning. The result register follows immediately after it. The ten logit registers are grouped together, and the large pixel input window is placed after the control and output region.

This separation improves readability, prevents accidental overlap, and makes address calculation simple.

### 8.6 Base Address Selection

The HPS software accesses the FPGA register interface through the lightweight bridge base address:

```
1 #define LW_BRIDGE_BASE 0xFF200000u
```

The C server maps this physical address using `/dev/mem` and `mmap()`, then accesses FPGA registers by adding word offsets.

## 8.7 Offset Address Definition

The software defines register addresses as word offsets:

```
1 #define REG_CONTROL_STATUS_WORD 0u
2 #define REG_RESULT_WORD         1u
3 #define REG_LOGIT_BASE_WORD     2u
4 #define REG_PIXEL_BASE_WORD     12u
```

Since each word is 4 bytes, the pixel input window starts at byte offset:

$$12 \times 4 = 0x0030$$

## 8.8 Control Register

The control register is written by the HPS at offset `0x0000`. It is used to send command pulses to the FPGA register interface.

Table 7: Control Register Bit Fields

Bit	Name	Function
bit 0	CTRL_START	Start one CNN inference
bit 1	CTRL_CLEAR_DONE	Clear the done flag inside the CNN core
bit 2	CTRL_CLEAR_INPUT_LOADED	Clear the input loaded flag
bit 31:3	Reserved	Unused

## 8.9 Status Register

The same offset `0x0000` is read as a status register. This means the register is write-control and read-status.

Table 8: Status Register Bit Fields

Bit	Name	Function
bit 0	STATUS_REGS_VALID	Output registers and logits are valid
bit 1	STATUS_BUSY	Accelerator is currently running
bit 2	STATUS_DONE	Accelerator has finished inference
bit 3	STATUS_INPUT_LOADED	All 1024 input pixels have been written
bit 31:4	Reserved	Read as zero

## 8.10 Result Register

The result register is located at offset `0x0004` and is read-only from the HPS side.

Table 9: Result Register Bit Fields

Bits	Field	Function
bit 3:0	predicted_digit	Final predicted digit, 0 to 9
bit 31:4	cycle_count	FPGA core cycle count

The HPS C server unpacks it using:

```

1 *predicted_digit = (int)(result_word & 0xfu);
2 *cycle_count = result_word >> 4;

```

Figure 5 gives the same protocol in bit-field form. It highlights that the write-side control bits and read-side status bits share the same base offset, while the result register packs the predicted digit with the cycle counter.

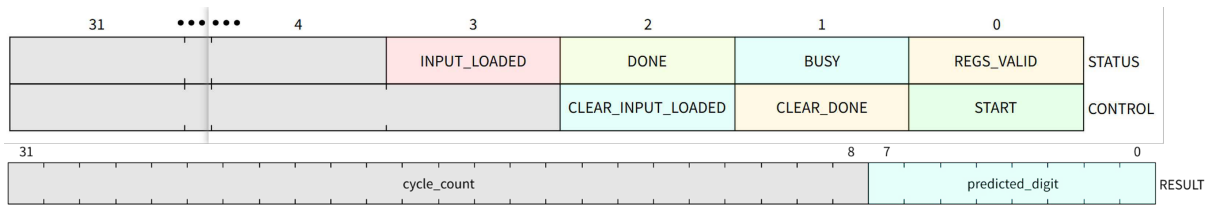


Figure 5: Detailed control, status, and result register bit-field layout.

### 8.11 LOGIT[0:9] Registers

The ten logit registers start at offset 0x0008. Each register stores one signed 32-bit final score.

Table 10: Logit Register Address Table

Register	Offset	Function
LOGIT_0	0x0008	Signed score for digit 0
LOGIT_1	0x000C	Signed score for digit 1
LOGIT_2	0x0010	Signed score for digit 2
LOGIT_3	0x0014	Signed score for digit 3
LOGIT_4	0x0018	Signed score for digit 4
LOGIT_5	0x001C	Signed score for digit 5
LOGIT_6	0x0020	Signed score for digit 6
LOGIT_7	0x0024	Signed score for digit 7
LOGIT_8	0x0028	Signed score for digit 8
LOGIT_9	0x002C	Signed score for digit 9

### 8.12 PIXEL\_INPUT[0:1023] Register Window

The pixel input window starts at offset 0x0030. It contains 1024 registers, one for each pixel of the 32 x 32 input image. Each pixel register is 32 bits wide, but only bits [7 : 0] are used.

The HPS writes pixels using:

```
1 regs[REG_PIXEL_BASE_WORD + index] = pixels[index];
```

The FPGA register interface writes the lower byte into the accelerator input memory.

### 8.13 Pixel Register Address Calculation

The pixel register byte offset is:

$$\text{PIXEL\_INPUT}[i] = 0x0030 + 4i$$

where  $i = 0, 1, \dots, 1023$ . In hardware, the byte address is converted into a word index, and the pixel index is calculated by subtracting the pixel base word.

### 8.14 Register Reset Values

On reset, the register interface clears command pulses, read/write state, and input status. The result and logit values are generated by the CNN accelerator core, so software should rely on `STATUS_DONE` and `STATUS_REGS_VALID` before treating result registers as meaningful.

### 8.15 Register Read / Write Permissions

Table 11: Register Read and Write Permissions

Region	Access	Description
<code>CONTROL_STATUS</code>	RW	Write commands, read status bits
<code>RESULT</code>	RO	Read predicted digit and cycle count
<code>LOGIT_0–LOGIT_9</code>	RO	Read final output logits
<code>PIXEL_INPUT[0:1023]</code>	RW	Write input pixels, optional debug readback

### 8.16 Start / Done Handshake Register Protocol

The HPS software follows a polling-based start/done protocol:

1. Read current status from `CONTROL_STATUS`.
2. Write `CTRL_CLEAR_DONE | CTRL_CLEAR_INPUT_LOADED`.
3. Write 1024 input pixels into `PIXEL_INPUT[0:1023]`.
4. Wait for the FPGA to set `STATUS_INPUT_LOADED`.
5. Write `CTRL_START | CTRL_CLEAR_DONE`.
6. Poll `CONTROL_STATUS` until `STATUS_DONE` becomes 1.
7. Continue checking until `STATUS_REGS_VALID` becomes 1.
8. Read `RESULT`.
9. Read `LOGIT_0` to `LOGIT_9`.

## 8.17 Complete Register Address Table

Table 12: Complete Register Address Table

Offset	Name	Function
0x0000	CONTROL_STATUS	Write control commands; read status
0x0004	RESULT	Predicted digit and cycle count
0x0008	LOGIT_0	Logit score for digit 0
0x000C	LOGIT_1	Logit score for digit 1
0x0010	LOGIT_2	Logit score for digit 2
0x0014	LOGIT_3	Logit score for digit 3
0x0018	LOGIT_4	Logit score for digit 4
0x001C	LOGIT_5	Logit score for digit 5
0x0020	LOGIT_6	Logit score for digit 6
0x0024	LOGIT_7	Logit score for digit 7
0x0028	LOGIT_8	Logit score for digit 8
0x002C	LOGIT_9	Logit score for digit 9
0x0030–0x102C	PIXEL_INPUT[0:1023]	Runtime input image window

## 8.18 Register Map Verification

The register map was verified from both the software and hardware sides. On the software side, the HPS C server uses constants from `hps_fpga_regs.h`. On the hardware side, `hps_lenet_regs.sv` decodes AXI addresses into word indices using `addr[13:2]`.

End-to-end testing confirmed that 1024 pixels can be written into the pixel window, the accelerator can be started through the control register, the HPS can poll the done bit, and all result/logit registers can be read back correctly. The final FPGA register-interface RTL is included with the accelerator source code in Appendix A.3.

# 9 FPGA Hardware Architecture

## 9.1 Top-Level FPGA Design

The FPGA hardware design integrates the HPS subsystem, a memory-mapped register interface, and a custom CNN accelerator core. The top-level structure connects the HPS lightweight AXI bridge to FPGA logic so that software running on the HPS can control the accelerator through memory-mapped registers.

The main FPGA-side components are summarized in Table 13.

Table 13: Top-Level FPGA Components

Component	Role
<code>soc_system_top</code>	Board-level integration wrapper
Platform Designer HPS subsystem	ARM, DDR, Ethernet, and lightweight bridge
<code>hps_lenet_regs</code>	Memory-mapped register interface
<code>lenet_int8_top</code>	CNN accelerator datapath and control FSM
BRAM / packed ROM memories	Runtime pixels, feature maps, weights, and biases

Figure 6 shows the detailed hardware organization. The diagram emphasizes the separation between the HPS subsystem, the memory-mapped register interface, and the `lenet_int8_top` inference core.

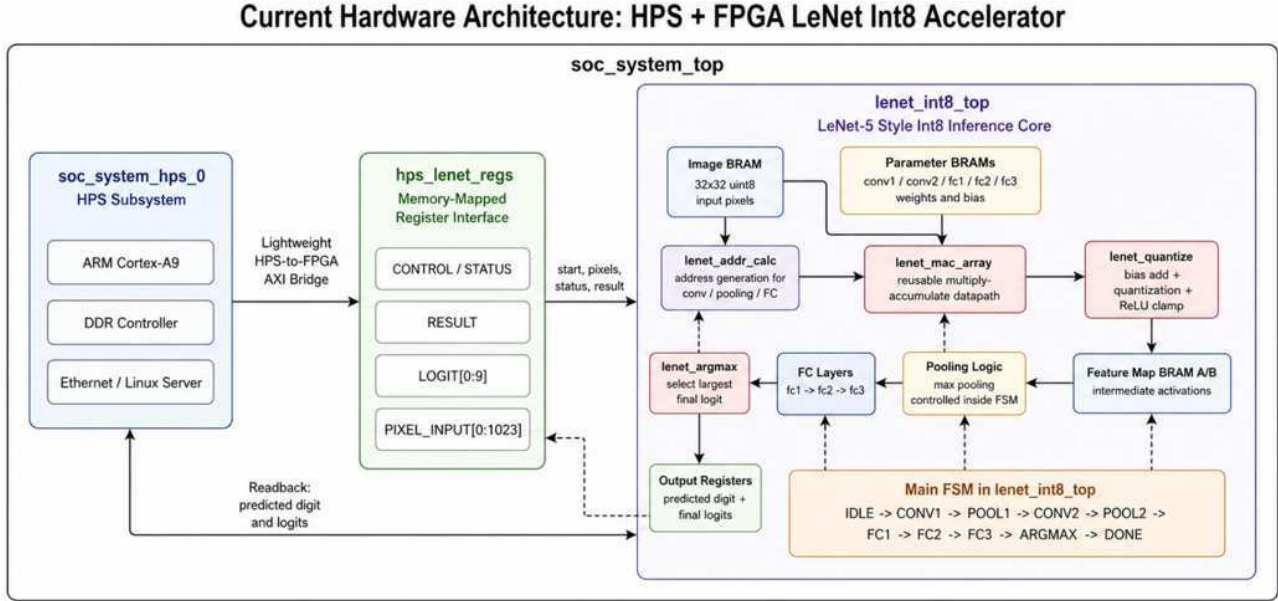


Figure 6: Current HPS and FPGA LeNet INT8 accelerator architecture.

## 9.2 soc\_system\_top Module

`soc_system_top.sv` is the board-level top module. It connects the physical DE1-SoC board pins, HPS subsystem signals, lightweight AXI bridge signals, and custom FPGA accelerator logic.

Its main responsibilities are to instantiate the Platform Designer HPS system, connect HPS peripherals, route lightweight AXI signals into the FPGA register interface, instantiate `hps_lenet_regs.sv`, instantiate `lenet_int8_top.sv`, and connect control/status/pixel/result/logit signals between the register interface and CNN core.

## 9.3 HPS Subsystem Integration

The HPS subsystem is generated by Platform Designer and instantiated inside the top-level design. It contains the ARM processor system and HPS peripherals required to run Linux and communicate with the outside world.

The HPS subsystem provides the ARM processor, DDR interface, Ethernet interface, SD card interface, UART/debug interface, and Lightweight HPS-to-FPGA bridge. In this project, the most important connection is the lightweight bridge, which allows the HPS C server to access FPGA registers.

## 9.4 hps\_lenet\_regs.sv Register Interface

`hps_lenet_regs.sv` implements the FPGA-side memory-mapped register interface. It receives AXI read and write transactions from the lightweight bridge and converts them into simple accelerator control signals.

The register interface decodes AXI byte addresses into 32-bit word indices, handles AXI read/write responses, generates `start_pulse`, writes input pixels into the CNN input buffer, and returns status, result, logits, and pixel readback data to the HPS.

## 9.5 lenet\_int8\_top.sv CNN Accelerator Core

lenet\_int8\_top.sv is the main CNN inference core. It receives input pixels from the register interface and executes the LeNet-style inference sequence:

$$\text{Conv1} \rightarrow \text{Pool1} \rightarrow \text{Conv2} \rightarrow \text{Pool2} \rightarrow \text{FC1} \rightarrow \text{FC2} \rightarrow \text{FC3}$$

The core uses integer arithmetic. Weights are stored as 8-bit values, activations are stored in fixed-width integer format, and MAC operations use wider accumulators to reduce overflow risk.

## 9.6 Clock and Reset Design

The accelerator uses the board 50 MHz clock as the main FPGA logic clock. This clock drives the register interface, CNN core, BRAMs, and control FSMs. The reset signal initializes the accelerator state, clears output registers, resets counters, and places the FSM into the idle state.

The design meets the 50 MHz target clock. The reported Fmax for clock\_50\_1 is 61.94 MHz, giving positive timing margin.

## 9.7 Input Pixel Buffer

The input image is stored in an on-chip BRAM-based pixel buffer. The input size is  $32 \times 32 = 1024$  pixels. Each pixel is an unsigned 8-bit grayscale value.

The HPS writes pixels through the register interface, which converts AXI writes into pixel\_wr\_en, pixel\_wr\_addr, and pixel\_wr\_data. These signals are connected to the CNN core.

## 9.8 Weight and Bias Memory

Weights and biases are stored in packed ROMs initialized from hex files. The design uses five parameter ROM groups: Conv1, Conv2, FC1, FC2, and FC3. Each packed ROM word contains multiple 8-bit parameters aligned with the MAC lane structure.

## 9.9 Intermediate Feature Map Storage

Intermediate activations are stored in on-chip feature map memories. The design uses two feature map buffer groups, allowing one buffer to act as source while another acts as destination. This ping-pong strategy reduces storage duplication and avoids off-chip intermediate memory.

## 9.10 Output Logit Storage

The final fully connected layer produces ten output logits. These logits are stored as signed 32-bit values and flattened into a packed bus called logits\_flat. The register interface maps this bus into ten read-only registers.

## 9.11 Top-Level Signal Connections

Table 14: Top-Level Signal Connections Between Register Interface and CNN Core

Signal	Direction	Purpose
start_pulse	hps.lenet_regs → lenet.int8_top	Start one inference
clear_done_pulse	hps.lenet_regs → lenet.int8_top	Clear previous done state
pixel_wr_en	hps.lenet_regs → lenet.int8_top	Input pixel write enable
pixel_wr_addr	hps.lenet_regs → lenet.int8_top	Input pixel write address
pixel_wr_data	hps.lenet_regs → lenet.int8_top	Input pixel write data
busy	lenet.int8_top → hps.lenet_regs	Accelerator is running
done	lenet.int8_top → hps.lenet_regs	Accelerator finished
predicted_digit	lenet.int8_top → hps.lenet_regs	Final predicted digit
logits_flat	lenet.int8_top → hps.lenet_regs	Ten final logits
cycle_count	lenet.int8_top → hps.lenet_regs	Number of inference cycles
regs_valid	lenet.int8_top → hps.lenet_regs	Result/logit registers are valid

## 9.12 FPGA Hardware Block Diagram

The simplified accelerator view in Figure 7 shows the main execution order inside the FPGA. The HPS writes pixels and control commands through the lightweight bridge, while the FPGA core advances through convolution, pooling, and fully connected stages before exposing the final answer and logits.

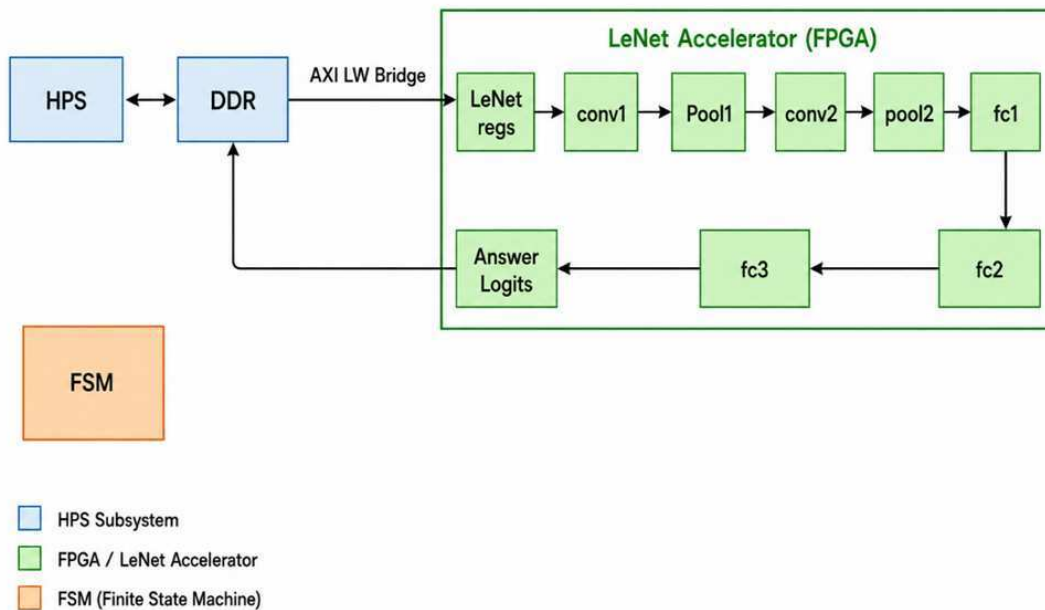


Figure 7: Simplified HPS-to-FPGA accelerator block diagram.

The final top-level FPGA wrapper and CNN RTL source files are listed in Appendix A.3.

## 10 CNN Accelerator Hardware Design

### 10.1 Execution Flow and Global FSM

The accelerator executes a fixed LeNet-style inference sequence:

$$\text{Conv1} \rightarrow \text{Pool1} \rightarrow \text{Conv2} \rightarrow \text{Pool2} \rightarrow \text{FC1} \rightarrow \text{FC2} \rightarrow \text{FC3} \rightarrow \text{Argmax}$$

This flow is implemented in `lenet_int8_top.sv` using a single global finite-state machine. The FSM contains explicit per-stage states for initialization, memory reads, accumulation, bias addition, quantization, writeback, argmax, and completion.

```
1 assign busy = (state != ST_IDLE) && (state != ST_DONE);  
2 assign done = (state == ST_DONE);
```

A start pulse launches one full inference pass, and the FSM returns to idle after completion.

Table 15 summarizes the major state-group transitions in the global CNN FSM. Each stage advances when the corresponding loop counters reach the final output element and the final MAC group for that output has completed.

Table 15: Overall CNN FSM State-Group Transitions

Current State Group	Transition Condition	Next State Group
ST_IDLE	<code>start == 1</code>	Conv1 states
Conv1 states	<code>out_ch == 5, out_y == 27, out_x == 27, term_idx + 3 &gt;= 25</code>	Pool1 states
Pool1 states	<code>pool_ch == 5, pool_y == 13, pool_x == 13, pool_read_idx == 3</code>	Conv2 states
Conv2 states	<code>out_ch == 15, out_y == 9, out_x == 9, term_idx + 3 &gt;= 150</code>	Pool2 states
Pool2 states	<code>pool_ch == 15, pool_y == 4, pool_x == 4, pool_read_idx == 3</code>	FC1 states
FC1 states	<code>neuron_idx == 119, term_idx + 3 &gt;= 400</code>	FC2 states
FC2 states	<code>neuron_idx == 83, term_idx + 3 &gt;= 120</code>	FC3 states
FC3 states	<code>neuron_idx == 9, term_idx + 3 &gt;= 84</code>	ST_ARGMAX
ST_ARGMAX	fixed 1 cycle	ST_DONE
ST_DONE	fixed 1 cycle	ST_IDLE

### 10.2 Convolution Datapath

Both convolution layers reuse the same datapath template. During read states, activation data and packed weight data are fetched from BRAM and ROM. During accumulation, lane inputs are fed into the MAC array. During writeback, bias and quantization are applied before storing the output activation.

Conv1 reads from image memory and writes to one feature map bank. Conv2 reads from a feature map bank and writes to another. Address generation is delegated to `lenet_addr_calc.sv`, avoiding hard-coded address logic inside the FSM.

Figure 8 records the earlier convolution and pooling datapath sketch. It is included as design context because it shows the same core ideas used in the final RTL: input BRAM, pixel buffering, kernel memory, MAC accumulation, bias/ReLU, pooling, and feature-map memory writeback.

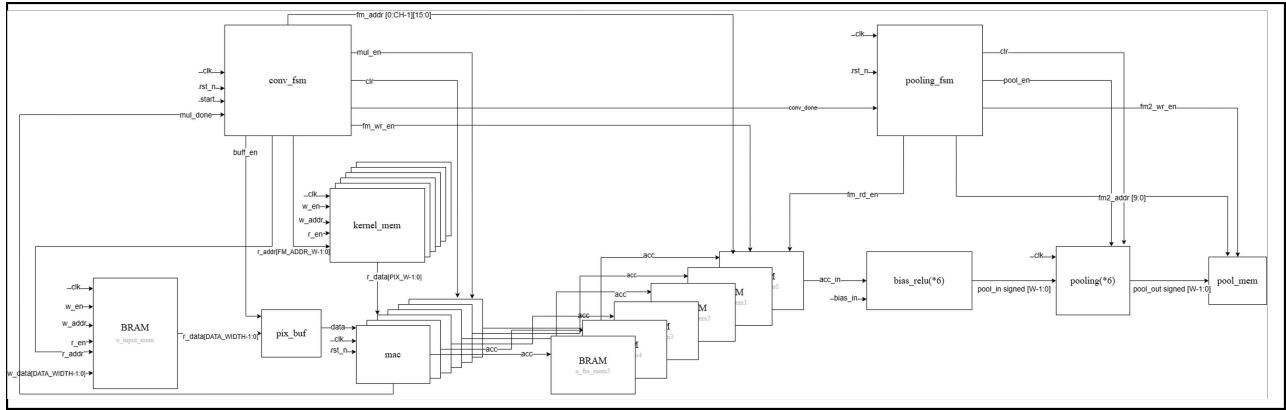


Figure 8: Datapath sketch for convolution, activation, pooling, and feature-map memory organization.

### 10.3 PE and MAC Design

The processing element is implemented as a lane-parallel MAC unit in `lenet_mac_array.sv`. The design uses `MAC_LANES = 3`. Each lane multiplies one activation value and one signed weight value per cycle. The lane products are then reduced into a shared 32-bit partial sum.

```

1 for (lane = 0; lane < MAC_LANES; lane = lane + 1) begin
2   if (valid_mask[lane]) begin
3     partial_sum = partial_sum + (act_value * weight_value);
4   end
5 end

```

A valid mask disables out-of-range tail lanes when the vector length is not divisible by the lane count.

### 10.4 Bias Addition and Quantization

Bias values are stored in the same packed parameter memory space as weights. For each output channel or neuron, the controller fetches the bias after the final MAC segment and adds it to the accumulated value.

The quantizer applies right-shift scaling with rounding and saturation. Negative values are clamped to zero for ReLU-style activation. Layer-specific shift constants are defined in `lenet_int8_pkg.sv`.

```

1 assign acc_with_bias = acc_reg +
2   {{(ACC_W-BIAS_W){current_bias[BIAS_W-1]}}, current_bias};
3
4 if (value_in <= 0)
5   value_out = '0;

```

### 10.5 Max Pooling Datapath

Pooling is implemented as  $2 \times 2$  max pooling. For each pooled output, four source activations are read sequentially, compared against a running maximum, and written to the next feature map bank.

This implementation is simple and deterministic. It reduces control complexity and matches the staged FSM structure.

## 10.6 Flatten / Fully Connected Datapath

There is no separate flatten module in hardware. Flattening is realized implicitly through linear indexing when FC layers read pooled feature maps. FC1, FC2, and FC3 reuse the same MAC and accumulator pattern as convolution, with different loop bounds and parameter address formulas.

FC1 and FC2 outputs are quantized activations written back to BRAM, while FC3 keeps full accumulated logits for final classification.

## 10.7 Output and Completion Signaling

Final prediction is produced by an argmax module that scans all ten logits and outputs the index of the maximum score.

```

1  if (score > best_score) begin
2      best_score = score;
3      best_idx = idx;
4  end
5  digit = best_idx[3:0];

```

The logits are exposed through register-mapped outputs for software-side inspection. Completion is indicated by entering `ST_DONE`; system-level logic latches completion so HPS polling remains robust.

## 10.8 Hardware Latency Analysis

Latency is measured in hardware using a cycle counter driven by the 50 MHz system clock. The HPS reads this counter and converts it to compute time. This number is the cleanest measurement of the CNN core itself. End-to-end response latency is reported separately because it also includes pixel writes, polling, TCP communication, JSON parsing, and browser-side handling.

Table 16 shows the cycle-level breakdown of the CNN core. The cycle estimate follows the staged datapath structure: each output element requires setup cycles, repeated three-lane MAC groups, and a final write or transition cycle. The convolution and fully connected stages dominate runtime because they perform the largest number of multiply-accumulate operations.

Table 16: CNN Core Stage Cycle Breakdown

Stage	Output Count	Cycles Needed per Output	Total Cycles
Start	1	1	1
Conv1	$6 \times 28 \times 28 = 4704$	$1 + \lceil 25/3 \rceil \times 3 + 1 = 29$	136,416
Pool1	$6 \times 14 \times 14 = 1176$	$1 + 4 \times 2 + 1 = 10$	11,760
Conv2	$16 \times 10 \times 10 = 1600$	$1 + \lceil 150/3 \rceil \times 3 + 1 = 152$	243,200
Pool2	$16 \times 5 \times 5 = 400$	10	4,000
FC1	120	$1 + \lceil 400/3 \rceil \times 3 + 1 = 404$	48,480
FC2	84	$1 + \lceil 120/3 \rceil \times 3 + 1 = 122$	10,248
FC3	10	$1 + \lceil 84/3 \rceil \times 3 + 1 = 86$	860

The table sums to 454,965 cycles. The measured counter value is 454,966 cycles because the implementation also includes the final completion/accounting cycle before `done` is observed by the surrounding control logic.

## 11 Memory Architecture

### 11.1 Memory Requirement Analysis

The design targets a  $32 \times 32$  grayscale input and multiple intermediate feature maps for convolution, pooling, and fully connected processing. Layer dimensions and parameter counts are fixed in `lenet_int8_pkg.sv`. The design uses on-chip memory for input pixels, feature maps, and parameters.

### 11.2 Input Pixel Storage

Input pixels are written by the HPS through the lightweight AXI memory-mapped pixel window. The pixel input window contains 1024 words for one complete input frame.

On the FPGA side, these writes drive the image BRAM interface through `pixel_wr_en`, `pixel_wr_addr`, and `pixel_wr_data`. This runtime loading path removes dependence on static input files and enables online inference requests.

### 11.3 Kernel / Bias Parameter Memory

Layer parameters are stored in dedicated packed ROM blocks. The ROMs use one initialization file per layer group. Data are lane-packed to match the MAC lane count, so one memory read returns multiple weights aligned to MAC consumption. Biases are appended after the weight region in the same address space.

```
1 (* ramstyle = "M10K" *) reg [LANES*DATA_WIDTH-1:0] mem [0:WORD_COUNT-1];
2
3 initial begin
4     if (INIT_FILE != "") begin
5         $readmemh(INIT_FILE, mem);
6     end
7 end
```

### 11.4 Feature Map Memory Organization

Intermediate activations are stored in two on-chip feature map banks using a ping-pong strategy. One bank acts as producer destination while the other acts as consumer source, then roles swap at stage boundaries. This minimizes conflicts and avoids off-chip intermediate storage.

### 11.5 BRAM / M10K Usage

Both writable memories and parameter ROMs request M10K mapping through synthesis attributes. This provides predictable timing and local bandwidth for the staged read-accumulate-write schedule.

```

1  (* ramstyle = "M10K" *) reg [DATA_WIDTH-1:0] mem [0:(1<<ADDR_WIDTH)-1];
2
3  always @(posedge clk) begin
4      if (w_en)
5          mem[w_addr] <= w_data;
6      if (r_en)
7          r_data <= mem[r_addr];
8  end

```

## 11.6 Parameter Initialization Using \$readmemh and .memh File Format

Parameters are initialized using `$readmemh`, enabling deterministic startup without runtime weight transfer from the HPS. The files are hex text files arranged to match the hardware packing format. The software reference pipeline follows the same signed-INT8 interpretation, supporting cross-validation between software and hardware behavior.

## 11.7 Addressing and Access Scheduling

Address generation for convolution and pooling is centralized in `lenet_addr_calc.sv`. Fully connected stages use linear indexing with per-neuron row offsets. Memory access is explicitly staged to match synchronous memory behavior. This improves correctness, readability, and cycle-level traceability.

## 11.8 Reuse Strategy and Bandwidth Limitations

The architecture emphasizes reuse. One MAC datapath serves all layers, two feature-map banks serve intermediate tensors, and preloaded ROMs serve parameter reads. This is area-efficient and robust for an educational FPGA platform.

The tradeoff is throughput. Bandwidth is constrained by sequential FSM scheduling, synchronous memory reads, and modest lane parallelism. Future speedups would require more lanes, deeper pipelining, or more memory banking.

# 12 Verification and Testing

## 12.1 Verification Overview

Verification was performed at multiple levels: software model verification, quantized model verification, RTL simulation, HPS register interface testing, FPGA on-board testing, and final browser-to-FPGA end-to-end testing.

The main purpose was to avoid debugging the whole system as one black box. The flow was checked in layers: first the software reference and preprocessing, then the quantized model, then RTL simulation, then register readback on the board, and finally the full browser-to-FPGA path.

## 12.2 Python Golden Model

A Python reference model was implemented in the web demo backend. It uses the same trained parameter files and performs the same LeNet-style structure: Conv1, Pool1, Conv2, Pool2, FC1, FC2, and FC3.

This model is used as the first-level reference because it is easier to inspect and debug than hardware. It allows preprocessing, parameter loading, layer dimensions, and final classification to be checked before running the FPGA.

### 12.3 Layer-by-Layer Output Comparison

Layer-by-layer verification was used during development to check whether each CNN stage produced reasonable intermediate values before moving on to the next stage. The main stages checked were input image, Conv1 output, Pool1 output, Conv2 output, Pool2 output, FC1 output, FC2 output, and final logits. This was especially helpful when distinguishing a quantization mismatch from an address-generation or parameter-packing bug.

For RTL simulation, internal signals such as FSM state, counters, accumulator, output channel, row/column index, and final logits can be observed in ModelSim waveforms. This helped confirm that the dataflow through the CNN follows the expected order.

### 12.4 Quantized Model Verification

Since the FPGA accelerator uses integer arithmetic, a quantized Python reference path was implemented. This path applies integer convolution and fully connected computation, followed by layer-specific shift-based rescaling and ReLU-style clamping.

The shift values match the hardware configuration:

CONV1.SHIFT	6
CONV2.SHIFT	8
FC1.SHIFT	8
FC2.SHIFT	7

The FPGA result should be compared against this quantized reference rather than only against the non-quantized software result.

### 12.5 Register Interface Test

The HPS-to-FPGA register interface was tested to confirm that software register offsets match FPGA address decoding logic. The test focused on verifying that control writes generate correct command pulses, status reads return valid busy/done/input-loaded bits, pixel writes go into the correct pixel index, and result/logit reads return correct data.

### 12.6 HPS Register Readback Test

HPS register readback was used to verify that the HPS can correctly read FPGA-side registers through the lightweight bridge. The C server reads the result register and extracts the predicted digit and cycle count. It also reads ten final logits from consecutive registers.

### 12.7 Pixel Register Write Test

The pixel write path was tested by sending a full  $32 \times 32$  image from the HPS to the FPGA. The HPS writes 1024 pixel values, one 32-bit MMIO word per pixel. The FPGA uses only the lower 8 bits of each write.

When the last pixel is written, the FPGA register interface sets the input-loaded status bit. This provides a simple way to confirm that the full image was transferred.

## 12.8 Start / Done Handshake Test

The start/done protocol was verified both in software and in RTL simulation. After writing all pixels, the C server writes the control register with `CTRL_START | CTRL_CLEAR_DONE`. The FPGA register interface converts this into a one-cycle start pulse. The HPS then polls the status register until the done bit becomes high.

## 12.9 ModelSim RTL Simulation

RTL simulation was performed using `tb_lenet_int8_top.sv`. This testbench directly instantiates the CNN core without the HPS or AXI bridge. The testbench generates a 50 MHz clock, applies reset, writes a simple  $32 \times 32$  input image, pulses start, waits for done, and prints the predicted digit, cycle count, and logits.

The purpose of this simulation is not to test recognition accuracy, but to confirm that the core can complete a full inference pass without deadlock or timeout. For this project, seeing the expected start, busy, done, and cycle-count behavior in ModelSim was the fastest way to catch FSM and counter mistakes before testing on the board. The final ModelSim testbench and run script are included in Appendix A.4.

Figure 9 shows the key waveform observations. The highlighted regions confirm the one-cycle `start` pulse, the asserted `busy` interval, the final done signal, and the measured cycle count of 454,966 cycles.

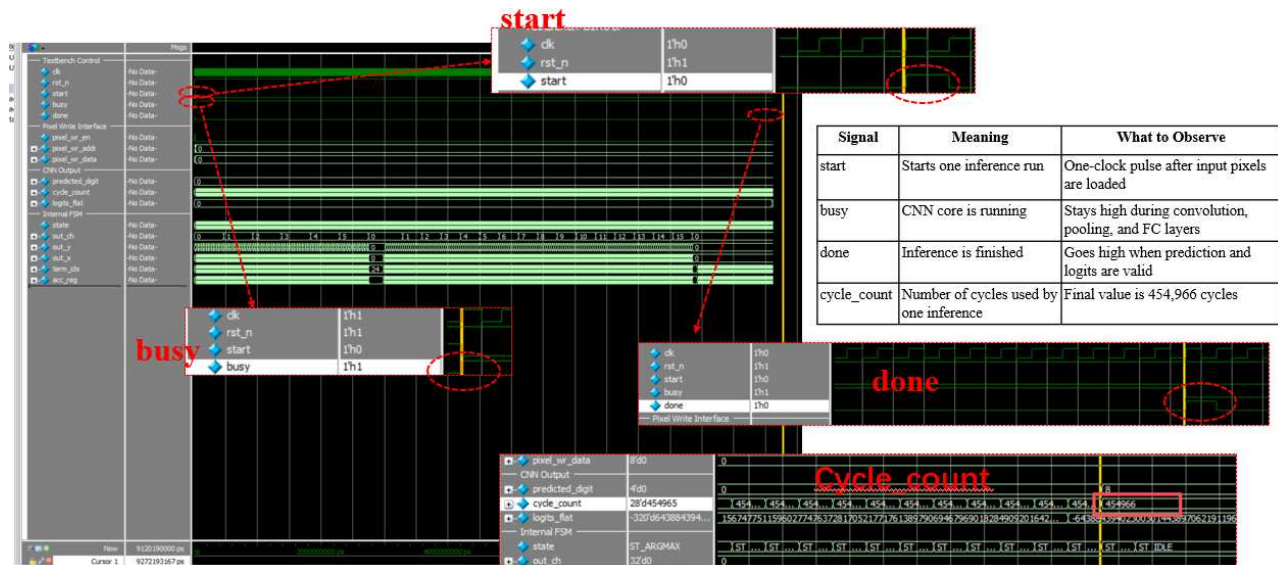


Figure 9: ModelSim waveform used to verify start, busy, done, and cycle-count behavior.

## 12.10 FPGA On-Board Testing

After RTL simulation, the design was compiled in Quartus and programmed onto the DE1-SoC board. The HPS Linux system was booted, Ethernet was configured, and the C inference server was executed on the board.

On-board testing verified that the bitstream can be programmed successfully, the HPS can access the lightweight bridge, pixel data can be written through MMIO, the accelerator can be started from software, and the final prediction/logits can be read back.

Figure 10 shows the DE1-SoC board during an on-board test. The seven-segment display is used as a visible sanity check for the predicted digit while the HPS/server path reports the full logits and

timing values.

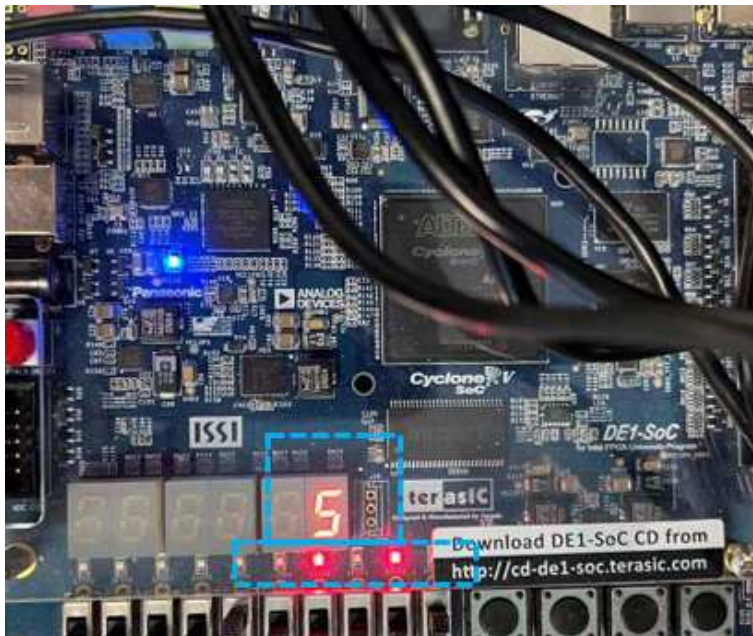


Figure 10: DE1-SoC board test showing a visible predicted digit on the seven-segment display.

### 12.11 End-to-End Browser-to-FPGA Test

The final system test was performed through the browser interface. The user draws or uploads a digit, Flask preprocesses the image, the TCP client sends pixels to the HPS C server, the HPS writes FPGA registers, the CNN core performs inference, and the result is displayed in the browser.

The web page displays the FPGA result, software quantized reference result, and software floating-point reference result. This allows the hardware result to be compared with software references during live testing.

### 12.12 Test Cases for Digits 0–9

Test cases were created by drawing or uploading digits from 0 to 9. The tests included clear digits, thin and thick strokes, centered and shifted digits, similar-looking digits such as 3/5 and 4/9, and uploaded image examples.

For each test, the predicted digit, top logits, and processed  $32 \times 32$  preview were observed.

Figure 11 shows representative digit test samples. The grid is useful for checking whether the pre-processing output still resembles the intended class and for identifying ambiguous handwritten shapes.



Figure 11: Representative processed digit samples used for functional testing.

### 12.13 Misclassification Case Analysis

Misclassification cases were mainly caused by preprocessing sensitivity and similarity between handwritten digit shapes. Common causes included poor centering, very thin or thick strokes, ambiguous digit shapes, mismatch with the training distribution, and close logits between similar classes.

If the FPGA result differs from the quantized software reference, the issue is likely in hardware computation, register transfer, or parameter initialization. If both agree but the prediction is wrong, the issue is likely caused by model accuracy or input quality.

## 13 Results and Performance Evaluation

### 13.1 Functional Correctness Results

The completed system successfully performs end-to-end handwritten digit inference from the browser interface to the FPGA accelerator. The user can draw or upload a digit image, and the software backend preprocesses it into a  $32 \times 32$  grayscale image. The processed pixel array is transmitted to the HPS, written into the FPGA input register window, and then processed by the CNN accelerator.

The FPGA returns the predicted digit, ten final logits, FPGA cycle count, and status/debug information. The web interface displays the FPGA result together with two software reference paths: the quantized software reference and the regular software model.

### 13.2 Recognition Accuracy

The trained LeNet-style model was evaluated using a digit test set containing digits 0 through 9. Based on the confusion matrix obtained during parameter testing, the model achieved approximately:

$$\frac{968}{1000} = 96.8\%$$

accuracy. Most digits were classified correctly, especially clear and centered examples. Accuracy was lower for ambiguous handwritten shapes and poorly preprocessed inputs.

### 13.3 Example Correct Predictions

Correct predictions were observed when the input digit was clearly drawn, centered, and had a reasonable stroke thickness. In these cases, the FPGA result, quantized software result, and regular software result usually agreed.

A correct prediction typically has the following behavior:

- The top logit corresponds to the correct digit.
- The top logit is much larger than the second-highest logit.
- The FPGA result matches the quantized software reference.
- The processed  $32 \times 32$  preview still resembles the intended digit.

### 13.4 Example Incorrect Predictions

Incorrect predictions occurred mainly for unclear or ambiguous inputs. Common failure cases included 3 confused with 5, 4 confused with 9, 8 confused with 9, and 0 confused with 6.

These errors were often caused by handwritten shape ambiguity, poor centering, or stroke thickness. The final logits are useful for analyzing these cases. If the top two logits are close, the model is uncertain.

### 13.5 FPGA Core Runtime

The FPGA core runtime is measured using the cycle counter inside the accelerator. The cycle count is packed into the result register together with the predicted digit.

A representative measured FPGA cycle count was:

454,966 cycles

At a 50 MHz FPGA clock, the core runtime is:

$$\frac{454,966}{50,000,000} = 9.10 \text{ ms}$$

This value represents pure FPGA CNN execution time after input pixels have already been written to the FPGA. It is lower than the browser-observed latency because it excludes the software and HPS-FPGA transfer work around the accelerator.

### 13.6 HPS Register Access Time

The HPS communicates with the FPGA through memory-mapped register accesses over the lightweight bridge. The largest register access overhead comes from writing 1024 pixel registers. Other overheads include writing the control register, polling the status register, reading the result register, and reading ten logit registers.

The register-window approach is simple and reliable but slower than a DMA-based transfer.

### 13.7 Ethernet Transmission Time

The browser/Flask side communicates with the HPS through Ethernet using TCP. Each inference request sends a JSON payload containing the command, request ID, image dimensions, and 1024 pixel values.

Because the image size is small, Ethernet communication is acceptable for an interactive demo. However, Ethernet transmission time can vary due to socket overhead, JSON serialization, operating system scheduling, and network buffering.

### 13.8 End-to-End Inference Latency

The end-to-end latency includes all software and hardware stages: Flask preprocessing, TCP transmission, HPS socket receiving, JSON parsing, MMIO pixel writes, FPGA computation, status polling, result/logit readback, TCP response, and browser display.

Table 17: Representative Inference Latency Comparison

Path	Runtime
CPU exact software path	33.68 ms
CPU quantized software path	34.89 ms
FPGA end-to-end path	12.65 ms
FPGA core only	9.10 ms

The FPGA end-to-end path is slower than pure FPGA core time because it includes communication and HPS control overhead. However, it is still faster than the software-only inference path in the measured example.

### 13.9 Runtime Variation Analysis

The FPGA core runtime is deterministic for a fixed design because the CNN accelerator follows a fixed FSM sequence. For the same bitstream, the number of stages and loop iterations does not change with the digit value, so the cycle count stays stable across inference runs.

The end-to-end latency is more variable because it includes software and communication overhead. Variation can come from Flask scheduling, Python JSON serialization, TCP connection setup, HPS Linux scheduling, socket receive timing, MMIO polling interval, and browser-side handling.

### 13.10 FPGA Resource Utilization

The final Quartus compilation was successful on the Cyclone V device 5CSEMA5F31C6. The fitted resource usage is shown in Table 18.

Table 18: FPGA Resource Utilization

Resource	Usage
Logic utilization	1,532 / 32,070 ALMs, 5%
Total registers	1,272
Total pins	362 / 457, 79%
Block memory bits	1,310,448 / 4,065,280, 32%
RAM blocks	165 / 397, 42%
DSP blocks	5 / 87, 6%
PLLs	0 / 6, 0%

The design uses relatively little logic and DSP resource, but a larger portion of memory resources. This is expected because the accelerator stores input images, intermediate feature maps, and packed parameter ROMs in on-chip memories.

Figure 12 visualizes the same resource profile. The chart makes the design tradeoff clear: the implementation is not logic- or DSP-limited, but it consumes a more noticeable fraction of the available M10K memory blocks.

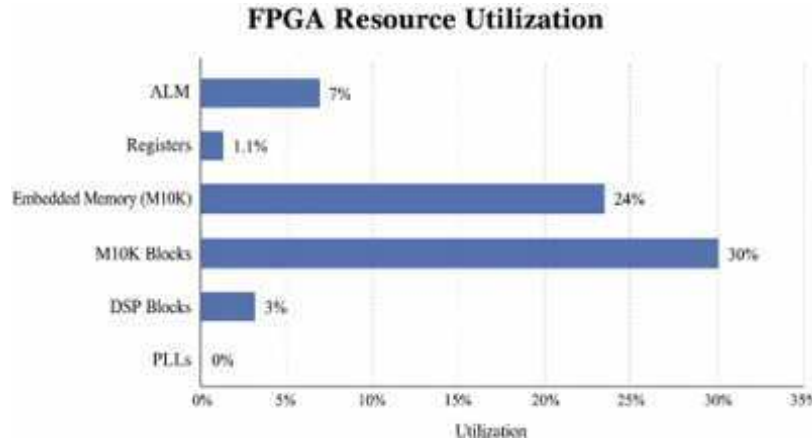


Figure 12: Quartus FPGA resource utilization summary for the fitted accelerator design.

### 13.11 Timing Analysis and Fmax

The design meets timing at the target 50 MHz clock. The Quartus Timing Analyzer reports the following timing results.

Table 19: Timing Analysis Summary

Clock Domain / Metric	Result
clock_50_1 Fmax	61.94 MHz
clock_50_1 setup slack	+3.855 ns
clock_50_1 hold slack	+0.314 ns
Worst overall setup slack	+1.730 ns

The Quartus Fmax report in Figure 13 confirms that the clock\_50\_1 domain has a restricted Fmax of 61.94 MHz, which is above the 50 MHz target used by the design.

```

+-----+-----+-----+
; Slow 1100mV 85C Model Fmax Summary
+-----+-----+-----+
; Fmax      ; Restricted Fmax ; Clock Name
+-----+-----+-----+
; 61.94 MHz ; 61.94 MHz      ; clock_50_1
; 1184.83 MHz ; 717.36 MHz    ; soc_system_hps_0:hps_0

```

Figure 13: Quartus Fmax summary for the 50 MHz FPGA clock domain.

An earlier critical path came from address-generation logic in the convolution datapath. The original expression computed the kernel row, kernel column, input-channel index, and packed ROM tap index directly from `term_idx` using division and modulo operations. Although this code was compact, synthesis mapped the divide/modulo expressions into relatively expensive combinational logic.

```

ky = term_idx / 5;
kx = term_idx % 5;
conv2_ic = term_idx / 25;
conv2_tap = term_idx % 25;
addr = channel_offset + (y + ky) * width + (x + kx);

```

Figure 14: Original convolution address-generation logic using division and modulo operations.

The optimized version avoids the expensive modulo-by-5 path for the kernel column by replacing it with a small lookup-style `case` structure. Since the kernel index pattern is fixed for a  $5 \times 5$  convolution, the hardware can decode the tap position with simple comparisons and constants instead of a general arithmetic remainder circuit.

```

case (idx)
  0, 5, 10, 15, 20: tap5_x = 3'd0;
  1, 6, 11, 16, 21: tap5_x = 3'd1;
  2, 7, 12, 17, 22: tap5_x = 3'd2;
  3, 8, 13, 18, 23: tap5_x = 3'd3;
  default:         tap5_x = 3'd4;
endcase

```

Figure 15: Optimized tap lookup logic used to reduce the old convolution address-generation critical path.

This change shortened the combinational path in the convolution address logic. A separate timing issue in the final output comparison was also improved by replacing a wide combinational `argmax` path with a sequential `argmax` scan.

### 13.12 Comparison with Software-Only Inference

Using the representative latency values:

$$\text{End-to-end speedup} = \frac{33.68}{12.65} = 2.66\times$$

$$\text{Compute speedup} = \frac{33.68}{9.10} = 3.70\times$$

The compute speedup is higher because it excludes Ethernet, HPS, and register access overhead.

### 13.13 Performance Bottleneck Analysis

The main computational bottleneck in the CNN core comes from multiply-accumulate operations in convolution and fully connected layers. To reduce resource usage, the design reuses a small number of MAC lanes instead of fully unrolling all operations.

At the system level, the main bottlenecks are 1024 separate MMIO pixel writes, TCP/JSON communication overhead, HPS polling, sequential MAC reuse, and on-chip memory bandwidth. Future improvements could include DMA-based input transfer, interrupt-based completion, persistent TCP connections, binary payload format, and increased CNN parallelism.

## 14 Failure Modes and Limitations

### 14.1 Image Preprocessing Errors

One major limitation is that the final prediction depends heavily on image preprocessing. If the crop, resize, centering, or contrast adjustment fails, the FPGA may receive a distorted  $32 \times 32$  input image. This type of failure is not caused by the FPGA computation itself.

### 14.2 Stroke Thickness and Digit Shape Sensitivity

The recognition result is sensitive to stroke thickness and digit shape. Very thin strokes may disappear after resizing, while very thick strokes may merge parts of the digit. The same digit may produce different logits if the stroke width, drawing speed, or shape changes.

### 14.3 Similar Digit Misclassification

Some digits are naturally difficult to distinguish when handwritten. Common confusing pairs include 3 and 5, 4 and 9, 7 and 1, 8 and 9, and 0 and 6. If two digits have close logits, small input changes or quantization differences can change the predicted class.

### 14.4 Quantization Error

The FPGA implementation uses integer arithmetic instead of floating-point arithmetic. This reduces hardware cost but introduces quantization error. If the FPGA result matches the quantized software reference but differs from the non-quantized reference, the difference is likely caused by quantization rather than an RTL bug.

### 14.5 Overflow Risk

CNN layers perform many multiply-accumulate operations. Even if inputs and weights are 8-bit, accumulated sums can become much larger. The design reduces overflow risk by using 32-bit accumulation and applying shift-based rescaling after each major layer. Saturation logic prevents wrap-around but may lose information if too many values saturate.

## 14.6 TCP Transmission Delay Variation

TCP provides reliable delivery but does not provide deterministic latency. End-to-end runtime can vary due to operating system scheduling, Ethernet buffering, socket overhead, and network conditions. Therefore, cycle-count-based FPGA timing is a better measure of pure hardware compute time.

## 14.7 Register Access Overhead

The current design transfers input pixels through a memory-mapped register window. Each pixel is written as one 32-bit word, even though only the lower 8 bits are used. This was a deliberate choice because it made address decoding and readback testing straightforward, but it is not the fastest possible transfer method. DMA or shared memory could reduce this overhead.

## 14.8 Polling-Based Control Limitation

The HPS currently waits for the FPGA to finish by polling the status register. Polling is simple but consumes CPU time and introduces latency depending on polling interval. An interrupt-based completion mechanism would be more efficient.

## 14.9 FPGA Resource Limitation

The DE1-SoC Cyclone V FPGA has limited logic, DSP blocks, and on-chip memory. This limits the amount of CNN parallelism that can be implemented. The design reuses MAC hardware to save resources, but this increases the number of cycles per inference.

## 14.10 Timing Closure Limitation

Timing closure is another limitation of FPGA implementation. Long combinational paths can reduce the maximum clock frequency. In this design, address arithmetic and output comparison both had to be watched carefully. The tap-lookup and sequential argmax changes helped reduce critical paths, but further improvements may require more pipelining and memory access optimization.

## 14.11 Scalability Limitation

The current system is designed for a small LeNet-style classifier with  $32 \times 32$  grayscale input and ten output classes. Scaling to larger images, larger CNNs, or real-time video would require DMA transfer, more memory bandwidth, more MAC lanes, and a more efficient communication protocol.

# 15 Future Work

The current system is functional and complete, but several parts could still be improved if the project were continued. The first direction would be reducing system-level latency. At the moment, the HPS writes 1024 pixels into the FPGA through individual MMIO accesses, which is simple and reliable but not very efficient. A DMA-based input path could move the full image as a bulk transfer. Similarly, replacing status polling with an FPGA interrupt would reduce wasted CPU time while the HPS waits for inference to finish.

The second direction would be improving compute throughput inside the FPGA. The current accelerator reuses a small number of MAC lanes to save resources, so convolution and fully connected layers take many cycles. Increasing the number of MAC lanes, adding more parallel processing elements, or

banking memories more aggressively could reduce the FPGA core cycle count, especially in Conv2 and FC1 where most of the multiply-accumulate work occurs.

The third direction would be improving the software-facing interface and recognition robustness. A persistent TCP connection or compact binary payload could reduce JSON serialization and connection setup overhead. On the accuracy side, more robust cropping, centering, contrast handling, and data augmentation would help the model handle off-center digits, unusual stroke widths, and ambiguous handwriting. After those improvements, the same pipeline could be extended from browser drawing and image upload to real-time camera input, although that would require faster preprocessing and more efficient data transfer.

## 16 Division of Work

### 16.1 Yizheng Tang

- **Software part:** built the web demo, Flask backend integration, sample saving, HPS client connection, and result/logit/timing display.
- **FPGA RTL part:** implemented and maintained the CNN core FSM, BRAM storage, MAC datapath integration, quantization connection, address calculation, and argmax logic.
- **Timing optimization part:** optimized the FPGA critical path in the later stage of the project, including replacing the long combinational argmax comparison with a sequential argmax scan, reducing timing pressure, improving Fmax, and checking the updated Quartus timing report.
- **HPS-FPGA interface part:** designed the compact register map and implemented the AXI register interface in `hps_lenet_regs.sv`.
- **HPS software part:** updated the C server logic for `/dev/mem`, `mmap()`, pixel upload, start/done polling, result readback, and logit readback.
- **Verification part:** created the RTL testbench, ran simulation, checked Quartus timing/resource reports, and debugged the full browser-to-FPGA demo.
- **Report and presentation part:** contributed to the report/PPT and took the main role in presentation and Q&A preparation.

### 16.2 Chenxi Shen

- **FPGA RTL part:** implemented key RTL modules for the later pipeline, including Conv2, quantization, and the FC1/FC2/FC3 computation flow, and completed their integration in the LeNet inference datapath.
- **Display/debug part:** implemented the seven-segment HEX display path to show prediction results directly on the board for hardware-level verification and demo usability.
- **Presentation part:** prepared the PPT sections for the above hardware modules, including architecture explanation and dataflow details.
- **Report writing part:** contributed to the final report by writing Sections 10 and 11, covering hardware-related content and implementation details.

## 16.3 Sirui Chen

- **CNN training part:** trained the network and exported the final weights and biases for FPGA use.
- **Hardware alignment part:** adjusted training and quantization so the exported parameters better aligned with hardware inference behavior.
- **Shift search part:** built a Python hardware-simulation testbench to evaluate different layer shift values and select suitable shifts for Conv1, Conv2, FC1, and FC2.
- **Verification part:** verified software/hardware consistency by comparing PyTorch inference results with hardware-style simulation and measuring prediction mismatch.
- **Report and slides part:** contributed to kernel parameter training, quantization, and shift-scaling documentation.

## 16.4 Weiwei Wu

- **Software part:** improved Flask-to-HPS data transfer and response return for prediction, logits, and timing display.
- **HPS-FPGA interface part:** improved the register map and AXI-side interface behavior for reliable control/status signaling and data exchange between HPS and FPGA.
- **HPS software part:** optimized the C-side MMIO workflow, including `/dev/mem` and `mmap()` access, pixel upload, start/done polling, and result/logit readback.
- **Report part:** contributed to the Software Input Processing and HPS Interface sections.

## 16.5 Tian Li

- **Second convolution layer RTL part:** implemented the hardware logic for the second convolution layer, including convolution computation and data control.
- **Hardware optimization part:** optimized the convolution datapath and control logic to reduce resource usage and improve timing.
- **FPGA integration part:** helped connect and debug the convolution module within the overall FPGA system.
- **Draw.io diagram part:** created the system architecture and hardware block diagrams for the report.
- **Verification part:** ran RTL simulation and checked hardware output correctness.
- **Report part:** organized the final report and wrote the hardware implementation and optimization sections.

# 17 Conclusion

## 17.1 Project Summary

This project implemented a complete FPGA-based CNN accelerator system for handwritten digit recognition on the DE1-SoC platform. The final demo accepts a digit drawn in a browser or uploaded as

an image, preprocesses it into a  $32 \times 32$  grayscale pixel array, transfers the data to the HPS through Ethernet TCP communication, and performs CNN inference on the FPGA fabric.

## 17.2 Main Technical Achievements

The main achievements include training and quantizing a LeNet-style classifier, building a browser-based digit input interface, implementing TCP communication between Flask and HPS, developing a C inference server, designing a compact memory-mapped register map, implementing a custom INT8 CNN accelerator, and verifying the design through software reference comparison, RTL simulation, register testing, and end-to-end hardware testing. The final system also exposes logits and cycle counts, which made it easier to understand a prediction rather than only displaying a single digit.

The final design meets timing at the 50 MHz target clock, with a reported Fmax of 61.94 MHz. The measured FPGA path also achieves lower latency than the software-only reference path.

## 17.3 Main Challenges

The main challenges were HPS-FPGA communication, quantization, timing closure, and full system integration. The design required correct coordination between Linux software, `/dev/mem`, `mmap()`, AXI register decoding, and the FPGA accelerator. Matching FPGA behavior with the quantized software reference also required careful handling of scaling, saturation, and layer output formats.

## 17.4 Final Remarks

Overall, the project achieved its goal of building a complete FPGA-based CNN inference system for handwritten digit recognition. The design is still limited by register-based input transfer, polling-based control, and modest CNN parallelism, but those limitations are now clearly isolated. That makes the project a useful foundation for future improvements such as DMA transfer, interrupt-based completion, higher MAC parallelism, improved preprocessing, and real-time camera input.

## 18 References

1. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
2. Intel, "DE1-SoC User Manual," Terasic Technologies.
3. Intel, "Cyclone V Hard Processor System Technical Reference Manual."
4. Intel, "Quartus Prime Standard Edition Handbook."

## A Final Source Code

This appendix includes the final hand-written project source files from `project.main`. Quartus-generated HDL, synthesis outputs, HPS handoff files, ModelSim work libraries, waveforms, transcripts, and run-time logs are intentionally excluded.

### A.1 Web Demo and Software Reference Code

Listing 1: Flask web server and request handler.

```

1  from __future__ import annotations
2
3  import base64
4  import io
5  import math
6  import os
7  import time
8  from pathlib import Path
9
10 from flask import Flask, jsonify, render_template, request
11 from PIL import Image
12
13 try:
14     from .predictor import DigitPredictor
15     from .hps_client import HpsInferClient
16 except ImportError:
17     from predictor import DigitPredictor
18     from hps_client import HpsInferClient
19
20
21 BASE_DIR = Path(__file__).resolve().parent
22 SAMPLE_DIR = BASE_DIR / "saved_samples"
23
24 app = Flask(__name__)
25 predictor = DigitPredictor(project_root=BASE_DIR.parent)
26 USE_HPS_INFER = os.getenv("USE_HPS_INFER", "0") == "1"
27 HPS_HOST = os.getenv("HPS_HOST", "127.0.0.1")
28 HPS_PORT = int(os.getenv("HPS_PORT", "9090"))
29 FPGA_CLOCK_HZ = 50_000_000
30 FPGA_INFERENCE_CYCLES = 312_909
31 FPGA_COMPUTE_MS = FPGA_INFERENCE_CYCLES * 1000.0 / FPGA_CLOCK_HZ
32 hps_client = HpsInferClient(host=HPS_HOST, port=HPS_PORT)
33
34 print(
35     f"[startup] USE_HPS_INFER={USE_HPS_INFER}"
36     f"HPS_HOST={HPS_HOST} HPS_PORT={HPS_PORT}",
37     flush=True,
38 )
39
40
41 def summarize_fpga_logits(logits: list[int]) -> tuple[float, list[dict[str, float | int]]]:
42     if not logits:
43         return 0.0, []
44
45     max_logit = max(logits)
46     max_abs_logit = max(abs(value) for value in logits) or 1
47     temperature = max(1.0, max_abs_logit / 16.0)
48     exps = [math.exp((value - max_logit) / temperature) for value in logits]
49     total = sum(exps) or 1.0
50     probabilities = [(value / total) * 100.0 for value in exps]
51
52     ranked = sorted(
53         (
54             {
55                 "digit": digit,
56                 "score": logits[digit],

```

```

57         "probability": round(probabilities[digit], 2),
58     }
59     for digit in range(len(logits))
60 ),
61     key=lambda item: item["score"],
62     reverse=True,
63 )
64 return ranked[0]["probability"], ranked[:3]
65
66
67 @app.get("/")
68 def index() -> str:
69     return render_template("index.html")
70
71
72 def decode_canvas_payload(payload: dict[str, object]) -> Image.Image:
73     image_data = payload.get("image")
74     if not isinstance(image_data, str) or "," not in image_data:
75         raise ValueError("Missing canvas image payload.")
76
77     _, encoded = image_data.split(",", 1)
78     raw = base64.b64decode(encoded)
79     return Image.open(io.BytesIO(raw)).convert("RGBA")
80
81
82 def elapsed_ms(start: float, end: float) -> float:
83     return round((end - start) * 1000.0, 2)
84
85
86 @app.post("/predict")
87 def predict():
88     payload = request.get_json(silent=True) or {}
89
90     try:
91         image = decode_canvas_payload(payload)
92     except Exception as exc: # pragma: no cover - defensive API response
93         print(f"[predict] Error: {exc}", flush=True)
94         return jsonify({"error": str(exc)}), 400
95
96     software_total_start = time.perf_counter()
97     processed = predictor.preprocess_canvas(image)
98     pixels = list(processed.getdata())
99     preview_image = predictor.render_preview_data_url(processed)
100
101     exact_start = time.perf_counter()
102     exact_logits = predictor._forward(pixels)
103     exact_end = time.perf_counter()
104
105     quantized_start = time.perf_counter()
106     quantized_logits = predictor._forward_quantized(pixels)
107     quantized_end = time.perf_counter()
108     software_total_end = time.perf_counter()
109
110     software_result = predictor._format_prediction(exact_logits, preview_image, "weights_bias_422")
111     software_result["quantized_reference"] = predictor._format_prediction(
112         quantized_logits,
113         preview_image,
114         "fpga_quantized_reference",

```

```

115 )
116 timing = {
117     "software_exact_ms": elapsed_ms(exact_start, exact_end),
118     "software_quantized_ms": elapsed_ms(quantized_start, quantized_end),
119     "software_total_ms": elapsed_ms(software_total_start, software_total_end),
120     "fpga_roundtrip_ms": None,
121     "fpga_compute_ms": None,
122     "fpga_cycles": None,
123     "fpga_clock_hz": None,
124     "speedup_vs_software": None,
125     "compute_speedup_vs_software": None,
126 }
127 response_payload = {
128     "preview_image": preview_image,
129     "software_result": software_result,
130     "software_quantized_result": software_result.get("quantized_reference"),
131     "timing": timing,
132 }
133
134 if USE_HPS_INFER:
135     print("[predict] Using HPS inference path.", flush=True)
136     try:
137         fpga_start = time.perf_counter()
138         hps_result = hps_client.infer(pixels)
139         fpga_end = time.perf_counter()
140         timing["fpga_roundtrip_ms"] = elapsed_ms(fpga_start, fpga_end)
141         debug_payload = hps_result.get("debug", {}) or {}
142         fpga_compute_ms = debug_payload.get("fpga_compute_ms")
143         fpga_cycles = debug_payload.get("fpga_cycles")
144         fpga_clock_hz = debug_payload.get("fpga_clock_hz")
145         if isinstance(fpga_compute_ms, (int, float)):
146             timing["fpga_compute_ms"] = round(float(fpga_compute_ms), 4)
147         else:
148             timing["fpga_compute_ms"] = round(FPGA_COMPUTE_MS, 4)
149         if isinstance(fpga_cycles, int):
150             timing["fpga_cycles"] = fpga_cycles
151         else:
152             timing["fpga_cycles"] = FPGA_INFERENCE_CYCLES
153         if isinstance(fpga_clock_hz, int):
154             timing["fpga_clock_hz"] = fpga_clock_hz
155         else:
156             timing["fpga_clock_hz"] = FPGA_CLOCK_HZ
157         if timing["fpga_roundtrip_ms"] and timing["fpga_roundtrip_ms"] > 0:
158             timing["speedup_vs_software"] = round(
159                 timing["software_exact_ms"] / timing["fpga_roundtrip_ms"],
160                 2,
161             )
162         if timing["fpga_compute_ms"] and timing["fpga_compute_ms"] > 0:
163             timing["compute_speedup_vs_software"] = round(
164                 timing["software_exact_ms"] / timing["fpga_compute_ms"],
165                 2,
166             )
167         print(f"[predict] HPS response: {hps_result}", flush=True)
168         logits = debug_payload.get("logits", []) or []
169         confidence, matches = summarize_fpga_logits(logits)
170         prediction = hps_result.get("prediction", "-")
171         if isinstance(prediction, int) and prediction < 0:
172             prediction = "-"

```

```

173         response_payload["fpga_result"] = {
174             "prediction": prediction,
175             "confidence": confidence,
176             "matches": matches,
177             "debug": hps_result.get("debug"),
178             "note": hps_result.get("note"),
179             "model": {
180                 "name": "hps_fpga_pipeline",
181                 "architecture": "Website_preprocess+Ethernet+HPS+FPGA_LeNet",
182             },
183         }
184     except Exception as exc: # pragma: no cover - defensive API response
185         print(f"[predict]_FPGA_path_error:_{exc}", flush=True)
186         response_payload["fpga_result"] = {
187             "prediction": "!",
188             "confidence": 0.0,
189             "matches": [],
190             "debug": None,
191             "note": str(exc),
192             "model": {
193                 "name": "fpga_error",
194                 "architecture": "HPS/FPGA_path_failed_during_this_request.",
195             },
196         }
197     else:
198         print("[predict]_Using_local_software_predictor_path.", flush=True)
199         response_payload["fpga_result"] = {
200             "prediction": "-",
201             "confidence": 0.0,
202             "matches": [],
203             "debug": None,
204             "note": "FPGA_path_disabled.",
205             "model": {
206                 "name": "fpga_path_disabled",
207                 "architecture": "Enable_USE_HPS_INFER_to_compare_hardware_output.",
208             },
209         }
210
211     return jsonify(response_payload)
212
213
214 @app.post("/save_sample")
215 def save_sample():
216     payload = request.get_json(silent=True) or {}
217     label = payload.get("label")
218
219     if label not in [str(digit) for digit in range(10)]:
220         return jsonify({"error": "Choose_a_digit_label_before_saving."}), 400
221
222     try:
223         image = decode_canvas_payload(payload)
224     except Exception as exc: # pragma: no cover - defensive API response
225         print(f"[save_sample]_Error:_{exc}", flush=True)
226         return jsonify({"error": str(exc)}), 400
227
228     target_dir = SAMPLE_DIR / f"digit_{label}"
229     target_dir.mkdir(parents=True, exist_ok=True)
230     filename = f"sample_{time.strftime('%Y%m%d_%H%M%S')}_{time.time_ns()/_1000000:06d}.png"

```

```

231 target_path = target_dir / filename
232 image.save(target_path)
233
234 return jsonify({
235     "saved": True,
236     "label": label,
237     "path": str(target_path.relative_to(BASE_DIR)),
238 })
239
240
241 if __name__ == "__main__":
242     app.run(debug=False)

```

Listing 2: TCP client used to send inference requests to the HPS server.

```

1  from __future__ import annotations
2
3  import json
4  import socket
5  import uuid
6  from typing import Any
7
8
9  class HpsInferClient:
10     def __init__(self, host: str, port: int, timeout_seconds: float = 35.0):
11         self.host = host
12         self.port = port
13         self.timeout_seconds = timeout_seconds
14
15     def infer(self, pixels: list[int], width: int = 32, height: int = 32) -> dict[str, Any]:
16         if len(pixels) != width * height:
17             raise ValueError(f"Expected_{width}*_{height}_pixels, got_{len(pixels)}.")
18
19         request = {
20             "command": "infer",
21             "request_id": str(uuid.uuid4()),
22             "width": width,
23             "height": height,
24             "pixels": pixels,
25         }
26
27         with socket.create_connection((self.host, self.port), timeout=self.timeout_seconds) as sock:
28             sock.sendall((json.dumps(request) + "\n").encode("utf-8"))
29             chunks: list[bytes] = []
30             while True:
31                 chunk = sock.recv(16384)
32                 if not chunk:
33                     break
34                 chunks.append(chunk)
35                 if b"\n" in chunk:
36                     break
37
38             response = b"".join(chunks).split(b"\n", 1)[0]
39
40             if not response:
41                 raise RuntimeError("No_response_from_HPS_inference_service.")
42
43             payload = json.loads(response.decode("utf-8"))

```

```

44     if not payload.get("ok"):
45         error = payload.get("error", "unknown_hps_error")
46         stage = payload.get("stage")
47         if stage:
48             raise RuntimeError(f"{error}_({stage})")
49         raise RuntimeError(error)
50     return payload

```

Listing 3: Software LeNet reference and quantized reference implementation.

```

1  from __future__ import annotations
2
3  import base64
4  import io
5  import math
6  from dataclasses import dataclass
7  from pathlib import Path
8
9  from PIL import Image, ImageFilter
10
11
12 def stretch_contrast(img: Image.Image) -> Image.Image:
13     pixels = list(img.getdata())
14     lo = min(pixels)
15     hi = max(pixels)
16     if hi <= lo:
17         return img.copy()
18     scale = 255.0 / (hi - lo)
19     return img.point(lambda value: int(round((value - lo) * scale)))
20
21
22 def denoise_image(img: Image.Image) -> Image.Image:
23     return img.filter(ImageFilter.MedianFilter(size=3))
24
25
26 def find_foreground_bbox(
27     img: Image.Image,
28     threshold: int,
29 ) -> tuple[int, int, int, int] | None:
30     pixels = img.load()
31     width, height = img.size
32     left, top = width, height
33     right, bottom = -1, -1
34
35     for y in range(height):
36         for x in range(width):
37             if pixels[x, y] >= threshold:
38                 left = min(left, x)
39                 top = min(top, y)
40                 right = max(right, x)
41                 bottom = max(bottom, y)
42
43     if right == -1:
44         return None
45     return left, top, right + 1, bottom + 1
46
47
48 def add_bbox_margin(

```

```

49     bbox: tuple[int, int, int, int],
50     image_size: tuple[int, int],
51     margin: int,
52 ) -> tuple[int, int, int, int]:
53     left, top, right, bottom = bbox
54     width, height = image_size
55     return (
56         max(0, left - margin),
57         max(0, top - margin),
58         min(width, right + margin),
59         min(height, bottom + margin),
60     )
61
62
63 def center_by_mass(img: Image.Image) -> Image.Image:
64     pixels = list(img.getdata())
65     width, height = img.size
66     total = sum(pixels)
67     if total == 0:
68         return img
69
70     cx = sum((idx % width) * value for idx, value in enumerate(pixels)) / total
71     cy = sum((idx // width) * value for idx, value in enumerate(pixels)) / total
72
73     shift_x = int(round((width - 1) / 2.0 - cx))
74     shift_y = int(round((height - 1) / 2.0 - cy))
75
76     shifted = Image.new("L", (width, height), color=0)
77     shifted.paste(img, (shift_x, shift_y))
78     return shifted
79
80
81 def preprocess_digit(
82     img: Image.Image,
83     out_size: int = 32,
84     inner_size: int = 24,
85     bbox_threshold: int = 24,
86     crop_margin: int = 2,
87 ) -> Image.Image:
88     working = img.convert("L")
89     working = denoise_image(working)
90     working = stretch_contrast(working)
91
92     bbox = find_foreground_bbox(working, bbox_threshold)
93     if bbox is None:
94         raise ValueError("No foreground digit detected. Please draw a number first.")
95
96     cropped = working.crop(add_bbox_margin(bbox, working.size, crop_margin))
97
98     scale = min(inner_size / cropped.width, inner_size / cropped.height)
99     new_width = max(1, int(round(cropped.width * scale)))
100    new_height = max(1, int(round(cropped.height * scale)))
101    resized = cropped.resize((new_width, new_height), Image.Resampling.LANCZOS)
102
103    canvas = Image.new("L", (out_size, out_size), color=0)
104    offset_x = (out_size - resized.width) // 2
105    offset_y = (out_size - resized.height) // 2
106    canvas.paste(resized, (offset_x, offset_y))

```

```

107
108     canvas = center_by_mass(canvas)
109     canvas = canvas.filter(ImageFilter.GaussianBlur(radius=0.35))
110     return stretch_contrast(canvas)
111
112
113 def image_to_vector(img: Image.Image) -> list[int]:
114     return list(img.getdata())
115
116
117 def parse_int8_hex_file(path: Path) -> list[int]:
118     values: list[int] = []
119     for line in path.read_text(encoding="utf-8").splitlines():
120         item = line.strip()
121         if not item:
122             continue
123         raw = int(item, 16)
124         if raw >= 128:
125             raw -= 256
126         values.append(raw)
127     return values
128
129
130 def max_pool_2x2(
131     feature_map: list[int],
132     channels: int,
133     input_size: int,
134 ) -> tuple[list[int], int]:
135     output_size = input_size // 2
136     pooled = [0] * (channels * output_size * output_size)
137
138     for channel in range(channels):
139         channel_offset = channel * input_size * input_size
140         pooled_offset = channel * output_size * output_size
141         for out_y in range(output_size):
142             in_y = out_y * 2
143             for out_x in range(output_size):
144                 in_x = out_x * 2
145                 base = channel_offset + in_y * input_size + in_x
146                 pooled[pooled_offset + out_y * output_size + out_x] = max(
147                     feature_map[base],
148                     feature_map[base + 1],
149                     feature_map[base + input_size],
150                     feature_map[base + input_size + 1],
151                 )
152     return pooled, output_size
153
154
155 def conv_valid(
156     inputs: list[int],
157     input_channels: int,
158     input_size: int,
159     output_channels: int,
160     kernel_size: int,
161     weights: list[int],
162     biases: list[int],
163 ) -> tuple[list[int], int]:
164     output_size = input_size - kernel_size + 1

```

```

165     outputs = [0] * (output_channels * output_size * output_size)
166     kernel_area = kernel_size * kernel_size
167
168     for out_channel in range(output_channels):
169         out_channel_offset = out_channel * output_size * output_size
170         weight_base = out_channel * input_channels * kernel_area
171         bias = biases[out_channel]
172
173         for out_y in range(output_size):
174             for out_x in range(output_size):
175                 acc = bias
176                 for in_channel in range(input_channels):
177                     in_channel_offset = in_channel * input_size * input_size
178                     channel_weight_base = weight_base + in_channel * kernel_area
179                     for ky in range(kernel_size):
180                         input_row = in_channel_offset + (out_y + ky) * input_size + out_x
181                         kernel_row = channel_weight_base + ky * kernel_size
182                         for kx in range(kernel_size):
183                             acc += inputs[input_row + kx] * weights[kernel_row + kx]
184                         outputs[out_channel_offset + out_y * output_size + out_x] = max(0, acc)
185
186     return outputs, output_size
187
188
189 def dense(
190     inputs: list[int],
191     outputs_count: int,
192     weights: list[int],
193     biases: list[int],
194     apply_relu: bool,
195 ) -> list[int]:
196     input_count = len(inputs)
197     outputs = [0] * outputs_count
198
199     for out_index in range(outputs_count):
200         weight_base = out_index * input_count
201         acc = biases[out_index]
202         for input_index, input_value in enumerate(inputs):
203             acc += input_value * weights[weight_base + input_index]
204         outputs[out_index] = max(0, acc) if apply_relu else acc
205
206     return outputs
207
208
209 def softmax_percentages(logits: list[int]) -> list[float]:
210     max_logit = max(logits)
211     max_abs_logit = max(abs(value) for value in logits) or 1
212     temperature = max(1.0, max_abs_logit / 16.0)
213     exps = [math.exp((value - max_logit) / temperature) for value in logits]
214     total = sum(exps) or 1.0
215     return [(value / total) * 100.0 for value in exps]
216
217
218 def quantize_relu(value: int, shift_right: int, clamp_max: int = 255) -> int:
219     if value <= 0:
220         return 0
221     if shift_right:
222         value += 1 << (shift_right - 1)

```

```

223         value >>= shift_right
224     return max(0, min(clamp_max, value))
225
226
227 @dataclass(frozen=True)
228 class Int8Layer:
229     weights: list[int]
230     biases: list[int]
231
232
233 @dataclass(frozen=True)
234 class QuantizationConfig:
235     act_bits: int = 16
236     conv1_shift: int = 6
237     conv2_shift: int = 8
238     fc1_shift: int = 8
239     fc2_shift: int = 7
240
241     @property
242     def act_max(self) -> int:
243         return (1 << self.act_bits) - 1
244
245
246 @dataclass(frozen=True)
247 class Int8LeNetModel:
248     conv1: Int8Layer
249     conv2: Int8Layer
250     fc1: Int8Layer
251     fc2: Int8Layer
252     fc3: Int8Layer
253
254
255 class DigitPredictor:
256     def __init__(self, project_root: Path):
257         self.project_root = project_root
258         self.model_dir = self._resolve_model_dir()
259         self.model = self._load_model()
260         self.quant = QuantizationConfig()
261
262     def _resolve_model_dir(self) -> Path:
263         candidates = [
264             self.project_root.parent / "parameter" / "weights_bias_422",
265             self.project_root / "parameter" / "weights_bias_422",
266         ]
267         for candidate in candidates:
268             if candidate.exists():
269                 return candidate
270         raise FileNotFoundError("Could not find parameter/weights_bias_422 for the web demo.")
271
272     def _split_layer(self, values: list[int], weight_count: int, bias_count: int) -> Int8Layer:
273         expected = weight_count + bias_count
274         if len(values) != expected:
275             raise ValueError(f"Expected {expected} values but found {len(values)}.")
276         return Int8Layer(
277             weights=values[:weight_count],
278             biases=values[weight_count:],
279         )
280

```

```

281 def _load_model(self) -> Int8LeNetModel:
282     conv1 = self._split_layer(parse_int8_hex_file(self.model_dir / "conv1.hex"), 6 * 5 * 5, 6)
283     conv2 = self._split_layer(parse_int8_hex_file(self.model_dir / "conv2.hex"), 16 * 6 * 5 * 5,
16)
284     fc1 = self._split_layer(parse_int8_hex_file(self.model_dir / "fc1.hex"), 120 * 400, 120)
285     fc2 = self._split_layer(parse_int8_hex_file(self.model_dir / "fc2.hex"), 84 * 120, 84)
286     fc3 = self._split_layer(parse_int8_hex_file(self.model_dir / "fc3.hex"), 10 * 84, 10)
287     return Int8LeNetModel(conv1=conv1, conv2=conv2, fc1=fc1, fc2=fc2, fc3=fc3)
288
289 def _forward(self, vector: list[int]) -> list[int]:
290     conv1_out, conv1_size = conv_valid(
291         inputs=vector,
292         input_channels=1,
293         input_size=32,
294         output_channels=6,
295         kernel_size=5,
296         weights=self.model.conv1.weights,
297         biases=self.model.conv1.biases,
298     )
299     pool1_out, pool1_size = max_pool_2x2(conv1_out, channels=6, input_size=conv1_size)
300
301     conv2_out, conv2_size = conv_valid(
302         inputs=pool1_out,
303         input_channels=6,
304         input_size=pool1_size,
305         output_channels=16,
306         kernel_size=5,
307         weights=self.model.conv2.weights,
308         biases=self.model.conv2.biases,
309     )
310     pool2_out, _ = max_pool_2x2(conv2_out, channels=16, input_size=conv2_size)
311
312     fc1_out = dense(pool2_out, 120, self.model.fc1.weights, self.model.fc1.biases, apply_relu=
True)
313     fc2_out = dense(fc1_out, 84, self.model.fc2.weights, self.model.fc2.biases, apply_relu=True)
314     return dense(fc2_out, 10, self.model.fc3.weights, self.model.fc3.biases, apply_relu=False)
315
316 def _forward_quantized(self, vector: list[int]) -> list[int]:
317     quant = self.quant
318     conv1_out, conv1_size = conv_valid(
319         inputs=vector,
320         input_channels=1,
321         input_size=32,
322         output_channels=6,
323         kernel_size=5,
324         weights=self.model.conv1.weights,
325         biases=self.model.conv1.biases,
326     )
327     conv1_q = [quantize_relu(value, quant.conv1_shift, quant.act_max) for value in conv1_out]
328     pool1_out, pool1_size = max_pool_2x2(conv1_q, channels=6, input_size=conv1_size)
329
330     conv2_out, conv2_size = conv_valid(
331         inputs=pool1_out,
332         input_channels=6,
333         input_size=pool1_size,
334         output_channels=16,
335         kernel_size=5,
336         weights=self.model.conv2.weights,

```

```

337         biases=self.model.conv2.biases,
338     )
339     conv2_q = [quantize_relu(value, quant.conv2_shift, quant.act_max) for value in conv2_out]
340     pool2_out, _ = max_pool_2x2(conv2_q, channels=16, input_size=conv2_size)
341
342     fc1_raw = dense(pool2_out, 120, self.model.fc1.weights, self.model.fc1.biases, apply_relu=
False)
343     fc1_out = [quantize_relu(value, quant.fc1_shift, quant.act_max) for value in fc1_raw]
344     fc2_raw = dense(fc1_out, 84, self.model.fc2.weights, self.model.fc2.biases, apply_relu=False)
345     )
346     fc2_out = [quantize_relu(value, quant.fc2_shift, quant.act_max) for value in fc2_raw]
347     return dense(fc2_out, 10, self.model.fc3.weights, self.model.fc3.biases, apply_relu=False)
348
349 def _format_prediction(self, logits: list[int], preview_image: str, model_name: str) -> dict[str
, object]:
350     probabilities = softmax_percentages(logits)
351
352     ranked = sorted(
353         (
354             {
355                 "digit": digit,
356                 "score": logits[digit],
357                 "probability": round(probabilities[digit], 2),
358             }
359             for digit in range(10)
360         ),
361         key=lambda item: item["score"],
362         reverse=True,
363     )
364
365     best = ranked[0]
366     return {
367         "prediction": best["digit"],
368         "confidence": best["probability"],
369         "matches": ranked[:3],
370         "preview_image": preview_image,
371         "logits": logits,
372         "model": {
373             "name": model_name,
374             "architecture": f"LeNet-style_{quantized}_{CNN}_{self.quant.act_bits}-bit_{activations}_{
8-bit}_{weights}",
375         },
376     }
377
378 def preprocess_canvas(self, canvas_image: Image.Image) -> Image.Image:
379     black_bg = Image.new("RGBA", canvas_image.size, (0, 0, 0, 255))
380     black_bg.alpha_composite(canvas_image)
381     return preprocess_digit(black_bg.convert("RGB"))
382
383 def render_preview_data_url(self, processed: Image.Image) -> str:
384     preview = processed.resize((160, 160), Image.Resampling.NEAREST)
385     preview_buffer = io.BytesIO()
386     preview.save(preview_buffer, format="PNG")
387     preview_base64 = base64.b64encode(preview_buffer.getvalue()).decode("ascii")
388     return f"data:image/png;base64,{preview_base64}"
389
390 def predict(self, canvas_image: Image.Image) -> dict[str, object]:
391     processed = self.preprocess_canvas(canvas_image)

```

```

391     vector = image_to_vector(processed)
392     preview_image = self.render_preview_data_url(processed)
393     exact = self._format_prediction(self._forward(vector), preview_image, "weights_bias_422")
394     exact["quantized_reference"] = self._format_prediction(
395         self._forward_quantized(vector),
396         preview_image,
397         "fpga_quantized_reference",
398     )
399     return exact

```

Listing 4: Browser interface HTML template.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  <meta charset="UTF-8" />
5  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6  <title>FPGA Handwritten Digit Recognition</title>
7  <link rel="stylesheet" href="{ url_for('static', filename='style.css') }" />
8  </head>
9  <body>
10 <main class="page">
11 <header class="topbar">
12 <div>
13 <h1>FPGA Digit Lab</h1>
14 </div>
15 <div class="status-strip" aria-label="Pipeline status">
16 <span>Web preprocess</span>
17 <span>Python reference</span>
18 <span>HPS FPGA</span>
19 </div>
20 </header>
21
22 <section class="console-grid">
23 <section class="panel input-panel">
24 <div class="panel-heading">
25 <p class="eyebrow">Input</p>
26 <h2>Drawing Pad</h2>
27 </div>
28
29 <div class="canvas-shell">
30 <canvas id="draw-canvas" width="360" height="360" aria-label="Digit drawing canvas"></
31 canvas>
32 </div>
33
34 <div class="control-row brush-control">
35 <label for="brush-size-slider">Stroke</label>
36 <input id="brush-size-slider" type="range" min="4" max="28" step="1" value="12" />
37 <span id="brush-size-value">12 px</span>
38 </div>
39
40 <div class="control-row invert-control">
41 <span>Invert</span>
42 <label class="switch" for="invert-toggle">
43 <input id="invert-toggle" type="checkbox" />
44 <span class="switch-track"></span>
45 </label>
46 <span id="invert-status">Off</span>

```

```

46     </div>
47
48     <div class="actions">
49         <button id="predict-btn" class="primary-btn">Predict</button>
50         <label class="upload-btn" for="upload-image-input">Upload</label>
51         <input id="upload-image-input" type="file" accept="image/*" />
52         <button id="clear-btn" class="ghost-btn">Clear</button>
53     </div>
54
55     <div class="label-save-row">
56         <label for="sample-label-select">Label</label>
57         <select id="sample-label-select">
58             <option value="">Choose</option>
59             <option value="0">0</option>
60             <option value="1">1</option>
61             <option value="2">2</option>
62             <option value="3">3</option>
63             <option value="4">4</option>
64             <option value="5">5</option>
65             <option value="6">6</option>
66             <option value="7">7</option>
67             <option value="8">8</option>
68             <option value="9">9</option>
69         </select>
70         <button id="save-sample-btn" class="ghost-btn" type="button">Save</button>
71     </div>
72     <p id="save-status" class="save-status">Not saved</p>
73
74 </section>
75
76 <section class="panel results-panel">
77     <div class="panel-heading">
78         <p class="eyebrow">Output</p>
79         <h2>Inference Results</h2>
80     </div>
81
82     <div class="result-stack">
83         <article class="result-block">
84             <p id="fpga-result-label" class="result-label">FPGA Result</p>
85             <div class="prediction-row">
86                 <div id="fpga-prediction-value" class="prediction-value">--</div>
87                 <div class="prediction-meta">
88                     <p id="fpga-confidence-text">Top-1 probability: --</p>
89                     <p id="fpga-top-match-text">Top scores: --</p>
90                 </div>
91             </div>
92         </article>
93
94         <article class="result-block compact-result">
95             <p id="software-quant-result-label" class="result-label">Quantized Software Reference
96 </p>
97             <div class="prediction-row">
98                 <div id="software-quant-prediction-value" class="prediction-value">--</div>
99                 <div class="prediction-meta">
100                     <p id="software-quant-confidence-text">Top-1 probability: --</p>
101                     <p id="software-quant-top-match-text">Top scores: --</p>
102                 </div>
103             </div>

```

```

103     </article>
104
105     <article class="result-block compact-result">
106         <p id="software-result-label" class="result-label">Software Result</p>
107         <div class="prediction-row">
108             <div id="software-prediction-value" class="prediction-value"></div>
109             <div class="prediction-meta">
110                 <p id="software-confidence-text">Top-1 probability: --</p>
111                 <p id="software-top-match-text">Top scores: --</p>
112             </div>
113         </div>
114     </article>
115 </div>
116 </section>
117 </section>
118
119 <section class="analysis-grid">
120     <section class="panel preview-block">
121         <div class="panel-heading">
122             <p class="eyebrow">Processed Input</p>
123             <h2>32x32 Preview</h2>
124         </div>
125         <div class="preview-frame">
126             <canvas id="preview-canvas" width="160" height="160"></canvas>
127         </div>
128     </section>
129
130     <section class="panel debug-block">
131         <div class="panel-heading">
132             <p class="eyebrow">FPGA Output</p>
133             <h2>Final Logits</h2>
134         </div>
135         <p id="debug-summary" class="debug-summary">Waiting for FPGA debug data.</p>
136         <div id="debug-grid" class="debug-grid"></div>
137     </section>
138
139     <section class="panel performance-panel">
140         <div class="panel-heading">
141             <p class="eyebrow">Performance</p>
142             <h2>Speed Compare</h2>
143         </div>
144         <div class="metric-grid">
145             <div>
146                 <span>CPU exact</span>
147                 <strong id="software-exact-time">--</strong>
148             </div>
149             <div>
150                 <span>CPU quant</span>
151                 <strong id="software-quant-time">--</strong>
152             </div>
153             <div>
154                 <span>FPGA path</span>
155                 <strong id="fpga-time">--</strong>
156             </div>
157             <div>
158                 <span>FPGA core</span>
159                 <strong id="fpga-core-time">--</strong>
160             </div>

```

```

161     <div>
162         <span>Speedup</span>
163         <strong id="speedup-value">--</strong>
164     </div>
165 </div>
166 <div class="speed-chart" aria-label="Speed comparison chart">
167     <div class="speed-bar-group">
168         <div class="speed-bar-track">
169             <div id="software-exact-bar" class="speed-bar cpu-exact-bar"></div>
170         </div>
171         <span>Exact</span>
172     </div>
173     <div class="speed-bar-group">
174         <div class="speed-bar-track">
175             <div id="software-quant-bar" class="speed-bar cpu-quant-bar"></div>
176         </div>
177         <span>Quant</span>
178     </div>
179     <div class="speed-bar-group">
180         <div class="speed-bar-track">
181             <div id="fpga-bar" class="speed-bar fpga-bar"></div>
182         </div>
183         <span>FPGA</span>
184     </div>
185 </div>
186 </section>
187 </section>
188 </main>
189
190 <script src="{{ url_for('static', filename='app.js') }}"></script>
191 </body>
192 </html>

```

Listing 5: Browser-side JavaScript for drawing, upload, prediction requests, and UI updates.

```

1  const drawCanvas = document.getElementById("draw-canvas");
2  const drawCtx = drawCanvas.getContext("2d");
3  const previewCanvas = document.getElementById("preview-canvas");
4  const previewCtx = previewCanvas.getContext("2d");
5  const predictBtn = document.getElementById("predict-btn");
6  const clearBtn = document.getElementById("clear-btn");
7  const uploadImageInput = document.getElementById("upload-image-input");
8  const invertToggle = document.getElementById("invert-toggle");
9  const invertStatus = document.getElementById("invert-status");
10 const brushSizeSlider = document.getElementById("brush-size-slider");
11 const brushSizeValue = document.getElementById("brush-size-value");
12 const sampleLabelSelect = document.getElementById("sample-label-select");
13 const saveSampleBtn = document.getElementById("save-sample-btn");
14 const saveStatus = document.getElementById("save-status");
15 const softwareResultLabel = document.getElementById("software-result-label");
16 const softwarePredictionValue = document.getElementById("software-prediction-value");
17 const softwareConfidenceText = document.getElementById("software-confidence-text");
18 const softwareTopMatchText = document.getElementById("software-top-match-text");
19 const softwareQuantResultLabel = document.getElementById("software-quant-result-label");
20 const softwareQuantPredictionValue = document.getElementById("software-quant-prediction-value");
21 const softwareQuantConfidenceText = document.getElementById("software-quant-confidence-text");
22 const softwareQuantTopMatchText = document.getElementById("software-quant-top-match-text");
23 const fpgaResultLabel = document.getElementById("fpga-result-label");

```

```

24 const fpgaPredictionValue = document.getElementById("fpga-prediction-value");
25 const fpgaConfidenceText = document.getElementById("fpga-confidence-text");
26 const fpgaTopMatchText = document.getElementById("fpga-top-match-text");
27 const softwareExactTime = document.getElementById("software-exact-time");
28 const softwareQuantTime = document.getElementById("software-quant-time");
29 const fpgaTime = document.getElementById("fpga-time");
30 const fpgaCoreTime = document.getElementById("fpga-core-time");
31 const speedupValue = document.getElementById("speedup-value");
32 const softwareExactBar = document.getElementById("software-exact-bar");
33 const softwareQuantBar = document.getElementById("software-quant-bar");
34 const fpgaBar = document.getElementById("fpga-bar");
35 const debugSummary = document.getElementById("debug-summary");
36 const debugGrid = document.getElementById("debug-grid");
37
38 let drawing = false;
39 let brushSize = Number(brushSizeSlider.value);
40
41 function applyInvertStatus() {
42     invertStatus.textContent = invertToggle.checked ? "On" : "Off";
43 }
44
45 function applyBrushSize() {
46     brushSize = Number(brushSizeSlider.value);
47     drawCtx.lineWidth = brushSize;
48     brushSizeValue.textContent = `${brushSize} px`;
49 }
50
51 function resetCanvas() {
52     drawCtx.fillStyle = "#000000";
53     drawCtx.fillRect(0, 0, drawCanvas.width, drawCanvas.height);
54     drawCtx.lineCap = "round";
55     drawCtx.lineJoin = "round";
56     drawCtx.strokeStyle = "#ffffff";
57     drawCtx.lineWidth = brushSize;
58 }
59
60 function resetSaveStatus() {
61     saveStatus.textContent = "Not saved";
62     saveStatus.style.color = "";
63 }
64
65 function clearPreview() {
66     previewCtx.fillStyle = "#000000";
67     previewCtx.fillRect(0, 0, previewCanvas.width, previewCanvas.height);
68 }
69
70 function getCanvasPayload() {
71     if (!invertToggle.checked) {
72         return drawCanvas.toDataURL("image/png");
73     }
74
75     const tempCanvas = document.createElement("canvas");
76     tempCanvas.width = drawCanvas.width;
77     tempCanvas.height = drawCanvas.height;
78     const tempCtx = tempCanvas.getContext("2d");
79     tempCtx.drawImage(drawCanvas, 0, 0);
80     const imageData = tempCtx.getImageData(0, 0, tempCanvas.width, tempCanvas.height);
81     const data = imageData.data;

```

```

82
83   for (let index = 0; index < data.length; index += 4) {
84       data[index] = 255 - data[index];
85       data[index + 1] = 255 - data[index + 1];
86       data[index + 2] = 255 - data[index + 2];
87       data[index + 3] = 255;
88   }
89
90   tempCtx.putImageData(imageData, 0, 0);
91   return tempCanvas.toDataURL("image/png");
92 }
93
94 function resetResultCards() {
95     softwarePredictionValue.textContent = "--";
96     softwareConfidenceText.textContent = "Top-1 probability: --";
97     softwareTopMatchText.textContent = "Top scores: --";
98     softwareResultLabel.textContent = "Software Result";
99     softwareQuantPredictionValue.textContent = "--";
100    softwareQuantConfidenceText.textContent = "Top-1 probability: --";
101    softwareQuantTopMatchText.textContent = "Top scores: --";
102    softwareQuantResultLabel.textContent = "Quantized Software Reference";
103    fpgaPredictionValue.textContent = "--";
104    fpgaConfidenceText.textContent = "Top-1 probability: --";
105    fpgaTopMatchText.textContent = "Top scores: --";
106    fpgaResultLabel.textContent = "FPGA Result";
107    softwareExactTime.textContent = "--";
108    softwareQuantTime.textContent = "--";
109    fpgaTime.textContent = "--";
110    fpgaCoreTime.textContent = "--";
111    speedupValue.textContent = "--";
112    softwareExactBar.style.height = "0%";
113    softwareQuantBar.style.height = "0%";
114    fpgaBar.style.height = "0%";
115    clearDebugView();
116 }
117
118 function getPoint(event) {
119     const rect = drawCanvas.getBoundingClientRect();
120     const touch = event.touches ? event.touches[0] : event;
121     const scaleX = drawCanvas.width / rect.width;
122     const scaleY = drawCanvas.height / rect.height;
123     return {
124         x: (touch.clientX - rect.left) * scaleX,
125         y: (touch.clientY - rect.top) * scaleY,
126     };
127 }
128
129 function startDrawing(event) {
130     drawing = true;
131     const { x, y } = getPoint(event);
132     drawCtx.beginPath();
133     drawCtx.moveTo(x, y);
134     event.preventDefault();
135 }
136
137 function draw(event) {
138     if (!drawing) {
139         return;

```

```

140     }
141     const { x, y } = getPoint(event);
142     drawCtx.lineTo(x, y);
143     drawCtx.stroke();
144     event.preventDefault();
145 }
146
147 function stopDrawing() {
148     drawing = false;
149 }
150
151 function renderPreview(dataUrl) {
152     const previewImage = new Image();
153     previewImage.onload = () => {
154         previewCtx.imageSmoothingEnabled = false;
155         previewCtx.clearRect(0, 0, previewCanvas.width, previewCanvas.height);
156         previewCtx.drawImage(previewImage, 0, 0, previewCanvas.width, previewCanvas.height);
157     };
158     previewImage.src = dataUrl;
159 }
160
161 function toTitleCase(text) {
162     return text
163         .replace(/[_-]+/g, " ")
164         .replace(/\b\w/g, (char) => char.toUpperCase());
165 }
166
167 function clearDebugView(message = "Waiting for FPGA debug data.") {
168     debugSummary.textContent = message;
169     debugGrid.innerHTML = "";
170 }
171
172 function renderLogitDebug(logits) {
173     debugSummary.textContent = "Showing final FPGA classification logits returned by the HPS/FPGA
174         pipeline.";
175     debugGrid.innerHTML = "";
176
177     const card = document.createElement("article");
178     card.className = "debug-card";
179
180     const title = document.createElement("p");
181     title.className = "debug-card-title";
182     title.textContent = "Final Logits";
183
184     const dataBlock = document.createElement("pre");
185     dataBlock.className = "debug-card-data";
186     dataBlock.textContent = logits
187         .map((value, index) => `digit ${index}: ${value}`)
188         .join("\n");
189
190     card.appendChild(title);
191     card.appendChild(dataBlock);
192     debugGrid.appendChild(card);
193 }
194
195 function formatHardwareTiming(debug) {
196     if (typeof debug?.fpga_cycles !== "number") {
197         return "FPGA core timing: unavailable";
198     }
199 }

```

```

197   }
198
199   const clockMhz =
200     typeof debug?.fpga_clock_hz === "number"
201     ? (debug.fpga_clock_hz / 1000000).toFixed(2)
202     : "--";
203   const computeMs =
204     typeof debug?.fpga_compute_ms === "number"
205     ? debug.fpga_compute_ms.toFixed(4)
206     : "--";
207   return `FPGA core timing: ${debug.fpga_cycles} cycles @ ${clockMhz} MHz = ${computeMs} ms`;
208 }
209
210 function renderResultCard(prefix, result) {
211   let labelEl;
212   let predictionEl;
213   let confidenceEl;
214   let matchesEl;
215
216   if (prefix === "software") {
217     labelEl = softwareResultLabel;
218     predictionEl = softwarePredictionValue;
219     confidenceEl = softwareConfidenceText;
220     matchesEl = softwareTopMatchText;
221   } else if (prefix === "software_quantized") {
222     labelEl = softwareQuantResultLabel;
223     predictionEl = softwareQuantPredictionValue;
224     confidenceEl = softwareQuantConfidenceText;
225     matchesEl = softwareQuantTopMatchText;
226   } else {
227     labelEl = fpgaResultLabel;
228     predictionEl = fpgaPredictionValue;
229     confidenceEl = fpgaConfidenceText;
230     matchesEl = fpgaTopMatchText;
231   }
232
233   labelEl.textContent = `${toTitleCase(prefix)} Result From ${toTitleCase(result?.model?.name || "
    unknown_source")}`;
234   predictionEl.textContent = result?.prediction ?? "-";
235   confidenceEl.textContent = `Top-1 probability: ${result?.confidence ?? 0}%`;
236
237   const nearest = (result?.matches || [])
238     .map((match) => `${match.digit} (score ${match.score}, ${match.probability}%)`)
239     .join(" | ");
240
241   if (nearest) {
242     matchesEl.textContent = `Top scores: ${nearest}`;
243   } else if (result?.note) {
244     matchesEl.textContent = `Status: ${result.note}`;
245   } else {
246     matchesEl.textContent = "Top scores: unavailable";
247   }
248 }
249
250 function renderDebugView(debug) {
251   const logits = debug?.logits;
252   if (debug?.logits_valid === false) {

```

```

253     clearDebugView('FPGA completed the run, but final logits were not marked valid yet. ${
        formatHardwareTiming(debug)}');
254     return;
255 }
256
257 if (Array.isArray(logits) && logits.length) {
258     renderLogitDebug(logits);
259     debugSummary.textContent = 'Showing final FPGA classification logits returned by the HPS/FPGA
        pipeline. ${formatHardwareTiming(debug)}';
260     return;
261 }
262
263 clearDebugView("No final FPGA logits returned.");
264 }
265
266 function formatMs(value) {
267     if (typeof value !== "number") {
268         return "--";
269     }
270     return `${value.toFixed(2)} ms`;
271 }
272
273 function renderTiming(timing) {
274     softwareExactTime.textContent = formatMs(timing?.software_exact_ms);
275     softwareQuantTime.textContent = formatMs(timing?.software_quantized_ms);
276     fpgaTime.textContent = formatMs(timing?.fpga_roundtrip_ms);
277     fpgaCoreTime.textContent = formatMs(timing?.fpga_compute_ms);
278
279     if (typeof timing?.speedup_vs_software === "number") {
280         speedupValue.textContent = `${timing.speedup_vs_software.toFixed(2)}x`;
281     } else if (timing?.fpga_roundtrip_ms === null) {
282         speedupValue.textContent = "FPGA off";
283     } else {
284         speedupValue.textContent = "--";
285     }
286
287     const values = [
288         timing?.software_exact_ms,
289         timing?.software_quantized_ms,
290         timing?.fpga_roundtrip_ms,
291     ].filter((value) => typeof value === "number" && value > 0);
292     const maxValue = values.length ? Math.max(...values) : 0;
293     const bars = [
294         [softwareExactBar, timing?.software_exact_ms],
295         [softwareQuantBar, timing?.software_quantized_ms],
296         [fpgaBar, timing?.fpga_roundtrip_ms],
297     ];
298
299     bars.forEach(([bar, value]) => {
300         if (typeof value !== "number" || value <= 0 || maxValue <= 0) {
301             bar.style.height = "0%";
302             return;
303         }
304         bar.style.height = `${Math.max(8, (value / maxValue) * 100)}%`;
305     });
306 }
307
308 async function predictDigit() {

```

```

309 softwarePredictionValue.textContent = "...";
310 softwareConfidenceText.textContent = "Top-1 probability: running";
311 softwareTopMatchText.textContent = "Top scores: calculating";
312 softwareQuantPredictionValue.textContent = "...";
313 softwareQuantConfidenceText.textContent = "Top-1 probability: running";
314 softwareQuantTopMatchText.textContent = "Top scores: calculating";
315 fpgaPredictionValue.textContent = "...";
316 fpgaConfidenceText.textContent = "Top-1 probability: running";
317 fpgaTopMatchText.textContent = "Top scores: calculating";
318 softwareExactTime.textContent = "running";
319 softwareQuantTime.textContent = "running";
320 fpgaTime.textContent = "running";
321 fpgaCoreTime.textContent = "running";
322 speedupValue.textContent = "--";
323 softwareExactBar.style.height = "0%";
324 softwareQuantBar.style.height = "0%";
325 fpgaBar.style.height = "0%";
326
327 try {
328     const response = await fetch("/predict", {
329         method: "POST",
330         headers: { "Content-Type": "application/json" },
331         body: JSON.stringify({ image: getCanvasPayload() }),
332     });
333
334     const data = await response.json();
335     if (!response.ok) {
336         throw new Error(data.error || "Prediction failed.");
337     }
338
339     renderResultCard("software", data.software_result);
340     renderResultCard("software_quantized", data.software_quantized_result);
341     renderResultCard("fpga", data.fpga_result);
342     renderPreview(data.preview_image);
343     renderDebugView(data.fpga_result?.debug);
344     renderTiming(data.timing);
345 } catch (error) {
346     softwarePredictionValue.textContent = "!";
347     softwareConfidenceText.textContent = error.message;
348     softwareTopMatchText.textContent = "Top scores: unavailable";
349     softwareResultLabel.textContent = "Software Result From Error";
350     softwareQuantPredictionValue.textContent = "!";
351     softwareQuantConfidenceText.textContent = error.message;
352     softwareQuantTopMatchText.textContent = "Top scores: unavailable";
353     softwareQuantResultLabel.textContent = "Quantized Software Reference From Error";
354     fpgaPredictionValue.textContent = "!";
355     fpgaConfidenceText.textContent = error.message;
356     fpgaTopMatchText.textContent = "Top scores: unavailable";
357     fpgaResultLabel.textContent = "FPGA Result From Error";
358     softwareExactTime.textContent = "--";
359     softwareQuantTime.textContent = "--";
360     fpgaTime.textContent = "--";
361     fpgaCoreTime.textContent = "--";
362     speedupValue.textContent = "--";
363     clearPreview();
364     clearDebugView("FPGA debug data unavailable.");
365 }
366 }

```

```

367
368 async function saveSample() {
369   const label = sampleLabelSelect.value;
370   if (!label) {
371     saveStatus.textContent = "Choose a label first";
372     saveStatus.style.color = "#fbbf24";
373     return;
374   }
375
376   saveSampleBtn.disabled = true;
377   saveStatus.textContent = "Saving...";
378   saveStatus.style.color = "";
379
380   try {
381     const response = await fetch("/save_sample", {
382       method: "POST",
383       headers: { "Content-Type": "application/json" },
384       body: JSON.stringify({
385         image: getCanvasPayload(),
386         label,
387       }),
388     });
389
390     const data = await response.json();
391     if (!response.ok) {
392       throw new Error(data.error || "Save failed.");
393     }
394
395     saveStatus.textContent = `Saved label ${data.label}`;
396     saveStatus.style.color = "#7dd3fc";
397   } catch (error) {
398     saveStatus.textContent = error.message;
399     saveStatus.style.color = "#f87171";
400   } finally {
401     saveSampleBtn.disabled = false;
402   }
403 }
404
405 function drawUploadedImage(file) {
406   if (!file) {
407     return;
408   }
409
410   const reader = new FileReader();
411   reader.onload = () => {
412     const uploadedImage = new Image();
413     uploadedImage.onload = () => {
414       const tempCanvas = document.createElement("canvas");
415       tempCanvas.width = drawCanvas.width;
416       tempCanvas.height = drawCanvas.height;
417       const tempCtx = tempCanvas.getContext("2d");
418
419       tempCtx.fillStyle = "#000000";
420       tempCtx.fillRect(0, 0, tempCanvas.width, tempCanvas.height);
421
422       const scale = Math.min(
423         tempCanvas.width / uploadedImage.width,
424         tempCanvas.height / uploadedImage.height,

```

```

425     );
426     const width = uploadedImage.width * scale;
427     const height = uploadedImage.height * scale;
428     const x = (tempCanvas.width - width) / 2;
429     const y = (tempCanvas.height - height) / 2;
430
431     tempCtx.imageSmoothingEnabled = true;
432     tempCtx.imageSmoothingQuality = "high";
433     tempCtx.drawImage(uploadedImage, x, y, width, height);
434
435     const imageData = tempCtx.getImageData(0, 0, tempCanvas.width, tempCanvas.height);
436     const data = imageData.data;
437     for (let index = 0; index < data.length; index += 4) {
438         const gray = Math.round((data[index] + data[index + 1] + data[index + 2]) / 3);
439         data[index] = gray;
440         data[index + 1] = gray;
441         data[index + 2] = gray;
442         data[index + 3] = 255;
443     }
444
445     drawCtx.putImageData(imageData, 0, 0);
446     applyBrushSize();
447     clearPreview();
448     resetResultCards();
449     resetSaveStatus();
450 };
451 uploadedImage.src = reader.result;
452 };
453 reader.readAsDataURL(file);
454 }
455
456 drawCanvas.addEventListener("mousedown", startDrawing);
457 drawCanvas.addEventListener("mousemove", draw);
458 window.addEventListener("mouseup", stopDrawing);
459 drawCanvas.addEventListener("mouseleave", stopDrawing);
460
461 drawCanvas.addEventListener("touchstart", startDrawing, { passive: false });
462 drawCanvas.addEventListener("touchmove", draw, { passive: false });
463 window.addEventListener("touchend", stopDrawing);
464
465 brushSizeSlider.addEventListener("input", applyBrushSize);
466 invertToggle.addEventListener("change", applyInvertStatus);
467 predictBtn.addEventListener("click", predictDigit);
468 saveSampleBtn.addEventListener("click", saveSample);
469 uploadImageInput.addEventListener("change", (event) => {
470     drawUploadedImage(event.target.files?.[0]);
471     event.target.value = "";
472 });
473 clearBtn.addEventListener("click", () => {
474     resetCanvas();
475     clearPreview();
476     resetResultCards();
477     resetSaveStatus();
478 });
479
480 resetCanvas();
481 applyInvertStatus();
482 applyBrushSize();

```

```
483 clearPreview();
484 clearDebugView();
485 resetSaveStatus();
```

Listing 6: Web interface CSS styling.

```
1  :root {
2    --bg: #080c12;
3    --panel: #101722;
4    --panel-strong: #141f2d;
5    --ink: #e6edf7;
6    --muted: #8fa1b7;
7    --subtle: #627184;
8    --accent: #38bdf8;
9    --accent-strong: #7dd3fc;
10   --success: #22c55e;
11   --line: #243244;
12   --line-strong: #33465f;
13   --black: #020408;
14   --shadow: 0 22px 50px rgba(0, 0, 0, 0.34);
15 }
16
17 * {
18   box-sizing: border-box;
19 }
20
21 body {
22   margin: 0;
23   min-height: 100vh;
24   font-family: "Segoe UI", "PingFang SC", "Microsoft YaHei", sans-serif;
25   color: var(--ink);
26   background: var(--bg);
27 }
28
29 .page {
30   width: min(1440px, calc(100% - 32px));
31   margin: 0 auto;
32   padding: 24px 0 32px;
33 }
34
35 .topbar {
36   min-height: 88px;
37   display: flex;
38   align-items: center;
39   justify-content: space-between;
40   gap: 18px;
41   border: 1px solid var(--line);
42   border-radius: 8px;
43   padding: 18px 20px;
44   background: #0d131d;
45   box-shadow: var(--shadow);
46 }
47
48 .eyebrow,
49 .result-label {
50   margin: 0 0 7px;
51   color: var(--muted);
52   font-size: 0.78rem;
```

```

53   font-weight: 800;
54   letter-spacing: 0.08em;
55   text-transform: uppercase;
56 }
57
58 h1,
59 h2,
60 h3,
61 p {
62   margin-top: 0;
63 }
64
65 h1 {
66   margin-bottom: 0;
67   font-size: clamp(2rem, 3vw, 3rem);
68   line-height: 1;
69   letter-spacing: 0;
70 }
71
72 h2 {
73   margin: 0;
74   font-size: 1.2rem;
75   line-height: 1.2;
76   letter-spacing: 0;
77 }
78
79 .status-strip {
80   display: flex;
81   flex-wrap: wrap;
82   justify-content: flex-end;
83   gap: 8px;
84 }
85
86 .status-strip span {
87   border: 1px solid var(--line-strong);
88   border-radius: 999px;
89   padding: 8px 11px;
90   background: #0a1018;
91   color: #bed0e4;
92   font-size: 0.86rem;
93   font-weight: 700;
94 }
95
96 .console-grid {
97   margin-top: 18px;
98   display: grid;
99   grid-template-columns: minmax(420px, 0.86fr) minmax(520px, 1.14fr);
100  gap: 18px;
101  align-items: stretch;
102 }
103
104 .analysis-grid {
105   margin-top: 18px;
106   display: grid;
107   grid-template-columns: 230px minmax(420px, 1fr) 300px;
108   gap: 18px;
109   align-items: stretch;
110 }

```

```

111
112 .panel {
113   border: 1px solid var(--line);
114   border-radius: 8px;
115   background: var(--panel);
116   box-shadow: var(--shadow);
117 }
118
119 .panel-heading {
120   min-height: 64px;
121   padding: 16px 18px 12px;
122   border-bottom: 1px solid var(--line);
123   background: var(--panel-strong);
124   border-radius: 8px 8px 0 0;
125 }
126
127 .input-panel {
128   display: grid;
129   justify-items: center;
130   align-content: start;
131   padding-bottom: 18px;
132 }
133
134 .input-panel .panel-heading,
135 .results-panel .panel-heading,
136 .debug-block .panel-heading,
137 .preview-block .panel-heading,
138 .performance-panel .panel-heading {
139   width: 100%;
140 }
141
142 .canvas-shell {
143   width: min(360px, calc(100% - 36px));
144   aspect-ratio: 1;
145   margin: 18px;
146   background: var(--black);
147   border: 1px solid #2d3c50;
148   border-radius: 8px;
149   overflow: hidden;
150   box-shadow:
151     inset 0 0 0 1px rgba(255, 255, 255, 0.05),
152     0 18px 30px rgba(0, 0, 0, 0.35);
153 }
154
155 #draw-canvas {
156   width: 100%;
157   height: 100%;
158   display: block;
159 }
160
161 #preview-canvas {
162   display: block;
163   image-rendering: pixelated;
164   background: var(--black);
165   border-radius: 6px;
166 }
167
168 .control-row {

```

```

169 width: min(360px, calc(100% - 36px));
170 display: grid;
171 gap: 10px;
172 color: #c5d5e7;
173 font-weight: 700;
174 }
175
176 .brush-control {
177   grid-template-columns: auto 1fr auto;
178 }
179
180 .invert-control {
181   margin-top: 10px;
182   grid-template-columns: auto auto 1fr;
183 }
184
185 .brush-control label,
186 .brush-control span {
187   white-space: nowrap;
188 }
189
190 .brush-control input[type="range"] {
191   width: 100%;
192   accent-color: var(--accent);
193 }
194
195 .switch {
196   width: 50px;
197   height: 28px;
198   display: inline-grid;
199   align-items: center;
200   cursor: pointer;
201 }
202
203 .switch input {
204   position: absolute;
205   opacity: 0;
206   pointer-events: none;
207 }
208
209 .switch-track {
210   position: relative;
211   width: 50px;
212   height: 28px;
213   border: 1px solid var(--line-strong);
214   border-radius: 999px;
215   background: #0a1018;
216   transition: background 0.16s ease, border-color 0.16s ease;
217 }
218
219 .switch-track::after {
220   content: "";
221   position: absolute;
222   top: 4px;
223   left: 4px;
224   width: 18px;
225   height: 18px;
226   border-radius: 50%;

```

```

227     background: #8fa1b7;
228     transition: transform 0.16s ease, background 0.16s ease;
229 }
230
231 .switch input:checked + .switch-track {
232     border-color: var(--accent);
233     background: #123247;
234 }
235
236 .switch input:checked + .switch-track::after {
237     transform: translateX(22px);
238     background: var(--accent);
239 }
240
241 #invert-status {
242     color: var(--muted);
243 }
244
245 .actions {
246     width: min(360px, calc(100% - 36px));
247     margin: 16px 0 12px;
248     display: grid;
249     grid-template-columns: repeat(3, 1fr);
250     gap: 10px;
251 }
252
253 .label-save-row {
254     width: min(360px, calc(100% - 36px));
255     display: grid;
256     grid-template-columns: auto minmax(0, 1fr) 72px;
257     align-items: center;
258     gap: 8px;
259     color: #d8e6f5;
260     font-size: 0.9rem;
261     font-weight: 800;
262 }
263
264 .label-save-row select,
265 #sample-label-select {
266     appearance: none;
267     -webkit-appearance: none;
268     min-width: 0;
269     width: 100%;
270     height: 34px;
271     border: 1px solid var(--line-strong);
272     border-radius: 6px;
273     padding: 0 34px 0 12px;
274     background:
275         linear-gradient(45deg, transparent 50%, #cfe0f1 50%) right 13px center / 6px 6px no-repeat,
276         linear-gradient(135deg, #cfe0f1 50%, transparent 50%) right 8px center / 6px 6px no-repeat,
277         #0a1018;
278     color: var(--ink);
279     font: inherit;
280 }
281
282 .label-save-row select option,
283 #sample-label-select option {
284     background: #0a1018;

```

```

285     color: var(--ink);
286 }
287
288 .label-save-row button {
289     width: 72px;
290     min-height: 34px;
291     padding: 7px 12px;
292     font-size: 0.86rem;
293 }
294
295 .save-status {
296     width: min(360px, calc(100% - 36px));
297     margin: 8px 0 0;
298     color: var(--muted);
299     font-size: 0.86rem;
300     font-weight: 700;
301 }
302
303 button,
304 .upload-btn {
305     min-height: 42px;
306     border: 1px solid transparent;
307     border-radius: 6px;
308     padding: 10px 12px;
309     font-size: 0.95rem;
310     font-weight: 800;
311     cursor: pointer;
312     transition: transform 0.14s ease, border-color 0.14s ease, background 0.14s ease;
313 }
314
315 button:hover,
316 .upload-btn:hover {
317     transform: translateY(-1px);
318 }
319
320 button:disabled {
321     cursor: wait;
322     opacity: 0.68;
323     transform: none;
324 }
325
326 .primary-btn {
327     background: var(--accent);
328     color: #041019;
329     box-shadow: 0 10px 24px rgba(56, 189, 248, 0.2);
330 }
331
332 .primary-btn:hover {
333     background: var(--accent-strong);
334 }
335
336 .ghost-btn {
337     background: #172232;
338     color: #d8e6f5;
339     border-color: #2f425b;
340 }
341
342 .ghost-btn:hover,

```

```

343 .upload-btn:hover {
344   border-color: var(--accent);
345 }
346
347 .upload-btn {
348   display: grid;
349   place-items: center;
350   background: #172232;
351   color: #d8e6f5;
352   border-color: #2f425b;
353   text-align: center;
354 }
355
356 #upload-image-input {
357   display: none;
358 }
359
360 .results-panel {
361   display: flex;
362   flex-direction: column;
363 }
364
365 .result-stack {
366   padding: 18px;
367   display: grid;
368   gap: 12px;
369 }
370
371 .result-block {
372   border: 1px solid var(--line);
373   border-radius: 8px;
374   padding: 16px;
375   background: #0c131d;
376 }
377
378 .result-block:first-child {
379   border-color: rgba(56, 189, 248, 0.55);
380   background: #0d1722;
381 }
382
383 .prediction-row {
384   display: grid;
385   grid-template-columns: 88px 1fr;
386   align-items: center;
387   gap: 15px;
388 }
389
390 .prediction-value {
391   width: 88px;
392   height: 88px;
393   border-radius: 8px;
394   display: grid;
395   place-items: center;
396   background: #132337;
397   border: 1px solid #28506c;
398   color: var(--accent-strong);
399   font-size: 3rem;
400   font-weight: 900;

```

```
401     line-height: 1;
402 }
403
404 .result-block:first-child .prediction-value {
405     background: var(--accent);
406     color: #041019;
407 }
408
409 .compact-result .prediction-value {
410     width: 68px;
411     height: 68px;
412     font-size: 2.2rem;
413 }
414
415 .compact-result .prediction-row {
416     grid-template-columns: 68px 1fr;
417 }
418
419 .prediction-meta {
420     color: #c6d5e7;
421     line-height: 1.6;
422     overflow-wrap: anywhere;
423 }
424
425 .prediction-meta p:last-child {
426     margin-bottom: 0;
427 }
428
429 .preview-block,
430 .debug-block,
431 .performance-panel {
432     overflow: hidden;
433 }
434
435 .preview-frame {
436     padding: 18px;
437     display: grid;
438     place-items: center;
439 }
440
441 .debug-block {
442     max-height: 520px;
443     overflow: auto;
444 }
445
446 .debug-summary {
447     margin: 14px 18px;
448     color: #c2d1e2;
449     line-height: 1.55;
450 }
451
452 .debug-grid {
453     padding: 0 18px 18px;
454     display: grid;
455     gap: 12px;
456 }
457
458 .debug-card {
```

```

459 | border: 1px solid var(--line);
460 | border-radius: 8px;
461 | padding: 12px;
462 | background: #0b111a;
463 | }
464 |
465 | .debug-card-title {
466 |   margin: 0 0 10px;
467 |   color: #b9c8da;
468 |   font-size: 0.9rem;
469 |   font-weight: 800;
470 |   text-transform: uppercase;
471 |   letter-spacing: 0.05em;
472 | }
473 |
474 | .debug-card-stats {
475 |   margin: 10px 0 0;
476 |   color: #c3d0df;
477 |   font-size: 0.92rem;
478 |   line-height: 1.5;
479 | }
480 |
481 | .debug-card-data {
482 |   margin: 0;
483 |   padding: 12px;
484 |   max-height: 280px;
485 |   overflow: auto;
486 |   border-radius: 6px;
487 |   background: #05080d;
488 |   color: #e5eef8;
489 |   font-size: 0.84rem;
490 |   line-height: 1.5;
491 |   font-family: Consolas, "Courier New", monospace;
492 |   white-space: pre;
493 | }
494 |
495 | .metric-grid {
496 |   padding: 18px;
497 |   display: grid;
498 |   gap: 10px;
499 | }
500 |
501 | .metric-grid div {
502 |   display: grid;
503 |   grid-template-columns: 92px 1fr;
504 |   align-items: center;
505 |   gap: 10px;
506 |   border-bottom: 1px solid var(--line);
507 |   padding-bottom: 10px;
508 | }
509 |
510 | .metric-grid div:last-child {
511 |   border-bottom: 0;
512 |   padding-bottom: 0;
513 | }
514 |
515 | .metric-grid span {
516 |   color: var(--subtle);

```

```
517     font-size: 0.9rem;
518     font-weight: 800;
519 }
520
521 .metric-grid strong {
522     color: var(--ink);
523     font-size: 0.98rem;
524     overflow-wrap: anywhere;
525 }
526
527 .speed-chart {
528     margin: 0 18px 18px;
529     min-height: 150px;
530     border-top: 1px solid var(--line);
531     padding-top: 18px;
532     display: grid;
533     grid-template-columns: repeat(3, 1fr);
534     gap: 14px;
535     align-items: end;
536 }
537
538 .speed-bar-group {
539     min-width: 0;
540     display: grid;
541     grid-template-rows: 112px auto;
542     gap: 9px;
543     justify-items: center;
544     color: var(--muted);
545     font-size: 0.82rem;
546     font-weight: 800;
547 }
548
549 .speed-bar-track {
550     width: 100%;
551     max-width: 54px;
552     height: 112px;
553     display: flex;
554     align-items: end;
555     border: 1px solid var(--line);
556     border-radius: 6px;
557     background: #0a1018;
558     overflow: hidden;
559 }
560
561 .speed-bar {
562     width: 100%;
563     height: 0%;
564     min-height: 0;
565     transition: height 0.28s ease;
566 }
567
568 .cpu-exact-bar {
569     background: #64748b;
570 }
571
572 .cpu-quant-bar {
573     background: #94a3b8;
574 }
```

```
575
576 .fpga-bar {
577     background: var(--accent);
578 }
579
580 @media (max-width: 1100px) {
581     .console-grid,
582     .analysis-grid {
583         grid-template-columns: 1fr;
584     }
585
586     .debug-block {
587         max-height: none;
588     }
589 }
590
591 @media (max-width: 720px) {
592     .page {
593         width: min(100% - 20px, 1440px);
594         padding-top: 10px;
595     }
596
597     .topbar {
598         align-items: flex-start;
599         flex-direction: column;
600     }
601
602     .status-strip {
603         justify-content: flex-start;
604     }
605
606     .actions {
607         grid-template-columns: 1fr 1fr;
608     }
609
610     .prediction-row,
611     .compact-result .prediction-row {
612         grid-template-columns: 64px 1fr;
613     }
614
615     .prediction-value,
616     .compact-result .prediction-value {
617         width: 64px;
618         height: 64px;
619         font-size: 2rem;
620     }
621 }
```

Listing 7: Python package requirements for the web demo.

```

1 Flask>=3.0.0
2 Pillow>=10.0.0

```

## A.2 HPS Software Code

Listing 8: HPS FPGA register constants shared with the C server.

```

1 #ifndef HPS_FPGA_REGS_H
2 #define HPS_FPGA_REGS_H
3
4 #define LW_BRIDGE_BASE          0xFF200000u
5 #define HPS_FPGA_MAP_SIZE      0x1030u
6
7 #define REG_CONTROL_STATUS_WORD 0u
8 #define REG_RESULT_WORD        1u
9 #define REG_LOGIT_BASE_WORD    2u
10 #define REG_PIXEL_BASE_WORD    12u
11
12 #define REG_CONTROL_STATUS_OFFSET (REG_CONTROL_STATUS_WORD * 4u)
13 #define REG_RESULT_OFFSET        (REG_RESULT_WORD * 4u)
14 #define REG_LOGIT_BASE_OFFSET    (REG_LOGIT_BASE_WORD * 4u)
15 #define REG_PIXEL_BASE_OFFSET    (REG_PIXEL_BASE_WORD * 4u)
16
17 /* CONTROL write bits */
18 #define CTRL_START              (1u << 0)
19 #define CTRL_CLEAR_DONE        (1u << 1)
20 #define CTRL_CLEAR_INPUT_LOADED (1u << 2)
21
22 /* STATUS read bits */
23 #define STATUS_REGS_VALID      (1u << 0)
24 #define STATUS_BUSY            (1u << 1)
25 #define STATUS_DONE            (1u << 2)
26 #define STATUS_INPUT_LOADED    (1u << 3)
27
28 #define FPGA_INPUT_PIXELS      1024u
29
30 #endif

```

Listing 9: HPS inference server for TCP receive, MMIO register access, polling, and result response.

```

1 #include <arpa/inet.h>
2 #include <errno.h>
3 #include <fcntl.h>
4 #include <netinet/in.h>
5 #include <signal.h>
6 #include <stdint.h>
7 #include <stdarg.h>
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <string.h>
11 #include <sys/mman.h>
12 #include <sys/socket.h>
13 #include <sys/types.h>
14 #include <unistd.h>
15

```

```

16 #include "hps_fpga_regs.h"
17
18 #define SERVER_PORT 9090
19 #define MAX_REQUEST_BYTES 16384
20 #define EXPECTED_PIXELS FPGA_INPUT_PIXELS
21 #define FPGA_POLL_LIMIT 30000
22 #define FPGA_POLL_DELAY_US 1000
23 #define FPGA_LOGIT_POLL_LIMIT 1000
24 #define FPGA_LOGIT_COUNT 10
25 #define FPGA_CLOCK_HZ 50000000.0
26 #define MAX_RESPONSE_BYTES 65536
27 #define INVALID_LW_BRIDGE_WORD 0xADADADADu
28
29 enum fpga_infer_result {
30     FPGA_INFER_OK = 0,
31     FPGA_INFER_DEVMEM_ERROR = -1,
32     FPGA_INFER_TIMEOUT = 1,
33     FPGA_INFER_INVALID_LW_READ = 2
34 };
35
36 static volatile sig_atomic_t keep_running = 1;
37
38 static void handle_sigint(int sig)
39 {
40     (void)sig;
41     keep_running = 0;
42 }
43
44 static int request_contains_infer_command(const char *text)
45 {
46     return strstr(text, "\command\") && strstr(text, "\infer\");
47 }
48
49 static int extract_request_id(const char *text, char *out, size_t out_size)
50 {
51     const char *tag = strstr(text, "\request_id\");
52     const char *colon;
53     const char *first_quote;
54     const char *second_quote;
55     size_t len;
56
57     if (!tag) {
58         out[0] = '\0';
59         return 0;
60     }
61
62     colon = strchr(tag, ':');
63     if (!colon) {
64         out[0] = '\0';
65         return 0;
66     }
67
68     first_quote = strchr(colon, '"');
69     if (!first_quote) {
70         out[0] = '\0';
71         return 0;
72     }
73     ++first_quote;

```

```

74
75 second_quote = strchr(first_quote, '"');
76 if (!second_quote) {
77     out[0] = '\0';
78     return 0;
79 }
80
81 len = (size_t)(second_quote - first_quote);
82 if (len >= out_size) {
83     len = out_size - 1;
84 }
85
86 memcpy(out, first_quote, len);
87 out[len] = '\0';
88 return 1;
89 }
90
91 static int parse_pixel_values(const char *text, uint8_t *pixels, int max_pixels)
92 {
93     int count = 0;
94     const char *pixels_tag = strstr(text, "\"pixels\"");
95     const char *p;
96
97     if (!pixels_tag) {
98         return -1;
99     }
100
101     p = strchr(pixels_tag, '[');
102     if (!p) {
103         return -1;
104     }
105     ++p;
106
107     while (*p && *p != ']') {
108         char *endptr;
109         long value;
110
111         while (*p == ' ' || *p == '\n' || *p == '\r' || *p == '\t' || *p == ',') {
112             ++p;
113         }
114         if (*p == ']') {
115             break;
116         }
117
118         errno = 0;
119         value = strtol(p, &endptr, 10);
120         if (endptr == p || errno != 0 || value < 0 || value > 255 || count >= max_pixels) {
121             return -1;
122         }
123
124         pixels[count] = (uint8_t)value;
125         ++count;
126         p = endptr;
127     }
128
129     return count;
130 }
131

```

```

132 static void write_response(int client_fd, const char *response)
133 {
134     size_t remaining = strlen(response);
135     const char *p = response;
136
137     while (remaining > 0) {
138         ssize_t written = send(client_fd, p, remaining, 0);
139         if (written <= 0) {
140             return;
141         }
142         p += written;
143         remaining -= (size_t)written;
144     }
145 }
146
147 static int open_fpga_regs(volatile uint32_t **regs_out, void **base_out, int *fd_out)
148 {
149     int fd;
150     void *base;
151
152     fd = open("/dev/mem", O_RDWR | O_SYNC);
153     if (fd < 0) {
154         return -1;
155     }
156
157     base = mmap(NULL, HPS_FPGA_MAP_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, LW_BRIDGE_BASE);
158     if (base == MAP_FAILED) {
159         close(fd);
160         return -1;
161     }
162
163     *regs_out = (volatile uint32_t *)base;
164     *base_out = base;
165     *fd_out = fd;
166     return 0;
167 }
168
169 static void close_fpga_regs(void *base, int fd)
170 {
171     if (base && base != MAP_FAILED) {
172         munmap(base, HPS_FPGA_MAP_SIZE);
173     }
174     if (fd >= 0) {
175         close(fd);
176     }
177 }
178
179 static void read_fpga_outputs(
180     volatile uint32_t *regs,
181     int *predicted_digit,
182     int32_t *logits,
183     uint32_t *cycle_count
184 )
185 {
186     int index;
187     uint32_t result_word;
188
189     result_word = regs[REG_RESULT_WORD];

```

```

190 *predicted_digit = (int)(result_word & 0xfu);
191 *cycle_count = result_word >> 4;
192 for (index = 0; index < FPGA_LOGIT_COUNT; ++index) {
193     logits[index] = (int32_t)regs[REG_LOGIT_BASE_WORD + index];
194 }
195 }
196
197 static int outputs_contain_invalid_lw_read(int predicted_digit, const int32_t *logits)
198 {
199     int index;
200
201     if (predicted_digit == (int)(INVALID_LW_BRIDGE_WORD & 0xfu)) {
202         for (index = 0; index < FPGA_LOGIT_COUNT; ++index) {
203             if ((uint32_t)logits[index] != INVALID_LW_BRIDGE_WORD) {
204                 return 0;
205             }
206         }
207         return 1;
208     }
209
210     return 0;
211 }
212
213 static int run_fpga_inference_with_logits(
214     const uint8_t *pixels,
215     uint32_t *status_before,
216     uint32_t *status_after,
217     int *predicted_digit,
218     int32_t *logits,
219     uint32_t *cycle_count,
220     int *debug_valid
221 )
222 {
223     volatile uint32_t *regs;
224     void *base = NULL;
225     int fd = -1;
226     int index;
227     int poll_count;
228
229     *debug_valid = 0;
230
231     if (open_fpga_regs(&regs, &base, &fd) != 0) {
232         return FPGA_INFER_DEVMEM_ERROR;
233     }
234
235     *status_before = regs[REG_CONTROL_STATUS_WORD];
236
237     regs[REG_CONTROL_STATUS_WORD] = CTRL_CLEAR_DONE | CTRL_CLEAR_INPUT_LOADED;
238
239     for (index = 0; index < EXPECTED_PIXELS; ++index) {
240         regs[REG_PIXEL_BASE_WORD + index] = pixels[index];
241     }
242
243     regs[REG_CONTROL_STATUS_WORD] = CTRL_START | CTRL_CLEAR_DONE;
244
245     for (poll_count = 0; poll_count < FPGA_POLL_LIMIT; ++poll_count) {
246         *status_after = regs[REG_CONTROL_STATUS_WORD];
247         if ((*status_after & STATUS_DONE) != 0u) {

```

```

248     for (index = 0; index < FPGA_LOGIT_POLL_LIMIT; ++index) {
249         *status_after = regs[REG_CONTROL_STATUS_WORD];
250         if ((*status_after & STATUS_REGS_VALID) != 0u) {
251             *debug_valid = 1;
252             break;
253         }
254         usleep(FPGA_POLL_DELAY_US);
255     }
256     if (*debug_valid) {
257         read_fpga_outputs(regs, predicted_digit, logits, cycle_count);
258         if (outputs_contain_invalid_lw_read(*predicted_digit, logits)) {
259             close_fpga_regs(base, fd);
260             return FPGA_INFER_INVALID_LW_READ;
261         }
262     }
263     close_fpga_regs(base, fd);
264     return FPGA_INFER_OK;
265 }
266     usleep(FPGA_POLL_DELAY_US);
267 }
268
269     close_fpga_regs(base, fd);
270     return FPGA_INFER_TIMEOUT;
271 }
272
273 static int append_text(char *buffer, size_t buffer_size, size_t *used, const char *text)
274 {
275     int written = snprintf(buffer + *used, buffer_size - *used, "%s", text);
276     if (written < 0 || (size_t)written >= buffer_size - *used) {
277         return -1;
278     }
279     *used += (size_t)written;
280     return 0;
281 }
282
283 static int append_format(char *buffer, size_t buffer_size, size_t *used, const char *fmt, ...)
284 {
285     va_list args;
286     int written;
287
288     va_start(args, fmt);
289     written = vsnprintf(buffer + *used, buffer_size - *used, fmt, args);
290     va_end(args);
291
292     if (written < 0 || (size_t)written >= buffer_size - *used) {
293         return -1;
294     }
295     *used += (size_t)written;
296     return 0;
297 }
298
299 static int build_success_response(
300     char *response,
301     size_t response_size,
302     const char *request_id,
303     uint32_t status_before,
304     uint32_t status_after,
305     int predicted_digit,

```

```

306     const int32_t *logits,
307     uint32_t cycle_count,
308     int debug_valid
309 )
310 {
311     size_t used = 0;
312     int index;
313     double fpga_compute_ms = ((double)cycle_count * 1000.0) / FPGA_CLOCK_HZ;
314
315     if (append_format(
316         response,
317         response_size,
318         &used,
319         "{\\"ok\\":true,\\"request_id\\":\\"%s\\",\\"source\\":\\"fpga\\",\\"prediction\\":%d,"
320         "\\"confidence\\":0.0,\\"status_before\\":%u,\\"status_after\\":%u,"
321         "\\"note\\":\\"%s\\",\\"debug\\":{\\"kind\\":\\"final_logits\\",\\"logits_valid\\":%s,"
322         "\\"fpga_cycles\\":%u,\\"fpga_clock_hz\\":50000000,\\"fpga_compute_ms\\":%.6f",
323         request_id,
324         predicted_digit,
325         status_before,
326         status_after,
327         debug_valid ? "hardware_run_completed_logits_available" : "
hardware_run_completed_but_logits_not_valid",
328         debug_valid ? "true" : "false",
329         cycle_count,
330         fpga_compute_ms
331     ) != 0) {
332         return -1;
333     }
334
335     if (!debug_valid) {
336         if (append_text(response, response_size, &used, "}}\n") != 0) {
337             return -1;
338         }
339         return 0;
340     }
341
342     if (append_text(response, response_size, &used, "\\"logits\\":[") != 0) {
343         return -1;
344     }
345
346     for (index = 0; index < FPGA_LOGIT_COUNT; ++index) {
347         if (append_format(
348             response,
349             response_size,
350             &used,
351             index == 0 ? "%d" : ",%d",
352             logits[index]
353         ) != 0) {
354             return -1;
355         }
356     }
357
358     if (append_text(response, response_size, &used, "]]\n") != 0) {
359         return -1;
360     }
361
362     return 0;

```

```

363 }
364
365 static void handle_client(int client_fd)
366 {
367     char buffer[MAX_REQUEST_BYTES + 1];
368     char response[MAX_RESPONSE_BYTES];
369     char request_id[128];
370     uint8_t pixels[EXPECTED_PIXELS];
371     int32_t logits[FPGA_LOGIT_COUNT];
372     ssize_t received;
373     int pixel_count;
374     uint32_t status_before;
375     uint32_t status_after;
376     uint32_t cycle_count;
377     int fpga_result;
378     int debug_valid;
379     int predicted_digit;
380
381     received = recv(client_fd, buffer, MAX_REQUEST_BYTES, 0);
382     if (received <= 0) {
383         return;
384     }
385
386     buffer[received] = '\0';
387     extract_request_id(buffer, request_id, sizeof(request_id));
388
389     if (!request_contains_infer_command(buffer)) {
390         write_response(
391             client_fd,
392             "{\"ok\":false,\"error\":\"unsupported_command\"}\n"
393         );
394         return;
395     }
396
397     pixel_count = parse_pixel_values(buffer, pixels, EXPECTED_PIXELS);
398     if (pixel_count != EXPECTED_PIXELS) {
399         snprintf(
400             response,
401             sizeof(response),
402             "{\"ok\":false,\"request_id\":\"%s\",\"error\":\"invalid_pixel_payload\",\"expected_pixels
403             \":1024}\n",
404             request_id
405         );
406         write_response(client_fd, response);
407         return;
408     }
409
410     printf("received_infer_request_id=%s_pixels=%d\n", request_id, pixel_count);
411     fflush(stdout);
412
413     predicted_digit = -1;
414     cycle_count = 0u;
415     memset(logits, 0, sizeof(logits));
416
417     fpga_result = run_fpga_inference_with_logits(
418         pixels,
419         &status_before,
420         &status_after,

```

```

420     &predicted_digit,
421     logits,
422     &cycle_count,
423     &debug_valid
424 );
425 if (fpga_result == FPGA_INFER_DEVMEM_ERROR) {
426     snprintf(
427         response,
428         sizeof(response),
429         "{\"ok\":false,\"request_id\":\"%s\",\"error\":\"devmem_access_failed\"}\n",
430         request_id
431     );
432     write_response(client_fd, response);
433     return;
434 }
435
436 if (fpga_result != FPGA_INFER_OK) {
437     if (fpga_result == FPGA_INFER_INVALID_LW_READ) {
438         snprintf(
439             response,
440             sizeof(response),
441             "{\"ok\":false,\"request_id\":\"%s\",\"error\":\"lw_bridge_invalid_read\",\"detail\":\"\n",
442             read_0xADADADAD_from_fpga_regs\",\"status_before\":%u,\"status_after\":%u}\n",
443             request_id,
444             status_before,
445             status_after
446         );
447         write_response(client_fd, response);
448         return;
449     }
450     snprintf(
451         response,
452         sizeof(response),
453         "{\"ok\":false,\"request_id\":\"%s\",\"error\":\"fpga_timeout\",\"status_before\":%u,\"",
454         status_after\":%u}\n",
455         request_id,
456         status_before,
457         status_after
458     );
459     write_response(client_fd, response);
460     return;
461 }
462
463 if (build_success_response(
464     response,
465     sizeof(response),
466     request_id,
467     status_before,
468     status_after,
469     predicted_digit,
470     logits,
471     cycle_count,
472     debug_valid
473 ) != 0) {
474     snprintf(
475         response,
476         sizeof(response),

```

```

476     {"ok":false,"request_id":"%s","error":"response_buffer_overflow"}\n",
477     request_id
478 );
479 }
480
481 printf(
482     "completed_request_id=%s_prediction=%d_status_before=%u_status_after=%u_debug_valid=%d_
483     fpga_cycles=%u\n",
484     request_id,
485     predicted_digit,
486     status_before,
487     status_after,
488     debug_valid,
489     cycle_count
490 );
491 fflush(stdout);
492 write_response(client_fd, response);
493 }
494
495 int main(void)
496 {
497     int server_fd;
498     int optval = 1;
499     struct sockaddr_in addr;
500
501     signal(SIGINT, handle_sigint);
502     signal(SIGTERM, handle_sigint);
503
504     server_fd = socket(AF_INET, SOCK_STREAM, 0);
505     if (server_fd < 0) {
506         perror("socket");
507         return 1;
508     }
509
510     if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &optval, sizeof(optval)) < 0) {
511         perror("setsockopt");
512         close(server_fd);
513         return 1;
514     }
515
516     memset(&addr, 0, sizeof(addr));
517     addr.sin_family = AF_INET;
518     addr.sin_addr.s_addr = htonl(INADDR_ANY);
519     addr.sin_port = htons(SERVER_PORT);
520
521     if (bind(server_fd, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
522         perror("bind");
523         close(server_fd);
524         return 1;
525     }
526
527     if (listen(server_fd, 4) < 0) {
528         perror("listen");
529         close(server_fd);
530         return 1;
531     }
532
533     printf("hps_infer_server_listening_on_port_%d\n", SERVER_PORT);

```

```

533
534 while (keep_running) {
535     int client_fd = accept(server_fd, NULL, NULL);
536     if (client_fd < 0) {
537         if (errno == EINTR) {
538             continue;
539         }
540         perror("accept");
541         break;
542     }
543
544     handle_client(client_fd);
545     close(client_fd);
546 }
547
548 close(server_fd);
549 return 0;
550 }

```

### A.3 FPGA RTL Code

Listing 10: Board-level HPS and FPGA top module.

```

1 // =====
2 // Copyright (c) 2013 by Terasic Technologies Inc.
3 // =====
4 //
5 // Modified 2019 by Stephen A. Edwards
6 // Modified 2026 to expose full LeNet inference registers to HPS LW bridge
7 //
8 module soc_system_top(
9
10 /////////////// ADC ///////////////
11 inout      ADC_CS_N,
12 output     ADC_DIN,
13 input      ADC_DOUT,
14 output     ADC_SCLK,
15
16 /////////////// AUD ///////////////
17 input      AUD_ADCDAT,
18 inout     AUD_ADCLRCK,
19 inout     AUD_BCLK,
20 output     AUD_DACDAT,
21 inout     AUD_DACLCK,
22 output     AUD_XCK,
23
24 /////////////// CLOCK2 ///////////////
25 input      CLOCK2_50,
26
27 /////////////// CLOCK3 ///////////////
28 input      CLOCK3_50,
29
30 /////////////// CLOCK4 ///////////////
31 input      CLOCK4_50,
32
33 /////////////// CLOCK ///////////////
34 input      CLOCK_50,

```

```

35
36 /////////////// DRAM ///////////////
37 output [12:0] DRAM_ADDR,
38 output [1:0] DRAM_BA,
39 output      DRAM_CAS_N,
40 output      DRAM_CKE,
41 output      DRAM_CLK,
42 output      DRAM_CS_N,
43 inout [15:0] DRAM_DQ,
44 output      DRAM_LDQM,
45 output      DRAM_RAS_N,
46 output      DRAM_UDQM,
47 output      DRAM_WE_N,
48
49 /////////////// FAN ///////////////
50 output      FAN_CTRL,
51
52 /////////////// FPGA ///////////////
53 output      FPGA_I2C_SCLK,
54 inout      FPGA_I2C_SDAT,
55
56 /////////////// GPIO ///////////////
57 inout [35:0] GPIO_0,
58 inout [35:0] GPIO_1,
59
60 /////////////// HEX0 ///////////////
61 output [6:0] HEX0,
62
63 /////////////// HEX1 ///////////////
64 output [6:0] HEX1,
65
66 /////////////// HEX2 ///////////////
67 output [6:0] HEX2,
68
69 /////////////// HEX3 ///////////////
70 output [6:0] HEX3,
71
72 /////////////// HEX4 ///////////////
73 output [6:0] HEX4,
74
75 /////////////// HEX5 ///////////////
76 output [6:0] HEX5,
77
78 /////////////// HPS ///////////////
79 inout      HPS_CONV_USB_N,
80 output [14:0] HPS_DDR3_ADDR,
81 output [2:0] HPS_DDR3_BA,
82 output      HPS_DDR3_CAS_N,
83 output      HPS_DDR3_CKE,
84 output      HPS_DDR3_CK_N,
85 output      HPS_DDR3_CK_P,
86 output      HPS_DDR3_CS_N,
87 output [3:0] HPS_DDR3_DM,
88 inout [31:0] HPS_DDR3_DQ,
89 inout [3:0] HPS_DDR3_DQS_N,
90 inout [3:0] HPS_DDR3_DQS_P,
91 output      HPS_DDR3_ODT,
92 output      HPS_DDR3_RAS_N,

```

```

93 output      HPS_DDR3_RESET_N,
94 input      HPS_DDR3_RZQ,
95 output      HPS_DDR3_WE_N,
96 output      HPS_ENET_GTX_CLK,
97 inout      HPS_ENET_INT_N,
98 output      HPS_ENET_MDC,
99 inout      HPS_ENET_MDIO,
100 input      HPS_ENET_RX_CLK,
101 input [3:0] HPS_ENET_RX_DATA,
102 input      HPS_ENET_RX_DV,
103 output [3:0] HPS_ENET_TX_DATA,
104 output      HPS_ENET_TX_EN,
105 inout      HPS_GSENSOR_INT,
106 inout      HPS_I2C1_SCLK,
107 inout      HPS_I2C1_SDAT,
108 inout      HPS_I2C2_SCLK,
109 inout      HPS_I2C2_SDAT,
110 inout      HPS_I2C_CONTROL,
111 inout      HPS_KEY,
112 inout      HPS_LED,
113 inout      HPS_LTC_GPIO,
114 output      HPS_SD_CLK,
115 inout      HPS_SD_CMD,
116 inout [3:0] HPS_SD_DATA,
117 output      HPS_SPIM_CLK,
118 input      HPS_SPIM_MISO,
119 output      HPS_SPIM_MOSI,
120 inout      HPS_SPIM_SS,
121 input      HPS_UART_RX,
122 output      HPS_UART_TX,
123 input      HPS_USB_CLKOUT,
124 inout [7:0] HPS_USB_DATA,
125 input      HPS_USB_DIR,
126 input      HPS_USB_NXT,
127 output      HPS_USB_STP,
128
129 ////////// IRDA //////////
130 input      IRDA_RXD,
131 output      IRDA_TXD,
132
133 ////////// KEY //////////
134 input [3:0] KEY,
135
136 ////////// LEDR //////////
137 output [9:0] LEDR,
138
139 ////////// PS2 //////////
140 inout      PS2_CLK,
141 inout      PS2_CLK2,
142 inout      PS2_DAT,
143 inout      PS2_DAT2,
144
145 ////////// SW //////////
146 input [9:0] SW,
147
148 ////////// TD //////////
149 input      TD_CLK27,
150 input [7:0] TD_DATA,

```

```

151 input      TD_HS,
152 output     TD_RESET_N,
153 input      TD_VS,
154
155
156 ////////// VGA //////////
157 output [7:0] VGA_B,
158 output     VGA_BLANK_N,
159 output     VGA_CLK,
160 output [7:0] VGA_G,
161 output     VGA_HS,
162 output [7:0] VGA_R,
163 output     VGA_SYNC_N,
164 output     VGA_VS
165 );
166
167 localparam integer LOGIT_COUNT = 10;
168 localparam integer PIXELS      = 32 * 32;
169
170 localparam [6:0] HEX_OFF        = 7'b1111111;
171 localparam [35:0] GPIO_HIZ     = {36{1'bZ}};
172 localparam [15:0] DRAM_DQ_HIZ  = {16{1'bZ}};
173
174 // All active logic in this top level runs from the board 50 MHz clock.
175 wire rst_n;
176 wire start_pulse;
177 wire clear_done_pulse;
178 reg  done_latched;
179 reg  accel_busy;
180 wire infer_done;
181 wire result_regs_valid;
182 wire [LOGIT_COUNT*32-1:0] logit_regs_flat;
183 wire pixel_wr_en;
184 wire [$clog2(PIXELS)-1:0] pixel_wr_addr;
185 wire [7:0] pixel_wr_data;
186 wire pixel_rd_en;
187 wire [$clog2(PIXELS)-1:0] pixel_rd_addr;
188 wire [7:0] pixel_rd_data;
189 wire input_loaded;
190 wire h2f_rst_n;
191 wire [3:0] predicted_digit;
192 wire signed [10*32-1:0] logits_flat;
193 wire [27:0] infer_cycle_count;
194 reg  [27:0] infer_cycle_count_latched;
195
196 // HPS lightweight AXI master connected to the LeNet register block.
197 wire [11:0] h2f_lw_awid;
198 wire [20:0] h2f_lw_awaddr;
199 wire [3:0]  h2f_lw_awlen;
200 wire [2:0]  h2f_lw_awsize;
201 wire [1:0]  h2f_lw_awburst;
202 wire [1:0]  h2f_lw_awlock;
203 wire [3:0]  h2f_lw_awcache;
204 wire [2:0]  h2f_lw_awprot;
205 wire       h2f_lw_awvalid;
206 wire       h2f_lw_awready;
207 wire [11:0] h2f_lw_wid;
208 wire [31:0] h2f_lw_wdata;

```

```

209 wire [3:0] h2f_lw_wstrb;
210 wire h2f_lw_wlast;
211 wire h2f_lw_wvalid;
212 wire h2f_lw_wready;
213 wire [11:0] h2f_lw_bid;
214 wire [1:0] h2f_lw_bresp;
215 wire h2f_lw_bvalid;
216 wire h2f_lw_bready;
217 wire [11:0] h2f_lw_arid;
218 wire [20:0] h2f_lw_araddr;
219 wire [3:0] h2f_lw_arlen;
220 wire [2:0] h2f_lw_arsize;
221 wire [1:0] h2f_lw_arburst;
222 wire [1:0] h2f_lw_arlock;
223 wire [3:0] h2f_lw_arcache;
224 wire [2:0] h2f_lw_arprot;
225 wire h2f_lw_arvalid;
226 wire h2f_lw_arready;
227 wire [11:0] h2f_lw_rid;
228 wire [31:0] h2f_lw_rdata;
229 wire [1:0] h2f_lw_rresp;
230 wire h2f_lw_rlast;
231 wire h2f_lw_rvalid;
232 wire h2f_lw_rready;
233
234 function automatic [6:0] hex_digit_to_7seg(input [3:0] value);
235     begin
236         case (value)
237             4'h0: hex_digit_to_7seg = 7'b1000000;
238             4'h1: hex_digit_to_7seg = 7'b1111001;
239             4'h2: hex_digit_to_7seg = 7'b0100100;
240             4'h3: hex_digit_to_7seg = 7'b0110000;
241             4'h4: hex_digit_to_7seg = 7'b0011001;
242             4'h5: hex_digit_to_7seg = 7'b0010010;
243             4'h6: hex_digit_to_7seg = 7'b0000010;
244             4'h7: hex_digit_to_7seg = 7'b1111000;
245             4'h8: hex_digit_to_7seg = 7'b0000000;
246             4'h9: hex_digit_to_7seg = 7'b0010000;
247             4'ha: hex_digit_to_7seg = 7'b0001000;
248             4'hb: hex_digit_to_7seg = 7'b0000011;
249             4'hc: hex_digit_to_7seg = 7'b1000110;
250             4'hd: hex_digit_to_7seg = 7'b0100001;
251             4'he: hex_digit_to_7seg = 7'b0000110;
252             default: hex_digit_to_7seg = 7'b0001110;
253         endcase
254     end
255 endfunction
256
257 assign rst_n = KEY[0] & h2f_rst_n;
258
259 always @(posedge CLOCK_50 or negedge rst_n) begin
260     if (!rst_n) begin
261         done_latched <= 1'b0;
262         accel_busy <= 1'b0;
263         infer_cycle_count_latched <= 28'd0;
264     end else begin
265         if (start_pulse || clear_done_pulse) begin
266             done_latched <= 1'b0;

```

```

267         end else if (infer_done) begin
268             done_latched             <= 1'b1;
269             infer_cycle_count_latched <= infer_cycle_count;
270         end
271
272         if (start_pulse) begin
273             accel_busy                 <= 1'b1;
274             infer_cycle_count_latched <= 28'd0;
275         end else if (infer_done) begin
276             accel_busy <= 1'b0;
277         end
278     end
279 end
280
281 // FPGA inference core.
282 lenet_int8_top dut (
283     .clk(CLOCK_50),
284     .rst_n(rst_n),
285     .start(start_pulse),
286     .pixel_wr_en(pixel_wr_en),
287     .pixel_wr_addr(pixel_wr_addr),
288     .pixel_wr_data(pixel_wr_data),
289     .pixel_dbg_rd_en(pixel_rd_en),
290     .pixel_dbg_rd_addr(pixel_rd_addr),
291     .pixel_dbg_rd_data(pixel_rd_data),
292     .busy(),
293     .done(infer_done),
294     .predicted_digit(predicted_digit),
295     .logits_flat(logits_flat),
296     .cycle_count(infer_cycle_count)
297 );
298
299 genvar logit_idx;
300 generate
301     for (logit_idx = 0; logit_idx < LOGIT_COUNT; logit_idx = logit_idx + 1) begin : g_logit_regs
302         assign logit_regs_flat[logit_idx * 32 +: 32] = logits_flat[logit_idx * 32 +: 32];
303     end
304 endgenerate
305
306 assign result_regs_valid = done_latched;
307
308 // Memory-mapped register window exposed to HPS at the lightweight bridge.
309 hps_lenet_regs #(
310     .REG_COUNT(LOGIT_COUNT),
311     .PIXELS(PIXELS)
312 ) u_regs (
313     .clk(CLOCK_50),
314     .rst_n(rst_n),
315     .regs_valid(result_regs_valid),
316     .accel_busy(accel_busy),
317     .accel_done(done_latched),
318     .predicted_digit(predicted_digit),
319     .cycle_count(infer_cycle_count_latched),
320     .regs_flat(logit_regs_flat),
321     .start_pulse(start_pulse),
322     .clear_done_pulse(clear_done_pulse),
323     .pixel_wr_en(pixel_wr_en),
324     .pixel_wr_addr(pixel_wr_addr),

```

```

325     .pixel_wr_data(pixel_wr_data),
326     .pixel_rd_en(pixel_rd_en),
327     .pixel_rd_addr(pixel_rd_addr),
328     .pixel_rd_data(pixel_rd_data),
329     .input_loaded(input_loaded),
330     .s_awid(h2f_lw_awid),
331     .s_awaddr(h2f_lw_awaddr),
332     .s_awvalid(h2f_lw_awvalid),
333     .s_awready(h2f_lw_awready),
334     .s_wdata(h2f_lw_wdata),
335     .s_wstrb(h2f_lw_wstrb),
336     .s_wlast(h2f_lw_wlast),
337     .s_wvalid(h2f_lw_wvalid),
338     .s_wready(h2f_lw_wready),
339     .s_bid(h2f_lw_bid),
340     .s_bresp(h2f_lw_bresp),
341     .s_bvalid(h2f_lw_bvalid),
342     .s_bready(h2f_lw_bready),
343     .s_arid(h2f_lw_arid),
344     .s_araddr(h2f_lw_araddr),
345     .s_arvalid(h2f_lw_arvalid),
346     .s_arready(h2f_lw_arready),
347     .s_rid(h2f_lw_rid),
348     .s_rdata(h2f_lw_rdata),
349     .s_rresp(h2f_lw_rresp),
350     .s_rlast(h2f_lw_rlast),
351     .s_rvalid(h2f_lw_rvalid),
352     .s_rready(h2f_lw_rready)
353 );
354
355 // HPS subsystem generated by Platform Designer.
356 soc_system_hps_0 #(
357     .F2S_Width(2),
358     .S2F_Width(2)
359 ) hps_0 (
360     .h2f_rst_n(h2f_rst_n),
361     .f2h_axi_clk(CLOCK_50),
362     .f2h_AWID(8'd0),
363     .f2h_AWADDR(32'd0),
364     .f2h_AWLEN(4'd0),
365     .f2h_AWSIZE(3'd0),
366     .f2h_AWBURST(2'd0),
367     .f2h_AWLOCK(2'd0),
368     .f2h_AWCACHE(4'd0),
369     .f2h_AWPROT(3'd0),
370     .f2h_AWVALID(1'b0),
371     .f2h_AWREADY(),
372     .f2h_AWUSER(5'd0),
373     .f2h_WID(8'd0),
374     .f2h_WDATA(64'd0),
375     .f2h_WSTRB(8'd0),
376     .f2h_WLAST(1'b0),
377     .f2h_WVALID(1'b0),
378     .f2h_WREADY(),
379     .f2h_BID(),
380     .f2h_BRESP(),
381     .f2h_BVALID(),
382     .f2h_BREADY(1'b0),

```

```

383     .f2h_ARID(8'd0),
384     .f2h_ARADDR(32'd0),
385     .f2h_ARLEN(4'd0),
386     .f2h_ARSIZE(3'd0),
387     .f2h_ARBURST(2'd0),
388     .f2h_ARLOCK(2'd0),
389     .f2h_ARCACHE(4'd0),
390     .f2h_ARPROT(3'd0),
391     .f2h_ARVALID(1'b0),
392     .f2h_ARREADY(),
393     .f2h_ARUSER(5'd0),
394     .f2h_RID(),
395     .f2h_RDATA(),
396     .f2h_RRESP(),
397     .f2h_RLAST(),
398     .f2h_RVALID(),
399     .f2h_RREADY(1'b0),
400     .h2f_lw_axi_clk(CLOCK_50),
401     .h2f_lw_AWID(h2f_lw_awid),
402     .h2f_lw_AWADDR(h2f_lw_awaddr),
403     .h2f_lw_AWLEN(h2f_lw_awlen),
404     .h2f_lw_AWSIZE(h2f_lw_awsiz),
405     .h2f_lw_AWBURST(h2f_lw_awburst),
406     .h2f_lw_AWLOCK(h2f_lw_awlock),
407     .h2f_lw_AWCACHE(h2f_lw_awcache),
408     .h2f_lw_AWPROT(h2f_lw_awprot),
409     .h2f_lw_AWVALID(h2f_lw_awvalid),
410     .h2f_lw_AWREADY(h2f_lw_awready),
411     .h2f_lw_WID(h2f_lw_wid),
412     .h2f_lw_WDATA(h2f_lw_wdata),
413     .h2f_lw_WSTRB(h2f_lw_wstrb),
414     .h2f_lw_WLAST(h2f_lw_wlast),
415     .h2f_lw_WVALID(h2f_lw_wvalid),
416     .h2f_lw_WREADY(h2f_lw_wready),
417     .h2f_lw_BID(h2f_lw_bid),
418     .h2f_lw_BRESP(h2f_lw_bresp),
419     .h2f_lw_BVALID(h2f_lw_bvalid),
420     .h2f_lw_BREADY(h2f_lw_bready),
421     .h2f_lw_ARID(h2f_lw_arid),
422     .h2f_lw_ARADDR(h2f_lw_araddr),
423     .h2f_lw_ARLEN(h2f_lw_arlen),
424     .h2f_lw_ARSIZE(h2f_lw_arsiz),
425     .h2f_lw_ARBURST(h2f_lw_arburst),
426     .h2f_lw_ARLOCK(h2f_lw_arlock),
427     .h2f_lw_ARCACHE(h2f_lw_arcache),
428     .h2f_lw_ARPROT(h2f_lw_arprot),
429     .h2f_lw_ARVALID(h2f_lw_arvalid),
430     .h2f_lw_ARREADY(h2f_lw_arready),
431     .h2f_lw_RID(h2f_lw_rid),
432     .h2f_lw_RDATA(h2f_lw_rdata),
433     .h2f_lw_RRESP(h2f_lw_rresp),
434     .h2f_lw_RLAST(h2f_lw_rlast),
435     .h2f_lw_RVALID(h2f_lw_rvalid),
436     .h2f_lw_RREADY(h2f_lw_rready),
437     .h2f_axi_clk(CLOCK_50),
438     .h2f_AWID(),
439     .h2f_AWADDR(),
440     .h2f_AWLEN(),

```

```

441 .h2f_AWSIZE(),
442 .h2f_AWBURST(),
443 .h2f_AWLOCK(),
444 .h2f_AWCACHE(),
445 .h2f_AWPROT(),
446 .h2f_AWVALID(),
447 .h2f_AWREADY(1'b0),
448 .h2f_WID(),
449 .h2f_WDATA(),
450 .h2f_WSTRB(),
451 .h2f_WLAST(),
452 .h2f_WVALID(),
453 .h2f_WREADY(1'b0),
454 .h2f_BID(12'd0),
455 .h2f_BRESP(2'd0),
456 .h2f_BVALID(1'b0),
457 .h2f_BREADY(),
458 .h2f_ARID(),
459 .h2f_ARADDR(),
460 .h2f_ARLEN(),
461 .h2f_ARSIZE(),
462 .h2f_ARBURST(),
463 .h2f_ARLOCK(),
464 .h2f_ARCACHE(),
465 .h2f_ARPROT(),
466 .h2f_ARVALID(),
467 .h2f_ARREADY(1'b0),
468 .h2f_RID(12'd0),
469 .h2f_RDATA(64'd0),
470 .h2f_RRESP(2'd0),
471 .h2f_RLAST(1'b0),
472 .h2f_RVALID(1'b0),
473 .h2f_RREADY(),
474 .mem_a(HPS_DDR3_ADDR),
475 .mem_ba(HPS_DDR3_BA),
476 .mem_ck(HPS_DDR3_CK_P),
477 .mem_ck_n(HPS_DDR3_CK_N),
478 .mem_cke(HPS_DDR3_CKE),
479 .mem_cs_n(HPS_DDR3_CS_N),
480 .mem_ras_n(HPS_DDR3_RAS_N),
481 .mem_cas_n(HPS_DDR3_CAS_N),
482 .mem_we_n(HPS_DDR3_WE_N),
483 .mem_reset_n(HPS_DDR3_RESET_N),
484 .mem_dq(HPS_DDR3_DQ),
485 .mem_dqs(HPS_DDR3_DQS_P),
486 .mem_dqs_n(HPS_DDR3_DQS_N),
487 .mem_odt(HPS_DDR3_ODT),
488 .mem_dm(HPS_DDR3_DM),
489 .oct_rzqin(HPS_DDR3_RZQ),
490 .hps_io_emac1_inst_TX_CLK(HPS_ENET_GTX_CLK),
491 .hps_io_emac1_inst_TXD0(HPS_ENET_TX_DATA[0]),
492 .hps_io_emac1_inst_TXD1(HPS_ENET_TX_DATA[1]),
493 .hps_io_emac1_inst_TXD2(HPS_ENET_TX_DATA[2]),
494 .hps_io_emac1_inst_TXD3(HPS_ENET_TX_DATA[3]),
495 .hps_io_emac1_inst_RXD0(HPS_ENET_RX_DATA[0]),
496 .hps_io_emac1_inst_MDIO(HPS_ENET_MDIO),
497 .hps_io_emac1_inst_MDC(HPS_ENET_MDC),
498 .hps_io_emac1_inst_RX_CTL(HPS_ENET_RX_DV),

```

```

499     .hps_io_emac1_inst_TX_CTL(HPS_ENET_TX_EN),
500     .hps_io_emac1_inst_RX_CLK(HPS_ENET_RX_CLK),
501     .hps_io_emac1_inst_RXD1(HPS_ENET_RX_DATA[1]),
502     .hps_io_emac1_inst_RXD2(HPS_ENET_RX_DATA[2]),
503     .hps_io_emac1_inst_RXD3(HPS_ENET_RX_DATA[3]),
504     .hps_io_sdio_inst_CMD(HPS_SD_CMD),
505     .hps_io_sdio_inst_D0(HPS_SD_DATA[0]),
506     .hps_io_sdio_inst_D1(HPS_SD_DATA[1]),
507     .hps_io_sdio_inst_CLK(HPS_SD_CLK),
508     .hps_io_sdio_inst_D2(HPS_SD_DATA[2]),
509     .hps_io_sdio_inst_D3(HPS_SD_DATA[3]),
510     .hps_io_usb1_inst_D0(HPS_USB_DATA[0]),
511     .hps_io_usb1_inst_D1(HPS_USB_DATA[1]),
512     .hps_io_usb1_inst_D2(HPS_USB_DATA[2]),
513     .hps_io_usb1_inst_D3(HPS_USB_DATA[3]),
514     .hps_io_usb1_inst_D4(HPS_USB_DATA[4]),
515     .hps_io_usb1_inst_D5(HPS_USB_DATA[5]),
516     .hps_io_usb1_inst_D6(HPS_USB_DATA[6]),
517     .hps_io_usb1_inst_D7(HPS_USB_DATA[7]),
518     .hps_io_usb1_inst_CLK(HPS_USB_CLKOUT),
519     .hps_io_usb1_inst_STP(HPS_USB_STP),
520     .hps_io_usb1_inst_DIR(HPS_USB_DIR),
521     .hps_io_usb1_inst_NXT(HPS_USB_NXT),
522     .hps_io_spim1_inst_CLK(HPS_SPIM_CLK),
523     .hps_io_spim1_inst_MOSI(HPS_SPIM_MOSI),
524     .hps_io_spim1_inst_MISO(HPS_SPIM_MISO),
525     .hps_io_spim1_inst_SS0(HPS_SPIM_SS),
526     .hps_io_uart0_inst_RX(HPS_UART_RX),
527     .hps_io_uart0_inst_TX(HPS_UART_TX),
528     .hps_io_i2c0_inst_SDA(HPS_I2C1_SDAT),
529     .hps_io_i2c0_inst_SCL(HPS_I2C1_SCLK),
530     .hps_io_i2c1_inst_SDA(HPS_I2C2_SDAT),
531     .hps_io_i2c1_inst_SCL(HPS_I2C2_SCLK),
532     .hps_io_gpio_inst_GPIO09(HPS_CONV_USB_N),
533     .hps_io_gpio_inst_GPIO35(HPS_ENET_INT_N),
534     .hps_io_gpio_inst_GPIO40(HPS_LTC_GPIO),
535     .hps_io_gpio_inst_GPIO48(HPS_I2C_CONTROL),
536     .hps_io_gpio_inst_GPIO53(HPS_LED),
537     .hps_io_gpio_inst_GPIO54(HPS_KEY),
538     .hps_io_gpio_inst_GPIO61(HPS_GSENSOR_INT)
539 );
540
541 // Tie unused board peripherals to stable values or high-Z so the DE1-Soc
542 // top-level pinout remains complete without changing the active design.
543 assign ADC_CS_N = SW[1] ? SW[0] : 1'bZ;
544 assign ADC_DIN = SW[0];
545 assign ADC_SCLK = SW[0];
546
547 assign AUD_ADCLRCK = SW[1] ? SW[0] : 1'bZ;
548 assign AUD_BCLK = SW[1] ? SW[0] : 1'bZ;
549 assign AUD_DACDAT = SW[0];
550 assign AUD_DACLCK = SW[1] ? SW[0] : 1'bZ;
551 assign AUD_XCK = SW[0];
552
553 assign DRAM_ADDR = { 13{ SW[0] } };
554 assign DRAM_BA = { 2{ SW[0] } };
555 assign DRAM_DQ = SW[1] ? { 16{ SW[0] } } : DRAM_DQ_HIZ;
556 assign {DRAM_CAS_N, DRAM_CKE, DRAM_CLK, DRAM_CS_N,

```

```

557     DRAM_LDQM, DRAM_RAS_N, DRAM_UDQM, DRAM_WE_N} = { 8{SW[0]} };
558
559     assign FAN_CTRL = SW[0];
560
561     assign FPGA_I2C_SCLK = SW[0];
562     assign FPGA_I2C_SDAT = SW[1] ? SW[0] : 1'bZ;
563
564     assign GPIO_0 = GPIO_HIZ;
565     assign GPIO_1 = GPIO_HIZ;
566
567     assign HEX0 = hex_digit_to_7seg(predicted_digit);
568     assign HEX1 = HEX_OFF;
569     assign HEX2 = HEX_OFF;
570     assign HEX3 = HEX_OFF;
571     assign HEX4 = HEX_OFF;
572     assign HEX5 = HEX_OFF;
573
574     assign IRDA_TXD = SW[0];
575
576     assign LEDR[0] = (predicted_digit == 4'd0);
577     assign LEDR[4:1] = predicted_digit;
578     assign LEDR[9:5] = 5'd0;
579
580     assign PS2_CLK = SW[1] ? SW[0] : 1'bZ;
581     assign PS2_CLK2 = SW[1] ? SW[0] : 1'bZ;
582     assign PS2_DAT = SW[1] ? SW[0] : 1'bZ;
583     assign PS2_DAT2 = SW[1] ? SW[0] : 1'bZ;
584
585     assign TD_RESET_N = SW[0];
586
587     assign {VGA_R, VGA_G, VGA_B} = { 24{ SW[0] } };
588     assign {VGA_BLANK_N, VGA_CLK,
589             VGA_HS, VGA_SYNC_N, VGA_VS} = { 5{ SW[0] } };
590 endmodule

```

Listing 11: Shared LeNet INT8 constants and dimensions.

```

1  package lenet_int8_pkg;
2  parameter int PIX_W = 8;
3  parameter int ACT_W = 16;
4  parameter int WGT_W = 8;
5  parameter int BIAS_W = 8;
6  parameter int ACC_W = 32;
7  parameter int MAC_LANES = 3;
8  parameter int SHIFT_W = 5;
9  parameter int ACT_MAX = (1 << ACT_W) - 1;
10
11 parameter int IMG_W = 32;
12 parameter int IMG_PIXELS = IMG_W * IMG_W;
13 parameter int MAX_FM_COUNT = 4704;
14
15 parameter int CONV1_IN_CH = 1;
16 parameter int CONV1_OUT_CH = 6;
17 parameter int CONV1_K = 5;
18 parameter int CONV1_CONV_W = 28;
19 parameter int CONV1_POOL_W = 14;
20 parameter int CONV1_POOL_COUNT = CONV1_OUT_CH * CONV1_POOL_W * CONV1_POOL_W;
21 parameter int CONV1_WEIGHT_COUNT = CONV1_OUT_CH * CONV1_IN_CH * CONV1_K * CONV1_K;

```

```

22 parameter int CONV1_BIAS_COUNT = CONV1_OUT_CH;
23 parameter int CONV1_PARAM_COUNT = CONV1_WEIGHT_COUNT + CONV1_BIAS_COUNT;
24
25 parameter int CONV2_IN_CH = 6;
26 parameter int CONV2_OUT_CH = 16;
27 parameter int CONV2_K = 5;
28 parameter int CONV2_CONV_W = 10;
29 parameter int CONV2_POOL_W = 5;
30 parameter int CONV2_POOL_COUNT = CONV2_OUT_CH * CONV2_POOL_W * CONV2_POOL_W;
31 parameter int CONV2_WEIGHT_COUNT = CONV2_OUT_CH * CONV2_IN_CH * CONV2_K * CONV2_K;
32 parameter int CONV2_BIAS_COUNT = CONV2_OUT_CH;
33 parameter int CONV2_PARAM_COUNT = CONV2_WEIGHT_COUNT + CONV2_BIAS_COUNT;
34
35 parameter int FC1_IN_COUNT = 400;
36 parameter int FC1_OUT_COUNT = 120;
37 parameter int FC1_WEIGHT_COUNT = FC1_IN_COUNT * FC1_OUT_COUNT;
38 parameter int FC1_BIAS_COUNT = FC1_OUT_COUNT;
39 parameter int FC1_PARAM_COUNT = FC1_WEIGHT_COUNT + FC1_BIAS_COUNT;
40
41 parameter int FC2_IN_COUNT = 120;
42 parameter int FC2_OUT_COUNT = 84;
43 parameter int FC2_WEIGHT_COUNT = FC2_IN_COUNT * FC2_OUT_COUNT;
44 parameter int FC2_BIAS_COUNT = FC2_OUT_COUNT;
45 parameter int FC2_PARAM_COUNT = FC2_WEIGHT_COUNT + FC2_BIAS_COUNT;
46
47 parameter int OUT_IN_COUNT = 84;
48 parameter int OUT_COUNT = 10;
49 parameter int OUT_WEIGHT_COUNT = OUT_IN_COUNT * OUT_COUNT;
50 parameter int OUT_BIAS_COUNT = OUT_COUNT;
51 parameter int OUT_PARAM_COUNT = OUT_WEIGHT_COUNT + OUT_BIAS_COUNT;
52
53 parameter int CONV1_SHIFT = 6;
54 parameter int CONV2_SHIFT = 8;
55 parameter int FC1_SHIFT = 8;
56 parameter int FC2_SHIFT = 7;
57 endpackage

```

Listing 12: Parameterized BRAM and ROM memory modules.

```

1 module BRAM #(
2     parameter DATA_WIDTH = 8,
3     parameter ADDR_WIDTH = 10,
4     parameter INIT_FILE = ""
5
6 ) (
7     input clk,
8     input w_en,
9     input [ADDR_WIDTH-1:0] w_addr,
10    input r_en,
11    input [ADDR_WIDTH-1:0] r_addr,
12    input [DATA_WIDTH-1:0] w_data,
13    output reg [DATA_WIDTH-1:0] r_data
14 );
15
16 (* ramstyle = "M10K" *) reg [DATA_WIDTH-1:0] mem [0:(1<<ADDR_WIDTH)-1];
17
18 initial begin
19     if (INIT_FILE != "") begin

```

```

20     $readmemh(INIT_FILE, mem);
21     end
22 end
23
24 always@(posedge clk)begin
25     if(w_en) mem[w_addr] <= w_data;
26 end
27
28 always@(posedge clk)begin
29     if(r_en) r_data <= mem[r_addr];
30 end
31
32 endmodule
33
34 module PACKED_ROM #(
35     parameter DATA_WIDTH = 8,
36     parameter ADDR_WIDTH = 10,
37     parameter LANES = 2,
38     parameter WORD_COUNT = (1 << ADDR_WIDTH),
39     parameter INIT_FILE = ""
40 ) (
41     input clk,
42     input r_en,
43     input [ADDR_WIDTH-1:0] r_addr,
44     output reg [LANES*DATA_WIDTH-1:0] r_data
45 );
46
47 (* ramstyle = "M10K" *) reg [LANES*DATA_WIDTH-1:0] mem [0:WORD_COUNT-1];
48
49 initial begin
50     if (INIT_FILE != "") begin
51         $readmemh(INIT_FILE, mem);
52     end
53 end
54
55 always@(posedge clk)begin
56     if(r_en) r_data <= mem[r_addr];
57 end
58
59 endmodule

```

Listing 13: HPS-to-FPGA memory-mapped register interface.

```

1  module hps_lenet_regs #(
2      parameter REG_COUNT = 10,
3      parameter PIXELS = 1024,
4      parameter PIX_BASE_WORD = 12
5  ) (
6      input wire          clk,
7      input wire          rst_n,
8      input wire          regs_valid,
9      input wire          accel_busy,
10     input wire          accel_done,
11     input wire [3:0]     predicted_digit,
12     input wire [27:0]    cycle_count,
13     input wire [REG_COUNT*32-1:0] regs_flat,
14     output reg          start_pulse,
15     output reg          clear_done_pulse,

```

```

16  output reg          pixel_wr_en,
17  output reg  [$clog2(PIXELS)-1:0] pixel_wr_addr,
18  output reg  [7:0]    pixel_wr_data,
19  output reg          pixel_rd_en,
20  output reg  [$clog2(PIXELS)-1:0] pixel_rd_addr,
21  input  wire  [7:0]    pixel_rd_data,
22  output reg          input_loaded,
23
24  input  wire  [11:0]    s_awid,
25  input  wire  [20:0]    s_awaddr,
26  input  wire          s_awvalid,
27  output wire          s_awready,
28  input  wire  [31:0]    s_wdata,
29  input  wire  [3:0]    s_wstrb,
30  input  wire          s_wlast,
31  input  wire          s_wvalid,
32  output wire          s_wready,
33  output reg  [11:0]    s_bid,
34  output reg  [1:0]    s_bresp,
35  output reg          s_bvalid,
36  input  wire          s_bready,
37
38  input  wire  [11:0]    s_arid,
39  input  wire  [20:0]    s_araddr,
40  input  wire          s_arvalid,
41  output wire          s_arready,
42  output reg  [11:0]    s_rid,
43  output reg  [31:0]    s_rdata,
44  output reg  [1:0]    s_rresp,
45  output reg          s_rlast,
46  output reg          s_rvalid,
47  input  wire          s_rready
48  );
49
50  localparam [1:0] AXI_RESP_OKAY = 2'b00;
51  localparam integer PIX_REG_COUNT = PIXELS;
52  localparam [1:0] READ_IDLE = 2'd0;
53  localparam [1:0] READ_PIXEL_WAIT_BRAM = 2'd1;
54
55  reg [11:0] awid_latched;
56  reg [20:0] awaddr_latched;
57  reg [31:0] wdata_latched;
58  reg      aw_seen;
59  reg      w_seen;
60  reg [1:0] read_state;
61  reg [11:0] pending_rid;
62
63  function automatic [31:0] read_register_word;//status reg
64  input [20:0] addr;
65  reg [11:0] word_index;
66  begin
67  word_index = addr[13:2];
68  if (word_index == 0) begin
69  read_register_word = {
70  28'd0,
71  input_loaded,
72  accel_done,
73  accel_busy,

```

```

74         regs_valid
75     };
76     end else if (word_index == 1) begin
77         read_register_word = {cycle_count, predicted_digit};
78     end else if ((word_index - 2) < REG_COUNT) begin
79         read_register_word = regs_flat[(word_index - 2) * 32 +: 32];
80     end else begin
81         read_register_word = 32'd0;
82     end
83 end
84 endfunction
85
86 function automatic is_pixel_window;
87     input [20:0] addr;
88     reg [11:0] word_index;
89     begin
90         word_index = addr[13:2];
91         is_pixel_window =
92             (word_index >= PIX_BASE_WORD) &&
93             ((word_index - PIX_BASE_WORD) < PIX_REG_COUNT);
94     end
95 endfunction
96
97 function automatic integer pixel_word_index;
98     input [20:0] addr;
99     reg [11:0] word_index;
100    begin
101        word_index = addr[13:2];
102        pixel_word_index = word_index - PIX_BASE_WORD;
103    end
104 endfunction
105
106 assign s_arready = !s_rvalid && (read_state == READ_IDLE);
107 assign s_awready = !s_bvalid && !aw_seen;
108 assign s_wready = !s_bvalid && !w_seen;
109
110 always @(posedge clk or negedge rst_n) begin
111     if (!rst_n) begin
112         s_rid      <= 12'd0;
113         s_rdata    <= 32'd0;
114         s_rresp    <= AXI_RESP_OKAY;
115         s_rlast    <= 1'b0;
116         s_rvalid   <= 1'b0;
117         pixel_rd_en <= 1'b0;
118         pixel_rd_addr <= {$clog2(PIXELS){1'b0}};
119         pending_rid <= 12'd0;
120         read_state <= READ_IDLE;
121     end else begin
122         pixel_rd_en <= 1'b0;
123
124         if (s_rvalid && s_rready) begin
125             s_rvalid <= 1'b0;
126             s_rlast  <= 1'b0;
127         end
128
129         case (read_state)
130             READ_IDLE: begin
131                 if (!s_rvalid && s_arvalid) begin

```

```

132     if (is_pixel_window(s_araddr)) begin
133         pending_rid <= s_arid; // pixel BRAM BRAM READ_PIXEL_WAIT_BRAM
134         pixel_rd_en <= 1'b1;
135         pixel_rd_addr <= pixel_word_index(s_araddr);
136         read_state <= READ_PIXEL_WAIT_BRAM;
137     end else begin
138         s_rid <= s_arid;
139         s_rdata <= read_register_word(s_araddr);
140         s_rresp <= AXI_RESP_OKAY;
141         s_rlast <= 1'b1;
142         s_rvalid <= 1'b1;
143     end
144 end
145 end
146
147 READ_PIXEL_WAIT_BRAM: begin
148     if (!s_rvalid) begin
149         s_rid <= pending_rid;
150         s_rdata <= {24'd0, pixel_rd_data};
151         s_rresp <= AXI_RESP_OKAY;
152         s_rlast <= 1'b1;
153         s_rvalid <= 1'b1;
154         read_state <= READ_IDLE;
155     end
156 end
157
158 default: begin
159     read_state <= READ_IDLE;
160 end
161 endcase
162 end
163 end
164
165 always @(posedge clk or negedge rst_n) begin
166     if (!rst_n) begin
167         start_pulse <= 1'b0;
168         clear_done_pulse <= 1'b0;
169         pixel_wr_en <= 1'b0;
170         pixel_wr_addr <= {$clog2(PIXELS){1'b0}};
171         pixel_wr_data <= 8'd0;
172         input_loaded <= 1'b0;
173         awid_latched <= 12'd0;
174         awaddr_latched <= 21'd0;
175         wdata_latched <= 32'd0;
176         aw_seen <= 1'b0;
177         w_seen <= 1'b0;
178         s_bid <= 12'd0;
179         s_bresp <= AXI_RESP_OKAY;
180         s_bvalid <= 1'b0;
181     end else begin
182         if (s_bvalid && s_bready) begin
183             s_bvalid <= 1'b0;
184         end
185
186         start_pulse <= 1'b0;
187         clear_done_pulse <= 1'b0;
188         pixel_wr_en <= 1'b0;
189

```

```

190     if (!aw_seen && s_awvalid) begin
191         awid_latched <= s_awid;
192         awaddr_latched <= s_awaddr;
193         aw_seen <= 1'b1;
194     end
195
196     if (!w_seen && s_wvalid) begin
197         wdata_latched <= s_wdata;
198         w_seen <= 1'b1;
199     end
200
201     if (!s_bvalid && aw_seen && w_seen) begin
202         if (is_pixel_window(awaddr_latched)) begin
203             pixel_wr_en <= 1'b1;
204             pixel_wr_addr <= pixel_word_index(awaddr_latched);
205             pixel_wr_data <= wdata_latched[7:0];
206             if (pixel_word_index(awaddr_latched) == (PIXELS - 1)) begin
207                 input_loaded <= 1'b1;
208             end
209         end else if (awaddr_latched[13:2] == 0) begin
210             if (wdata_latched[0]) begin
211                 start_pulse <= 1'b1;
212             end
213             if (wdata_latched[1]) begin
214                 clear_done_pulse <= 1'b1;
215             end
216             if (wdata_latched[2]) begin
217                 input_loaded <= 1'b0;
218             end
219         end
220
221         s_bid <= awid_latched;
222         s_bresp <= AXI_RESP_OKAY;
223         s_bvalid <= 1'b1;
224         aw_seen <= 1'b0;
225         w_seen <= 1'b0;
226     end
227 end
228 end
229
230 endmodule

```

Listing 14: Convolution and pooling address-generation logic.

```

1  module lenet_addr_calc (
2      input logic [3:0] out_ch,
3      input logic [4:0] out_y,
4      input logic [4:0] out_x,
5      input logic [3:0] pool_ch,
6      input logic [3:0] pool_y,
7      input logic [3:0] pool_x,
8      input logic [8:0] term_idx,
9      input logic [1:0] pool_read_idx,
10     output logic [clog2(lenet_int8_pkg::IMG_PIXELS)-1:0] conv1_src_addr,
11     output logic [clog2(lenet_int8_pkg::MAX_FM_COUNT)-1:0] conv1_dst_addr,
12     output logic [clog2(lenet_int8_pkg::MAX_FM_COUNT)-1:0] pool1_src_addr,
13     output logic [clog2(lenet_int8_pkg::MAX_FM_COUNT)-1:0] pool1_dst_addr,
14     output logic [clog2(lenet_int8_pkg::MAX_FM_COUNT)-1:0] conv2_src_addr,

```

```

15 output logic [clog2(lenet_int8_pkg::MAX_FM_COUNT)-1:0] conv2_dst_addr,
16 output logic [clog2(lenet_int8_pkg::MAX_FM_COUNT)-1:0] pool2_src_addr,
17 output logic [clog2(lenet_int8_pkg::MAX_FM_COUNT)-1:0] pool2_dst_addr
18 );
19 import lenet_int8_pkg::*;
20
21 logic [2:0] conv1_ky;
22 logic [2:0] conv1_kx;
23 logic      pool1_dy;
24 logic      pool1_dx;
25 logic [2:0] conv2_ic;
26 logic [4:0] conv2_tap;
27 logic [2:0] conv2_ky;
28 logic [2:0] conv2_kx;
29 logic      pool2_dy;
30 logic      pool2_dx;
31
32 function automatic logic [2:0] tap5_y(input logic [8:0] idx);
33     begin
34         if (idx < 5) begin
35             tap5_y = 3'd0;
36         end else if (idx < 10) begin
37             tap5_y = 3'd1;
38         end else if (idx < 15) begin
39             tap5_y = 3'd2;
40         end else if (idx < 20) begin
41             tap5_y = 3'd3;
42         end else begin
43             tap5_y = 3'd4;
44         end
45     end
46 endfunction
47
48 function automatic logic [2:0] tap5_x(input logic [8:0] idx);
49     begin
50         case (idx)
51             0, 5, 10, 15, 20: tap5_x = 3'd0;
52             1, 6, 11, 16, 21: tap5_x = 3'd1;
53             2, 7, 12, 17, 22: tap5_x = 3'd2;
54             3, 8, 13, 18, 23: tap5_x = 3'd3;
55             default:         tap5_x = 3'd4;
56         endcase
57     end
58 endfunction
59
60 function automatic logic [2:0] conv2_in_ch(input logic [8:0] idx);
61     begin
62         if (idx < 25) begin
63             conv2_in_ch = 3'd0;
64         end else if (idx < 50) begin
65             conv2_in_ch = 3'd1;
66         end else if (idx < 75) begin
67             conv2_in_ch = 3'd2;
68         end else if (idx < 100) begin
69             conv2_in_ch = 3'd3;
70         end else if (idx < 125) begin
71             conv2_in_ch = 3'd4;
72         end else begin

```

```

73     conv2_in_ch = 3'd5;
74     end
75     end
76 endfunction
77
78 function automatic logic [4:0] conv2_tap_idx(input logic [8:0] idx);
79     begin
80         if (idx < 25) begin
81             conv2_tap_idx = idx[4:0];
82         end else if (idx < 50) begin
83             conv2_tap_idx = idx[4:0] - 5'd25;
84         end else if (idx < 75) begin
85             conv2_tap_idx = idx[4:0] - 5'd18;
86         end else if (idx < 100) begin
87             conv2_tap_idx = idx[4:0] - 5'd11;
88         end else if (idx < 125) begin
89             conv2_tap_idx = idx[4:0] - 5'd4;
90         end else begin
91             conv2_tap_idx = idx[4:0] - 5'd29;
92         end
93     end
94 endfunction
95
96 always_comb begin
97     conv1_ky = tap5_y(term_idx);
98     conv1_kx = tap5_x(term_idx);
99     conv1_src_addr = ((out_y + conv1_ky) * IMG_W) + (out_x + conv1_kx);
100    conv1_dst_addr = (out_ch * CONV1_CONV_W * CONV1_CONV_W) + (out_y * CONV1_CONV_W) + out_x;
101
102    pool1_dy = (pool_read_idx >= 2);
103    pool1_dx = pool_read_idx[0];
104    pool1_src_addr = (pool_ch * CONV1_CONV_W * CONV1_CONV_W) + ((pool_y * 2 + pool1_dy) *
CONV1_CONV_W) + (pool_x * 2 + pool1_dx);
105    pool1_dst_addr = (pool_ch * CONV1_POOL_W * CONV1_POOL_W) + (pool_y * CONV1_POOL_W) + pool_x;
106
107    conv2_ic = conv2_in_ch(term_idx);
108    conv2_tap = conv2_tap_idx(term_idx);
109    conv2_ky = tap5_y(conv2_tap);
110    conv2_kx = tap5_x(conv2_tap);
111    conv2_src_addr = (conv2_ic * CONV1_POOL_W * CONV1_POOL_W) + ((out_y + conv2_ky) * CONV1_POOL_W)
+ (out_x + conv2_kx);
112    conv2_dst_addr = (out_ch * CONV2_CONV_W * CONV2_CONV_W) + (out_y * CONV2_CONV_W) + out_x;
113
114    pool2_dy = (pool_read_idx >= 2);
115    pool2_dx = pool_read_idx[0];
116    pool2_src_addr = (pool_ch * CONV2_CONV_W * CONV2_CONV_W) + ((pool_y * 2 + pool2_dy) *
CONV2_CONV_W) + (pool_x * 2 + pool2_dx);
117    pool2_dst_addr = (pool_ch * CONV2_POOL_W * CONV2_POOL_W) + (pool_y * CONV2_POOL_W) + pool_x;
118 end
119 endmodule

```

Listing 15: Lane-parallel multiply-accumulate array.

```

1 module lenet_mac_array (
2   input  logic [lenet_int8_pkg::MAC_LANES * lenet_int8_pkg::ACT_W-1:0] activations_flat,
3   input  logic signed [lenet_int8_pkg::MAC_LANES * lenet_int8_pkg::WGT_W-1:0] weights_flat,
4   input  logic [lenet_int8_pkg::MAC_LANES-1:0] valid_mask,
5   output logic signed [lenet_int8_pkg::ACC_W-1:0] partial_sum
6 );
7   import lenet_int8_pkg::*;
8
9   integer lane;
10  logic signed [ACT_W:0] act_value;
11  logic signed [WGT_W-1:0] weight_value;
12
13  always_comb begin
14    partial_sum = '0;
15    for (lane = 0; lane < MAC_LANES; lane = lane + 1) begin
16      act_value = $signed({1'b0, activations_flat[lane * ACT_W +: ACT_W]});
17      weight_value = weights_flat[lane * WGT_W +: WGT_W];
18      if (valid_mask[lane]) begin
19        partial_sum = partial_sum + (act_value * weight_value);
20      end
21    end
22  end
23 endmodule

```

Listing 16: Shift-based quantization and ReLU clamp logic.

```

1 module lenet_quantize (
2   input  logic signed [lenet_int8_pkg::ACC_W-1:0] value_in,
3   input  logic [lenet_int8_pkg::SHIFT_W-1:0] shift_right,
4   output logic [lenet_int8_pkg::ACT_W-1:0] value_out
5 );
6   import lenet_int8_pkg::*;
7
8   logic signed [ACC_W-1:0] rounded_value;
9   logic signed [ACC_W-1:0] scaled_value;
10  logic signed [ACC_W-1:0] round_offset;
11
12  always_comb begin
13    value_out = '0;
14    rounded_value = value_in;
15    scaled_value = value_in;
16    round_offset = '0;
17
18    if (value_in <= 0) begin
19      value_out = '0;
20    end else begin
21      if (shift_right != 0) begin
22        round_offset = $signed(32'd1 <<< (shift_right - 1'b1));
23        rounded_value = value_in + round_offset;
24        scaled_value = rounded_value >>> shift_right;
25      end
26
27      if (scaled_value > ACT_MAX) begin
28        value_out = ACT_W'(ACT_MAX);
29      end else begin
30        value_out = scaled_value[ACT_W-1:0];

```

```

31     end
32     end
33     end
34 endmodule

```

Listing 17: Sequential argmax module for final digit prediction.

```

1 module lenet_argmax (
2   input logic signed [lenet_int8_pkg::OUT_COUNT * lenet_int8_pkg::ACC_W-1:0] logits_flat,
3   output logic [3:0] digit
4 );
5   import lenet_int8_pkg::*;
6
7   integer idx;
8   integer best_idx;
9   logic signed [ACC_W-1:0] best_score;
10  logic signed [ACC_W-1:0] score;
11
12  always_comb begin
13    best_score = $signed(logits_flat[0 +: ACC_W]);
14    best_idx = 0;
15    for (idx = 1; idx < OUT_COUNT; idx = idx + 1) begin
16      score = $signed(logits_flat[idx * ACC_W +: ACC_W]);
17      if (score > best_score) begin
18        best_score = score;
19        best_idx = idx;
20      end
21    end
22    digit = best_idx[3:0];
23  end
24 endmodule

```

Listing 18: Top-level INT8 LeNet accelerator core.

```

1 module lenet_int8_top #(
2   parameter CONV1_PARAM_FILE = "D:/Coursefile/ee4840_embedded/embedded_lab/parameter/
3     weights_bias_422/conv1_pack3.hex",
4   parameter CONV2_PARAM_FILE = "D:/Coursefile/ee4840_embedded/embedded_lab/parameter/
5     weights_bias_422/conv2_pack3.hex",
6   parameter FC1_PARAM_FILE   = "D:/Coursefile/ee4840_embedded/embedded_lab/parameter/
7     weights_bias_422/fc1_pack3.hex",
8   parameter FC2_PARAM_FILE   = "D:/Coursefile/ee4840_embedded/embedded_lab/parameter/
9     weights_bias_422/fc2_pack3.hex",
10  parameter OUT_PARAM_FILE    = "D:/Coursefile/ee4840_embedded/embedded_lab/parameter/
11    weights_bias_422/fc3_pack3.hex"
12 ) (
13   input  logic clk,
14   input  logic rst_n,
15   input  logic start,
16   input  logic pixel_wr_en,
17   input  logic [$clog2(lenet_int8_pkg::IMG_PIXELS)-1:0] pixel_wr_addr,
18   input  logic [lenet_int8_pkg::PIX_W-1:0] pixel_wr_data,
19   input  logic pixel_dbg_rd_en,
20   input  logic [$clog2(lenet_int8_pkg::IMG_PIXELS)-1:0] pixel_dbg_rd_addr,
21   output logic [lenet_int8_pkg::PIX_W-1:0] pixel_dbg_rd_data,
22   output logic busy,
23   output logic done,
24   output logic [3:0] predicted_digit,

```

```

20  output logic signed [lenet_int8_pkg::OUT_COUNT * lenet_int8_pkg::ACC_W-1:0] logits_flat,
21  output logic [27:0] cycle_count
22  );
23  import lenet_int8_pkg::*;
24
25  localparam int IMG_ADDR_W = $clog2(IMG_PIXELS);
26  localparam int FM_ADDR_W = $clog2(MAX_FM_COUNT);
27  localparam int CONV1_PARAM_ADDR_W = $clog2(CONV1_PARAM_COUNT);
28  localparam int CONV2_PARAM_ADDR_W = $clog2(CONV2_PARAM_COUNT);
29  localparam int FC1_PARAM_ADDR_W = $clog2(FC1_PARAM_COUNT);
30  localparam int FC2_PARAM_ADDR_W = $clog2(FC2_PARAM_COUNT);
31  localparam int OUT_PARAM_ADDR_W = $clog2(OUT_PARAM_COUNT);
32  localparam int CONV1_WORDS_PER_ROW = ((CONV1_IN_CH * CONV1_K * CONV1_K) + MAC_LANES - 1) /
    MAC_LANES;
33  localparam int CONV2_WORDS_PER_ROW = ((CONV2_IN_CH * CONV2_K * CONV2_K) + MAC_LANES - 1) /
    MAC_LANES;
34  localparam int FC1_WORDS_PER_ROW = (FC1_IN_COUNT + MAC_LANES - 1) / MAC_LANES;
35  localparam int FC2_WORDS_PER_ROW = (FC2_IN_COUNT + MAC_LANES - 1) / MAC_LANES;
36  localparam int OUT_WORDS_PER_ROW = (OUT_IN_COUNT + MAC_LANES - 1) / MAC_LANES;
37  localparam int CONV1_WEIGHT_WORD_COUNT = CONV1_OUT_CH * CONV1_WORDS_PER_ROW;
38  localparam int CONV2_WEIGHT_WORD_COUNT = CONV2_OUT_CH * CONV2_WORDS_PER_ROW;
39  localparam int FC1_WEIGHT_WORD_COUNT = FC1_OUT_COUNT * FC1_WORDS_PER_ROW;
40  localparam int FC2_WEIGHT_WORD_COUNT = FC2_OUT_COUNT * FC2_WORDS_PER_ROW;
41  localparam int OUT_WEIGHT_WORD_COUNT = OUT_COUNT * OUT_WORDS_PER_ROW;
42  localparam int CONV1_PARAM_WORD_COUNT = CONV1_WEIGHT_WORD_COUNT + CONV1_BIAS_COUNT;
43  localparam int CONV2_PARAM_WORD_COUNT = CONV2_WEIGHT_WORD_COUNT + CONV2_BIAS_COUNT;
44  localparam int FC1_PARAM_WORD_COUNT = FC1_WEIGHT_WORD_COUNT + FC1_BIAS_COUNT;
45  localparam int FC2_PARAM_WORD_COUNT = FC2_WEIGHT_WORD_COUNT + FC2_BIAS_COUNT;
46  localparam int OUT_PARAM_WORD_COUNT = OUT_WEIGHT_WORD_COUNT + OUT_BIAS_COUNT;
47  localparam int CONV1_PARAM_WORD_ADDR_W = $clog2(CONV1_PARAM_WORD_COUNT);
48  localparam int CONV2_PARAM_WORD_ADDR_W = $clog2(CONV2_PARAM_WORD_COUNT);
49  localparam int FC1_PARAM_WORD_ADDR_W = $clog2(FC1_PARAM_WORD_COUNT);
50  localparam int FC2_PARAM_WORD_ADDR_W = $clog2(FC2_PARAM_WORD_COUNT);
51  localparam int OUT_PARAM_WORD_ADDR_W = $clog2(OUT_PARAM_WORD_COUNT);
52
53  typedef enum logic [5:0] {
54      ST_IDLE,
55      ST_CONV1_INIT,
56      ST_CONV1_READ,
57      ST_CONV1_ACCUM,
58      ST_CONV1_ADD,
59      ST_CONV1_WRITE,
60      ST_POOL1_INIT,
61      ST_POOL1_READ,
62      ST_POOL1_ACCUM,
63      ST_POOL1_WRITE,
64      ST_CONV2_INIT,
65      ST_CONV2_READ,
66      ST_CONV2_ACCUM,
67      ST_CONV2_ADD,
68      ST_CONV2_WRITE,
69      ST_POOL2_INIT,
70      ST_POOL2_READ,
71      ST_POOL2_ACCUM,
72      ST_POOL2_WRITE,
73      ST_FC1_INIT,
74      ST_FC1_READ,
75      ST_FC1_ACCUM,

```

```

76     ST_FC1_ADD,
77     ST_FC1_WRITE,
78     ST_FC2_INIT,
79     ST_FC2_READ,
80     ST_FC2_ACCUM,
81     ST_FC2_ADD,
82     ST_FC2_WRITE,
83     ST_FC3_INIT,
84     ST_FC3_READ,
85     ST_FC3_ACCUM,
86     ST_FC3_ADD,
87     ST_FC3_WRITE,
88     ST_ARGMAX_INIT,
89     ST_ARGMAX_SCAN,
90     ST_DONE
91 } state_t;
92
93 state_t state;
94
95 logic [MAC_LANES-1:0] img_r_en;
96 logic [IMG_ADDR_W-1:0] img_r_addr [0:MAC_LANES-1];
97 logic [PIX_W-1:0] img_r_data [0:MAC_LANES-1];
98
99 logic [MAC_LANES-1:0] fm_a_r_en;
100 logic [FM_ADDR_W-1:0] fm_a_r_addr [0:MAC_LANES-1];
101 logic [ACT_W-1:0] fm_a_r_data [0:MAC_LANES-1];
102 logic fm_a_w_en;
103 logic [FM_ADDR_W-1:0] fm_a_w_addr;
104 logic [ACT_W-1:0] fm_a_w_data;
105
106 logic [MAC_LANES-1:0] fm_b_r_en;
107 logic [FM_ADDR_W-1:0] fm_b_r_addr [0:MAC_LANES-1];
108 logic [ACT_W-1:0] fm_b_r_data [0:MAC_LANES-1];
109 logic fm_b_w_en;
110 logic [FM_ADDR_W-1:0] fm_b_w_addr;
111 logic [ACT_W-1:0] fm_b_w_data;
112
113 logic signed [ACC_W-1:0] logits_mem [0:OUT_COUNT-1];
114
115 logic [MAC_LANES-1:0] conv1_param_r_en;
116 logic [CONV1_PARAM_WORD_ADDR_W-1:0] conv1_param_r_addr [0:MAC_LANES-1];
117 logic [WGT_W-1:0] conv1_param_r_data [0:MAC_LANES-1];
118 logic [MAC_LANES-1:0] conv2_param_r_en;
119 logic [CONV2_PARAM_WORD_ADDR_W-1:0] conv2_param_r_addr [0:MAC_LANES-1];
120 logic [WGT_W-1:0] conv2_param_r_data [0:MAC_LANES-1];
121 logic [MAC_LANES-1:0] fc1_param_r_en;
122 logic [FC1_PARAM_WORD_ADDR_W-1:0] fc1_param_r_addr [0:MAC_LANES-1];
123 logic [WGT_W-1:0] fc1_param_r_data [0:MAC_LANES-1];
124 logic [MAC_LANES-1:0] fc2_param_r_en;
125 logic [FC2_PARAM_WORD_ADDR_W-1:0] fc2_param_r_addr [0:MAC_LANES-1];
126 logic [WGT_W-1:0] fc2_param_r_data [0:MAC_LANES-1];
127 logic [MAC_LANES-1:0] out_param_r_en;
128 logic [OUT_PARAM_WORD_ADDR_W-1:0] out_param_r_addr [0:MAC_LANES-1];
129 logic [WGT_W-1:0] out_param_r_data [0:MAC_LANES-1];
130 logic [MAC_LANES * WGT_W-1:0] conv1_param_r_data_flat;
131 logic [MAC_LANES * WGT_W-1:0] conv2_param_r_data_flat;
132 logic [MAC_LANES * WGT_W-1:0] fc1_param_r_data_flat;
133 logic [MAC_LANES * WGT_W-1:0] fc2_param_r_data_flat;

```

```

134 logic [MAC_LANES * WGT_W-1:0] out_param_r_data_flat;
135
136 logic [MAC_LANES * ACT_W-1:0] mac_activation;
137 logic signed [MAC_LANES * WGT_W-1:0] mac_weight;
138 logic [MAC_LANES-1:0] mac_valid;
139 logic signed [ACC_W-1:0] mac_partial_sum;
140 logic signed [ACC_W-1:0] mac_partial_sum_r;
141 logic signed [ACC_W-1:0] acc_reg;
142 logic signed [BIAS_W-1:0] current_bias;
143 logic [SHIFT_W-1:0] current_shift;
144 logic signed [ACC_W-1:0] acc_with_bias;
145 logic [ACT_W-1:0] quantized_value;
146 logic [ACT_W-1:0] pool_max_reg;
147 logic [18:0] cycle_counter;
148
149 integer out_ch;
150 integer out_y;
151 integer out_x;
152 integer pool_ch;
153 integer pool_y;
154 integer pool_x;
155 integer neuron_idx;
156 logic [8:0] term_idx;
157 logic [8:0] term_word_idx;
158 integer lane_idx;
159 integer pool_read_idx;
160 integer flat_idx;
161 integer reset_idx;
162 logic [3:0] argmax_index;
163 logic [3:0] argmax_best_digit;
164 logic signed [ACC_W-1:0] argmax_best_score;
165 logic [8:0] term_idx_lane [0:MAC_LANES-1];
166 logic [MAC_LANES-1:0] term_valid;
167 logic [IMG_ADDR_W-1:0] addr_conv1_src [0:MAC_LANES-1];
168 logic [FM_ADDR_W-1:0] addr_conv1_dst [0:MAC_LANES-1];
169 logic [FM_ADDR_W-1:0] addr_pool1_src [0:MAC_LANES-1];
170 logic [FM_ADDR_W-1:0] addr_pool1_dst [0:MAC_LANES-1];
171 logic [FM_ADDR_W-1:0] addr_conv2_src [0:MAC_LANES-1];
172 logic [FM_ADDR_W-1:0] addr_conv2_dst [0:MAC_LANES-1];
173 logic [FM_ADDR_W-1:0] addr_pool2_src [0:MAC_LANES-1];
174 logic [FM_ADDR_W-1:0] addr_pool2_dst [0:MAC_LANES-1];
175
176 genvar mem_lane;
177 generate
178   for (mem_lane = 0; mem_lane < MAC_LANES; mem_lane = mem_lane + 1) begin : g_lane_mem
179     BRAM #(
180       .DATA_WIDTH(PIX_W),
181       .ADDR_WIDTH(IMG_ADDR_W)
182     ) u_image_mem (
183       .clk(clk),
184       .w_en(pixel_wr_en),
185       .w_addr(pixel_wr_addr),
186       .r_en((mem_lane == 0) ? (pixel_dbg_rd_en | img_r_en[mem_lane]) : img_r_en[mem_lane]),
187       .r_addr((mem_lane == 0 && pixel_dbg_rd_en) ? pixel_dbg_rd_addr : img_r_addr[mem_lane]),
188       .w_data(pixel_wr_data),
189       .r_data(img_r_data[mem_lane])
190     );
191

```

```

192     BRAM #(
193         .DATA_WIDTH(ACT_W),
194         .ADDR_WIDTH(FM_ADDR_W)
195     ) u_fm_a (
196         .clk(clk),
197         .w_en(fm_a_w_en),
198         .w_addr(fm_a_w_addr),
199         .r_en(fm_a_r_en[mem_lane]),
200         .r_addr(fm_a_r_addr[mem_lane]),
201         .w_data(fm_a_w_data),
202         .r_data(fm_a_r_data[mem_lane])
203     );
204
205     BRAM #(
206         .DATA_WIDTH(ACT_W),
207         .ADDR_WIDTH(FM_ADDR_W)
208     ) u_fm_b (
209         .clk(clk),
210         .w_en(fm_b_w_en),
211         .w_addr(fm_b_w_addr),
212         .r_en(fm_b_r_en[mem_lane]),
213         .r_addr(fm_b_r_addr[mem_lane]),
214         .w_data(fm_b_w_data),
215         .r_data(fm_b_r_data[mem_lane])
216     );
217     end
218     endgenerate
219
220     PACKED_ROM #(
221         .DATA_WIDTH(WGT_W),
222         .ADDR_WIDTH(CONV1_PARAM_WORD_ADDR_W),
223         .LANES(MAC_LANES),
224         .WORD_COUNT(CONV1_PARAM_WORD_COUNT),
225         .INIT_FILE(CONV1_PARAM_FILE)
226     ) u_conv1_params (
227         .clk(clk),
228         .r_en(|conv1_param_r_en),
229         .r_addr(conv1_param_r_addr[0]),
230         .r_data(conv1_param_r_data_flat)
231     );
232
233     PACKED_ROM #(
234         .DATA_WIDTH(WGT_W),
235         .ADDR_WIDTH(CONV2_PARAM_WORD_ADDR_W),
236         .LANES(MAC_LANES),
237         .WORD_COUNT(CONV2_PARAM_WORD_COUNT),
238         .INIT_FILE(CONV2_PARAM_FILE)
239     ) u_conv2_params (
240         .clk(clk),
241         .r_en(|conv2_param_r_en),
242         .r_addr(conv2_param_r_addr[0]),
243         .r_data(conv2_param_r_data_flat)
244     );
245
246     PACKED_ROM #(
247         .DATA_WIDTH(WGT_W),
248         .ADDR_WIDTH(FC1_PARAM_WORD_ADDR_W),
249         .LANES(MAC_LANES),

```

```

250     .WORD_COUNT(FC1_PARAM_WORD_COUNT),
251     .INIT_FILE(FC1_PARAM_FILE)
252 ) u_fc1_params (
253     .clk(clk),
254     .r_en(|fc1_param_r_en),
255     .r_addr(fc1_param_r_addr[0]),
256     .r_data(fc1_param_r_data_flat)
257 );
258
259 PACKED_ROM #(
260     .DATA_WIDTH(WGT_W),
261     .ADDR_WIDTH(FC2_PARAM_WORD_ADDR_W),
262     .LANES(MAC_LANES),
263     .WORD_COUNT(FC2_PARAM_WORD_COUNT),
264     .INIT_FILE(FC2_PARAM_FILE)
265 ) u_fc2_params (
266     .clk(clk),
267     .r_en(|fc2_param_r_en),
268     .r_addr(fc2_param_r_addr[0]),
269     .r_data(fc2_param_r_data_flat)
270 );
271
272 PACKED_ROM #(
273     .DATA_WIDTH(WGT_W),
274     .ADDR_WIDTH(OUT_PARAM_WORD_ADDR_W),
275     .LANES(MAC_LANES),
276     .WORD_COUNT(OUT_PARAM_WORD_COUNT),
277     .INIT_FILE(OUT_PARAM_FILE)
278 ) u_out_params (
279     .clk(clk),
280     .r_en(|out_param_r_en),
281     .r_addr(out_param_r_addr[0]),
282     .r_data(out_param_r_data_flat)
283 );
284
285 genvar param_lane;
286 generate
287     for (param_lane = 0; param_lane < MAC_LANES; param_lane = param_lane + 1) begin : g_param_unpack
288         assign conv1_param_r_data[param_lane] = conv1_param_r_data_flat[param_lane * WGT_W +: WGT_W];
289         assign conv2_param_r_data[param_lane] = conv2_param_r_data_flat[param_lane * WGT_W +: WGT_W];
290         assign fc1_param_r_data[param_lane] = fc1_param_r_data_flat[param_lane * WGT_W +: WGT_W];
291         assign fc2_param_r_data[param_lane] = fc2_param_r_data_flat[param_lane * WGT_W +: WGT_W];
292         assign out_param_r_data[param_lane] = out_param_r_data_flat[param_lane * WGT_W +: WGT_W];
293     end
294 endgenerate
295
296 lenet_mac_array u_mac (
297     .activations_flat(mac_activation),
298     .weights_flat(mac_weight),
299     .valid_mask(mac_valid),
300     .partial_sum(mac_partial_sum)
301 );
302
303 lenet_quantize u_quantize (
304     .value_in(acc_with_bias),
305     .shift_right(current_shift),
306     .value_out(quantized_value)
307 );

```

```

308
309 generate
310   for (mem_lane = 0; mem_lane < MAC_LANES; mem_lane = mem_lane + 1) begin : g_addr_calc
311     lenet_addr_calc u_addr_calc (
312       .out_ch(out_ch[3:0]),
313       .out_y(out_y[4:0]),
314       .out_x(out_x[4:0]),
315       .pool_ch(pool_ch[3:0]),
316       .pool_y(pool_y[3:0]),
317       .pool_x(pool_x[3:0]),
318       .term_idx(term_idx_lane[mem_lane]),
319       .pool_read_idx(pool_read_idx[1:0]),
320       .conv1_src_addr(addr_conv1_src[mem_lane]),
321       .conv1_dst_addr(addr_conv1_dst[mem_lane]),
322       .pool1_src_addr(addr_pool1_src[mem_lane]),
323       .pool1_dst_addr(addr_pool1_dst[mem_lane]),
324       .conv2_src_addr(addr_conv2_src[mem_lane]),
325       .conv2_dst_addr(addr_conv2_dst[mem_lane]),
326       .pool2_src_addr(addr_pool2_src[mem_lane]),
327       .pool2_dst_addr(addr_pool2_dst[mem_lane])
328     );
329   end
330 endgenerate
331
332 always_comb begin
333   pixel_dbg_rd_data = img_r_data[0];
334
335   img_r_en = '0;
336   fm_a_r_en = '0;
337   fm_a_w_en = 1'b0;
338   fm_a_w_addr = '0;
339   fm_a_w_data = '0;
340   fm_b_r_en = '0;
341   fm_b_w_en = 1'b0;
342   fm_b_w_addr = '0;
343   fm_b_w_data = '0;
344   conv1_param_r_en = '0;
345   conv2_param_r_en = '0;
346   fc1_param_r_en = '0;
347   fc2_param_r_en = '0;
348   out_param_r_en = '0;
349   mac_activation = '0;
350   mac_weight = '0;
351   mac_valid = '0;
352   current_bias = '0;
353   current_shift = '0;
354   term_valid = '0;
355
356   for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
357     img_r_addr[lane_idx] = '0;
358     fm_a_r_addr[lane_idx] = '0;
359     fm_b_r_addr[lane_idx] = '0;
360     conv1_param_r_addr[lane_idx] = '0;
361     conv2_param_r_addr[lane_idx] = '0;
362     fc1_param_r_addr[lane_idx] = '0;
363     fc2_param_r_addr[lane_idx] = '0;
364     out_param_r_addr[lane_idx] = '0;
365     term_idx_lane[lane_idx] = term_idx + lane_idx;

```

```

366     end
367
368     logits_flat = '0;
369     for (flat_idx = 0; flat_idx < OUT_COUNT; flat_idx = flat_idx + 1) begin
370         logits_flat[flat_idx * ACC_W +: ACC_W] = logits_mem[flat_idx];
371     end
372
373     case (state)
374     ST_CONV1_READ: begin
375         for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
376             term_valid[lane_idx] = ((term_idx + lane_idx) < (CONV1_K * CONV1_K));
377             img_r_en[lane_idx] = term_valid[lane_idx];
378             img_r_addr[lane_idx] = addr_conv1_src[lane_idx];
379             conv1_param_r_en[lane_idx] = term_valid[lane_idx];
380         end
381         conv1_param_r_addr[0] = (out_ch * CONV1_WORDS_PER_ROW) + term_word_idx;
382     end
383
384     ST_CONV1_ACCUM: begin
385         for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
386             mac_activation[lane_idx * ACT_W +: ACT_W] = img_r_data[lane_idx];
387             mac_weight[lane_idx * WGT_W +: WGT_W] = conv1_param_r_data[lane_idx];
388             mac_valid[lane_idx] = ((term_idx + lane_idx) < (CONV1_K * CONV1_K));
389         end
390         current_shift = CONV1_SHIFT;
391         if ((term_idx + MAC_LANES) >= (CONV1_K * CONV1_K)) begin
392             conv1_param_r_en[0] = 1'b1;
393             conv1_param_r_addr[0] = CONV1_WEIGHT_WORD_COUNT + out_ch;
394         end
395     end
396
397     ST_CONV1_WRITE: begin
398         current_bias = conv1_param_r_data[0];
399         current_shift = CONV1_SHIFT;
400         fm_a_w_en = 1'b1;
401         fm_a_w_addr = addr_conv1_dst[0];
402         fm_a_w_data = quantized_value;
403     end
404
405     ST_POOL1_READ: begin
406         fm_a_r_en[0] = 1'b1;
407         fm_a_r_addr[0] = addr_pool1_src[0];
408     end
409
410     ST_POOL1_WRITE: begin
411         fm_b_w_en = 1'b1;
412         fm_b_w_addr = addr_pool1_dst[0];
413         fm_b_w_data = pool_max_reg;
414     end
415
416     ST_CONV2_READ: begin
417         for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
418             term_valid[lane_idx] = ((term_idx + lane_idx) < (CONV2_IN_CH * CONV2_K * CONV2_K));
419             fm_b_r_en[lane_idx] = term_valid[lane_idx];
420             fm_b_r_addr[lane_idx] = addr_conv2_src[lane_idx];
421             conv2_param_r_en[lane_idx] = term_valid[lane_idx];
422         end
423         conv2_param_r_addr[0] = (out_ch * CONV2_WORDS_PER_ROW) + term_word_idx;

```

```

424     end
425
426 ST_CONV2_ACCUM: begin
427     for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
428         mac_activation[lane_idx * ACT_W +: ACT_W] = fm_b_r_data[lane_idx];
429         mac_weight[lane_idx * WGT_W +: WGT_W] = conv2_param_r_data[lane_idx];
430         mac_valid[lane_idx] = ((term_idx + lane_idx) < (CONV2_IN_CH * CONV2_K * CONV2_K));
431     end
432     current_shift = CONV2_SHIFT;
433     if ((term_idx + MAC_LANES) >= (CONV2_IN_CH * CONV2_K * CONV2_K)) begin
434         conv2_param_r_en[0] = 1'b1;
435         conv2_param_r_addr[0] = CONV2_WEIGHT_WORD_COUNT + out_ch;
436     end
437 end
438
439 ST_CONV2_WRITE: begin
440     current_bias = conv2_param_r_data[0];
441     current_shift = CONV2_SHIFT;
442     fm_a_w_en = 1'b1;
443     fm_a_w_addr = addr_conv2_dst[0];
444     fm_a_w_data = quantized_value;
445 end
446
447 ST_POOL2_READ: begin
448     fm_a_r_en[0] = 1'b1;
449     fm_a_r_addr[0] = addr_pool2_src[0];
450 end
451
452 ST_POOL2_WRITE: begin
453     fm_b_w_en = 1'b1;
454     fm_b_w_addr = addr_pool2_dst[0];
455     fm_b_w_data = pool_max_reg;
456 end
457
458 ST_FC1_READ: begin
459     for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
460         term_valid[lane_idx] = ((term_idx + lane_idx) < FC1_IN_COUNT);
461         fm_b_r_en[lane_idx] = term_valid[lane_idx];
462         fm_b_r_addr[lane_idx] = term_idx + lane_idx;
463         fc1_param_r_en[lane_idx] = term_valid[lane_idx];
464     end
465     fc1_param_r_addr[0] = (neuron_idx * FC1_WORDS_PER_ROW) + term_word_idx;
466 end
467
468 ST_FC1_ACCUM: begin
469     for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
470         mac_activation[lane_idx * ACT_W +: ACT_W] = fm_b_r_data[lane_idx];
471         mac_weight[lane_idx * WGT_W +: WGT_W] = fc1_param_r_data[lane_idx];
472         mac_valid[lane_idx] = ((term_idx + lane_idx) < FC1_IN_COUNT);
473     end
474     current_shift = FC1_SHIFT;
475     if ((term_idx + MAC_LANES) >= FC1_IN_COUNT) begin
476         fc1_param_r_en[0] = 1'b1;
477         fc1_param_r_addr[0] = FC1_WEIGHT_WORD_COUNT + neuron_idx;
478     end
479 end
480
481 ST_FC1_WRITE: begin

```

```

482     current_bias = fc1_param_r_data[0];
483     current_shift = FC1_SHIFT;
484     fm_a_w_en = 1'b1;
485     fm_a_w_addr = neuron_idx;
486     fm_a_w_data = quantized_value;
487 end
488
489 ST_FC2_READ: begin
490     for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
491         term_valid[lane_idx] = ((term_idx + lane_idx) < FC2_IN_COUNT);
492         fm_a_r_en[lane_idx] = term_valid[lane_idx];
493         fm_a_r_addr[lane_idx] = term_idx + lane_idx;
494         fc2_param_r_en[lane_idx] = term_valid[lane_idx];
495     end
496     fc2_param_r_addr[0] = (neuron_idx * FC2_WORDS_PER_ROW) + term_word_idx;
497 end
498
499 ST_FC2_ACCUM: begin
500     for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
501         mac_activation[lane_idx * ACT_W +: ACT_W] = fm_a_r_data[lane_idx];
502         mac_weight[lane_idx * WGT_W +: WGT_W] = fc2_param_r_data[lane_idx];
503         mac_valid[lane_idx] = ((term_idx + lane_idx) < FC2_IN_COUNT);
504     end
505     current_shift = FC2_SHIFT;
506     if ((term_idx + MAC_LANES) >= FC2_IN_COUNT) begin
507         fc2_param_r_en[0] = 1'b1;
508         fc2_param_r_addr[0] = FC2_WEIGHT_WORD_COUNT + neuron_idx;
509     end
510 end
511
512 ST_FC2_WRITE: begin
513     current_bias = fc2_param_r_data[0];
514     current_shift = FC2_SHIFT;
515     fm_b_w_en = 1'b1;
516     fm_b_w_addr = neuron_idx;
517     fm_b_w_data = quantized_value;
518 end
519
520 ST_FC3_READ: begin
521     for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
522         term_valid[lane_idx] = ((term_idx + lane_idx) < OUT_IN_COUNT);
523         fm_b_r_en[lane_idx] = term_valid[lane_idx];
524         fm_b_r_addr[lane_idx] = term_idx + lane_idx;
525         out_param_r_en[lane_idx] = term_valid[lane_idx];
526     end
527     out_param_r_addr[0] = (neuron_idx * OUT_WORDS_PER_ROW) + term_word_idx;
528 end
529
530 ST_FC3_ACCUM: begin
531     for (lane_idx = 0; lane_idx < MAC_LANES; lane_idx = lane_idx + 1) begin
532         mac_activation[lane_idx * ACT_W +: ACT_W] = fm_b_r_data[lane_idx];
533         mac_weight[lane_idx * WGT_W +: WGT_W] = out_param_r_data[lane_idx];
534         mac_valid[lane_idx] = ((term_idx + lane_idx) < OUT_IN_COUNT);
535     end
536     if ((term_idx + MAC_LANES) >= OUT_IN_COUNT) begin
537         out_param_r_en[0] = 1'b1;
538         out_param_r_addr[0] = OUT_WEIGHT_WORD_COUNT + neuron_idx;
539     end

```

```

540     end
541
542     ST_FC3_WRITE: begin
543         current_bias = out_param_r_data[0];
544     end
545
546     default: begin
547     end
548 endcase
549 end
550
551 assign acc_with_bias = acc_reg + {{(ACC_W-BIAS_W){current_bias[BIAS_W-1]}}, current_bias};
552
553 always_ff @(posedge clk or negedge rst_n) begin
554     if (!rst_n) begin
555         state <= ST_IDLE;
556         acc_reg <= '0;
557         predicted_digit <= '0;
558         argmax_index <= '0;
559         argmax_best_digit <= '0;
560         argmax_best_score <= '0;
561         out_ch <= 0;
562         out_y <= 0;
563         out_x <= 0;
564         pool_ch <= 0;
565         pool_y <= 0;
566         pool_x <= 0;
567         neuron_idx <= 0;
568         term_idx <= 0;
569         term_word_idx <= 0;
570         pool_read_idx <= 0;
571         pool_max_reg <= '0;
572         mac_partial_sum_r <= '0;
573         for (reset_idx = 0; reset_idx < OUT_COUNT; reset_idx = reset_idx + 1) begin
574             logits_mem[reset_idx] <= '0;
575         end
576         cycle_counter <= 19'd0;
577     end else begin
578         if ((state == ST_IDLE) && start) begin
579             cycle_counter <= 19'd1;
580         end else if ((state != ST_IDLE) && (state != ST_DONE)) begin
581             cycle_counter <= cycle_counter + 19'd1;
582         end
583
584         case (state)
585             ST_IDLE: begin
586                 acc_reg <= '0;
587                 if (start) begin
588                     predicted_digit <= '0;
589                     argmax_index <= '0;
590                     argmax_best_digit <= '0;
591                     argmax_best_score <= '0;
592                     out_ch <= 0;
593                     out_y <= 0;
594                     out_x <= 0;
595                     term_idx <= 0;
596                     term_word_idx <= 0;
597                     for (reset_idx = 0; reset_idx < OUT_COUNT; reset_idx = reset_idx + 1) begin

```

```

598         logits_mem[reset_idx] <= '0;
599     end
600     state <= ST_CONV1_INIT;
601 end
602 end
603
604 ST_CONV1_INIT: begin
605     acc_reg <= '0;
606     term_idx <= 0;
607     term_word_idx <= 0;
608     state <= ST_CONV1_READ;
609 end
610 ST_CONV1_READ: state <= ST_CONV1_ACCUM;
611 ST_CONV1_ACCUM: begin
612     mac_partial_sum_r <= mac_partial_sum;
613     state <= ST_CONV1_ADD;
614 end
615 ST_CONV1_ADD: begin
616     acc_reg <= acc_reg + mac_partial_sum_r;
617     if ((term_idx + MAC_LANES) >= (CONV1_K * CONV1_K)) begin
618         state <= ST_CONV1_WRITE;
619     end else begin
620         term_idx <= term_idx + MAC_LANES;
621         term_word_idx <= term_word_idx + 1;
622         state <= ST_CONV1_READ;
623     end
624 end
625 ST_CONV1_WRITE: begin
626     if (out_x == (CONV1_CONV_W - 1)) begin
627         out_x <= 0;
628         if (out_y == (CONV1_CONV_W - 1)) begin
629             out_y <= 0;
630             if (out_ch == (CONV1_OUT_CH - 1)) begin
631                 out_ch <= 0;
632                 pool_ch <= 0;
633                 pool_y <= 0;
634                 pool_x <= 0;
635                 state <= ST_POOL1_INIT;
636             end else begin
637                 out_ch <= out_ch + 1;
638                 state <= ST_CONV1_INIT;
639             end
640         end else begin
641             out_y <= out_y + 1;
642             state <= ST_CONV1_INIT;
643         end
644     end else begin
645         out_x <= out_x + 1;
646         state <= ST_CONV1_INIT;
647     end
648 end
649
650 ST_POOL1_INIT: begin
651     pool_read_idx <= 0;
652     pool_max_reg <= '0;
653     state <= ST_POOL1_READ;
654 end
655 ST_POOL1_READ: state <= ST_POOL1_ACCUM;

```

```

656 ST_POOL1_ACCUM: begin
657     if ((pool_read_idx == 0) || (fm_a_r_data[0] > pool_max_reg)) begin
658         pool_max_reg <= fm_a_r_data[0];
659     end
660     if (pool_read_idx == 3) begin
661         state <= ST_POOL1_WRITE;
662     end else begin
663         pool_read_idx <= pool_read_idx + 1;
664         state <= ST_POOL1_READ;
665     end
666 end
667 ST_POOL1_WRITE: begin
668     if (pool_x == (CONV1_POOL_W - 1)) begin
669         pool_x <= 0;
670         if (pool_y == (CONV1_POOL_W - 1)) begin
671             pool_y <= 0;
672             if (pool_ch == (CONV1_OUT_CH - 1)) begin
673                 pool_ch <= 0;
674                 out_ch <= 0;
675                 out_y <= 0;
676                 out_x <= 0;
677                 state <= ST_CONV2_INIT;
678             end else begin
679                 pool_ch <= pool_ch + 1;
680                 state <= ST_POOL1_INIT;
681             end
682         end else begin
683             pool_y <= pool_y + 1;
684             state <= ST_POOL1_INIT;
685         end
686     end else begin
687         pool_x <= pool_x + 1;
688         state <= ST_POOL1_INIT;
689     end
690 end
691
692 ST_CONV2_INIT: begin
693     acc_reg <= '0;
694     term_idx <= 0;
695     term_word_idx <= 0;
696     state <= ST_CONV2_READ;
697 end
698 ST_CONV2_READ: state <= ST_CONV2_ACCUM;
699 ST_CONV2_ACCUM: begin
700     mac_partial_sum_r <= mac_partial_sum;
701     state <= ST_CONV2_ADD;
702 end
703 ST_CONV2_ADD: begin
704     acc_reg <= acc_reg + mac_partial_sum_r;
705     if ((term_idx + MAC_LANES) >= (CONV2_IN_CH * CONV2_K * CONV2_K)) begin
706         state <= ST_CONV2_WRITE;
707     end else begin
708         term_idx <= term_idx + MAC_LANES;
709         term_word_idx <= term_word_idx + 1;
710         state <= ST_CONV2_READ;
711     end
712 end
713 ST_CONV2_WRITE: begin

```

```

714     if (out_x == (CONV2_CONV_W - 1)) begin
715         out_x <= 0;
716         if (out_y == (CONV2_CONV_W - 1)) begin
717             out_y <= 0;
718             if (out_ch == (CONV2_OUT_CH - 1)) begin
719                 out_ch <= 0;
720                 pool_ch <= 0;
721                 pool_y <= 0;
722                 pool_x <= 0;
723                 state <= ST_POOL2_INIT;
724             end else begin
725                 out_ch <= out_ch + 1;
726                 state <= ST_CONV2_INIT;
727             end
728         end else begin
729             out_y <= out_y + 1;
730             state <= ST_CONV2_INIT;
731         end
732     end else begin
733         out_x <= out_x + 1;
734         state <= ST_CONV2_INIT;
735     end
736 end
737
738 ST_POOL2_INIT: begin
739     pool_read_idx <= 0;
740     pool_max_reg <= '0;
741     state <= ST_POOL2_READ;
742 end
743 ST_POOL2_READ: state <= ST_POOL2_ACCUM;
744 ST_POOL2_ACCUM: begin
745     if ((pool_read_idx == 0) || (fm_a_r_data[0] > pool_max_reg)) begin
746         pool_max_reg <= fm_a_r_data[0];
747     end
748     if (pool_read_idx == 3) begin
749         state <= ST_POOL2_WRITE;
750     end else begin
751         pool_read_idx <= pool_read_idx + 1;
752         state <= ST_POOL2_READ;
753     end
754 end
755 ST_POOL2_WRITE: begin
756     if (pool_x == (CONV2_POOL_W - 1)) begin
757         pool_x <= 0;
758         if (pool_y == (CONV2_POOL_W - 1)) begin
759             pool_y <= 0;
760             if (pool_ch == (CONV2_OUT_CH - 1)) begin
761                 pool_ch <= 0;
762                 neuron_idx <= 0;
763                 state <= ST_FC1_INIT;
764             end else begin
765                 pool_ch <= pool_ch + 1;
766                 state <= ST_POOL2_INIT;
767             end
768         end else begin
769             pool_y <= pool_y + 1;
770             state <= ST_POOL2_INIT;
771         end

```

```

772     end else begin
773         pool_x <= pool_x + 1;
774         state <= ST_POOL2_INIT;
775     end
776 end
777
778 ST_FC1_INIT: begin
779     acc_reg <= '0;
780     term_idx <= 0;
781     term_word_idx <= 0;
782     state <= ST_FC1_READ;
783 end
784 ST_FC1_READ: state <= ST_FC1_ACCUM;
785 ST_FC1_ACCUM: begin
786     mac_partial_sum_r <= mac_partial_sum;
787     state <= ST_FC1_ADD;
788 end
789 ST_FC1_ADD: begin
790     acc_reg <= acc_reg + mac_partial_sum_r;
791     if ((term_idx + MAC_LANES) >= FC1_IN_COUNT) begin
792         state <= ST_FC1_WRITE;
793     end else begin
794         term_idx <= term_idx + MAC_LANES;
795         term_word_idx <= term_word_idx + 1;
796         state <= ST_FC1_READ;
797     end
798 end
799 ST_FC1_WRITE: begin
800     if (neuron_idx == (FC1_OUT_COUNT - 1)) begin
801         neuron_idx <= 0;
802         state <= ST_FC2_INIT;
803     end else begin
804         neuron_idx <= neuron_idx + 1;
805         state <= ST_FC1_INIT;
806     end
807 end
808
809 ST_FC2_INIT: begin
810     acc_reg <= '0;
811     term_idx <= 0;
812     term_word_idx <= 0;
813     state <= ST_FC2_READ;
814 end
815 ST_FC2_READ: state <= ST_FC2_ACCUM;
816 ST_FC2_ACCUM: begin
817     mac_partial_sum_r <= mac_partial_sum;
818     state <= ST_FC2_ADD;
819 end
820 ST_FC2_ADD: begin
821     acc_reg <= acc_reg + mac_partial_sum_r;
822     if ((term_idx + MAC_LANES) >= FC2_IN_COUNT) begin
823         state <= ST_FC2_WRITE;
824     end else begin
825         term_idx <= term_idx + MAC_LANES;
826         term_word_idx <= term_word_idx + 1;
827         state <= ST_FC2_READ;
828     end
829 end

```

```

830 ST_FC2_WRITE: begin
831     if (neuron_idx == (FC2_OUT_COUNT - 1)) begin
832         neuron_idx <= 0;
833         state <= ST_FC3_INIT;
834     end else begin
835         neuron_idx <= neuron_idx + 1;
836         state <= ST_FC2_INIT;
837     end
838 end
839
840 ST_FC3_INIT: begin
841     acc_reg <= '0;
842     term_idx <= 0;
843     term_word_idx <= 0;
844     state <= ST_FC3_READ;
845 end
846 ST_FC3_READ: state <= ST_FC3_ACCUM;
847 ST_FC3_ACCUM: begin
848     mac_partial_sum_r <= mac_partial_sum;
849     state <= ST_FC3_ADD;
850 end
851 ST_FC3_ADD: begin
852     acc_reg <= acc_reg + mac_partial_sum_r;
853     if ((term_idx + MAC_LANES) >= OUT_IN_COUNT) begin
854         state <= ST_FC3_WRITE;
855     end else begin
856         term_idx <= term_idx + MAC_LANES;
857         term_word_idx <= term_word_idx + 1;
858         state <= ST_FC3_READ;
859     end
860 end
861 ST_FC3_WRITE: begin
862     logits_mem[neuron_idx] <= acc_with_bias;
863     if (neuron_idx == (OUT_COUNT - 1)) begin
864         state <= ST_ARGMAX_INIT;
865     end else begin
866         neuron_idx <= neuron_idx + 1;
867         state <= ST_FC3_INIT;
868     end
869 end
870
871 ST_ARGMAX_INIT: begin
872     argmax_index <= 4'd1;
873     argmax_best_digit <= 4'd0;
874     argmax_best_score <= logits_mem[0];
875     state <= ST_ARGMAX_SCAN;
876 end
877
878 ST_ARGMAX_SCAN: begin
879     if (logits_mem[argmax_index] > argmax_best_score) begin
880         argmax_best_score <= logits_mem[argmax_index];
881         argmax_best_digit <= argmax_index;
882     end
883
884     if (argmax_index == (OUT_COUNT - 1)) begin
885         predicted_digit <= (logits_mem[argmax_index] > argmax_best_score) ? argmax_index :
argmax_best_digit;
886         state <= ST_DONE;

```

```

887         end else begin
888             argmax_index <= argmax_index + 4'd1;
889         end
890     end
891
892     ST_DONE: begin
893         state <= ST_IDLE;
894     end
895
896     default: begin
897         state <= ST_IDLE;
898     end
899 endcase
900 end
901 end
902
903 assign busy = (state != ST_IDLE) && (state != ST_DONE);
904 assign done = (state == ST_DONE);
905 assign cycle_count = {9'd0, cycle_counter};
906 endmodule

```

## A.4 Testbench Code

Listing 19: ModelSim testbench for the INT8 LeNet accelerator.

```

1  `timescale 1ns/1ps
2
3  module tb_lenet_int8_top;
4      import lenet_int8_pkg::*;
5
6      logic clk;
7      logic rst_n;
8      logic start;
9
10     logic pixel_wr_en;
11     logic [$clog2(IMG_PIXELS)-1:0] pixel_wr_addr;
12     logic [PIX_W-1:0] pixel_wr_data;
13
14     logic pixel_dbg_rd_en;
15     logic [$clog2(IMG_PIXELS)-1:0] pixel_dbg_rd_addr;
16     logic [PIX_W-1:0] pixel_dbg_rd_data;
17
18     logic busy;
19     logic done;
20     logic [3:0] predicted_digit;
21     logic signed [OUT_COUNT * ACC_W-1:0] logits_flat;
22     logic [27:0] cycle_count;
23
24     integer i;
25     integer wait_cycles;
26     integer logit_index;
27     logic signed [ACC_W-1:0] one_logit;
28
29     lenet_int8_top dut (
30         .clk(clk),
31         .rst_n(rst_n),
32         .start(start),

```

```

33     .pixel_wr_en(pixel_wr_en),
34     .pixel_wr_addr(pixel_wr_addr),
35     .pixel_wr_data(pixel_wr_data),
36     .pixel_dbg_rd_en(pixel_dbg_rd_en),
37     .pixel_dbg_rd_addr(pixel_dbg_rd_addr),
38     .pixel_dbg_rd_data(pixel_dbg_rd_data),
39     .busy(busy),
40     .done(done),
41     .predicted_digit(predicted_digit),
42     .logits_flat(logits_flat),
43     .cycle_count(cycle_count)
44 );
45
46 initial begin
47     clk = 1'b0;
48     forever #10 clk = ~clk;
49 end
50
51 initial begin
52     rst_n = 1'b0;
53     start = 1'b0;
54     pixel_wr_en = 1'b0;
55     pixel_wr_addr = '0;
56     pixel_wr_data = '0;
57     pixel_dbg_rd_en = 1'b0;
58     pixel_dbg_rd_addr = '0;
59
60     repeat (5) @(posedge clk);
61     rst_n = 1'b1;
62     repeat (2) @(posedge clk);
63
64     // Write a simple 32x32 test image. Every pixel is 1.
65     for (i = 0; i < IMG_PIXELS; i = i + 1) begin
66         pixel_wr_en = 1'b1;
67         pixel_wr_addr = i;
68         pixel_wr_data = 8'd1;
69         @(posedge clk);
70     end
71
72     pixel_wr_en = 1'b0;
73     pixel_wr_addr = '0;
74     pixel_wr_data = '0;
75     repeat (2) @(posedge clk);
76
77     // Start one inference.
78     start = 1'b1;
79     @(posedge clk);
80     start = 1'b0;
81
82     wait_cycles = 0;
83     while ((done == 1'b0) && (wait_cycles < 2000000)) begin
84         @(posedge clk);
85         wait_cycles = wait_cycles + 1;
86     end
87
88     if (done == 1'b1) begin
89         $display("Inference finished.");
90         $display("predicted_digit = %0d", predicted_digit);

```

```

91     $display("cycle_count_#####=%0d", cycle_count);
92
93     for (logit_index = 0; logit_index < OUT_COUNT; logit_index = logit_index + 1) begin
94         one_logit = logits_flat[logit_index * ACC_W +: ACC_W];
95         $display("logit[%0d]=%0d", logit_index, one_logit);
96     end
97 end else begin
98     $display("ERROR: timeout_waiting_for_done.");
99 end
100
101 repeat (10) @(posedge clk);
102 $finish;
103 end
104 endmodule

```

Listing 20: ModelSim run script for compiling and simulating the accelerator testbench.

```

1 transcript on
2
3 if {[file exists work]} {
4     vdel -lib work -all
5 }
6 vlib work
7 vmap work work
8
9 vlog -sv ../RTL/lenet_int8_pkg.sv
10 vlog -sv ../RTL/BRAM.sv
11 vlog -sv ../RTL/lenet_addr_calc.sv
12 vlog -sv ../RTL/lenet_mac_array.sv
13 vlog -sv ../RTL/lenet_quantize.sv
14 vlog -sv ../RTL/lenet_argmax.sv
15 vlog -sv ../RTL/lenet_int8_top.sv
16 vlog -sv tb_lenet_int8_top.sv
17
18 vsim -voptargs=+acc work.tb_lenet_int8_top
19
20 add wave -divider "Testbench Control"
21 add wave sim:/tb_lenet_int8_top/clk
22 add wave sim:/tb_lenet_int8_top/rst_n
23 add wave sim:/tb_lenet_int8_top/start
24 add wave sim:/tb_lenet_int8_top/busy
25 add wave sim:/tb_lenet_int8_top/done
26
27 add wave -divider "Pixel Write Interface"
28 add wave sim:/tb_lenet_int8_top/pixel_wr_en
29 add wave -radix unsigned sim:/tb_lenet_int8_top/pixel_wr_addr
30 add wave -radix unsigned sim:/tb_lenet_int8_top/pixel_wr_data
31
32 add wave -divider "CNN Output"
33 add wave -radix unsigned sim:/tb_lenet_int8_top/predicted_digit
34 add wave -radix unsigned sim:/tb_lenet_int8_top/cycle_count
35 add wave -radix signed sim:/tb_lenet_int8_top/logits_flat
36
37 add wave -divider "Internal FSM"
38 add wave sim:/tb_lenet_int8_top/dut/state
39 add wave -radix unsigned sim:/tb_lenet_int8_top/dut/out_ch
40 add wave -radix unsigned sim:/tb_lenet_int8_top/dut/out_y
41 add wave -radix unsigned sim:/tb_lenet_int8_top/dut/out_x

```

```
42 add wave -radix unsigned sim:/tb_lenet_int8_top/dut/term_idx
43 add wave -radix signed sim:/tb_lenet_int8_top/dut/acc_reg
44
45 run -all
46 wave zoom full
```

## B Final FPGA Parameter Files

The following files are the final lane-packed INT8 parameter memories used by the FPGA accelerator through \$readmemh. Unpacked software-reference parameter files are summarized in Table 3; the packed pack3 files are listed here because they are the files directly consumed by the final RTL.

Listing 21: Packed Conv1 weights and biases for three-lane FPGA ROM loading.

```
1 bedff0
2 24eecd
3 e9cbc7
4 f73408
5 09ebdd
6 062d3f
7 4bff42
8 4c584c
9 000026
10 33200f
11 ab0731
12 c0e7c1
13 8fe8e7
14 dfe081
15 f10310
16 32d7de
17 232a29
18 00002e
19 262814
20 001816
21 3ded2e
22 fed406
23 3e1d32
24 294605
25 27f730
26 3437fe
27 000004
28 ea24fa
29 3ccffd
30 23274c
31 34023e
32 1c48f7
33 ed06d9
34 c50e21
35 29fa09
36 0000ec
37 f5ec1a
38 f6b9e9
39 d00933
40 2215ed
41 03f319
42 311747
```

```
43 35420b
44 16de22
45 00000b
46 130000
47 cd2e1c
48 0b38e1
49 fb102b
50 292150
51 350a2f
52 e242f8
53 f419d5
54 000010
55 0000fb
56 00007f
57 000004
58 000001
59 000027
60 000007
```

Listing 22: Packed Conv2 weights and biases for three-lane FPGA ROM loading.

```
1 2228ee
2 eee800
3 2312ea
4 f6bcdf
5 f50b20
6 1615f7
7 e12913
8 05f8e3
9 fcea11
10 f3f1dc
11 fbceb1
12 def2e9
13 0cefd1
14 cfe9e6
15 10ead8
16 e00ee6
17 22cef2
18 d0eeee
19 281ded
20 ecd80d
21 f22602
22 1dfce6
23 1b210a
24 07f211
25 eb1c25
26 e41410
27 f6e403
28 dce200
29 fe1d00
30 efe71b
31 3714e3
32 f0e508
33 1a221a
34 1ee905
35 ee10fa
36 24f2ee
37 e5e3e6
```

38 ff150d  
39 0fdf04  
40 111f11  
41 08eb0a  
42 1e0900  
43 fafefa  
44 fc1f1d  
45 11eacc  
46 e81cfc  
47 00e5ed  
48 f80f12  
49 180af6  
50 111dfe  
51 3934fc  
52 bbd0fe  
53 345917  
54 c0ce13  
55 074729  
56 fdccd9  
57 ea2439  
58 0ff6f0  
59 12e219  
60 e70c2f  
61 3ce3ca  
62 e50429  
63 13c79d  
64 c8e52a  
65 e3f2c4  
66 e7f0e4  
67 15d1e1  
68 e5dd14  
69 0808e1  
70 e5e918  
71 001b07  
72 e1d6e5  
73 151d18  
74 f5d6f7  
75 e7e920  
76 00f617  
77 2712da  
78 fb231a  
79 1202e8  
80 09122a  
81 3ff9dd  
82 ccf26  
83 f2ffdd  
84 1614f0  
85 edff0e  
86 0b29e3  
87 c2e713  
88 163806  
89 efebfd  
90 fe0825  
91 03dfe3  
92 110d14  
93 e0f2e1  
94 221fed  
95 f2dfdf

96 fe261a  
97 dae402  
98 ff231d  
99 ede4de  
100 e5e70c  
101 1103fc  
102 c1222f  
103 ede4cc  
104 ddc608  
105 d5d8db  
106 040305  
107 fee8ea  
108 0e2d1f  
109 321322  
110 0d1415  
111 0c1a0b  
112 38f6fc  
113 354141  
114 360928  
115 492c34  
116 2c1bf8  
117 000e1f  
118 201c16  
119 e60627  
120 0c050b  
121 ecd5d3  
122 d41e10  
123 d1d6e0  
124 f2e106  
125 f50a0e  
126 fcfcfc  
127 241414  
128 0f280a  
129 dbf10a  
130 f800ea  
131 eba5bd  
132 0ed4e1  
133 eaf1eb  
134 242ad5  
135 1d081d  
136 f5ee16  
137 d8fffc  
138 ddd1da  
139 f212d7  
140 d5e8d6  
141 f8fdf6  
142 0d04fd  
143 19f0fa  
144 11190d  
145 280023  
146 ddd2dd  
147 ef01ef  
148 e1d5b3  
149 ed13da  
150 01e6f5  
151 e8fb1b  
152 17fefb  
153 f2d72a

154 1d0afe  
155 e6ebd2  
156 c4c2f7  
157 d1e5f0  
158 16edca  
159 fa17dc  
160 fafa16  
161 000400  
162 f1df21  
163 e9f700  
164 0dfda9  
165 99d715  
166 270ff9  
167 1e9df4  
168 f6da11  
169 f93901  
170 25e4cd  
171 c4f026  
172 05441a  
173 0dc9f2  
174 e7f43c  
175 293be1  
176 ddfb14  
177 1506ec  
178 f7e71c  
179 e92c1c  
180 2a11cd  
181 d1e310  
182 fe140a  
183 03d9dc  
184 19f70b  
185 1df6ef  
186 e1362a  
187 3ffb00  
188 f5e617  
189 0d2300  
190 f9ffd9  
191 d9e80a  
192 ff3a00  
193 e8f7e3  
194 ea060b  
195 1904dd  
196 c7fd35  
197 001f09  
198 3adecc  
199 bc0e3b  
200 322713  
201 01fafa  
202 f2fbed  
203 2402e0  
204 22efff  
205 f5f514  
206 000d24  
207 24f7cd  
208 cde81a  
209 f1f301  
210 e0f513  
211 190e00

212 01ddd5  
213 c7e92a  
214 040cb1  
215 d7c1b5  
216 bddaf2  
217 1d18ed  
218 ffd1dc  
219 d10320  
220 010f0e  
221 0ed9e4  
222 f7032f  
223 201f05  
224 1e04eb  
225 e010f0  
226 f4df09  
227 111911  
228 f80fe2  
229 ddd621  
230 fc0613  
231 fe0001  
232 ef0e22  
233 0b10ff  
234 010efd  
235 03fd0b  
236 e4f415  
237 dd1e1b  
238 0b0fea  
239 f60a23  
240 161402  
241 1908f4  
242 201417  
243 02f809  
244 00f412  
245 212616  
246 01dcf0  
247 d9ed25  
248 230615  
249 2602d3  
250 e10b1e  
251 4e4419  
252 1f321e  
253 394215  
254 cbe8ec  
255 bcb4a6  
256 f80000  
257 fb1e0b  
258 454025  
259 341b20  
260 031637  
261 0dfafb  
262 02d8db  
263 1a041a  
264 54052f  
265 565e3c  
266 1114f0  
267 07e406  
268 0e1bf4  
269 10f630

270 26192b  
271 ef0d0b  
272 00ff1c  
273 dec7d1  
274 00e3dd  
275 f0f5d6  
276 fefb03  
277 09ebf6  
278 281e1d  
279 2f0b15  
280 022e18  
281 b7f111  
282 efdac5  
283 dfeded  
284 13fbe2  
285 25131c  
286 431f2a  
287 f81d1e  
288 bdd0d9  
289 ef01c8  
290 fedcef  
291 f812f5  
292 f20318  
293 01ffdf  
294 0eff28  
295 1c0818  
296 1dff02  
297 08ecf9  
298 e1c9c9  
299 dee9e4  
300 f3e1ce  
301 b8e4d4  
302 1faad7  
303 e5ed18  
304 211ed6  
305 2e2904  
306 090ff7  
307 0b283b  
308 0b1a14  
309 150af3  
310 e8ec13  
311 173b34  
312 1b4b36  
313 2f1e0e  
314 ee0960  
315 06ff1e  
316 10ffee  
317 ea1d01  
318 e9d1c0  
319 d0cf0b  
320 cdb2e5  
321 e0f4e2  
322 17e3e0  
323 1500f9  
324 020c28  
325 130f12  
326 efcbe1  
327 c6f5e4

328 afd8b6  
329 fcedd0  
330 dcd8f3  
331 1a060e  
332 fe2b20  
333 001422  
334 e1f728  
335 cfd9d8  
336 efd9f6  
337 1bcce1  
338 200204  
339 130a1f  
340 2d2400  
341 100518  
342 df0016  
343 f5e8d4  
344 b8ba1b  
345 fab4d7  
346 e5f3ef  
347 0be1ea  
348 f5f11a  
349 f30915  
350 202d01  
351 1921f6  
352 12dcf4  
353 eedb0b  
354 fad01d  
355 131a1a  
356 4c0be9  
357 49d42e  
358 e6383b  
359 e809af  
360 15f2cf  
361 fc0303  
362 da2909  
363 3610ef  
364 2e0bf7  
365 aad91d  
366 b6f617  
367 efd9b4  
368 e40a20  
369 eeffd7  
370 effbd7  
371 cfd7e1  
372 0730f9  
373 3721f6  
374 231920  
375 d82709  
376 030afb  
377 fc1311  
378 f40bea  
379 fe0ce4  
380 f508f4  
381 000ee1  
382 1ef904  
383 213002  
384 240fe9  
385 df11fc

386 fd16e1  
387 e1faf3  
388 08dee1  
389 09f800  
390 0ff920  
391 1a331e  
392 e4f0f7  
393 16031e  
394 0a0d12  
395 00eeec  
396 d0d6dc  
397 d618f5  
398 260be3  
399 130018  
400 d91f3b  
401 f4f5fe  
402 0a161a  
403 a9afc0  
404 2f38cb  
405 051227  
406 441af9  
407 ec1126  
408 2d240d  
409 dfe5eb  
410 eac7ee  
411 152527  
412 3f0b2e  
413 58645d  
414 fdf557  
415 26162f  
416 e7e800  
417 0018ff  
418 3c2310  
419 d0de26  
420 11ecce  
421 c4f00d  
422 f4d3db  
423 ffd4f0  
424 1e05e2  
425 f5241b  
426 3414fc  
427 db2426  
428 19e7d1  
429 db0115  
430 d9c69a  
431 ed0434  
432 1de4f0  
433 1d1428  
434 e6fafd  
435 2e0227  
436 beb9d8  
437 07e5b4  
438 dbf1eb  
439 2624d1  
440 040c10  
441 0f1bef  
442 f4080e  
443 2c242a

444 e5e43f  
445 00f7ff  
446 c0addd  
447 f200d6  
448 eae4e9  
449 fc15dd  
450 000424  
451 09e6db  
452 f20104  
453 fff010  
454 192b12  
455 1e1b11  
456 f1da14  
457 e8482e  
458 f924dc  
459 f6f4e5  
460 ead5dd  
461 eedb0c  
462 070ff7  
463 fbeted  
464 f1e41b  
465 2304f8  
466 2ceef3  
467 e8e11b  
468 0b00eb  
469 0ad1df  
470 faff12  
471 090f0f  
472 17f006  
473 f10ff2  
474 0eeb18  
475 13210d  
476 f603e5  
477 010eea  
478 0b00e4  
479 f40000  
480 e0160d  
481 16060a  
482 0bf6fa  
483 0c05f3  
484 cef208  
485 f0180c  
486 15e4d8  
487 1006fd  
488 1b1af4  
489 160810  
490 10101e  
491 01e6db  
492 e1f81d  
493 f6f207  
494 f8f2f0  
495 04eef7  
496 061bf7  
497 eb01ea  
498 fbf61c  
499 13fd00  
500 f7ffe7  
501 12f50d

502 1a220c  
503 504c14  
504 f2e740  
505 0c131b  
506 cbe0e9  
507 f3d5d8  
508 27fd19  
509 ead50f  
510 1a090b  
511 0cf6d2  
512 e6293c  
513 d0e4e9  
514 2900d4  
515 5c452d  
516 3039f7  
517 082242  
518 ffe814  
519 0916e3  
520 0d2af7  
521 0016e9  
522 ff25f6  
523 f1f1ee  
524 cfda1f  
525 f4f5c5  
526 ee0ee8  
527 f4ecd7  
528 f9f5fa  
529 2d0508  
530 2a2638  
531 08141b  
532 f1fbfa  
533 d4dbca  
534 fce3e7  
535 f0fddd  
536 21fc12  
537 1b1317  
538 1a330a  
539 d3f70e  
540 e5cbca  
541 eeeb00  
542 eee7f0  
543 e2e50f  
544 0c07f5  
545 f820e7  
546 251c10  
547 051a35  
548 e9fa09  
549 d1d5f5  
550 e6ead9  
551 1e190a  
552 e0e4fd  
553 1b1bea  
554 faeef6  
555 02251a  
556 e0f7df  
557 1c340c  
558 e6cc13  
559 090d0a

560 fd121f  
561 ebc3fb  
562 c6172a  
563 ddcfb1  
564 fb0dfe  
565 17e6d8  
566 ecef00  
567 ed0bd4  
568 ff03f4  
569 f702eb  
570 d702f5  
571 0f0c0b  
572 ead0e9  
573 1219fd  
574 0a08f8  
575 0111fc  
576 06f91a  
577 27feee  
578 ed0626  
579 f2f204  
580 fbf023  
581 lefae4  
582 edfcf7  
583 lbee12  
584 1f16e9  
585 fb0709  
586 1e1cfa  
587 04f1fe  
588 1c1912  
589 06f418  
590 fdfe fd  
591 f308f4  
592 fb0c1d  
593 fa12fa  
594 211500  
595 d9d0eb  
596 11ffe8  
597 0a03d9  
598 131d12  
599 0cd9da  
600 e80116  
601 c9fd01  
602 22bae7  
603 342817  
604 462706  
605 464744  
606 e90ece  
607 b52007  
608 b2b5dc  
609 e3eece  
610 c3e4c4  
611 2a2be8  
612 f12e0f  
613 1b2935  
614 ded43a  
615 1f04ea  
616 0e0606  
617 f92213

618 f5eed9  
619 f300e5  
620 f6d303  
621 241f1f  
622 1a0405  
623 1bfc1b  
624 d40808  
625 0be6f1  
626 f5cd09  
627 050517  
628 d9e9ef  
629 0e0ac3  
630 e51913  
631 222028  
632 f81a32  
633 f3f5f8  
634 fd0406  
635 cfe5cb  
636 f2f11d  
637 3af706  
638 33261a  
639 0d0334  
640 11f8ef  
641 e6d3d0  
642 e1e5c7  
643 0beb00  
644 e8eb04  
645 f5c0ef  
646 041102  
647 f9141e  
648 16150d  
649 d1fd16  
650 f6f8f9  
651 000502  
652 190d03  
653 dfdcfb  
654 1fe60e  
655 fced00  
656 4725c3  
657 e0f4fd  
658 1c3e2f  
659 d0e2f3  
660 f201c7  
661 efb997  
662 810c04  
663 0b17cd  
664 ee9702  
665 000645  
666 28c9c1  
667 102631  
668 eeed2a  
669 f529d9  
670 01ebd9  
671 d20c19  
672 2cf4e1  
673 ec0008  
674 20070c  
675 df0111

676 fc1628  
677 27d50b  
678 dc0bef  
679 fc0a07  
680 0e0bf4  
681 0c1514  
682 29f303  
683 e50121  
684 fbf2e8  
685 f3091c  
686 020e13  
687 1debf1  
688 d70706  
689 1df4ed  
690 e3f6fe  
691 1306ea  
692 190403  
693 d9e819  
694 160ffb  
695 02d7d5  
696 e4de0d  
697 30e80e  
698 edfaec  
699 0930fc  
700 0c0213  
701 1a3216  
702 fa0ae9  
703 270ef5  
704 dae616  
705 e40911  
706 fb0d34  
707 2fef05  
708 eeb509  
709 f3013a  
710 e0120e  
711 08dff4  
712 10e409  
713 cad301  
714 ed0fdf  
715 ddd6ac  
716 cde30f  
717 0114e1  
718 e4ebff  
719 07f2e2  
720 f71216  
721 fdeed9  
722 dd0d16  
723 08fe18  
724 1d02ea  
725 d6ef0e  
726 022401  
727 ff1802  
728 1b0207  
729 ebf50a  
730 031114  
731 0817d3  
732 e8e9f2  
733 0d03f4

734 15f2f5  
735 0c1e19  
736 e0eeff  
737 fd0012  
738 1a17f4  
739 ff05fc  
740 0b1ffb  
741 00e8e7  
742 f1ff13  
743 ef0504  
744 ee0708  
745 00dfed  
746 1b0afb  
747 d60bf8  
748 f21d04  
749 2b0aea  
750 cfe011  
751 26fef1  
752 e803fa  
753 2a15f9  
754 e2dd3e  
755 0fd1d5  
756 c803e0  
757 cdd9c9  
758 fbe2cd  
759 f2fc2e  
760 0222f7  
761 12cfee  
762 dc493c  
763 1014c7  
764 c5bff4  
765 29270a  
766 01dacc  
767 e71c00  
768 ec0909  
769 090ae5  
770 10f619  
771 ff0f1c  
772 032115  
773 09fef1  
774 df26f4  
775 fee4b2  
776 0010e6  
777 0606e3  
778 0f1202  
779 f8efdc  
780 3d42fe  
781 d2fa1d  
782 060905  
783 dcb503  
784 16e9e0  
785 e1071e  
786 021cef  
787 ed271a  
788 f506fd  
789 120e0c  
790 07b9db  
791 e01418

```
792 de0dcd
793 e3ef01
794 0d21e0
795 fb140e
796 131227
797 1f27fd
798 10eed3
799 d7ff10
800 eae0e4
801 0000a3
802 000081
803 00001d
804 00005c
805 0000bf
806 0000fd
807 000043
808 0000ce
809 00009b
810 00008f
811 000091
812 0000dd
813 000096
814 00003f
815 000001
816 0000de
```

Listing 23: Packed FC1 weights and biases for three-lane FPGA ROM loading.

```
1 020a0d
2 f00cfc
3 f6faf3
4 000804
5 ed02f4
6 f7f800
7 0debf5
8 0512ee
9 07f8f7
10 04fcee
11 f4f100
12 0206fa
13 f80bee
14 04f7ed
15 f2fa07
16 000b05
17 010306
18 fe05fb
19 f200fb
20 ec0300
21 f305f0
22 0aebf5
23 edf805
24 f60010
25 fcf8f9
26 08fb0b
27 0a01f8
28 11020f
29 0e00ff
30 ef00fa
```

31 f3f307  
32 f105f1  
33 fbf308  
34 f10c00  
35 f105fd  
36 faf108  
37 01fa0f  
38 05ecf3  
39 06f70d  
40 050cf9  
41 fa080a  
42 ee0e05  
43 eef40c  
44 060af8  
45 0d07ef  
46 f31006  
47 080201  
48 edf203  
49 f2f5f8  
50 ff000a  
51 000af4  
52 fa08f2  
53 01fdf6  
54 0e010b  
55 07f500  
56 fe1109  
57 f9fc06  
58 0b0af4  
59 effbfe  
60 f9ffee  
61 ecfeeb  
62 f8f103  
63 fafdf6  
64 0df5f9  
65 f8140b  
66 070cef  
67 09f3f0  
68 0a0e08  
69 07120d  
70 0e0e01  
71 08ee07  
72 05ea10  
73 f40df5  
74 050a07  
75 fd10f8  
76 efec04  
77 fb0804  
78 f1f002  
79 0efbed  
80 f500f2  
81 070105  
82 00f5ef  
83 000102  
84 f2f2f2  
85 0ef0ee  
86 fafeed  
87 f707f0  
88 02faec

89 f9f60c  
90 0c02ed  
91 f600f9  
92 0205fb  
93 f5f60c  
94 eb0af4  
95 f90cee  
96 020aec  
97 09f7f0  
98 06f1ec  
99 effef0  
100 fd0006  
101 0a0e0d  
102 ef07f4  
103 ee0afc  
104 0b040d  
105 0d0f04  
106 0707f1  
107 01eded  
108 0cfa05  
109 f8f903  
110 050c01  
111 f5fc03  
112 f0020b  
113 04f7ef  
114 f7f70a  
115 fff4f9  
116 f4f304  
117 09f200  
118 0209f9  
119 f7f5f5  
120 f0ee08  
121 fd070b  
122 fd0cf2  
123 f8eef7  
124 f100f4  
125 f3f7ed  
126 f0fd09  
127 f10309  
128 f700f9  
129 080a08  
130 f2fcf8  
131 090b01  
132 0efd00  
133 ec080b  
134 0000f5  
135 13fe17  
136 e90a1c  
137 f703e9  
138 10e513  
139 06eae8  
140 00f4ee  
141 100603  
142 02fb09  
143 e8f5f5  
144 362e12  
145 04f1f7  
146 eb1c0f

147 f5fe05  
148 df01e9  
149 00fc00  
150 15ff05  
151 340716  
152 0d3036  
153 fb16c6  
154 c5fe0b  
155 f31118  
156 e8dbfe  
157 0dfe0f  
158 08f113  
159 2c1403  
160 daebe4  
161 f0d6f4  
162 32dbdf  
163 faf432  
164 1905f6  
165 ee0705  
166 1df5f1  
167 03fbfc  
168 f1f100  
169 f8fef2  
170 1e0402  
171 ec1f27  
172 24160e  
173 03fa25  
174 f80df3  
175 ff0f12  
176 4407e8  
177 e51630  
178 131bdf  
179 e4e602  
180 f411fa  
181 e50c25  
182 152603  
183 fd42e  
184 08f503  
185 eeef25  
186 0408f7  
187 ecdf0c  
188 1708f4  
189 f3db13  
190 00e701  
191 ffeefa  
192 10d3e4  
193 fd2719  
194 dcedf3  
195 11fc02  
196 0afa20  
197 1c1e2e  
198 000a03  
199 05ee06  
200 03090e  
201 3f07ec  
202 f20634  
203 262216  
204 d3f62a

205 f2090b  
206 f3e100  
207 f802ff  
208 08df12  
209 29f1e9  
210 0fff05  
211 ee0601  
212 eff40a  
213 0cfa09  
214 1904fc  
215 f90fff  
216 2608fd  
217 0ae304  
218 070cf3  
219 f00f2c  
220 f00b30  
221 f509f7  
222 f804f5  
223 fbf517  
224 1dfff4  
225 fef70a  
226 f40cf7  
227 2417e9  
228 e9fb16  
229 26ff00  
230 fdf402  
231 0df6f5  
232 030c0b  
233 1112f5  
234 f407f8  
235 1923ff  
236 27190d  
237 dcf0e9  
238 1fff01  
239 12f616  
240 f409ea  
241 fbf30a  
242 07edf3  
243 ce06f4  
244 0112ee  
245 fbe8cc  
246 d30bf8  
247 0af4ec  
248 ec0e0d  
249 0014f6  
250 0d131e  
251 04fdf8  
252 230700  
253 040b31  
254 1e0c01  
255 1b03eb  
256 10f603  
257 0205ff  
258 fb0108  
259 13fef7  
260 130f06  
261 2ce218  
262 f0f714

263 14f1f8  
264 1c0fff  
265 fcfdfc  
266 f11215  
267 12f200  
268 000002  
269 0eee08  
270 ca0712  
271 17e8e3  
272 fadf0b  
273 070bed  
274 fcfe07  
275 0d0c09  
276 062118  
277 e60102  
278 192919  
279 09e3e0  
280 d00e10  
281 02e2e1  
282 00e802  
283 f8f6ed  
284 03f6fd  
285 1500fa  
286 e90c34  
287 e701b2  
288 d9eaed  
289 f9dbee  
290 0417e7  
291 00f8f9  
292 ea06e2  
293 fedff1  
294 f1f5eb  
295 f7ea00  
296 ecf702  
297 f92128  
298 33f305  
299 ffec11  
300 200515  
301 f1c511  
302 00fa00  
303 eb06fb  
304 100c0a  
305 e7fd00  
306 03061b  
307 0501ef  
308 12f80a  
309 fe1e03  
310 3329ee  
311 fc053d  
312 152ed3  
313 d3ff1b  
314 0103ed  
315 fd18f3  
316 fbfe08  
317 ed05fe  
318 13fcfb  
319 e61f13  
320 1908fa

321 0cd513  
322 f2e712  
323 10fae0  
324 05ee00  
325 0c0316  
326 0a1215  
327 091605  
328 efe4ee  
329 0412de  
330 f4e71b  
331 070d05  
332 f7f9fc  
333 f116fd  
334 2e180f  
335 0d0929  
336 0cf808  
337 17ea0c  
338 e50603  
339 cdeff0  
340 14fcd8  
341 f90c06  
342 10feffa  
343 dd0df9  
344 01080b  
345 ef0319  
346 f1fdf6  
347 feee03  
348 efee04  
349 06faec  
350 23f4ff  
351 131e04  
352 2c1b0d  
353 fd2135  
354 f1f518  
355 0202f3  
356 02faf6  
357 f8e9f5  
358 ffece3  
359 0c0bfa  
360 e2faf8  
361 1c16fb  
362 daf800  
363 191c10  
364 05f9f1  
365 f50af1  
366 f805f6  
367 0112f8  
368 0ff514  
369 221d09  
370 081820  
371 01f3fb  
372 e6d7f1  
373 01e606  
374 ee02ef  
375 07f1f5  
376 2003fd  
377 c2ec07  
378 0e09f2

379 ebe1d9  
380 f1211e  
381 17f800  
382 e6111c  
383 12ff1a  
384 01ff24  
385 120609  
386 000508  
387 f60913  
388 ff171b  
389 f4f1ed  
390 09ff04  
391 0d0afe  
392 05f115  
393 0a0600  
394 283309  
395 01ea10  
396 04f0d9  
397 f9f412  
398 09f9eb  
399 d5f6ff  
400 fafdf8  
401 08f7f8  
402 000005  
403 f9faf7  
404 e51c04  
405 00f2f0  
406 020911  
407 e6fa0a  
408 04faff  
409 eaafc19  
410 0201ef  
411 d70014  
412 260011  
413 1a01fd  
414 190703  
415 190031  
416 1b13ff  
417 0d1007  
418 100d07  
419 f303ef  
420 05f3ff  
421 0eed20  
422 e7f4f6  
423 01f316  
424 fc04e5  
425 090104  
426 f41200  
427 ede5ed  
428 b8ec11  
429 0dd9d4  
430 ebe008  
431 1b0515  
432 0807e6  
433 e60812  
434 09212d  
435 2ffc17  
436 0d1300

437 e5f916  
438 15f4f3  
439 f10f06  
440 00150d  
441 0ffc0e  
442 e7fffc  
443 042004  
444 0e00fe  
445 221930  
446 0f080e  
447 e40217  
448 0fe7f0  
449 f52c2c  
450 fc1bf6  
451 0ef101  
452 e7e9f6  
453 0c122c  
454 e92507  
455 f9d5d5  
456 f3dd07  
457 ece8ea  
458 ced2f2  
459 e2f9d5  
460 d2ebd7  
461 1b08fd  
462 160bfd  
463 16ded6  
464 eb1d34  
465 160e16  
466 1004d1  
467 fb03f6  
468 07f801  
469 000b13  
470 16f1fa  
471 05cc35  
472 042634  
473 e11b0a  
474 ecf6f4  
475 f4ebdf  
476 fdedee  
477 d4eaeb  
478 0305fb  
479 140720  
480 f70905  
481 fd0317  
482 02fff1  
483 0b00ed  
484 11e9fd  
485 0809f7  
486 27c604  
487 08333e  
488 0119fc  
489 f41511  
490 08f405  
491 fefc06  
492 d6ff0b  
493 fd0004  
494 00d0ee

495 140a01  
496 effd11  
497 120d16  
498 171402  
499 00f50a  
500 010b05  
501 03f9f7  
502 f11316  
503 0f11f8  
504 161323  
505 00ee03  
506 11f721  
507 ff09fa  
508 eaf803  
509 e2fbd5  
510 fbbc06  
511 dbf5ea  
512 0d00fc  
513 fafffe  
514 09f000  
515 09f117  
516 0aeafb  
517 010002  
518 1c0300  
519 fd1003  
520 f30213  
521 f70907  
522 1b0e24  
523 1e0003  
524 f8fa01  
525 00fe1a  
526 0c17f4  
527 09060a  
528 1126cf  
529 0af22c  
530 fbdb09  
531 f416fa  
532 0a0ff1  
533 dfeb1a  
534 09f004  
535 de0009  
536 0000dc  
537 160000  
538 ebf00c  
539 170ae3  
540 e5ef13  
541 090cec  
542 f3f700  
543 fb0a0d  
544 08f6f9  
545 0a01e0  
546 ed0312  
547 f3f108  
548 d1ec18  
549 f2dfd3  
550 edf4ed  
551 e704ec  
552 14e8e1

553 361e09  
554 ef092a  
555 f521e9  
556 dafe07  
557 05fcf2  
558 feddfc  
559 1e0e01  
560 0d022a  
561 2b2b18  
562 4f14e9  
563 fe102b  
564 2c5507  
565 f9111c  
566 112730  
567 05230b  
568 15f304  
569 e24923  
570 ec00d6  
571 eb0604  
572 0ff3e9  
573 e5df12  
574 0d000e  
575 06f2ea  
576 ffec05  
577 ed07f0  
578 1dccee  
579 eadaff  
580 0127e9  
581 e5e9ff  
582 e60c10  
583 fceef6  
584 08fdfa  
585 16ed27  
586 290f11  
587 f61608  
588 0febfd  
589 182a28  
590 14031c  
591 03181d  
592 f60c00  
593 150712  
594 03e00f  
595 f1e533  
596 f5eaf6  
597 0d1009  
598 f8f4f7  
599 f8010a  
600 18f910  
601 0f0201  
602 db1636  
603 12f6e2  
604 fd1d19  
605 1d21fa  
606 f4eaff  
607 e7eaf8  
608 e6e1e9  
609 00f4f0  
610 f01806

611 292bed  
612 f3120d  
613 fbe2fa  
614 11f504  
615 0bf3ff  
616 0d1116  
617 1607e2  
618 fa1c28  
619 02f804  
620 0b0d00  
621 e0070a  
622 03001e  
623 10f0f4  
624 e8e1e8  
625 02f813  
626 11170d  
627 10f8ff  
628 100421  
629 fb0b08  
630 fdece9  
631 f614f3  
632 01faf9  
633 fbfc0b  
634 f3f5fc  
635 fd03ed  
636 f9f0fd  
637 110dfd  
638 1104f5  
639 f4f605  
640 fe01f8  
641 120103  
642 f60006  
643 051415  
644 1bfae9  
645 000d28  
646 060900  
647 ecd7f1  
648 f81b18  
649 14e2da  
650 fe0a0d  
651 0c0903  
652 1cf114  
653 f9ed11  
654 021508  
655 eceff8  
656 e10d03  
657 ffeae9  
658 f1dff4  
659 eb140a  
660 f709df  
661 04ea0a  
662 160029  
663 030902  
664 0004fa  
665 e0fbf9  
666 160bfa  
667 0ffce1  
668 eb0b14

669 1a1def  
670 000011  
671 f400eb  
672 fdf902  
673 e9e4f3  
674 0afb2f  
675 00fbfc  
676 f5ec0e  
677 00f407  
678 060ef8  
679 ece7f4  
680 f000f5  
681 040a00  
682 07eff6  
683 0b0d06  
684 0e0d05  
685 f00309  
686 f8f400  
687 02ec01  
688 ff08ff  
689 080a06  
690 f5f70f  
691 feee01  
692 fafc09  
693 edeff3  
694 02fb09  
695 040006  
696 0ef40b  
697 ed0f03  
698 edf8fd  
699 0604f3  
700 05fbeb  
701 f7f303  
702 eff2f6  
703 05eced  
704 fa0fe6  
705 f60cee  
706 f4f7ee  
707 f80af3  
708 f402f9  
709 0cf6ee  
710 010a03  
711 f50aff  
712 0fed02  
713 07faf9  
714 f2fd00  
715 f40d07  
716 e9f7fc  
717 f400f4  
718 f0fe02  
719 f1f5f2  
720 ebf8f3  
721 eefff1  
722 eaeeec  
723 f70905  
724 080304  
725 fc0ffc  
726 f90a0b

727 07ebfc  
728 f5f7ed  
729 f505f6  
730 05f2fc  
731 0407fb  
732 01f9fd  
733 f4f310  
734 09f3eb  
735 0908f8  
736 02040c  
737 06f5f3  
738 04edf8  
739 e4ee07  
740 eefde3  
741 f0fbeb  
742 04ebfc  
743 ebffed  
744 f40a0e  
745 0500f8  
746 fa08f8  
747 060200  
748 0dfaea  
749 f7f500  
750 f10703  
751 000706  
752 080d0b  
753 08ffef  
754 06fa06  
755 0001fa  
756 00f7f9  
757 fcf3f7  
758 f5ebf0  
759 f8ffed  
760 05fd00  
761 0808f4  
762 01eff2  
763 09fcfb  
764 09f201  
765 fe00ec  
766 ff02fe  
767 fe0e0c  
768 effcf0  
769 f7f50a  
770 000cee  
771 f0f7ea  
772 f8fc05  
773 03f203  
774 0afc0b  
775 0ef301  
776 00f603  
777 faf3f6  
778 f50f03  
779 0001ed  
780 ec00f8  
781 0aea0c  
782 0507f8  
783 fbf700  
784 edfeff

785 ebf608  
786 0cf90e  
787 f4f5f5  
788 03f904  
789 00fdee  
790 ebee06  
791 ef0cfe  
792 040b00  
793 0e0af5  
794 fe0000  
795 090904  
796 f60300  
797 e5f900  
798 f3f6f2  
799 f60810  
800 fb00ed  
801 f2f5f3  
802 0ffef3  
803 fd08ea  
804 0000f4  
805 ea1104  
806 0bfa0b  
807 090f02  
808 1e11fe  
809 11f005  
810 ed0bef  
811 e8f2e4  
812 eaf5dc  
813 2a1c00  
814 00eff6  
815 000714  
816 19fc02  
817 ec0001  
818 feff09  
819 f2fc0b  
820 fdecdb  
821 0bf609  
822 f1e3f5  
823 f8ecfa  
824 060110  
825 07ddf3  
826 0ef21e  
827 0bf807  
828 f216fa  
829 05fef6  
830 4c2220  
831 273c2d  
832 21451a  
833 180bf6  
834 09f104  
835 1128f7  
836 0b03fc  
837 171a40  
838 14f30a  
839 241405  
840 f102fa  
841 18020d  
842 0c0c00

843 fa071b  
844 f3081b  
845 0912f4  
846 03dae8  
847 eff2e2  
848 d8ec15  
849 00f80a  
850 f4fef9  
851 0ef4fd  
852 19f0f4  
853 f2fe10  
854 f80e12  
855 08090c  
856 f30618  
857 e0f2de  
858 fce203  
859 e90107  
860 24e5e5  
861 e7f0f8  
862 d406cd  
863 06fbf0  
864 1f13f1  
865 0d0403  
866 f70f00  
867 230be9  
868 02f725  
869 e7dbef  
870 beee06  
871 eee9c5  
872 ecedfa  
873 f208ff  
874 fff3e0  
875 2702d7  
876 dbdfef  
877 dbefef  
878 d715e4  
879 e8f9ee  
880 01fe09  
881 f8fdf9  
882 0803f3  
883 28131b  
884 101317  
885 0603f3  
886 cd081b  
887 f500ea  
888 fa0cf1  
889 f8e8e3  
890 e9edd4  
891 0c0e16  
892 08f816  
893 0a1a06  
894 0e1915  
895 fc05e4  
896 1bfe00  
897 f50811  
898 1613ee  
899 1409ef  
900 000d06

901 1a0b03  
902 0a02f6  
903 f8fb07  
904 020ff7  
905 f00003  
906 ea06e7  
907 f613e2  
908 09f6fd  
909 fe000f  
910 15240c  
911 cf0015  
912 d3e2ea  
913 1012df  
914 15e7f2  
915 122122  
916 0af9f0  
917 051110  
918 3307fb  
919 f7080d  
920 1ce603  
921 05091c  
922 160e08  
923 1309fb  
924 f4f1e2  
925 04050d  
926 fd1107  
927 01f8ff  
928 1001fd  
929 e4110e  
930 f31d1f  
931 e832e3  
932 f12509  
933 14eefe  
934 f3faec  
935 042110  
936 eaf521  
937 160800  
938 0000f1  
939 15fa02  
940 df04f5  
941 ededf7  
942 090be9  
943 fifef8  
944 04f2e5  
945 f8000d  
946 0cf508  
947 f5ef08  
948 1908fe  
949 12f103  
950 110912  
951 0efd02  
952 0e08eb  
953 fd0611  
954 ebee0e  
955 0df8f6  
956 f7effb  
957 1010e9  
958 fff90f

959 060007  
960 fd37ff  
961 ee0009  
962 0e10d4  
963 ddd1e9  
964 a7eef9  
965 00cfc5  
966 c5c4fb  
967 2a0b34  
968 2201f3  
969 02eb17  
970 0d0e25  
971 1aec08  
972 f8ff08  
973 f70311  
974 0eef00  
975 f0ed00  
976 e5090d  
977 18f0f1  
978 ebed00  
979 0a0a16  
980 f4000f  
981 0ff5fc  
982 0ffbe3  
983 1efef8  
984 1bf5e7  
985 f41211  
986 d9f91d  
987 00ebcd  
988 e7e2ec  
989 fd0f04  
990 022416  
991 d9f0f5  
992 ee10e7  
993 f4edf5  
994 becefc  
995 030ed8  
996 eb0fe6  
997 1208ed  
998 0008f6  
999 1104e0  
1000 f1e607  
1001 f8eb2c  
1002 000de8  
1003 dcfed7  
1004 0d1c00  
1005 0beefc  
1006 fb1703  
1007 f0e808  
1008 f3170c  
1009 f33009  
1010 fa0cf8  
1011 efec00  
1012 0dfbfd  
1013 c3c909  
1014 0c07fd  
1015 e8111c  
1016 ee14f8

1017 e21301  
1018 00ed08  
1019 18efee  
1020 08f1f4  
1021 08edf7  
1022 27def7  
1023 071b1b  
1024 0cfe0f  
1025 e71602  
1026 00effa  
1027 f70beb  
1028 e4fbf0  
1029 00ebe6  
1030 05d4d5  
1031 02f0fe  
1032 f4eb01  
1033 090c16  
1034 110aff  
1035 f1041a  
1036 050e08  
1037 f4ff03  
1038 f605f6  
1039 2a2419  
1040 e51831  
1041 06df00  
1042 f4f201  
1043 e7eae6  
1044 fff50d  
1045 0300ff  
1046 e9f5ef  
1047 d7e7ec  
1048 f0f7e9  
1049 f80bf0  
1050 03f40a  
1051 26e807  
1052 04e615  
1053 0108fc  
1054 0e10fa  
1055 ff0b09  
1056 ed08f9  
1057 fd0010  
1058 f80018  
1059 0b09e8  
1060 fef9f3  
1061 02f700  
1062 02f9eb  
1063 17fc0f  
1064 f909f8  
1065 efebea  
1066 12e404  
1067 f9fc04  
1068 e502fd  
1069 fc0719  
1070 e6e00c  
1071 02f5eb  
1072 0000f9  
1073 13190a  
1074 e2f603

1075 e8ffda  
1076 02ee00  
1077 f0f0fe  
1078 ebfacb  
1079 f41822  
1080 0a2a1d  
1081 fef608  
1082 092125  
1083 1e0502  
1084 de0321  
1085 04e3d6  
1086 dbd2e4  
1087 0dfdfd  
1088 1bfde8  
1089 331220  
1090 120c1c  
1091 f80eea  
1092 e40010  
1093 ddf612  
1094 dc02f7  
1095 f0eae6  
1096 f7f102  
1097 f6e8e7  
1098 0ffe0e  
1099 f3f212  
1100 123fdf  
1101 fb0a0a  
1102 00112e  
1103 1c1e10  
1104 17fdec  
1105 f61030  
1106 090800  
1107 fd0bf9  
1108 0df4ec  
1109 0cec00  
1110 f92100  
1111 1b0af2  
1112 f4f707  
1113 0c1f01  
1114 24fbf2  
1115 001410  
1116 2d0bf8  
1117 f1051f  
1118 dbf713  
1119 effcec  
1120 f5f6fe  
1121 d9e3fa  
1122 0fe8df  
1123 080608  
1124 0f09ea  
1125 00e800  
1126 2632fb  
1127 00f303  
1128 fbe820  
1129 f20afc  
1130 12e6c4  
1131 0e081a  
1132 eaecfd

1133 edfa04  
1134 0be602  
1135 0b2134  
1136 f31ffe  
1137 04fbf5  
1138 071506  
1139 0b040f  
1140 edf70e  
1141 1a1aff  
1142 eaf10e  
1143 f6020b  
1144 fde6e0  
1145 0ecfc7  
1146 ddcc0e  
1147 0efafb  
1148 19140d  
1149 ebfefb  
1150 f0eded  
1151 f6eef8  
1152 000c04  
1153 18e7e2  
1154 21071b  
1155 11031f  
1156 2e0d0b  
1157 f6072a  
1158 161724  
1159 01070f  
1160 eedbeb  
1161 ebf7f2  
1162 00fbed  
1163 edff06  
1164 04f814  
1165 0c1d0e  
1166 f4f5ff  
1167 160201  
1168 f105fc  
1169 dc0201  
1170 fd110d  
1171 0ff6fb  
1172 0cfcfe  
1173 1c2d0a  
1174 0df31f  
1175 04e2f1  
1176 e9f3f3  
1177 eddcca  
1178 eee7f2  
1179 dc19f3  
1180 24fdf2  
1181 de1a2a  
1182 f7080b  
1183 0402f1  
1184 e01310  
1185 15efef  
1186 f5e80f  
1187 070516  
1188 212b0f  
1189 0bf205  
1190 0c08f9

1191 06f906  
1192 121304  
1193 130ff6  
1194 f8e3ef  
1195 0307f3  
1196 0410eb  
1197 0d030e  
1198 290701  
1199 ff0a19  
1200 010ced  
1201 c20513  
1202 ff05f7  
1203 00e0f0  
1204 d407fa  
1205 0912f0  
1206 000019  
1207 f4ee09  
1208 08ef0b  
1209 1213f4  
1210 001e03  
1211 1af913  
1212 ec1616  
1213 f3fcfe  
1214 fed1da  
1215 ff030b  
1216 00ff0c  
1217 efed0f  
1218 100405  
1219 f3f310  
1220 0e1200  
1221 ed0bfb  
1222 f406fd  
1223 f711f9  
1224 eee0de  
1225 fbded0  
1226 e2051b  
1227 1018fe  
1228 25cdd8  
1229 f51c2f  
1230 0b1600  
1231 f70127  
1232 fbfd0a  
1233 130a18  
1234 1507f7  
1235 fa070c  
1236 361cfb  
1237 1ef3fa  
1238 09293a  
1239 526304  
1240 0900e9  
1241 e3ef04  
1242 0cf909  
1243 fbfe10  
1244 0900e6  
1245 fcf7f9  
1246 e3f70c  
1247 fcece8  
1248 fb00ef

1249 efecf7  
1250 05fdf2  
1251 f3d8ff  
1252 0009ea  
1253 0c03ee  
1254 1b2b18  
1255 06fef8  
1256 eff61d  
1257 1109fd  
1258 1705fb  
1259 d0e3d7  
1260 ddc1f1  
1261 f2f724  
1262 18f3f7  
1263 32fae6  
1264 d8043a  
1265 0f142f  
1266 0401f9  
1267 27f815  
1268 e00911  
1269 0004e8  
1270 13f6ec  
1271 dadbf7  
1272 ea1b16  
1273 101be7  
1274 fdef11  
1275 d8daf5  
1276 0fecf8  
1277 f608e0  
1278 fcdfcc  
1279 d52329  
1280 2936c0  
1281 f0f603  
1282 fffeed  
1283 f50701  
1284 f8ff1a  
1285 08f505  
1286 f9ff10  
1287 e5ea05  
1288 f40a05  
1289 f5e8d5  
1290 e9f5e8  
1291 f7f909  
1292 ebebda  
1293 e9e7e0  
1294 f82812  
1295 210c06  
1296 f70d20  
1297 f5e4f4  
1298 fae6f6  
1299 09fcf8  
1300 0c0103  
1301 110cfb  
1302 0d1006  
1303 fcf20f  
1304 04f5e3  
1305 00f80d  
1306 f10bed

1307 0efcf2  
1308 ea0afc  
1309 e2dde6  
1310 060400  
1311 13ff1a  
1312 062813  
1313 f7e1e5  
1314 d5fae7  
1315 e6e3fa  
1316 0601fd  
1317 01e511  
1318 2708fc  
1319 fe02fb  
1320 18220d  
1321 1907dc  
1322 f2c808  
1323 0103f7  
1324 0f0007  
1325 041700  
1326 05180f  
1327 12f7ff  
1328 fe0002  
1329 00f5f9  
1330 fef2eb  
1331 02f3f2  
1332 140deb  
1333 00f4ee  
1334 edd3f5  
1335 070d01  
1336 04ebdd  
1337 f01c29  
1338 0509fd  
1339 111013  
1340 000000  
1341 01fe10  
1342 f10c00  
1343 04fff1  
1344 f70ff4  
1345 01f4ed  
1346 0401ef  
1347 07fbeb  
1348 eee5ff  
1349 fcfe04  
1350 fdf9f2  
1351 0708f5  
1352 00f6eb  
1353 04f1ec  
1354 10fe03  
1355 00ff0b  
1356 efec01  
1357 f7fa04  
1358 f50bf5  
1359 0f01e9  
1360 0014f3  
1361 f6ee01  
1362 e4f40c  
1363 f601ed  
1364 0009ff

1365 fd0908  
1366 01f9ef  
1367 f7fcf9  
1368 08f5f3  
1369 11e70c  
1370 0bf500  
1371 01feeb  
1372 fbf105  
1373 e7ecfd  
1374 0509fe  
1375 fbdeea  
1376 e90709  
1377 f9f1f5  
1378 efe700  
1379 000307  
1380 08f1f2  
1381 f0f5f5  
1382 e409f3  
1383 f004e5  
1384 f3f4f5  
1385 00f402  
1386 fcef08  
1387 e60700  
1388 06f701  
1389 00f4eb  
1390 f9ffee  
1391 fc0dff  
1392 f8fc08  
1393 f1f902  
1394 030ae8  
1395 e9040f  
1396 fbde5  
1397 fd0305  
1398 08f1fe  
1399 00050a  
1400 f1f808  
1401 05fbef  
1402 eff6f5  
1403 f412fb  
1404 fe000b  
1405 0401ee  
1406 f8fffd  
1407 fcf3fd  
1408 f70301  
1409 f707f1  
1410 0a0a05  
1411 f7f5f1  
1412 fd0d14  
1413 0deb02  
1414 e9f4ff  
1415 01ece9  
1416 edfb00  
1417 f20a04  
1418 ffeb02  
1419 f3f205  
1420 fdfe0e  
1421 f2e705  
1422 f0f0fe

1423 eafaec  
1424 f6e902  
1425 f3100e  
1426 0ef504  
1427 0bfaf5  
1428 01f400  
1429 fdf1fc  
1430 0803fd  
1431 ecf8f0  
1432 f4f8f0  
1433 fdfded  
1434 04f8f9  
1435 08f9fa  
1436 ec04f1  
1437 ebeafc  
1438 f4fdf3  
1439 e6f3fd  
1440 ebf9f7  
1441 e3e5ed  
1442 02ede6  
1443 08f9fd  
1444 f6fdfe  
1445 fdfbee  
1446 f8eee7  
1447 f7eff8  
1448 06f700  
1449 ff0dfc  
1450 e7f6fb  
1451 02f4f1  
1452 0efce5  
1453 f104f7  
1454 f503fe  
1455 f0f7fe  
1456 e809f4  
1457 0ce6fa  
1458 ff04e7  
1459 02f0f8  
1460 faf0f0  
1461 eaea0c  
1462 02f7ee  
1463 000a00  
1464 050800  
1465 040206  
1466 07f4eb  
1467 fb0013  
1468 fbfd01  
1469 0df1f6  
1470 f2eb0a  
1471 fa0704  
1472 f205f7  
1473 ec020b  
1474 00000d  
1475 f8ef05  
1476 12fa0e  
1477 10f1f4  
1478 f21101  
1479 f90d0f  
1480 fb0e00

1481 0d0411  
1482 0d0bf6  
1483 0f0afd  
1484 0f0fed  
1485 fcfb04  
1486 05f5f6  
1487 01000a  
1488 10f702  
1489 fa0af1  
1490 f9fced  
1491 f90fed  
1492 0d0c11  
1493 f10fee  
1494 f20000  
1495 f708f0  
1496 03ed09  
1497 f501ee  
1498 f700ef  
1499 f30d00  
1500 02f900  
1501 10f512  
1502 f606ef  
1503 f8fc00  
1504 ff09f9  
1505 01f309  
1506 f5faf9  
1507 1102ff  
1508 01fc00  
1509 01f0f1  
1510 f4f4fd  
1511 f50e05  
1512 10fd02  
1513 fdfdf9  
1514 0500f9  
1515 fe0d01  
1516 ed0505  
1517 0706f4  
1518 fef210  
1519 04f4fe  
1520 ee040e  
1521 1000fd  
1522 10f0f6  
1523 fb0e0b  
1524 f3f6ec  
1525 eef4fd  
1526 050ff2  
1527 02f2f2  
1528 f1f707  
1529 0310fa  
1530 02f210  
1531 f909fc  
1532 ee030b  
1533 010607  
1534 05eef7  
1535 fe05f3  
1536 10eff1  
1537 f7f4f2  
1538 00f1ee

1539 fbf307  
1540 0c10ee  
1541 f3f70f  
1542 fd0206  
1543 fa02f2  
1544 05faf7  
1545 0b030d  
1546 0600f2  
1547 f2f10e  
1548 ff0af0  
1549 f7ecfb  
1550 fd00fb  
1551 f90a05  
1552 f70702  
1553 f10bf0  
1554 eef609  
1555 fdf700  
1556 f60ff2  
1557 f00410  
1558 0e0603  
1559 060df4  
1560 07f5fd  
1561 f20e0b  
1562 080ff6  
1563 ed0dfb  
1564 eeef8  
1565 f9ff0d  
1566 ecedfa  
1567 f9f7ed  
1568 00f8fa  
1569 07ef0c  
1570 fb06fe  
1571 edf6ee  
1572 ecfcf0  
1573 07eff5  
1574 030900  
1575 0c0801  
1576 fdf8fc  
1577 f5feed  
1578 0eeff9  
1579 f607f3  
1580 f0f506  
1581 04f0f7  
1582 0becfd  
1583 f600f1  
1584 08ed07  
1585 01fbfe  
1586 05f0f7  
1587 03f702  
1588 f10d0f  
1589 03f6ef  
1590 f90df2  
1591 fef110  
1592 04f506  
1593 f90f08  
1594 10f209  
1595 f40c0d  
1596 ee0800

1597 00edf6  
1598 07f00d  
1599 0807f4  
1600 ff1105  
1601 f600f5  
1602 04fe0d  
1603 ff10f7  
1604 0d01ef  
1605 12f00d  
1606 fff5f8  
1607 000c00  
1608 0000ff  
1609 eef002  
1610 0712fa  
1611 fe090b  
1612 2a1dfd  
1613 0ef105  
1614 f90d0a  
1615 ddf001  
1616 0c0700  
1617 0eff0a  
1618 00def3  
1619 021d28  
1620 29f6e0  
1621 0a2830  
1622 120509  
1623 01f716  
1624 0efb05  
1625 f4ee16  
1626 eedebf  
1627 f9d31a  
1628 23fbe6  
1629 f10407  
1630 0307e8  
1631 fafff5  
1632 f006f4  
1633 fedaeb  
1634 fd1129  
1635 250f22  
1636 f60404  
1637 23f9eb  
1638 03c60c  
1639 fd2d0f  
1640 142305  
1641 18070c  
1642 1b0c17  
1643 0524fe  
1644 0e1712  
1645 050e04  
1646 fd0dfc  
1647 faee17  
1648 ebe8fe  
1649 15f9f3  
1650 0d07f9  
1651 0b0f05  
1652 efe42a  
1653 230ee9  
1654 0002de

1655 13321e  
1656 2727f1  
1657 f4000d  
1658 f70407  
1659 060804  
1660 eb0200  
1661 e5e8eb  
1662 e6cf0d  
1663 f400f8  
1664 f3dff0  
1665 f4edd2  
1666 f400d3  
1667 ef08fa  
1668 281c03  
1669 0afc13  
1670 df2f12  
1671 1100fb  
1672 0403f6  
1673 f800f2  
1674 e8feec  
1675 f80afb  
1676 f9f2d2  
1677 d4e218  
1678 1efddf  
1679 01fade  
1680 05f4f3  
1681 000014  
1682 fd06fd  
1683 e4f008  
1684 fdf600  
1685 311709  
1686 f7fb0c  
1687 250302  
1688 0efcf2  
1689 1209f8  
1690 dde5f8  
1691 fdf7ee  
1692 fad7f4  
1693 0704e7  
1694 03e7c9  
1695 0d11ec  
1696 041e25  
1697 1e0729  
1698 081605  
1699 0cfee6  
1700 21fafd  
1701 f5e50d  
1702 1c36f5  
1703 01f1fc  
1704 011116  
1705 16190a  
1706 fa10f6  
1707 03fb08  
1708 f61618  
1709 0aedf2  
1710 f90af5  
1711 080800  
1712 280b16

1713 10210e  
1714 0717f7  
1715 dceef6  
1716 fdf5eb  
1717 160305  
1718 0ff7f9  
1719 10261f  
1720 280608  
1721 fcfc31  
1722 2b07f5  
1723 060f0e  
1724 23f8d8  
1725 170817  
1726 f4ef15  
1727 f614fa  
1728 01fdf5  
1729 1e0d20  
1730 091718  
1731 051718  
1732 faf4ff  
1733 fe0c0f  
1734 fa1adb  
1735 c824f9  
1736 edf00f  
1737 253114  
1738 11fff9  
1739 f40006  
1740 0cefe2  
1741 ec02fb  
1742 0000e5  
1743 ff1503  
1744 1bf4f3  
1745 f31c31  
1746 1b0000  
1747 120c10  
1748 010fe2  
1749 ed03ea  
1750 1a0afb  
1751 211a07  
1752 f3e20b  
1753 0c2a30  
1754 2af900  
1755 fe2621  
1756 15f9fe  
1757 0811ff  
1758 f60b04  
1759 f6f3fa  
1760 0dd5d2  
1761 00f8e4  
1762 dfecdb  
1763 040200  
1764 0c00f7  
1765 1b0202  
1766 fd0c23  
1767 fceddf  
1768 161b23  
1769 0e252e  
1770 321a11

1771 28edf2  
1772 dfe50d  
1773 ee4710  
1774 08f4f1  
1775 120b13  
1776 1e1211  
1777 2125f5  
1778 fcf917  
1779 23350d  
1780 fa1a05  
1781 f7fafc  
1782 d9e601  
1783 170b0d  
1784 0cf2f1  
1785 dd02e2  
1786 dcdf11  
1787 06e4d8  
1788 f70af2  
1789 0c2d13  
1790 291300  
1791 e2e138  
1792 f808f9  
1793 05fe01  
1794 f90802  
1795 e50deb  
1796 ee13eb  
1797 dbdd02  
1798 e4d4f8  
1799 b7ecc7  
1800 d90ecd  
1801 e80406  
1802 0c231c  
1803 01f11a  
1804 051007  
1805 070f16  
1806 fa0112  
1807 08ebea  
1808 e2fde9  
1809 0c04ed  
1810 e6fa01  
1811 10fdf6  
1812 d5e611  
1813 070500  
1814 e4edf2  
1815 04dcf0  
1816 f7d9fe  
1817 f30af7  
1818 010905  
1819 3108fc  
1820 070826  
1821 0f0016  
1822 1115fd  
1823 0dee00  
1824 100c10  
1825 fd0000  
1826 e1eaeb  
1827 f5fcee  
1828 0b00f0

1829 2dfdfb  
1830 190814  
1831 130430  
1832 0c1c0d  
1833 f3050d  
1834 0ff608  
1835 041109  
1836 21351b  
1837 0cf619  
1838 040f1c  
1839 fa1a17  
1840 11fe0e  
1841 120400  
1842 fc1107  
1843 03f906  
1844 fbfeffc  
1845 fa2010  
1846 230710  
1847 0d1812  
1848 13f0eb  
1849 fb1d11  
1850 070301  
1851 372df3  
1852 f5f600  
1853 272206  
1854 fe040e  
1855 0f0803  
1856 1cf90d  
1857 181d08  
1858 192901  
1859 16f00b  
1860 09010c  
1861 110522  
1862 0e1003  
1863 191105  
1864 1c1018  
1865 f40815  
1866 f20dfd  
1867 f2ef17  
1868 dff005  
1869 f31fe6  
1870 ec06fe  
1871 34351e  
1872 022601  
1873 000204  
1874 02e706  
1875 040204  
1876 0000e6  
1877 0a0007  
1878 f3f617  
1879 02dfd2  
1880 f6ff07  
1881 07ea11  
1882 fafef7  
1883 fb0ef3  
1884 03f6ea  
1885 f00813  
1886 09180d

1887 f1e0d5  
1888 fa0614  
1889 fbfbe4  
1890 fdfcf5  
1891 0af8f6  
1892 19fd04  
1893 1a0301  
1894 0d3b45  
1895 0709b5  
1896 b4f606  
1897 11ff09  
1898 09f208  
1899 f92506  
1900 1c1e1d  
1901 13130d  
1902 00e7dc  
1903 f3f8eb  
1904 00f5ec  
1905 d6e4f9  
1906 ff0119  
1907 1de1e3  
1908 0f0002  
1909 fa0dee  
1910 fbf9d9  
1911 f0dfed  
1912 ed08f9  
1913 03f50c  
1914 130600  
1915 ebfef4  
1916 160a12  
1917 f40104  
1918 2b1502  
1919 f10114  
1920 fd2dc2  
1921 d9cf13  
1922 ec1520  
1923 17fefc  
1924 f90b1c  
1925 160af6  
1926 060909  
1927 ede7ec  
1928 fdeee3  
1929 faf507  
1930 100bf1  
1931 191a31  
1932 2d3214  
1933 17191b  
1934 2ded1d  
1935 eef91b  
1936 d5d4df  
1937 080912  
1938 0ff3f6  
1939 081bed  
1940 fd1200  
1941 0ffd16  
1942 f711fd  
1943 181e12  
1944 e21106

1945	061edd
1946	e3f41d
1947	31201e
1948	28f0f1
1949	0e1828
1950	1814fb
1951	2d1013
1952	fceff4
1953	dbe004
1954	0607f0
1955	15f7f1
1956	0c0006
1957	e10d10
1958	0e18f5
1959	00ea0d
1960	291f18
1961	e6f214
1962	07150f
1963	00efea
1964	eb0309
1965	0e0bfc
1966	0ef204
1967	f01a1b
1968	0126fc
1969	080d0c
1970	07e1fe
1971	0512f6
1972	0705eb
1973	fe000b
1974	03f7fc
1975	0a0308
1976	fa0805
1977	ff10fa
1978	fa0204
1979	d4f0ee
1980	1a04cd
1981	0c082a
1982	120703
1983	fff608
1984	fa1903
1985	f2f01b
1986	ff00ed
1987	f7dce4
1988	da1509
1989	dbfbe9
1990	f3dfffa
1991	0906f6
1992	ede507
1993	13fb01
1994	0b000e
1995	f2eb00
1996	eafb00
1997	e2faf4
1998	10fd10
1999	140a11
2000	140417
2001	1402e5
2002	03050b

2003	0fc0d6
2004	faf5e3
2005	0e01ea
2006	0c0cfd
2007	1624fb
2008	042102
2009	f4f106
2010	000028
2011	f91a06
2012	041f13
2013	eff1fd
2014	f5ea0b
2015	f3f90d
2016	1cf9ea
2017	2801f2
2018	000e28
2019	fefaf6
2020	1dff15
2021	2418fd
2022	100908
2023	fb080e
2024	130b08
2025	e6ec18
2026	ff1516
2027	fad9ef
2028	0afefc
2029	01f1da
2030	f807ed
2031	01eee9
2032	e50a00
2033	04dbe9
2034	eef500
2035	14fcfb
2036	e4170a
2037	0dd9f9
2038	1ec9f0
2039	ed0b0f
2040	150916
2041	12d7f8
2042	eb1df4
2043	d1c2e8
2044	0f19fc
2045	0f06f2
2046	011a04
2047	0816de
2048	fbf118
2049	fa281c
2050	1219f6
2051	00ff18
2052	e70811
2053	fcebfd
2054	f0f8e4
2055	f30808
2056	110905
2057	e90d0b
2058	09f906
2059	f3e9fb
2060	100be0

2061 0a0700  
2062 111818  
2063 fff520  
2064 d401d8  
2065 d8fd08  
2066 06f0d1  
2067 a3f712  
2068 ed2701  
2069 130b04  
2070 f2f1f9  
2071 df0106  
2072 1afdd0  
2073 1afef7  
2074 ed0df2  
2075 1409fe  
2076 3407dc  
2077 022523  
2078 0514f6  
2079 17fb18  
2080 e70000  
2081 f7eaed  
2082 d8dfffd  
2083 efe5e4  
2084 fde00d  
2085 0e1124  
2086 160007  
2087 1b270e  
2088 efff0a  
2089 0affea  
2090 06d601  
2091 0600ff  
2092 1003f1  
2093 24191f  
2094 03f6f1  
2095 2013fe  
2096 ff07f0  
2097 f90702  
2098 f8e7ff  
2099 ebeb10  
2100 f8ecff  
2101 ef0521  
2102 0af5f6  
2103 060c02  
2104 22130d  
2105 100622  
2106 f81111  
2107 f6f40c  
2108 080700  
2109 1f2af6  
2110 fbfe14  
2111 230a00  
2112 040e2c  
2113 0a190b  
2114 f5faf9  
2115 e1cfea  
2116 0006f9  
2117 feede7  
2118 fe111d

2119 fe030c  
2120 16f515  
2121 fb0d05  
2122 ed0b05  
2123 0e161a  
2124 3205ff  
2125 f6f615  
2126 122e1b  
2127 fd0c0e  
2128 0203fa  
2129 14141d  
2130 26f103  
2131 000d0f  
2132 0f10f8  
2133 feebf7  
2134 ef0f06  
2135 e60aff  
2136 18fede  
2137 fbf009  
2138 111708  
2139 f5d902  
2140 12ea07  
2141 e5f5ec  
2142 ee08ef  
2143 010107  
2144 000001  
2145 07ede6  
2146 f1f700  
2147 ff020c  
2148 f50aeb  
2149 e2ff1d  
2150 07f400  
2151 d3020f  
2152 fbf1e8  
2153 0601ee  
2154 edeceb  
2155 04000c  
2156 fe01f7  
2157 0d151c  
2158 fbf4ee  
2159 181801  
2160 08f3eb  
2161 fc0414  
2162 1c02eb  
2163 021754  
2164 3f090a  
2165 ebf303  
2166 f11d02  
2167 14ef00  
2168 fafe03  
2169 090603  
2170 f4fbfc  
2171 ee1af1  
2172 db0802  
2173 1a04f2  
2174 deffff  
2175 f72210  
2176 08e400

2177 f2f4e5  
2178 f61000  
2179 f300fe  
2180 fcf9fa  
2181 e5f500  
2182 f10bfd  
2183 1d0fff  
2184 0700f0  
2185 ee0ff6  
2186 e4f405  
2187 271c0f  
2188 03072a  
2189 281725  
2190 e6f714  
2191 d9ef08  
2192 02eaf7  
2193 250cff  
2194 000d18  
2195 fc00f5  
2196 df0ae5  
2197 271efd  
2198 3e1712  
2199 3e2608  
2200 e4fb24  
2201 d62bfc  
2202 12e4e8  
2203 f600ed  
2204 f1fceb  
2205 0f04de  
2206 fd0220  
2207 f50110  
2208 020d08  
2209 081bf b  
2210 06e4d2  
2211 cdf7f6  
2212 0401c7  
2213 1ffe12  
2214 2e3838  
2215 121128  
2216 083935  
2217 f5e5dc  
2218 e9e228  
2219 f411e8  
2220 00f6f8  
2221 04e7f3  
2222 f6f205  
2223 e91609  
2224 f81706  
2225 22fee7  
2226 00eaea  
2227 020ff9  
2228 06dd0d  
2229 15010c  
2230 fffc15  
2231 0e1100  
2232 f700ea  
2233 e3ddea  
2234 01f0f1

2235 000a05  
2236 0df311  
2237 ecefed  
2238 0a0df9  
2239 e10705  
2240 fd0a0d  
2241 f1f813  
2242 181611  
2243 fc0309  
2244 050415  
2245 e60ee2  
2246 03e4fa  
2247 15f809  
2248 2220fa  
2249 101f09  
2250 00e718  
2251 f60813  
2252 1a0af7  
2253 09f210  
2254 f50b15  
2255 020fee  
2256 fce2f9  
2257 0e0cee  
2258 e3f7f5  
2259 071e06  
2260 0d06f7  
2261 0af006  
2262 08faf5  
2263 e5f5f6  
2264 f40004  
2265 00f5f7  
2266 010701  
2267 01f910  
2268 040b01  
2269 1500f2  
2270 fdf4df  
2271 ff0a02  
2272 17ed03  
2273 f2160f  
2274 eb1701  
2275 f8f5fb  
2276 10ec07  
2277 0e0001  
2278 000009  
2279 0004eb  
2280 edef05  
2281 f202f6  
2282 090b05  
2283 07eaae  
2284 e60513  
2285 01f7ef  
2286 fbfd6  
2287 ef0205  
2288 09070c  
2289 e9fce2  
2290 0ce900  
2291 f903ec  
2292 e2fb07

2293 e8fdf3  
2294 0008f5  
2295 05e401  
2296 f70302  
2297 f4f9e9  
2298 f4f9ff  
2299 ee00ff  
2300 ef0c08  
2301 030f05  
2302 ff0909  
2303 00edec  
2304 f6fc02  
2305 e8e8e4  
2306 ea06fa  
2307 0ce2f2  
2308 f6fe0c  
2309 fbe3f9  
2310 f5e50c  
2311 f7f6ef  
2312 000cf4  
2313 e1fdf2  
2314 f6f7f6  
2315 07f800  
2316 f3f702  
2317 f6faef  
2318 0be3e4  
2319 e6edfb  
2320 f003fe  
2321 080ff9  
2322 0105ec  
2323 eff2f6  
2324 010004  
2325 ed0a0e  
2326 07f5fe  
2327 06fbef  
2328 f1f7fd  
2329 e1f4ec  
2330 dbeae6  
2331 0416ef  
2332 06ee0b  
2333 feeefb  
2334 fd0b01  
2335 0001f3  
2336 1b0e11  
2337 f10cf7  
2338 f5f5f5  
2339 f8eef1  
2340 effdfd  
2341 fdf803  
2342 fb0700  
2343 10ebe4  
2344 f8e9eb  
2345 f308ee  
2346 f2eae9  
2347 16ea05  
2348 0b04f9  
2349 04ebf3  
2350 f409ed

2351 10f8f4  
2352 08ecf1  
2353 020fff  
2354 f400ef  
2355 eae604  
2356 05eb09  
2357 020002  
2358 fd04f6  
2359 e70001  
2360 00e9e5  
2361 ff02ee  
2362 14040d  
2363 f8f001  
2364 e804ea  
2365 f8eb04  
2366 f8fcf3  
2367 010405  
2368 f9e6ef  
2369 f5fcfb  
2370 ec04ff  
2371 ebf108  
2372 f5f8fb  
2373 e9e5f4  
2374 e6f4f1  
2375 f105ef  
2376 eaf5e9  
2377 ef08ed  
2378 f2e0e9  
2379 07e903  
2380 0300fb  
2381 f90107  
2382 02edfd  
2383 fee801  
2384 fa00ea  
2385 fcf2f1  
2386 06f1f9  
2387 ebf80a  
2388 f10bfe  
2389 0207e6  
2390 fcece9  
2391 f4e1e5  
2392 0a06e0  
2393 e8f3fa  
2394 f60510  
2395 f9e5dc  
2396 06ebf3  
2397 00f7e8  
2398 e7e5f0  
2399 e5edff  
2400 0006e7  
2401 f2e9eb  
2402 edf8f5  
2403 04e2fe  
2404 140f12  
2405 02fbfe  
2406 f6fdfb  
2407 fd00f5  
2408 11fdf8

2409 fbf7f0  
2410 00dfec  
2411 0e10fa  
2412 0000f5  
2413 f0110d  
2414 0100ff  
2415 04f904  
2416 f90000  
2417 eb07fe  
2418 fe122e  
2419 0deff5  
2420 00f600  
2421 1011f2  
2422 eafae4  
2423 f20511  
2424 2005e3  
2425 05f2eb  
2426 152d07  
2427 fd0809  
2428 f10206  
2429 e7f1ee  
2430 fffc0  
2431 ede129  
2432 372711  
2433 ffe7ee  
2434 2a1625  
2435 fefe0b  
2436 2019cf  
2437 f60706  
2438 3f19fb  
2439 171719  
2440 ed2e24  
2441 1019fe  
2442 101e00  
2443 0900fb  
2444 f8382d  
2445 113631  
2446 fbe826  
2447 17090a  
2448 0ff807  
2449 02f100  
2450 ecf601  
2451 f30307  
2452 1a080f  
2453 0c02ee  
2454 05010b  
2455 12fd0e  
2456 0ee81b  
2457 2e210d  
2458 eff7e6  
2459 04e400  
2460 ece6e5  
2461 0e0bd9  
2462 f11603  
2463 0a12f2  
2464 f7fe06  
2465 17281a  
2466 d9ed0a

2467 0114e8  
2468 1828eb  
2469 52eb10  
2470 f72042  
2471 fdf0f4  
2472 090113  
2473 ee2409  
2474 f4e3e3  
2475 e4e9ed  
2476 14ec0c  
2477 db1119  
2478 02f410  
2479 e6d8f8  
2480 eff1e4  
2481 e0e206  
2482 2419e8  
2483 0ee6d6  
2484 161512  
2485 0a081d  
2486 0c53d5  
2487 aee8fe  
2488 04fcf3  
2489 fb0000  
2490 0a02fb  
2491 00040c  
2492 ecfa17  
2493 000821  
2494 f60305  
2495 000900  
2496 f214fe  
2497 01fa0e  
2498 0304fc  
2499 04031a  
2500 f1edec  
2501 1413f6  
2502 03f300  
2503 02f7ee  
2504 1ee1fd  
2505 f5f8fc  
2506 f50e01  
2507 100efc  
2508 0df220  
2509 2218f8  
2510 fff7f4  
2511 0ce900  
2512 01f9f3  
2513 fa0802  
2514 de01fc  
2515 0e0a02  
2516 fbe50d  
2517 00f6f3  
2518 00152c  
2519 0d08fe  
2520 0206ff  
2521 0f0fd1  
2522 12eff5  
2523 fd0414  
2524 58f706

2525 101025  
2526 1440f6  
2527 07e9fc  
2528 0de90f  
2529 0300f7  
2530 0412f5  
2531 1113f3  
2532 e9ff01  
2533 e8f8f3  
2534 e303fc  
2535 f60f0a  
2536 0af0fb  
2537 e7091b  
2538 fe1c16  
2539 e50720  
2540 16f814  
2541 eff000  
2542 f0f203  
2543 0d0408  
2544 19f412  
2545 00ff08  
2546 0000fd  
2547 0908e4  
2548 f6e900  
2549 080100  
2550 00f9e8  
2551 ef01ff  
2552 f6f107  
2553 fdf80b  
2554 25f900  
2555 f711fe  
2556 fd0c00  
2557 f6eaf4  
2558 fdf500  
2559 fe01d3  
2560 e70204  
2561 fa0b00  
2562 13ff06  
2563 151021  
2564 05071e  
2565 08251f  
2566 0edb0a  
2567 18111f  
2568 06fdfe  
2569 133026  
2570 201bfc  
2571 f6fd0b  
2572 2ffdf3  
2573 ef0a44  
2574 f41d0c  
2575 140a02  
2576 001af0  
2577 140bed  
2578 f4fc17  
2579 f52c06  
2580 f6e709  
2581 f91715  
2582 0ffde2

2583 d4f016  
2584 e71710  
2585 02e8d8  
2586 07f40f  
2587 f2fb06  
2588 01140a  
2589 20faf9  
2590 f71200  
2591 0e0ce3  
2592 15f000  
2593 1f001a  
2594 f1fd29  
2595 0d24e0  
2596 01081d  
2597 feeee5  
2598 0df5fd  
2599 002217  
2600 18210d  
2601 2200f5  
2602 f8252b  
2603 4c0b21  
2604 200740  
2605 f5d12d  
2606 f81202  
2607 211de7  
2608 fb0209  
2609 f0daf4  
2610 211806  
2611 1818f7  
2612 fb0b0e  
2613 feebfe  
2614 eaf300  
2615 080fdf  
2616 fcf2ff  
2617 fe0829  
2618 3003f0  
2619 081333  
2620 1c2328  
2621 270b18  
2622 ffeee0  
2623 e2eefc  
2624 0afef2  
2625 f9f6ed  
2626 fe1811  
2627 fdfe0e  
2628 fcf914  
2629 f22103  
2630 ee13ee  
2631 fffaf9  
2632 04ec1c  
2633 e50200  
2634 0c19fc  
2635 0300f4  
2636 d9f9fb  
2637 06f90b  
2638 f1f81f  
2639 fc07f1  
2640 00de0e

2641 ebfaf9  
2642 05f3f7  
2643 f7ef19  
2644 1b16ff  
2645 fdf107  
2646 f906fc  
2647 ebf3e5  
2648 efee02  
2649 0eee0f  
2650 0c1df4  
2651 fa0af0  
2652 e20327  
2653 3a04f9  
2654 0cf407  
2655 f4f622  
2656 0cff07  
2657 0ae31e  
2658 140f20  
2659 1df8d3  
2660 da1309  
2661 1a23f9  
2662 01f00c  
2663 e5f704  
2664 fc09e7  
2665 e8dd0c  
2666 ed080b  
2667 10faf1  
2668 f9de02  
2669 000df3  
2670 07fafe  
2671 0ffe02  
2672 c7f301  
2673 190ee5  
2674 17d1f0  
2675 f5f71c  
2676 021712  
2677 12f105  
2678 10ecf3  
2679 0018f6  
2680 000009  
2681 f811fe  
2682 f20bf1  
2683 f900fd  
2684 f6feeb  
2685 04ed11  
2686 1306f6  
2687 0d0315  
2688 0f0015  
2689 fff408  
2690 1c1415  
2691 1401eb  
2692 f3fa01  
2693 fd0909  
2694 0f1710  
2695 0dfd04  
2696 f60315  
2697 00f201  
2698 ff3018

2699 0bfff3  
2700 2afdff  
2701 fee902  
2702 f51f00  
2703 f200ff  
2704 f4f705  
2705 f7fde8  
2706 b2de08  
2707 fcd88e  
2708 d4d6e2  
2709 d21214  
2710 10de06  
2711 f0b802  
2712 102108  
2713 16b0d3  
2714 0d0efe  
2715 e400f9  
2716 f005ff  
2717 14e1e3  
2718 11ea16  
2719 f9fe00  
2720 ff15ef  
2721 fffe07  
2722 f730ff  
2723 13051f  
2724 1110f9  
2725 131722  
2726 15eb23  
2727 fe2313  
2728 e3ed1c  
2729 f3fdf3  
2730 05fe0d  
2731 0f1715  
2732 050ff3  
2733 e4f3fc  
2734 ff250a  
2735 0a0e01  
2736 faf709  
2737 0e1a24  
2738 250909  
2739 f6f404  
2740 effcfc  
2741 06eaf9  
2742 11ece0  
2743 eafe17  
2744 0e0806  
2745 16ec24  
2746 4c25f9  
2747 e61832  
2748 150efa  
2749 1210f9  
2750 0300fc  
2751 251a2d  
2752 2a211d  
2753 201604  
2754 3cfc18  
2755 0a0718  
2756 030d01

2757 060017  
2758 ebeff7  
2759 ff0ce2  
2760 f7f510  
2761 f9fc10  
2762 0ff5ec  
2763 15fb14  
2764 0af71f  
2765 24130b  
2766 030801  
2767 080402  
2768 e6eeef  
2769 fb04e9  
2770 00f7eb  
2771 f1fa01  
2772 e70cf0  
2773 0909f5  
2774 fef00b  
2775 f406fd  
2776 0a0e07  
2777 f5eff4  
2778 030ffd  
2779 1304f7  
2780 07fdf6  
2781 1208f1  
2782 fe1a17  
2783 fe0600  
2784 0cec06  
2785 f0f505  
2786 090dfd  
2787 0ceaf3  
2788 f90806  
2789 d9ed0d  
2790 02fa17  
2791 f913ec  
2792 e3fb01  
2793 f80ef7  
2794 f1db00  
2795 fc0d11  
2796 f10509  
2797 1a0e06  
2798 0a0416  
2799 fef315  
2800 10f6fc  
2801 f5040b  
2802 1204f5  
2803 07f50f  
2804 080bf9  
2805 1cf504  
2806 1007f8  
2807 e5e303  
2808 061ff9  
2809 f6ffff  
2810 0ef9ef  
2811 04f0f1  
2812 0dfeee  
2813 06e908  
2814 00000f

2815 fe1b03  
2816 0c1af5  
2817 13fe0b  
2818 f3f50f  
2819 130f0e  
2820 10240b  
2821 21faf4  
2822 01ed17  
2823 fe0202  
2824 0beff5  
2825 0d0719  
2826 1b2607  
2827 00fb06  
2828 051106  
2829 030a03  
2830 001917  
2831 eee4f9  
2832 15e3ca  
2833 d4ce20  
2834 2c1dea  
2835 0207dc  
2836 0cfe13  
2837 efd4f5  
2838 ff1bee  
2839 021217  
2840 272718  
2841 37fcf0  
2842 1f381e  
2843 ec0fec  
2844 15223f  
2845 1b02e5  
2846 ff25f2  
2847 e9f920  
2848 0dfd21  
2849 2b0e15  
2850 ea0803  
2851 29fce1  
2852 07ef15  
2853 fa0425  
2854 041d04  
2855 15fafa  
2856 b8f204  
2857 0f15da  
2858 dad529  
2859 161b01  
2860 080be6  
2861 19eff7  
2862 e8c7c2  
2863 f5f909  
2864 faf7f7  
2865 0605f3  
2866 1a120e  
2867 2a321e  
2868 ebe522  
2869 f80e00  
2870 10faf2  
2871 eedb23  
2872 eb061a

2873 0ff9ea  
2874 2519f0  
2875 f6fd0a  
2876 0cf2c3  
2877 f612db  
2878 13f301  
2879 f50c22  
2880 09fbf9  
2881 e7ef16  
2882 110af4  
2883 f2f206  
2884 2a01e3  
2885 07cdce  
2886 f7ee23  
2887 e810f8  
2888 062bdf  
2889 ea0314  
2890 05f6f6  
2891 190bef  
2892 03fef6  
2893 1c010c  
2894 ebf313  
2895 f20f05  
2896 f1070f  
2897 01fafd  
2898 d3cd00  
2899 140ad7  
2900 07e5d5  
2901 fefb19  
2902 e4fa04  
2903 fe08ec  
2904 09fcea  
2905 f70af0  
2906 0eee0a  
2907 f210f8  
2908 1413f4  
2909 0bfd0d  
2910 ff0c18  
2911 fd17f4  
2912 f00106  
2913 fa0df8  
2914 fb0afe  
2915 fbe3ff  
2916 fc0518  
2917 212af0  
2918 d7d821  
2919 ebddf2  
2920 06f909  
2921 01f406  
2922 f8fd02  
2923 2419e2  
2924 11f9fd  
2925 021505  
2926 23050f  
2927 ed1913  
2928 251300  
2929 10f716  
2930 f10af7

2931 ee090b  
2932 fe0d01  
2933 0c01f1  
2934 fe000c  
2935 f20703  
2936 fb1e11  
2937 080203  
2938 0400f1  
2939 ee1cfd  
2940 1febd6  
2941 bd0913  
2942 152ef8  
2943 e7def8  
2944 fff1ea  
2945 efe4e2  
2946 071100  
2947 f8e512  
2948 0000e9  
2949 0af512  
2950 f100f5  
2951 f7fbf2  
2952 f51eef  
2953 0907ed  
2954 fa060b  
2955 f8f701  
2956 f201f0  
2957 07fdff  
2958 04f2ef  
2959 070dfe  
2960 14ff11  
2961 110207  
2962 151afb  
2963 f3f4e7  
2964 f70909  
2965 f7f8e3  
2966 0e15f3  
2967 06fe28  
2968 38fc00  
2969 1e0406  
2970 321620  
2971 120c23  
2972 1104f5  
2973 cfedf5  
2974 eb03f7  
2975 0110c9  
2976 eff81d  
2977 0d19fa  
2978 072104  
2979 23f4ed  
2980 f63848  
2981 2e3108  
2982 fa091e  
2983 fcf1c7  
2984 10ebec  
2985 f5f1f6  
2986 eef40f  
2987 f1e8e5  
2988 df0202

2989 1b0de8  
2990 f50103  
2991 1afd02  
2992 09020e  
2993 1a0206  
2994 09fa03  
2995 0be913  
2996 000d14  
2997 07f7ed  
2998 f1fd07  
2999 111808  
3000 fc0014  
3001 f2f000  
3002 f8e5e6  
3003 0015ed  
3004 ff1122  
3005 4c09f6  
3006 0a1a3a  
3007 04e6f0  
3008 041815  
3009 020cd8  
3010 e501f8  
3011 f7e908  
3012 0cfff2  
3013 e4fff6  
3014 002619  
3015 08db01  
3016 fafd01  
3017 04ecfc  
3018 131e0e  
3019 f50afb  
3020 1ff2e1  
3021 022e25  
3022 2431c8  
3023 bcc7f7  
3024 1b00fa  
3025 0c171f  
3026 fe13f1  
3027 07f6f7  
3028 08fff6  
3029 08f719  
3030 dee802  
3031 fbf8f6  
3032 0a0a0c  
3033 fe1222  
3034 19fff5  
3035 070e0c  
3036 f91705  
3037 08fffa  
3038 f6e6f5  
3039 fce4df  
3040 04e7e1  
3041 f70801  
3042 ed04ff  
3043 faf813  
3044 13f50d  
3045 050318  
3046 fbf2fa

3047 01daf2  
3048 0dfb06  
3049 0d1c16  
3050 fd0309  
3051 0af1f6  
3052 05f014  
3053 00e8e0  
3054 02151c  
3055 0003f6  
3056 f4f9f3  
3057 00f7d7  
3058 0a11ee  
3059 fb091c  
3060 23f716  
3061 1f08ff  
3062 fd110b  
3063 03faed  
3064 02ea05  
3065 f10819  
3066 0004ee  
3067 00f502  
3068 f7fd23  
3069 04f2fb  
3070 e4fd01  
3071 f70703  
3072 fa07ed  
3073 e1fd1f  
3074 061801  
3075 e3060b  
3076 14f302  
3077 efff17  
3078 ebecf8  
3079 f0fc17  
3080 0ff1f3  
3081 f7f803  
3082 0000fa  
3083 f104ff  
3084 fe1f01  
3085 ec05eb  
3086 0f0b19  
3087 fde114  
3088 faff0e  
3089 33f7f9  
3090 ebf308  
3091 e4e30b  
3092 2a08f7  
3093 f7fdf3  
3094 0c12fa  
3095 f6190a  
3096 001412  
3097 f2f20a  
3098 07160e  
3099 030df7  
3100 170306  
3101 f0f4eb  
3102 e616f6  
3103 cfddda  
3104 01f0dd

3105 05faf2  
3106 f30702  
3107 1c05fb  
3108 daee1c  
3109 10d1ef  
3110 26cde5  
3111 f71501  
3112 16f6fb  
3113 f2f2f8  
3114 f0f80e  
3115 17dee1  
3116 12000d  
3117 da1203  
3118 e80f1b  
3119 262906  
3120 26130c  
3121 f5281f  
3122 1202f4  
3123 00ee00  
3124 ed08db  
3125 2322fd  
3126 f6f0f9  
3127 dded06  
3128 fbf6ea  
3129 051003  
3130 281d29  
3131 0f1d2a  
3132 faff1c  
3133 f1f513  
3134 1a06e4  
3135 dfe807  
3136 f8f5f7  
3137 e00521  
3138 f2d3de  
3139 d8f1f3  
3140 eceffd  
3141 1b15fd  
3142 fce705  
3143 130b23  
3144 1b0d1c  
3145 101119  
3146 06fdff  
3147 0ee103  
3148 f41500  
3149 12fcd7  
3150 fe0229  
3151 e8ea0f  
3152 fa0712  
3153 0c1100  
3154 cef706  
3155 02eefb  
3156 1701ee  
3157 02ea2b  
3158 01e8f5  
3159 051502  
3160 020024  
3161 0f140d  
3162 1a0af5

3163 060312  
3164 25f5ef  
3165 eee901  
3166 eed9fa  
3167 0420fb  
3168 ecf5e6  
3169 e9f3f1  
3170 0b040d  
3171 0ffd1f  
3172 08fef8  
3173 010f1a  
3174 ea17f5  
3175 0df2ec  
3176 f5fe18  
3177 000106  
3178 040611  
3179 1afdea  
3180 0cfa0d  
3181 0d2108  
3182 eef30d  
3183 13efe8  
3184 0c0ffc  
3185 e008fd  
3186 1400ff  
3187 fd1c11  
3188 0208fe  
3189 13f1fc  
3190 08190b  
3191 00f714  
3192 10eafa  
3193 fdebd8  
3194 faf1fb  
3195 09fa0c  
3196 2e1c0d  
3197 fefe11  
3198 071414  
3199 f409ea  
3200 0a0015  
3201 fe121f  
3202 320e08  
3203 f90721  
3204 06ff15  
3205 130c0d  
3206 0518f7  
3207 0fe3e8  
3208 1fd8da  
3209 000409  
3210 f2f4fd  
3211 ffd9eb  
3212 15f8ee  
3213 021907  
3214 fffff2  
3215 fe0b15  
3216 000002  
3217 e7e000  
3218 0e08fb  
3219 110815  
3220 200d01

3221 f80518  
3222 03f81f  
3223 fe1909  
3224 f6f0e3  
3225 f1f510  
3226 fc0c00  
3227 ee06f0  
3228 0b01fc  
3229 03021e  
3230 212502  
3231 0f170e  
3232 ecf4fe  
3233 fcf8ea  
3234 c1e806  
3235 0bf1d5  
3236 f4d9de  
3237 02fd0d  
3238 160cdb  
3239 ed142a  
3240 ed0dd6  
3241 cbe1df  
3242 e5efeb  
3243 f51707  
3244 dbc61b  
3245 1903f5  
3246 f1d3b0  
3247 f6e8f7  
3248 fb1f27  
3249 39efe2  
3250 0ff025  
3251 00fafc  
3252 000e03  
3253 e51505  
3254 09f00a  
3255 f80efe  
3256 edfc06  
3257 0a0011  
3258 f83126  
3259 ddf110  
3260 f6f9fe  
3261 09fbf4  
3262 0b02fb  
3263 180e21  
3264 fc0f0f  
3265 fb1adf  
3266 e4020d  
3267 09e4fb  
3268 eaff0a  
3269 d4e7e2  
3270 efd6e3  
3271 f2e4e9  
3272 03fb0e  
3273 2b0ad5  
3274 e80908  
3275 f400f0  
3276 0c0b12  
3277 16eeee  
3278 d50c0f

3279 fcec09  
3280 05f4f2  
3281 e80f05  
3282 1e0aff  
3283 fbf01f  
3284 f5f601  
3285 f8dd03  
3286 fd0014  
3287 0b2809  
3288 26101a  
3289 043b25  
3290 3848f5  
3291 c2da15  
3292 fa0102  
3293 f714ff  
3294 f61318  
3295 122103  
3296 00fa15  
3297 f20305  
3298 fb0af2  
3299 faea05  
3300 0a040f  
3301 fa0a0a  
3302 00ede4  
3303 f51408  
3304 181007  
3305 0e1404  
3306 ef0522  
3307 ef05f4  
3308 f5f7d8  
3309 0be8f9  
3310 f30200  
3311 100ef4  
3312 1d0e1c  
3313 11fe19  
3314 f90efa  
3315 fefa1c  
3316 17120a  
3317 f5fa10  
3318 070e0a  
3319 f2e5fe  
3320 15edf8  
3321 f2152a  
3322 010b1f  
3323 170a06  
3324 d1ed11  
3325 f5d7ec  
3326 04fbfb  
3327 f0fe19  
3328 33150e  
3329 170013  
3330 d4f316  
3331 0d04ef  
3332 fec7e4  
3333 0a0806  
3334 09f105  
3335 0afdfd  
3336 00f614

3337 fe12ee  
3338 ecff14  
3339 1df402  
3340 0ef522  
3341 10000b  
3342 f4f409  
3343 06fcde  
3344 10d7e9  
3345 0e060b  
3346 e6fee3  
3347 091121  
3348 1a04d7  
3349 f9edf4  
3350 0000fe  
3351 f3f2fa  
3352 0becf1  
3353 0f0bfa  
3354 000b0a  
3355 fcf10  
3356 f30609  
3357 f60dfd  
3358 edf0fd  
3359 0b0c03  
3360 fb09f9  
3361 0eec06  
3362 0aff00  
3363 f5f902  
3364 0df4ed  
3365 f60000  
3366 000606  
3367 f503f7  
3368 f31203  
3369 f1ecfa  
3370 fdf103  
3371 0200ed  
3372 0cfcf7  
3373 ec0502  
3374 f4f1f2  
3375 0df4f1  
3376 080702  
3377 03fdf6  
3378 f0fd0c  
3379 f5ef0e  
3380 0e0cfb  
3381 08fe0f  
3382 01edf6  
3383 00090e  
3384 08ecfb  
3385 f70af5  
3386 eceffd  
3387 f9edf4  
3388 10f000  
3389 0001fe  
3390 05faec  
3391 090aed  
3392 00f4f9  
3393 fdf3ef  
3394 fafcf1

3395 0103f9  
3396 f6edf8  
3397 0efaf9  
3398 040bf8  
3399 f509f5  
3400 faecf7  
3401 0004fd  
3402 0ff2ff  
3403 f20bf3  
3404 f0f900  
3405 ee01ed  
3406 0cf7ff  
3407 ee090c  
3408 eeedeb  
3409 f8f0f7  
3410 ed03f2  
3411 080603  
3412 0efdff  
3413 efeeec  
3414 0eefed  
3415 ef080e  
3416 030cf5  
3417 080000  
3418 0cf3e5  
3419 fb030e  
3420 faf10a  
3421 fef900  
3422 f10d00  
3423 f5f7e9  
3424 0d0dfb  
3425 06f5ed  
3426 ef0c07  
3427 eef9f2  
3428 0af30d  
3429 ecf5f2  
3430 0c0600  
3431 f7040b  
3432 ec0a0b  
3433 fbecfa  
3434 0000fb  
3435 01eeea  
3436 000901  
3437 fff5ec  
3438 09fdf4  
3439 0bea0f  
3440 fb040b  
3441 eef9f4  
3442 0506ef  
3443 fa06f2  
3444 00f30a  
3445 fffcf5  
3446 eeecf1  
3447 0c0e03  
3448 07f8fe  
3449 f0f2f4  
3450 fd0ced  
3451 0a0c0e  
3452 050f08

3453 040605  
3454 f8f603  
3455 ec0603  
3456 f501f2  
3457 0dfa06  
3458 08f700  
3459 0efd10  
3460 ff0d06  
3461 0b0def  
3462 fbf3f4  
3463 09f00e  
3464 06f4ed  
3465 f91003  
3466 f801ec  
3467 eff903  
3468 f0edf5  
3469 fbf80a  
3470 0807f7  
3471 fb10fa  
3472 0cffee  
3473 f5f10d  
3474 f7f5f4  
3475 f7f509  
3476 ef060f  
3477 f80008  
3478 fbfb02  
3479 03ecf8  
3480 f202f9  
3481 03eeea  
3482 070b07  
3483 ef09f1  
3484 000001  
3485 0608f3  
3486 030107  
3487 00f0fc  
3488 0cfbed  
3489 0f0e08  
3490 0808fe  
3491 0f08f9  
3492 000cf5  
3493 f8f0f2  
3494 0d06f7  
3495 faf5f3  
3496 fc03ec  
3497 fd0800  
3498 0204f8  
3499 030807  
3500 02ee0c  
3501 02fedd  
3502 0ff500  
3503 f2fffe  
3504 f204f8  
3505 f20fef  
3506 ff0003  
3507 f1f10d  
3508 f802ee  
3509 f710fa  
3510 f6f20d

3511 080ef8  
3512 11fc10  
3513 0005f8  
3514 01f701  
3515 f010ed  
3516 ff0af4  
3517 0f0109  
3518 f9090c  
3519 fb06fa  
3520 f9feef  
3521 0703ef  
3522 00f3f6  
3523 faf4ed  
3524 f200f0  
3525 06fe03  
3526 01fbed  
3527 030ef0  
3528 03f4ed  
3529 0a05f3  
3530 ecfa08  
3531 05eeef  
3532 f10601  
3533 0cf3ed  
3534 f2f108  
3535 fbf7f7  
3536 0006fe  
3537 050504  
3538 00fa00  
3539 f1f800  
3540 0ef10e  
3541 06080a  
3542 faf9fc  
3543 fafef8  
3544 0dfff8  
3545 06f200  
3546 0007f3  
3547 f0f9fe  
3548 f208f1  
3549 080fed  
3550 0bf5f3  
3551 060005  
3552 f60d07  
3553 f20cec  
3554 fcfeed  
3555 02ecf3  
3556 fe00f1  
3557 0ff60b  
3558 ed0703  
3559 f10bff  
3560 0e0305  
3561 ecf4fe  
3562 f20dfc  
3563 ebedf8  
3564 fc0c00  
3565 f3ee0c  
3566 fe0a0a  
3567 f7ef0d  
3568 07fb03

3569 fdeff1  
3570 f805fa  
3571 fb00fc  
3572 fafffe  
3573 0af200  
3574 f9ec0a  
3575 ff01ff  
3576 f3fcf2  
3577 f90906  
3578 07f1fb  
3579 07f500  
3580 02efee  
3581 f8f7f0  
3582 0007ff  
3583 0e0af2  
3584 fd0af5  
3585 f3f80e  
3586 00fbf2  
3587 00ed0f  
3588 02ef04  
3589 fcf4f9  
3590 f3fd0f  
3591 efedf7  
3592 f600f6  
3593 f2faff  
3594 00f2ef  
3595 0d05f2  
3596 03fefafa  
3597 fbfd0c  
3598 ef00ff  
3599 0300fe  
3600 f2ed04  
3601 f80007  
3602 fa0cf2  
3603 04efee  
3604 f10dfe  
3605 eff80a  
3606 f0fe08  
3607 eb0600  
3608 0000f9  
3609 f307ff  
3610 010af3  
3611 00faf6  
3612 0ef8ef  
3613 00f1f3  
3614 f90300  
3615 f70dff  
3616 edfef9  
3617 080d06  
3618 0000f0  
3619 040fec  
3620 0a00f8  
3621 08fd0a  
3622 f80cf3  
3623 0cf8fe  
3624 0d02ef  
3625 03f708  
3626 0901fa

3627 0df6fb  
3628 f7f6f9  
3629 fd01f6  
3630 fbef08  
3631 ec0cec  
3632 eff0ed  
3633 050e03  
3634 f9f5f4  
3635 f8f9f3  
3636 11f9f8  
3637 0800f7  
3638 ff0ff8  
3639 ee070d  
3640 edf407  
3641 f4f70f  
3642 fcf8f3  
3643 fefafd  
3644 0ef8f5  
3645 f10af6  
3646 f0f0ff  
3647 0dee07  
3648 09f407  
3649 eefcf7  
3650 04f9f7  
3651 0000fa  
3652 f1f603  
3653 09010a  
3654 f20701  
3655 f8fe0e  
3656 07fbf5  
3657 ed0ff8  
3658 05fd04  
3659 fd0100  
3660 0207f0  
3661 0211f6  
3662 04faff  
3663 eff1f3  
3664 00f50d  
3665 fc01f4  
3666 fe0800  
3667 fb05f1  
3668 fe100b  
3669 02edf4  
3670 fefaf0  
3671 04effd  
3672 fff0ff  
3673 f008f8  
3674 f70cf5  
3675 0709fd  
3676 f607fd  
3677 03eefc  
3678 ec06f0  
3679 f707f6  
3680 03f110  
3681 f00af7  
3682 ef0904  
3683 09f4f5  
3684 f1f30a

3685 ef0b00  
3686 05f407  
3687 0cf8f7  
3688 f4fe00  
3689 04f1f9  
3690 0800f8  
3691 040eec  
3692 fa0604  
3693 fefcf2  
3694 07060e  
3695 0e0907  
3696 eef9f6  
3697 f5f307  
3698 0d00fd  
3699 010ef3  
3700 fbfd04  
3701 ee02f9  
3702 0b0409  
3703 f5ffee  
3704 f1f9fb  
3705 ef010d  
3706 f4f9f9  
3707 fbfbf9  
3708 f7eff8  
3709 f800ec  
3710 f1ec08  
3711 f0edf5  
3712 f707f1  
3713 f106ed  
3714 fd030d  
3715 fa0df2  
3716 f6fb10  
3717 edf803  
3718 00f807  
3719 f3030b  
3720 fffc00  
3721 01ed0d  
3722 070af8  
3723 edeef4  
3724 0ffcf d  
3725 00ee0c  
3726 ec0bf0  
3727 08f101  
3728 f90000  
3729 0a0001  
3730 0fff09  
3731 10f9eb  
3732 0eef0e  
3733 0310f8  
3734 f3faee  
3735 03030f  
3736 0d0cf7  
3737 f3fd00  
3738 01f60e  
3739 0001fe  
3740 faeced  
3741 f500f2  
3742 10fafd

3743 07040f  
3744 05ff03  
3745 faedf0  
3746 f9f703  
3747 050907  
3748 f3fd09  
3749 080df9  
3750 ff00ef  
3751 f6f909  
3752 0000f6  
3753 0fe6e9  
3754 e1ff18  
3755 0304ee  
3756 fa0a0a  
3757 0202f8  
3758 faed00  
3759 fc0402  
3760 0c06fe  
3761 d8e20f  
3762 232716  
3763 fbe7ca  
3764 f500ff  
3765 0e1604  
3766 07fd09  
3767 f50cfb  
3768 1209f7  
3769 210905  
3770 ec1910  
3771 0c14ef  
3772 e400f6  
3773 dce7fd  
3774 deeee2  
3775 1304f0  
3776 110a24  
3777 0f1bfe  
3778 f6e701  
3779 04f208  
3780 04d2dd  
3781 fefd26  
3782 24fded  
3783 dddf41  
3784 0bf207  
3785 fbd0ce  
3786 0be8f1  
3787 e3fc05  
3788 0ced12  
3789 0f161f  
3790 fe2713  
3791 270e1d  
3792 050701  
3793 d9090b  
3794 0bfef8  
3795 10151a  
3796 0f20e7  
3797 d1f600  
3798 f100f2  
3799 ef06ff  
3800 2a1a16

3801 f1001d  
3802 07fc13  
3803 db0206  
3804 1800ec  
3805 ffec01  
3806 0c161d  
3807 0d0225  
3808 fae50a  
3809 d22011  
3810 0ddee4  
3811 f72012  
3812 ecf8ef  
3813 2ef203  
3814 021420  
3815 0b062e  
3816 f40fe6  
3817 110deb  
3818 ece204  
3819 20100f  
3820 fe0b0d  
3821 1ffd05  
3822 eb011f  
3823 06231e  
3824 d3fdf6  
3825 df6e7  
3826 0dc6fa  
3827 471016  
3828 06e3de  
3829 f6eb02  
3830 e90412  
3831 fc12f8  
3832 1dfdfc  
3833 0ef900  
3834 200cf3  
3835 e90711  
3836 fdfa21  
3837 fc0d12  
3838 f708f2  
3839 d802fd  
3840 14020e  
3841 0cf510  
3842 000a04  
3843 f91600  
3844 e31d0f  
3845 00fbf7  
3846 e1ff15  
3847 091100  
3848 1df8fd  
3849 f60907  
3850 fe1114  
3851 1d030d  
3852 12f904  
3853 1704f5  
3854 11f706  
3855 e2dc0c  
3856 1616f3  
3857 1a2a3a  
3858 e7e5f2

3859 e70600  
3860 071011  
3861 c5fc15  
3862 000df0  
3863 04d1ed  
3864 ef0705  
3865 07fff0  
3866 06030a  
3867 f91c10  
3868 121f05  
3869 0c09fb  
3870 1ff607  
3871 fde70f  
3872 0b1afd  
3873 1f0604  
3874 04ebef  
3875 0e091b  
3876 1619fb  
3877 0ae5e8  
3878 0b14d0  
3879 ffef18  
3880 f2e2f7  
3881 21ecf9  
3882 1016f7  
3883 04fbf2  
3884 06fdfe  
3885 f0010a  
3886 0000f2  
3887 f8f208  
3888 e7ef07  
3889 02f902  
3890 fd0cfe  
3891 0bfc01  
3892 fef00a  
3893 fa0aff  
3894 0dece5  
3895 db0115  
3896 2215fd  
3897 01f2e1  
3898 f413f4  
3899 edf7e7  
3900 f20efb  
3901 090601  
3902 07f6f4  
3903 25131a  
3904 d50d11  
3905 0c04c4  
3906 d8f1f8  
3907 010e1a  
3908 f6f7f6  
3909 e00d16  
3910 f3fcf3  
3911 e9f60e  
3912 e2d6e0  
3913 e9040d  
3914 00f1f2  
3915 08000e  
3916 1b06e2

3917 e9f8f0  
3918 011422  
3919 143713  
3920 f1f007  
3921 e60ff9  
3922 0dfd01  
3923 e51928  
3924 170ff0  
3925 f6eaf8  
3926 0f08fc  
3927 091109  
3928 2af6ee  
3929 eafe19  
3930 1117e5  
3931 e5ddf9  
3932 14fa15  
3933 fd1417  
3934 00181d  
3935 0f00f0  
3936 f3ee0e  
3937 f9f20d  
3938 0e0ff9  
3939 efe0f3  
3940 0bfa08  
3941 fcd9fc  
3942 f2f319  
3943 27feef  
3944 f0fb09  
3945 0e0c20  
3946 fbeaf8  
3947 390bff  
3948 e51329  
3949 ef0d1b  
3950 f40b03  
3951 0bf1fe  
3952 08110e  
3953 2608f6  
3954 0fff21  
3955 13ef02  
3956 e0ec16  
3957 ed1120  
3958 1cfef1  
3959 0b0126  
3960 1219ff  
3961 faf7f7  
3962 f60c02  
3963 fafb03  
3964 080700  
3965 feee0d  
3966 f900ff  
3967 f2000e  
3968 fd01f2  
3969 0605e7  
3970 160613  
3971 f8001b  
3972 010d1b  
3973 f406f2  
3974 17ff14

3975 10f20d  
3976 effc04  
3977 e9f4de  
3978 e8ffed  
3979 1d16e1  
3980 db520  
3981 1afaf6  
3982 f3f7fc  
3983 f9f40c  
3984 fd15fd  
3985 fcf508  
3986 ff04f2  
3987 0904f8  
3988 12f802  
3989 fffb00  
3990 10010c  
3991 f10412  
3992 fffdff  
3993 f21103  
3994 e8eaf4  
3995 d5e209  
3996 1513fd  
3997 f7ecef  
3998 fc01fd  
3999 02eadf  
4000 dbfa17  
4001 1012f6  
4002 06d3ff  
4003 f7ff09  
4004 0d15f3  
4005 e70b03  
4006 021711  
4007 0ef3fd  
4008 e4f708  
4009 0e0c10  
4010 02f8ee  
4011 0ff5fe  
4012 13171e  
4013 15f600  
4014 e8d5fa  
4015 021610  
4016 02ecea  
4017 ec0104  
4018 0a0a09  
4019 f4120f  
4020 000011  
4021 f71301  
4022 0b09f5  
4023 0df0ef  
4024 040cf2  
4025 0209f2  
4026 1006f9  
4027 f7fde7  
4028 f90602  
4029 fe02f8  
4030 01f7f1  
4031 041009  
4032 fbf606

4033 0603fd  
4034 fff100  
4035 f90b0a  
4036 00f304  
4037 f006ea  
4038 0c0afe  
4039 0c09f7  
4040 00f20a  
4041 f6fe0d  
4042 f509fb  
4043 0e06f7  
4044 0c0801  
4045 02f80d  
4046 0dff0d  
4047 ef000f  
4048 f20007  
4049 070efe  
4050 0bf502  
4051 f0f6f0  
4052 f406fd  
4053 0bf3f9  
4054 06ecee  
4055 0cf9ed  
4056 edf5ed  
4057 00fd02  
4058 04efeb  
4059 03f900  
4060 0c0207  
4061 fc04fe  
4062 03f304  
4063 040ef0  
4064 f002f5  
4065 f2f50b  
4066 040700  
4067 03edf2  
4068 f90107  
4069 050002  
4070 09ebef  
4071 000601  
4072 07f909  
4073 ee0af4  
4074 fb08f7  
4075 ef0a05  
4076 0ff9ee  
4077 f106ee  
4078 0af2f3  
4079 f80c11  
4080 f6eff4  
4081 fcf90f  
4082 f0ff0f  
4083 eff008  
4084 0807f0  
4085 0610ec  
4086 04fc10  
4087 fb00fb  
4088 f9f2f0  
4089 0009f3  
4090 f9faf6

4091 eafafa  
4092 070bfc  
4093 f203f5  
4094 02ed08  
4095 030c02  
4096 09fffb  
4097 1102f2  
4098 ee0d0d  
4099 0af0f9  
4100 0bebf8  
4101 f003f0  
4102 fd07f2  
4103 f2f5f3  
4104 03020a  
4105 fd0106  
4106 f20009  
4107 f609fe  
4108 08fef7  
4109 11ec00  
4110 fbed0c  
4111 060300  
4112 ee040d  
4113 ee0cf0  
4114 ff0409  
4115 05f706  
4116 070eeb  
4117 ee0fa08  
4118 01fef8  
4119 fa0105  
4120 0102ed  
4121 04f8ec  
4122 00fdfe  
4123 09f1f8  
4124 0600f6  
4125 fd0500  
4126 060004  
4127 fe0e02  
4128 f3ec00  
4129 12fa03  
4130 1003ec  
4131 020cee  
4132 07030f  
4133 f301f0  
4134 f5080b  
4135 0f07f3  
4136 faf0fc  
4137 ee0af3  
4138 f90afc  
4139 fd08f4  
4140 eb01f5  
4141 06f3f0  
4142 ee0ef9  
4143 fbf5ec  
4144 0af4ed  
4145 f50200  
4146 06f405  
4147 f6fb0e  
4148 f70cfc

4149 f904f2  
4150 0c010b  
4151 f3f905  
4152 ed05fd  
4153 f70c04  
4154 000003  
4155 09f410  
4156 f804fc  
4157 fdecf6  
4158 0af3e5  
4159 0201f7  
4160 d4ffe6  
4161 c9dffb  
4162 fe19cc  
4163 f4f2f4  
4164 0ffc00  
4165 fceb1c  
4166 f3fdf6  
4167 0bf9e0  
4168 ea03db  
4169 d305eb  
4170 0cfccc  
4171 25f5f2  
4172 362b1b  
4173 000022  
4174 111a20  
4175 04fefc  
4176 09fe17  
4177 fbfafe  
4178 fcf203  
4179 06faf4  
4180 f91bf3  
4181 021dff  
4182 2a2304  
4183 f6fe05  
4184 1d4f20  
4185 223011  
4186 fe1d32  
4187 194f33  
4188 fle3ee  
4189 eff41b  
4190 f708ee  
4191 0bfc09  
4192 fc1dfb  
4193 ff080a  
4194 e2d817  
4195 02081a  
4196 00d2e0  
4197 2e0bf2  
4198 05f719  
4199 f91228  
4200 e7fc04  
4201 fc04ed  
4202 10f6f5  
4203 dae508  
4204 f2f4de  
4205 0115fb  
4206 2a0c0c

4207 181e1b  
4208 311e0e  
4209 f31013  
4210 f1ebf5  
4211 e8f6ec  
4212 00d0b1  
4213 f6fb17  
4214 f4fa15  
4215 f40afa  
4216 f4f117  
4217 091b14  
4218 10f8ec  
4219 cdcee2  
4220 aaea04  
4221 18f0bf  
4222 0cf70b  
4223 090eef  
4224 053012  
4225 fcfdfb  
4226 fae3dc  
4227 c2b2d6  
4228 b3e6ea  
4229 01f7b4  
4230 000707  
4231 f90008  
4232 0efb08  
4233 06f205  
4234 e812f1  
4235 faddee  
4236 dbef19  
4237 ecf9e4  
4238 ffdedf  
4239 2d2a0b  
4240 050007  
4241 1003fe  
4242 0ffc00  
4243 f5e9f4  
4244 0f010d  
4245 d3cac6  
4246 e5dbfb  
4247 110800  
4248 f31007  
4249 080415  
4250 f40102  
4251 03fffb  
4252 04eaf6  
4253 04f3de  
4254 ecfb0d  
4255 1b11eb  
4256 000918  
4257 08fd00  
4258 eb0c06  
4259 00fbee  
4260 f60c02  
4261 caf907  
4262 eec2ad  
4263 e618d4  
4264 f809ee

4265 edf516  
4266 08ed1c  
4267 00fb00  
4268 f401fe  
4269 ec08ff  
4270 190afd  
4271 f1fd1f  
4272 001bf5  
4273 f20709  
4274 160606  
4275 1806f6  
4276 07fd0d  
4277 dc00f8  
4278 0b19de  
4279 e6f703  
4280 2a07ec  
4281 ed1142  
4282 1620ef  
4283 ddfb0e  
4284 fe2502  
4285 f1f9f8  
4286 bef325  
4287 eeebde  
4288 0000e2  
4289 070016  
4290 00fe04  
4291 f0fe00  
4292 f6fff6  
4293 ecf1fc  
4294 e8f9ff  
4295 04f7ed  
4296 edfdfe  
4297 f407f3  
4298 edffe6  
4299 f2f711  
4300 f304ea  
4301 060802  
4302 feeffb  
4303 ecf4eb  
4304 fa08fc  
4305 020205  
4306 0503e9  
4307 f30efa  
4308 080cf6  
4309 10f909  
4310 f3e501  
4311 f1f405  
4312 01f700  
4313 00f8f0  
4314 f6e404  
4315 0906ef  
4316 03f8f9  
4317 fff300  
4318 f90bfd  
4319 090707  
4320 f7f3de  
4321 def204  
4322 f9f607

4323 fffa06  
4324 e60c0c  
4325 0b00e7  
4326 f0f1e9  
4327 f4f7f9  
4328 01edf1  
4329 f60fe3  
4330 04e5ec  
4331 0706f7  
4332 f00e07  
4333 011006  
4334 f809fc  
4335 0bf9f0  
4336 0ef1f0  
4337 e4fe01  
4338 e9eee8  
4339 02f001  
4340 f50b0d  
4341 0f0002  
4342 ffeff0  
4343 ef00f6  
4344 eff510  
4345 f5f9f8  
4346 0502f4  
4347 07030a  
4348 f3ff06  
4349 f0f1f4  
4350 edfcfa  
4351 ef0bff  
4352 e8f7ea  
4353 e50cf7  
4354 0ffefc  
4355 16f010  
4356 f50402  
4357 0011fa  
4358 f1f0f2  
4359 fff6fb  
4360 06f0fe  
4361 0408fd  
4362 0f05ec  
4363 ec0d02  
4364 0b12fb  
4365 11fbfa  
4366 030102  
4367 00eff0  
4368 f107e6  
4369 eee2e9  
4370 0903f4  
4371 0b0800  
4372 f9030c  
4373 f00cf1  
4374 0f1109  
4375 110d13  
4376 06eafe  
4377 060000  
4378 1002e6  
4379 e9eb04  
4380 0d0801

4381 f2e9fc  
4382 fe0207  
4383 0aeeef  
4384 f7f404  
4385 fc0503  
4386 f7eaf6  
4387 f40cf6  
4388 09ece5  
4389 ff0a09  
4390 ff0509  
4391 f4f617  
4392 e10307  
4393 fef8f4  
4394 dce3e3  
4395 0bf3eb  
4396 000001  
4397 0304f0  
4398 e4f90b  
4399 fb0800  
4400 ef03f9  
4401 04feee  
4402 f3ebf2  
4403 edeffb  
4404 de0afb  
4405 0403fa  
4406 eefced  
4407 fe10f8  
4408 09f7f6  
4409 f1f6eb  
4410 ecfff7  
4411 f0f8f7  
4412 f1fb0d  
4413 f4fef1  
4414 f7eff2  
4415 120904  
4416 f20a1f  
4417 f405fe  
4418 fd0009  
4419 eb06fa  
4420 fff100  
4421 fe0be2  
4422 000006  
4423 16040d  
4424 2611fe  
4425 fad2eb  
4426 e0f900  
4427 09d9e5  
4428 fa15ff  
4429 09d7ed  
4430 c90c1a  
4431 f0f8d3  
4432 260cf9  
4433 e60af9  
4434 fe22fc  
4435 eef0f5  
4436 ec0afa  
4437 ebd2d1  
4438 110808

4439 18ebd1  
4440 3b0809  
4441 ded2f2  
4442 fa472a  
4443 11efe2  
4444 0c0528  
4445 f4e0ed  
4446 e50612  
4447 2124f7  
4448 291e00  
4449 261300  
4450 1e3e05  
4451 182f4a  
4452 3c6d41  
4453 25191f  
4454 1f3329  
4455 0c4c28  
4456 03e7e7  
4457 15070c  
4458 0305fc  
4459 1003f1  
4460 00041a  
4461 01ff09  
4462 1b1706  
4463 f60a13  
4464 07e7f8  
4465 fc040a  
4466 e8d600  
4467 ed2919  
4468 fbf701  
4469 01f1d2  
4470 f4e7dc  
4471 05ede8  
4472 fff70b  
4473 2121fc  
4474 1c0408  
4475 f42150  
4476 ff00e0  
4477 d2f316  
4478 1de4f3  
4479 fbcd25  
4480 0e0b02  
4481 12f9e3  
4482 090f16  
4483 ee2304  
4484 0acaba  
4485 030ee9  
4486 1401e3  
4487 f8f702  
4488 dcf42f  
4489 06c8f7  
4490 0a0e0f  
4491 fb090f  
4492 d7f20c  
4493 d2dcc5  
4494 d3d1e1  
4495 dfdcdf  
4496 d600ba

4497 d2f5e3  
4498 f90ef9  
4499 162a14  
4500 f6f804  
4501 fc0d16  
4502 0bedf0  
4503 f5fff9  
4504 ecf801  
4505 f9fdfb  
4506 e5d701  
4507 1c1111  
4508 08e3dc  
4509 0f1604  
4510 dcf012  
4511 e106f0  
4512 fc0ae2  
4513 ed0100  
4514 fcefff  
4515 0612f6  
4516 10181f  
4517 0c13f8  
4518 e3f10f  
4519 0c0de9  
4520 db203  
4521 020ff5  
4522 f7dcec  
4523 2509db  
4524 ef2821  
4525 f905ee  
4526 ef0b0d  
4527 ead0d2  
4528 dee7cc  
4529 ebd8d8  
4530 dbf2e9  
4531 ea0fb4  
4532 0bf4d0  
4533 cc08f3  
4534 040ef0  
4535 eaf21a  
4536 30170c  
4537 01e21a  
4538 323222  
4539 e6f107  
4540 fc09ee  
4541 061505  
4542 031afb  
4543 00fd25  
4544 0b0710  
4545 04fa13  
4546 1e1e05  
4547 f1f6ed  
4548 20fe00  
4549 ca1a21  
4550 0f2e10  
4551 f5e509  
4552 2af101  
4553 01d1e5  
4554 d40f1c

4555 fc0d08  
4556 0000ef  
4557 f7edf0  
4558 f00100  
4559 fbedf0  
4560 ff0201  
4561 ebfdf8  
4562 0ff9f0  
4563 09e4ed  
4564 e9ef18  
4565 04e9f6  
4566 f7f505  
4567 e9f2ea  
4568 f4060e  
4569 0000fc  
4570 0df9f7  
4571 eefafc  
4572 0005f2  
4573 03fff7  
4574 140c0f  
4575 050521  
4576 141404  
4577 c5e6df  
4578 e3ef10  
4579 fae6dd  
4580 0f000d  
4581 372400  
4582 3021e7  
4583 0cf3ea  
4584 e81f25  
4585 09220d  
4586 18fe0e  
4587 f10634  
4588 0f10e4  
4589 e8d8e9  
4590 000cf3  
4591 f7090d  
4592 f9f2f8  
4593 01e602  
4594 0feef6  
4595 17fe15  
4596 0c11f7  
4597 fefcfa  
4598 1cecea  
4599 f90b0a  
4600 161dfc  
4601 ee081b  
4602 d9d0e6  
4603 eedadc  
4604 1a05ec  
4605 16fc14  
4606 050b1a  
4607 f9ed02  
4608 120df5  
4609 250831  
4610 0a1322  
4611 200c06  
4612 fefbc4

4613 c8031e  
4614 fbf2eb  
4615 0400ed  
4616 15f402  
4617 fb0400  
4618 110ff5  
4619 04fcf5  
4620 e8ee1e  
4621 010c09  
4622 01e6da  
4623 f00a04  
4624 2a00fa  
4625 040c2a  
4626 331100  
4627 00e3e6  
4628 b50415  
4629 f4f9cf  
4630 ebfef13  
4631 121709  
4632 f6e9e8  
4633 fe06fc  
4634 fb0706  
4635 fcf700  
4636 f5150f  
4637 021ff7  
4638 051104  
4639 f8f9f6  
4640 0bdfd1  
4641 03f5ea  
4642 efefed  
4643 c90504  
4644 dff2e8  
4645 fef801  
4646 00f4f7  
4647 11160c  
4648 f60018  
4649 01f6ed  
4650 febf7  
4651 0ef2ec  
4652 0504f5  
4653 0506ee  
4654 00f114  
4655 0a060e  
4656 fdfcfe  
4657 e8e1f7  
4658 04eaf7  
4659 18f4e6  
4660 0bfd14  
4661 1afc02  
4662 f7f4f8  
4663 cc0611  
4664 03fbee  
4665 160308  
4666 0c0008  
4667 e4e607  
4668 f80307  
4669 0106ff  
4670 05f7f9

4671 f3f106  
4672 fc12f4  
4673 08f0fe  
4674 ec0b0d  
4675 f7f5e9  
4676 e9f7f9  
4677 04fbfa  
4678 f80004  
4679 13031c  
4680 0a00f5  
4681 f6f7fe  
4682 0d0503  
4683 dfee1d  
4684 0202f9  
4685 0dcaf4  
4686 ff030a  
4687 f6f1ff  
4688 eb1b0c  
4689 1403f5  
4690 0000f9  
4691 f5f200  
4692 1df9ef  
4693 010102  
4694 2109fb  
4695 eede16  
4696 f20e13  
4697 f51208  
4698 14efda  
4699 e3ed0d  
4700 07fced  
4701 f8020b  
4702 1910f1  
4703 e90534  
4704 010414  
4705 121000  
4706 0014ef  
4707 f20705  
4708 fcdecc  
4709 e3bc30  
4710 090cfd  
4711 e3f902  
4712 22fbe8  
4713 1c1505  
4714 0a05f4  
4715 ec0f04  
4716 070804  
4717 0a110a  
4718 1c14fa  
4719 f809e8  
4720 1a0010  
4721 2515f7  
4722 0cfe29  
4723 2f3e18  
4724 1708ee  
4725 010711  
4726 020016  
4727 191c04  
4728 180ddd

4729 e8f710  
4730 fdffff  
4731 0d0b00  
4732 0d02e5  
4733 1a1412  
4734 f9d722  
4735 fef100  
4736 0000fb  
4737 1610fa  
4738 231600  
4739 060e09  
4740 fafd08  
4741 00f90f  
4742 fb0300  
4743 ff07ee  
4744 e5bf08  
4745 f70a1d  
4746 0bfcee  
4747 0be2f2  
4748 ecfb0b  
4749 fc1b0e  
4750 250005  
4751 21f90a  
4752 ec2b22  
4753 0002fa  
4754 03f2f0  
4755 e300f7  
4756 dbfb12  
4757 fc09e2  
4758 f5ecf0  
4759 f9ccfa  
4760 230c1f  
4761 1af6c3  
4762 0ae302  
4763 f50d2b  
4764 fa10f4  
4765 f4f100  
4766 e20205  
4767 180207  
4768 09f620  
4769 191dfc  
4770 16070e  
4771 ff08f9  
4772 d90405  
4773 10f0dc  
4774 d7c6ee  
4775 fefd0e  
4776 feded0  
4777 02f4f0  
4778 050517  
4779 25090e  
4780 05f81e  
4781 f9fe08  
4782 f3f60f  
4783 fef7fd  
4784 031af4  
4785 00faf0  
4786 050918

4787 0209ee  
4788 f50306  
4789 fbf3ed  
4790 13fcef  
4791 fiebf4  
4792 eff9e4  
4793 ecf4f4  
4794 lafc13  
4795 130332  
4796 1025f4  
4797 f2e7f8  
4798 01ed0a  
4799 02010a  
4800 000cfd  
4801 01160c  
4802 1005ec  
4803 fa0b2c  
4804 1ff6fd  
4805 0017f0  
4806 f6d8e9  
4807 030c16  
4808 e9f50c  
4809 ff13ec  
4810 0cf600  
4811 f2ec15  
4812 fb090d  
4813 100cfe  
4814 00fa00  
4815 09fff6  
4816 121fbb  
4817 acf71b  
4818 f9daee  
4819 211301  
4820 2104fb  
4821 040e10  
4822 1b02f5  
4823 0fff0e  
4824 0000ff  
4825 0de9ec  
4826 02fdfd  
4827 f1fcf6  
4828 effe0f  
4829 f5eefa  
4830 04f1f1  
4831 f407ec  
4832 f80109  
4833 eb00f4  
4834 faf1ed  
4835 faf706  
4836 f6f10a  
4837 feeb02  
4838 fbec03  
4839 0104f3  
4840 1101ec  
4841 f6f2e6  
4842 f302f1  
4843 0805f4  
4844 0e00e9

4845 0ef2fd  
4846 08f415  
4847 eff7ef  
4848 edf1f4  
4849 06ef0d  
4850 05faf0  
4851 fb0d02  
4852 f5eefc  
4853 f509f3  
4854 fff5fc  
4855 0d0500  
4856 0cf0fe  
4857 f8f6f8  
4858 e50303  
4859 05fbf3  
4860 f4fe07  
4861 0cf8f0  
4862 0cf9fe  
4863 0808f5  
4864 04e7ed  
4865 f0fbf2  
4866 ebeec  
4867 09e70b  
4868 f50805  
4869 02f20b  
4870 fdeef7  
4871 f3ebf4  
4872 06fefa  
4873 edfdef  
4874 f1f3ef  
4875 fbeb09  
4876 e70007  
4877 fa0200  
4878 f5f70b  
4879 fc0105  
4880 00fb04  
4881 fbefed  
4882 ee0504  
4883 fd0a0d  
4884 e6fdee  
4885 f9fe09  
4886 fdf5ec  
4887 ecea00  
4888 fa00f3  
4889 f400ef  
4890 fff5fe  
4891 eaf000  
4892 fff2f2  
4893 0c0bec  
4894 ee0708  
4895 eb0bec  
4896 f801f3  
4897 150ce8  
4898 f307f1  
4899 07f314  
4900 ecebf0  
4901 f00009  
4902 effff2

4903 edf20f  
4904 eefeed  
4905 07ff02  
4906 f209ee  
4907 05edfa  
4908 f704ef  
4909 f5e8fa  
4910 0d0800  
4911 fce601  
4912 f809f5  
4913 08feee  
4914 eaf408  
4915 f9f20d  
4916 06f7e8  
4917 e9ecea  
4918 fdecff  
4919 f50006  
4920 fa0bfe  
4921 f000fc  
4922 0502fe  
4923 01f00a  
4924 fff3f3  
4925 07f5fe  
4926 f208fa  
4927 06f502  
4928 fe0eed  
4929 f305f3  
4930 fc09f6  
4931 fdf607  
4932 0e0e00  
4933 0508fd  
4934 0300f0  
4935 fafc06  
4936 ee00f2  
4937 f5edf2  
4938 09f30d  
4939 02f80d  
4940 f20b00  
4941 fa0000  
4942 00faf7  
4943 01080a  
4944 02eaf9  
4945 0aed06  
4946 f00fef  
4947 000af0  
4948 010d07  
4949 e9e705  
4950 f9fff6  
4951 18e9e8  
4952 fe0e07  
4953 fdf400  
4954 f701f4  
4955 efee00  
4956 ee0ae7  
4957 fbf110  
4958 000001  
4959 1008ff  
4960 141200

4961 0c191f  
4962 0c06da  
4963 db1504  
4964 0bfae5  
4965 ede917  
4966 0d3e06  
4967 ed06e9  
4968 fefd13  
4969 231a16  
4970 02f7fd  
4971 051913  
4972 0000ef  
4973 010b1d  
4974 0802e5  
4975 ec070e  
4976 16f2e1  
4977 fa1326  
4978 1edf04  
4979 fa101d  
4980 ed17de  
4981 1ef2e8  
4982 e3e826  
4983 03fdef  
4984 152c21  
4985 050c2f  
4986 040213  
4987 25111e  
4988 03ffe8  
4989 e4412c  
4990 28fcf9  
4991 fbec12  
4992 0410ec  
4993 1ef80d  
4994 0ffe02  
4995 0900fb  
4996 eb2910  
4997 1d1df4  
4998 f7ef05  
4999 00060d  
5000 ed02e6  
5001 0901fa  
5002 03eb2e  
5003 1ff4fa  
5004 e4e814  
5005 c5260e  
5006 190dee  
5007 f4ec1b  
5008 fd00e4  
5009 f70c0f  
5010 dc0e0b  
5011 18faf6  
5012 252b09  
5013 f50d0d  
5014 d8f91d  
5015 c00b01  
5016 01e5b2  
5017 08fe19  
5018 fd0a13

5019 f700e9  
5020 f5002d  
5021 15032b  
5022 fc1deb  
5023 1808d2  
5024 f8d0f1  
5025 f6190c  
5026 d6dcd1  
5027 0000f3  
5028 ff1c14  
5029 e50d0b  
5030 d9fff5  
5031 dab8c5  
5032 bc9a21  
5033 1e25e5  
5034 f9efed  
5035 000c0f  
5036 0f080c  
5037 e316e7  
5038 eaf903  
5039 1afef1  
5040 eff0fb  
5041 e9220b  
5042 e9eeed  
5043 0df8ff  
5044 e6eff8  
5045 03faee  
5046 1afb01  
5047 eee707  
5048 f51310  
5049 13fd02  
5050 f5000a  
5051 090c0a  
5052 101611  
5053 09040b  
5054 1d160d  
5055 fef4fb  
5056 1f0708  
5057 0012fa  
5058 01ff0f  
5059 e103f3  
5060 070c12  
5061 02ef03  
5062 150503  
5063 ee12f7  
5064 0b01ec  
5065 dbf308  
5066 1018dd  
5067 1b181a  
5068 e80818  
5069 0f171a  
5070 00f618  
5071 0a1601  
5072 f2da05  
5073 071528  
5074 2b22eb  
5075 f9fb19  
5076 00fd14

5077 f3eb15  
5078 091015  
5079 2217e7  
5080 00ecfe  
5081 f01e20  
5082 1200e3  
5083 00e814  
5084 efe4c6  
5085 fa1529  
5086 f2e2ff  
5087 180afd  
5088 f50dfe  
5089 f9f1dd  
5090 05da10  
5091 1008ea  
5092 0000f9  
5093 000019  
5094 f0110c  
5095 f0f0d2  
5096 09e308  
5097 0500f7  
5098 090df3  
5099 f30cfd  
5100 f00c1a  
5101 0e0e0a  
5102 161d04  
5103 ff08fc  
5104 eb0405  
5105 0ffaf7  
5106 f6ea0e  
5107 f5eeee  
5108 01f109  
5109 1beef  
5110 25402b  
5111 fd12ec  
5112 12340a  
5113 fcf00e  
5114 fd0b0c  
5115 ecf9ec  
5116 faf1fa  
5117 09fdef  
5118 d5dde7  
5119 e9e594  
5120 eff9e1  
5121 e4151d  
5122 fa1b34  
5123 25f5fe  
5124 060fe2  
5125 eaf002  
5126 f70ff3  
5127 f4e9f5  
5128 fdecfe  
5129 f00de5  
5130 18fa11  
5131 06ff18  
5132 190b01  
5133 0bfb09  
5134 28fb18

5135 114330  
5136 140fde  
5137 e81637  
5138 000a24  
5139 e8dce2  
5140 04e706  
5141 dfe9fb  
5142 09f3f0  
5143 04ff0a  
5144 0ffd1c  
5145 16fcf8  
5146 0d1d11  
5147 1d1315  
5148 fcf206  
5149 e00e19  
5150 0412ed  
5151 050b0d  
5152 f70103  
5153 000400  
5154 0109f8  
5155 201305  
5156 01f125  
5157 02f823  
5158 0a0d02  
5159 13fe03  
5160 132e17  
5161 3f2322  
5162 20361b  
5163 0eec16  
5164 081222  
5165 f9f9dd  
5166 fafff6  
5167 0e1902  
5168 171e1a  
5169 f81d0b  
5170 0cef07  
5171 0804f6  
5172 0cf5fb  
5173 fa0200  
5174 f50a06  
5175 0eee0d  
5176 4825f8  
5177 043036  
5178 041617  
5179 0cf91c  
5180 e0e200  
5181 01f3e8  
5182 0f11e6  
5183 fd2eb  
5184 fb1000  
5185 101900  
5186 0cf816  
5187 00f6ff  
5188 f0f0ff  
5189 e0fcfb  
5190 0702f6  
5191 05f8fc  
5192 0e0e00

5193 392c10  
5194 101a21  
5195 110400  
5196 e70615  
5197 01ecfa  
5198 11f900  
5199 f90e01  
5200 e904ee  
5201 ea17ff  
5202 e9f304  
5203 02f5e1  
5204 d0f1f3  
5205 0b0005  
5206 e8f3f5  
5207 09e405  
5208 f5f31a  
5209 100500  
5210 eef7fc  
5211 fd1b0b  
5212 fb1604  
5213 ebfc16  
5214 ea0908  
5215 f6f00f  
5216 fefa11  
5217 f9fcfe  
5218 3c450e  
5219 22d20e  
5220 072808  
5221 f60a06  
5222 03dc03  
5223 eafdfa  
5224 d61122  
5225 fdfef4  
5226 000013  
5227 060c0d  
5228 f2f00d  
5229 f3fff7  
5230 f9fd06  
5231 081409  
5232 db0302  
5233 e0000a  
5234 12ebde  
5235 151aff  
5236 06f814  
5237 0a0a15  
5238 dc0b03  
5239 07ebdb  
5240 dff4f2  
5241 f00400  
5242 05f7f8  
5243 150214  
5244 da1925  
5245 0516c9  
5246 d8ed00  
5247 080310  
5248 f4dbee  
5249 fe021a  
5250 e5ef0e

5251 11fcff  
5252 25f7ef  
5253 ee0125  
5254 1e23e1  
5255 fafedd  
5256 d619fe  
5257 fc1703  
5258 01dde9  
5259 da3116  
5260 05effe  
5261 0a050d  
5262 16e7e3  
5263 f80005  
5264 fd0df0  
5265 f7fee7  
5266 01fc13  
5267 1101f0  
5268 0bdcee  
5269 e2f4fb  
5270 0cfde0  
5271 00f1f0  
5272 f8051c  
5273 1d0a0a  
5274 15130f  
5275 fdeb06  
5276 fe04ff  
5277 0eec06  
5278 04f3f8  
5279 14191a  
5280 1635fe  
5281 0207f1  
5282 faf2fb  
5283 080efb  
5284 06f4e4  
5285 03e6f8  
5286 e5f205  
5287 0d0005  
5288 0e0b1c  
5289 091104  
5290 09ea14  
5291 e5fafb  
5292 eb1705  
5293 1fd6f5  
5294 ed0824  
5295 1033e8  
5296 e2ec23  
5297 fbfc00  
5298 0600e8  
5299 00e5f6  
5300 fc000e  
5301 1004d8  
5302 f5f614  
5303 e40bf4  
5304 01faf6  
5305 fbe911  
5306 fe1af8  
5307 00f6f4  
5308 f10b15

5309 f5effb  
5310 060e00  
5311 e6f804  
5312 f40b0f  
5313 0d0f0e  
5314 0706f1  
5315 f0fdff  
5316 211b12  
5317 f9d8df  
5318 09f8ea  
5319 06f709  
5320 f2fbfe  
5321 0af4fc  
5322 03f3e7  
5323 e900ed  
5324 edfae6  
5325 f4ed07  
5326 edfc0d  
5327 04f510  
5328 1ff0f7  
5329 fc011d  
5330 f303fe  
5331 07170f  
5332 f20100  
5333 dc0f00  
5334 f8dcf5  
5335 1433f8  
5336 f2f9fb  
5337 10dee6  
5338 db0416  
5339 ffe5f9  
5340 e306f6  
5341 ec03e7  
5342 0cfef4  
5343 f4f61b  
5344 1b03fc  
5345 f0f814  
5346 f718ff  
5347 01eae9  
5348 edf7fa  
5349 f2fdef  
5350 00fafd  
5351 f00cf6  
5352 f9041c  
5353 060002  
5354 e100f6  
5355 e5f104  
5356 fa0ffb  
5357 10020b  
5358 f00b0d  
5359 17eedb  
5360 0000f8  
5361 f814f4  
5362 fdfb03  
5363 fdf9fd  
5364 1517ec  
5365 04f61a  
5366 0ed7f7

5367	021411
5368	101216
5369	16e219
5370	ebf714
5371	1727fe
5372	1af207
5373	23122b
5374	f8eb08
5375	111312
5376	f00e09
5377	fffcf0
5378	2c1f1d
5379	1d152b
5380	1807db
5381	fbf50b
5382	f21501
5383	02f1f1
5384	f9fd1c
5385	f5e6eb
5386	a7ff1c
5387	05e286
5388	a6a4f3
5389	0b0bf5
5390	fce7f3
5391	e9d606
5392	1a2100
5393	0394c6
5394	0c1d0c
5395	e1f2dd
5396	e9ff03
5397	090a03
5398	f30315
5399	160fec
5400	f20cfd
5401	e3ea12
5402	0e1f04
5403	0b2808
5404	090018
5405	191a0e
5406	09061f
5407	ea0b1e
5408	eff70d
5409	0b01fb
5410	f3e4fd
5411	ff0603
5412	ff17f8
5413	02f001
5414	0c40ff
5415	1afb1b
5416	efda09
5417	c201db
5418	0106ef
5419	e8f7fd
5420	040bf4
5421	0ef6f8
5422	210d13
5423	0ffe0a
5424	0b11fb

5425 0ae40a  
5426 2408d0  
5427 f52604  
5428 1a14ef  
5429 2f1416  
5430 222f3e  
5431 0f1738  
5432 002b31  
5433 18efe5  
5434 0be229  
5435 141a0e  
5436 f6070c  
5437 20fceb  
5438 fcf322  
5439 f61b08  
5440 fefef8  
5441 0117ff  
5442 21f5eb  
5443 faf622  
5444 0a0408  
5445 101504  
5446 160606  
5447 12200c  
5448 001100  
5449 05ef06  
5450 02f203  
5451 fe1e09  
5452 f906e9  
5453 f6fc08  
5454 0bfef9  
5455 e80c12  
5456 001903  
5457 f611fa  
5458 1bfd06  
5459 f6101a  
5460 080e01  
5461 1d1007  
5462 f714fd  
5463 020507  
5464 2d220e  
5465 160129  
5466 ebe5f6  
5467 fb0bef  
5468 e912fc  
5469 00e8f8  
5470 eff20c  
5471 1024fa  
5472 c9eefd  
5473 0ff5e4  
5474 fddaed  
5475 070219  
5476 040bf0  
5477 1ff8f2  
5478 f3e003  
5479 0c0afe  
5480 070302  
5481 fb0e00  
5482 150009

5483 Obfa11  
5484 Obfc25  
5485 1f01fa  
5486 1106ed  
5487 12dff0  
5488 f91af2  
5489 Ob1200  
5490 Oa0712  
5491 eeeaeef  
5492 210007  
5493 edde07  
5494 000011  
5495 000bf3  
5496 eff704  
5497 f50cf0  
5498 10f10b  
5499 edfe07  
5500 10f20b  
5501 00f9f2  
5502 030dfe  
5503 f80702  
5504 f8f0ef  
5505 070cf9  
5506 f4f1f3  
5507 f0fa07  
5508 fe01ef  
5509 f1f701  
5510 01f2fb  
5511 0510fe  
5512 0efffd  
5513 f7edfd  
5514 f4ff04  
5515 09f50e  
5516 fb0003  
5517 f0f8fb  
5518 05ef06  
5519 0c11f5  
5520 f9f8f2  
5521 ee0dea  
5522 f6e907  
5523 fdf904  
5524 fbf4f5  
5525 030b0c  
5526 f30def  
5527 021009  
5528 0d09f5  
5529 0003ee  
5530 Ob0701  
5531 f50900  
5532 08eef7  
5533 ec0af0  
5534 0703f8  
5535 f6070f  
5536 000600  
5537 0d02fc  
5538 f70d07  
5539 faf404  
5540 0302f4

5541 02fa06  
5542 fdeffd  
5543 f40ef4  
5544 0cf60d  
5545 fc07f7  
5546 fef406  
5547 fdf5f7  
5548 01ea07  
5549 06f00e  
5550 0d0c09  
5551 00efef  
5552 faf6fd  
5553 fff3f7  
5554 fb05f3  
5555 07f1fd  
5556 08fae9  
5557 fbefef  
5558 02f2f4  
5559 0bf0f7  
5560 f4f909  
5561 0af2f4  
5562 070502  
5563 f3f0f8  
5564 01f0ea  
5565 05fb09  
5566 f010f9  
5567 0f01f3  
5568 fc0403  
5569 fb090b  
5570 f2f2f4  
5571 080502  
5572 100bef  
5573 05ffec  
5574 f4fe07  
5575 100100  
5576 fefbee  
5577 0c0400  
5578 ecfe0a  
5579 0a0002  
5580 fdf2f6  
5581 f3f3f6  
5582 09eff2  
5583 f7f109  
5584 f70f0e  
5585 00f3fb  
5586 02ef0f  
5587 0bf9fa  
5588 01f20f  
5589 eeede  
5590 01eff5  
5591 0507ee  
5592 0efcf7  
5593 f2fa00  
5594 010ffc  
5595 fdecf4  
5596 07ed07  
5597 fafd06  
5598 040cf1

5599 05f30f  
5600 0eff07  
5601 f200fc  
5602 ff03f2  
5603 f60b00  
5604 0cf3ff  
5605 08f105  
5606 06fffa  
5607 f000f6  
5608 0aef06  
5609 0ffe06  
5610 f5fdef  
5611 01f0ed  
5612 080505  
5613 07f404  
5614 ea0c07  
5615 fefc0b  
5616 0e0bef  
5617 ee0df7  
5618 ecfbfb  
5619 05f00b  
5620 f8ffed  
5621 fdedfd  
5622 08f703  
5623 f50ff0  
5624 fbfafe  
5625 f4f0f3  
5626 f9f400  
5627 000c01  
5628 000006  
5629 17faf4  
5630 01fbf5  
5631 fa0c12  
5632 f513ff  
5633 020d03  
5634 f101ee  
5635 090720  
5636 1a2c0f  
5637 24090a  
5638 0d040e  
5639 172200  
5640 090b06  
5641 1a1212  
5642 fdff00  
5643 0026ff  
5644 071800  
5645 f312f4  
5646 eccfce  
5647 fc1cd3  
5648 ecbbc4  
5649 200804  
5650 e51403  
5651 1a11f1  
5652 ddf3f6  
5653 ffe8d2  
5654 160329  
5655 0ee924  
5656 e7f30a

5657 33fd1f  
5658 03e3db  
5659 f00e06  
5660 0ef818  
5661 fafcf0  
5662 f8fe18  
5663 131bf7  
5664 1510e6  
5665 0c031f  
5666 e30f06  
5667 0917ea  
5668 fef2f1  
5669 08ff0d  
5670 ef120c  
5671 d9eafd  
5672 cb21fb  
5673 18fea8  
5674 1907f6  
5675 eb321e  
5676 fd1704  
5677 ece8de  
5678 01ebd6  
5679 efe0fa  
5680 d80001  
5681 cb07db  
5682 1029d8  
5683 f5d7d6  
5684 faef05  
5685 d912f1  
5686 e1f2dd  
5687 dcdffe  
5688 111916  
5689 0e03ef  
5690 13d21e  
5691 d3d503  
5692 070003  
5693 0f0cdf  
5694 240ce4  
5695 f6ea1d  
5696 ddf4ec  
5697 0a09eb  
5698 e2de0d  
5699 c9143b  
5700 e91bdf  
5701 13f505  
5702 0ae018  
5703 e1f712  
5704 0a0012  
5705 0301fa  
5706 181a1d  
5707 e4e405  
5708 fc08f1  
5709 1dfcf8  
5710 20faf9  
5711 fb0415  
5712 001400  
5713 ebdef5  
5714 08f112

5715 ef10ff  
5716 1d0ffb  
5717 e8e9fe  
5718 daf007  
5719 faeef7  
5720 1003ff  
5721 0e161e  
5722 24060e  
5723 0d1012  
5724 18ecef  
5725 081107  
5726 2113f6  
5727 0a2001  
5728 04f308  
5729 deff05  
5730 000804  
5731 fd131c  
5732 05f9ef  
5733 fd05fb  
5734 d1f1ee  
5735 fe02fd  
5736 12faf5  
5737 29faec  
5738 f5f526  
5739 12150d  
5740 f11503  
5741 19e9e0  
5742 130709  
5743 050822  
5744 052c0c  
5745 fbf809  
5746 0bf9fa  
5747 fefa08  
5748 f40517  
5749 0c1d0e  
5750 13ec0b  
5751 effaf5  
5752 0700f9  
5753 0904fd  
5754 b8e01a  
5755 3913d8  
5756 05d407  
5757 280a21  
5758 fa090b  
5759 1107f7  
5760 02e504  
5761 fe1008  
5762 000007  
5763 000511  
5764 ef000a  
5765 fafdea  
5766 00ef01  
5767 02faed  
5768 d5f009  
5769 fcf502  
5770 080102  
5771 cdf312  
5772 1d130b

5773 00dff3  
5774 fe0b0b  
5775 06f0e8  
5776 e8020b  
5777 ecf4e0  
5778 fd0300  
5779 24f3ef  
5780 173822  
5781 f001e5  
5782 d8e70b  
5783 f0e710  
5784 f1f8f1  
5785 e5f1f3  
5786 f6e9de  
5787 07e4f2  
5788 cff5dc  
5789 00e7ca  
5790 f3f70a  
5791 001d14  
5792 190bf7  
5793 0a0a1d  
5794 111a09  
5795 0b2728  
5796 00e70c  
5797 f4fbf8  
5798 1e02fb  
5799 ff0c1b  
5800 031608  
5801 1400e5  
5802 fefef1  
5803 141ffb  
5804 1bdcff  
5805 1e2828  
5806 050f04  
5807 f8041d  
5808 01fd00  
5809 e30c15  
5810 0603f3  
5811 f6fdf4  
5812 0af904  
5813 140b0a  
5814 0c0c06  
5815 f2f3f3  
5816 030802  
5817 e1ee08  
5818 fbea0b  
5819 f4fdda  
5820 e8f6da  
5821 0d02f4  
5822 f90f14  
5823 220af5  
5824 feee1e  
5825 02121c  
5826 f8efe3  
5827 caf9ea  
5828 e91403  
5829 1afeee  
5830 0e071e

5831 14ea16  
5832 f21d12  
5833 ce0d13  
5834 eee8e0  
5835 f600ee  
5836 f303cb  
5837 c0e9e8  
5838 18fd0c  
5839 e10e08  
5840 f4f5ed  
5841 e200fd  
5842 f505f7  
5843 0efb02  
5844 0af3f4  
5845 ea02fc  
5846 1cf400  
5847 f43847  
5848 f51313  
5849 e00002  
5850 fdf6f1  
5851 fef0ee  
5852 f3fdfe  
5853 e1dee9  
5854 fcebfd  
5855 fb11fb  
5856 d9fa0a  
5857 0efe17  
5858 0af702  
5859 f0f206  
5860 02f2f4  
5861 0c08f3  
5862 11fc0c  
5863 252513  
5864 140a22  
5865 f402ee  
5866 fbef12  
5867 eaf1df  
5868 0df5fd  
5869 ea0ce6  
5870 e4e0fb  
5871 e908d7  
5872 0c0ad6  
5873 efe0e7  
5874 e703f6  
5875 00eef8  
5876 f70111  
5877 081bed  
5878 fe040c  
5879 0f1119  
5880 05f8f0  
5881 f4f705  
5882 1e1310  
5883 1ceaf2  
5884 fceff8  
5885 fc04f2  
5886 0e0d01  
5887 fa0017  
5888 3a1606

5889 0de70a  
5890 f4f0f8  
5891 f7e612  
5892 faedf0  
5893 ed02ff  
5894 daf5f0  
5895 effdf9  
5896 0000fb  
5897 e40109  
5898 eb030e  
5899 f0fcfb  
5900 00fd05  
5901 010311  
5902 04e40c  
5903 f60000  
5904 03e9f4  
5905 1c0f04  
5906 031812  
5907 fc16fe  
5908 e7f202  
5909 0f20ec  
5910 020a12  
5911 2a1b23  
5912 09e907  
5913 2715fc  
5914 d71131  
5915 0324c4  
5916 bfcee8  
5917 fffe11  
5918 00fdf9  
5919 170d28  
5920 f900fa  
5921 00efdb  
5922 f9e4de  
5923 e01a23  
5924 dec5f9  
5925 1002f4  
5926 c6c6a2  
5927 c6f9de  
5928 0c1004  
5929 15d8e0  
5930 01f310  
5931 0bfaf7  
5932 18fce7  
5933 e91227  
5934 f90e19  
5935 0f0aee  
5936 f408f6  
5937 f810f3  
5938 2d260d  
5939 d90a21  
5940 e91df4  
5941 15fce5  
5942 08fb00  
5943 052026  
5944 0b2830  
5945 f6f516  
5946 13f7f9

5947 ecf2f1  
5948 e3f6f9  
5949 ebd8df  
5950 f80ce3  
5951 00e7e6  
5952 040719  
5953 0bf5e8  
5954 f4faef  
5955 e0f7d0  
5956 00e8fa  
5957 170102  
5958 191436  
5959 f80112  
5960 f8f10e  
5961 ee0701  
5962 0d1af2  
5963 0ad1f6  
5964 e6ff2e  
5965 1a18f1  
5966 d8fa1c  
5967 27262b  
5968 2a2b33  
5969 27271a  
5970 3222fa  
5971 c4d300  
5972 14fe03  
5973 e3edf3  
5974 120aef  
5975 f8fd03  
5976 1e150f  
5977 eff0f3  
5978 f4f500  
5979 ebf000  
5980 1c3cfc  
5981 e902fd  
5982 e71222  
5983 010d04  
5984 2a2c1c  
5985 212610  
5986 1d2122  
5987 e3eaff  
5988 1a14f9  
5989 fe9f5  
5990 fde70f  
5991 11f909  
5992 1602ef  
5993 f407fd  
5994 1f0dee  
5995 f5fc21  
5996 1a0b0a  
5997 f1ff0a  
5998 fd16fb  
5999 e70313  
6000 2416fa  
6001 292d20  
6002 f50c23  
6003 08180c  
6004 f4fc0d

6005 f303d9  
6006 0808fb  
6007 1ce0f4  
6008 fe1410  
6009 0003dc  
6010 f9fafd  
6011 f204e8  
6012 e6d904  
6013 0c0d05  
6014 27edec  
6015 f2f605  
6016 f516fd  
6017 1af9ef  
6018 faf40d  
6019 110bf5  
6020 11060b  
6021 061e00  
6022 d31842  
6023 3f02c0  
6024 f7f9eb  
6025 192d05  
6026 dd12f7  
6027 fb231e  
6028 191cff  
6029 0adfff  
6030 000007  
6031 f409fc  
6032 0e0c08  
6033 fbf206  
6034 0e04ef  
6035 fd18f4  
6036 ea030e  
6037 f912f9  
6038 00e3f8  
6039 0e0e05  
6040 01edff  
6041 04120f  
6042 2bf40c  
6043 041a2d  
6044 091110  
6045 19f9f6  
6046 03f6fd  
6047 edf6ee  
6048 1bfa0f  
6049 01011c  
6050 27f0ef  
6051 11102c  
6052 31100f  
6053 03081b  
6054 fe0ecd  
6055 d6dd01  
6056 aee6f4  
6057 00db89  
6058 abb908  
6059 fb0809  
6060 0ff5f3  
6061 0fc7dd  
6062 f92a46

6063 3c2303  
6064 f20d10  
6065 fb0300  
6066 06000d  
6067 eafdfc  
6068 0205f2  
6069 e2ebd7  
6070 d8f7ef  
6071 03f6fc  
6072 f92727  
6073 192217  
6074 01eb0f  
6075 2e07fe  
6076 350e04  
6077 210b0e  
6078 dd0210  
6079 000dca  
6080 e4dbf0  
6081 180c13  
6082 eff90e  
6083 ebeef0  
6084 f3f309  
6085 0afcd0  
6086 08102c  
6087 34f4f3  
6088 043029  
6089 0e0401  
6090 100eff  
6091 11deeb  
6092 e0f608  
6093 ecffec  
6094 fafbd9  
6095 e2ff0c  
6096 332607  
6097 e5120a  
6098 1b3101  
6099 efe711  
6100 2f2619  
6101 1b3619  
6102 33090e  
6103 fd302d  
6104 3541dc  
6105 bbd106  
6106 1906fe  
6107 fc0401  
6108 051a1a  
6109 f4fffb  
6110 07e8fd  
6111 f4f218  
6112 e4f8e8  
6113 0af00a  
6114 25060d  
6115 17181d  
6116 190d1b  
6117 182a02  
6118 160c15  
6119 0d21f0  
6120 e7e1ea

6121 f7ffea  
6122 fdf2d7  
6123 f5ef03  
6124 001104  
6125 020d05  
6126 15160b  
6127 0f170b  
6128 110202  
6129 f6f90f  
6130 0d0c07  
6131 1d080d  
6132 fc1a10  
6133 091dfb  
6134 08ee12  
6135 fee6f3  
6136 0d2412  
6137 1afce0  
6138 cbed0c  
6139 f3d6ea  
6140 0c0500  
6141 ed1f0e  
6142 0bfa10  
6143 19fff3  
6144 dff506  
6145 0b11fa  
6146 f0ccff  
6147 090cfc  
6148 ebed02  
6149 f7f9e5  
6150 00150d  
6151 f70dfa  
6152 ebf0c3  
6153 0e07e5  
6154 0aec16  
6155 061316  
6156 0b1efe  
6157 02dbfd  
6158 021512  
6159 122f01  
6160 f8ff04  
6161 f20b25  
6162 20e9fc  
6163 f0e408  
6164 0000f1  
6165 0f14ff  
6166 f0e719  
6167 fb0000  
6168 e2e500  
6169 041204  
6170 d009f2  
6171 000fe0  
6172 f4e2da  
6173 f10306  
6174 0118f3  
6175 0efef9  
6176 fd0810  
6177 edf1d5  
6178 e7f5ed

6179 ff06e3  
6180 edf901  
6181 3110f3  
6182 194641  
6183 ff07e7  
6184 e5231b  
6185 f303f0  
6186 1ff300  
6187 ef1c00  
6188 14f712  
6189 2e3521  
6190 fe03e9  
6191 ff15fb  
6192 041017  
6193 f11d19  
6194 2c1f30  
6195 3d00ef  
6196 fc2825  
6197 28672c  
6198 fafce6  
6199 efea11  
6200 0cf4e5  
6201 f3eff4  
6202 18f80a  
6203 f6ee0a  
6204 070004  
6205 14f9e8  
6206 25d6f5  
6207 04fb0b  
6208 f802e6  
6209 f5ff02  
6210 f31a0d  
6211 19f3fa  
6212 1a05fa  
6213 09f51f  
6214 1bfd1e  
6215 0af6f3  
6216 22fd15  
6217 071431  
6218 03021b  
6219 0f121e  
6220 1cfaf5  
6221 38e6f8  
6222 ffd920  
6223 07f305  
6224 e2f3f3  
6225 001716  
6226 17f5eb  
6227 ecffe7  
6228 15e615  
6229 e7cafb  
6230 bf2235  
6231 20e9b7  
6232 112210  
6233 1c2bf5  
6234 02040a  
6235 f7daed  
6236 f7ebf0

6237 d4f9fa  
6238 ff3adf  
6239 f4e1f6  
6240 110d07  
6241 eceb0a  
6242 1001fc  
6243 0af8f7  
6244 071605  
6245 e8ee05  
6246 e0001f  
6247 08e6dc  
6248 2221e2  
6249 fa1d0a  
6250 00ff0f  
6251 13f80f  
6252 00ee15  
6253 111e08  
6254 2e0f07  
6255 f4f5dd  
6256 f707ec  
6257 fe0002  
6258 f6f80a  
6259 051103  
6260 fbfe05  
6261 f502f7  
6262 eceff2  
6263 f9df04  
6264 f006ee  
6265 171502  
6266 f9f310  
6267 0a0d04  
6268 eff9ed  
6269 090505  
6270 0016ed  
6271 de1804  
6272 e4dde9  
6273 f400f3  
6274 0a00f1  
6275 f6e1f4  
6276 f20afa  
6277 fdfcfe  
6278 0612fc  
6279 0700e1  
6280 ede021  
6281 ebf517  
6282 1007f5  
6283 0808f6  
6284 081300  
6285 00fc13  
6286 00fe06  
6287 f900ed  
6288 13f8f0  
6289 f6e9f3  
6290 11061d  
6291 010009  
6292 fa09ea  
6293 08e2f5  
6294 16030d

6295 14180c  
6296 e70c33  
6297 230de3  
6298 000008  
6299 f30001  
6300 f7edec  
6301 f0ebf8  
6302 f3f5fa  
6303 ebf7f5  
6304 f7f0f1  
6305 f3ee00  
6306 f0f901  
6307 f2f8f2  
6308 04f8f7  
6309 eef006  
6310 f9f0f2  
6311 e7f8ed  
6312 090906  
6313 f7ff0e  
6314 0004f5  
6315 fbfffd  
6316 080afb  
6317 03e605  
6318 fef8f1  
6319 e6eaf2  
6320 f9f4f7  
6321 f608f9  
6322 0aef01  
6323 f9f0f4  
6324 fc1109  
6325 fde9ff  
6326 fc02fa  
6327 e8f3f4  
6328 f90df5  
6329 f8ff00  
6330 00ecf7  
6331 02f9f4  
6332 f504fb  
6333 0beff7  
6334 fef7e9  
6335 000406  
6336 e7f6ec  
6337 f502fe  
6338 f8f5ec  
6339 ef0402  
6340 0516fd  
6341 eaaff5  
6342 f8e903  
6343 fafde8  
6344 faf9e7  
6345 f603fa  
6346 ec00f2  
6347 f8f9f6  
6348 03faff  
6349 ededea  
6350 ebec08  
6351 04e809  
6352 0102fc

6353 15fbf5  
6354 f6f709  
6355 eb0104  
6356 000bf7  
6357 f3f3ee  
6358 02eef0  
6359 f6e9ff  
6360 f001f5  
6361 0609e7  
6362 e6f1ee  
6363 eaf1f6  
6364 f20908  
6365 090b01  
6366 f809ef  
6367 00f703  
6368 fde6e6  
6369 fd07fe  
6370 070902  
6371 06efef  
6372 00e9ed  
6373 f41ae9  
6374 f1f4f5  
6375 00f5f9  
6376 f0ffe8  
6377 f5ff04  
6378 fbeefb  
6379 f80ff3  
6380 e5faf2  
6381 02e7f5  
6382 f1eb10  
6383 0fe9e9  
6384 f101f1  
6385 f6f6f3  
6386 ecf5f2  
6387 f303f3  
6388 0b0c0a  
6389 e8f7f8  
6390 fdebe9  
6391 f0fc02  
6392 ffff04  
6393 05e9f4  
6394 0503f4  
6395 ff09f8  
6396 eafbf2  
6397 01e709  
6398 f9f9ee  
6399 f904fb  
6400 07f5fe  
6401 f7f7e6  
6402 f6f503  
6403 0102ee  
6404 00f6f7  
6405 fde914  
6406 e5f9f4  
6407 f307f0  
6408 04f3eb  
6409 f4f707  
6410 0300ed

6411 fcedff  
6412 05f90b  
6413 09f705  
6414 02f010  
6415 e6f7fd  
6416 e9fcea  
6417 f9ece9  
6418 f5ec07  
6419 09eef2  
6420 e908fb  
6421 f4ff00  
6422 0002fd  
6423 00e8f1  
6424 f5e4ff  
6425 0912fd  
6426 e5f1fd  
6427 f6f601  
6428 f50608  
6429 09f201  
6430 f2fcfa  
6431 07fdeb  
6432 0000ee  
6433 0aeff1  
6434 fc0f13  
6435 09fcfc  
6436 070401  
6437 fd0009  
6438 00f501  
6439 f5faf2  
6440 10f9f7  
6441 f3f80e  
6442 00f1f0  
6443 ed0c0e  
6444 f2f3f6  
6445 ec0509  
6446 ec060d  
6447 f9f2f6  
6448 0e0f04  
6449 0becf3  
6450 fcf0fa  
6451 07f8f0  
6452 0609f4  
6453 f5faf7  
6454 fa0b08  
6455 07f0fd  
6456 fd0a04  
6457 00f50c  
6458 f9eef6  
6459 f1f3ee  
6460 fd0af0  
6461 fc0e08  
6462 1005f0  
6463 fef305  
6464 f7efef  
6465 f8faff  
6466 0f0ded  
6467 f7ef01  
6468 f9fe09

6469 f30cfb  
6470 fdeeed  
6471 0ff8fb  
6472 0af210  
6473 0704f5  
6474 040f0a  
6475 fdeff0  
6476 f006fe  
6477 000b0d  
6478 f4fbef  
6479 ec05f4  
6480 fd0df1  
6481 0aeffc  
6482 fff7fa  
6483 f608f3  
6484 f70300  
6485 07ff10  
6486 f8f6fc  
6487 090ef5  
6488 f6f508  
6489 0b00f7  
6490 ef00f2  
6491 f1f2fb  
6492 f206ed  
6493 03ee05  
6494 030b00  
6495 09f8fe  
6496 f3f2fc  
6497 f1f5ff  
6498 fcfbfc  
6499 f703fa  
6500 f9fbf3  
6501 f2fffa  
6502 f905ee  
6503 fef5f4  
6504 0f0bfb  
6505 0d0cf4  
6506 01f4fb  
6507 0500f6  
6508 00050f  
6509 fcf02  
6510 0e0d02  
6511 f80900  
6512 090002  
6513 fff0f9  
6514 0eee10  
6515 fe09ec  
6516 ee02f7  
6517 01fbee  
6518 06fbf3  
6519 0cf04  
6520 08fb03  
6521 04ef0a  
6522 fef8ef  
6523 f408f1  
6524 0f07f5  
6525 050f03  
6526 f2ef09

6527 0c09ee  
6528 09faf1  
6529 f2ef08  
6530 0bef09  
6531 010900  
6532 0e0409  
6533 05f106  
6534 04fbf5  
6535 f2f90b  
6536 ececf7  
6537 090400  
6538 f4f700  
6539 0c06f7  
6540 01f2fb  
6541 09fc0c  
6542 02f401  
6543 0eef01  
6544 0b00f8  
6545 ecf908  
6546 fc0af9  
6547 ed0205  
6548 0502f7  
6549 03070c  
6550 09edf6  
6551 07fb05  
6552 ed1111  
6553 f70a05  
6554 faf2ff  
6555 f0ef00  
6556 f5f9ec  
6557 050e09  
6558 14fd03  
6559 0bf0ee  
6560 f5fdf7  
6561 f00705  
6562 00f5f3  
6563 ee07ef  
6564 1002f2  
6565 eff6fe  
6566 000000  
6567 fd1616  
6568 fb0dff  
6569 080f1a  
6570 2915e8  
6571 f40506  
6572 00edf6  
6573 ddf819  
6574 1a0de5  
6575 130c23  
6576 0700fe  
6577 241f2d  
6578 2d08ef  
6579 0e3253  
6580 21f204  
6581 051efb  
6582 f3f6e0  
6583 f8fd0e  
6584 09d7e7

6585 00fd2a  
6586 19edfb  
6587 0c121e  
6588 091efe  
6589 f70e01  
6590 000fed  
6591 cdd0e3  
6592 dff136  
6593 10f5fb  
6594 08cb03  
6595 180a0e  
6596 f9e2f3  
6597 ff0605  
6598 140a16  
6599 2a1f01  
6600 1f161d  
6601 0e00f1  
6602 fef405  
6603 13141b  
6604 f5200a  
6605 06f5eb  
6606 e2e318  
6607 140c02  
6608 05f607  
6609 12fef7  
6610 f5e722  
6611 2606f5  
6612 12fcf5  
6613 053835  
6614 f60b1d  
6615 e2eefc  
6616 e4e8e5  
6617 1e0506  
6618 e5142b  
6619 f2f2c8  
6620 f9dfd9  
6621 eefaec  
6622 dad60a  
6623 ff0fe8  
6624 ee13e8  
6625 fb03ed  
6626 101ef7  
6627 12f303  
6628 e01c1a  
6629 0f0d03  
6630 0009f6  
6631 f2fcef  
6632 fa0f08  
6633 02fbff  
6634 0be9e1  
6635 e0f424  
6636 e7fdfe  
6637 f117f4  
6638 17fa12  
6639 04f320  
6640 f9fde8  
6641 d4e4e8  
6642 fc00fd

6643 0d0cfa  
6644 f3fb1a  
6645 00fefafa  
6646 f30d0d  
6647 1a09f2  
6648 00df0f  
6649 f90bf6  
6650 00eafb  
6651 2e01ff  
6652 faead1  
6653 161703  
6654 1c161a  
6655 05101d  
6656 dc110e  
6657 0001e7  
6658 0cd8ee  
6659 0e0efe  
6660 172006  
6661 050720  
6662 0e1013  
6663 0cfc03  
6664 0f0700  
6665 03f100  
6666 001316  
6667 13fb02  
6668 e5180a  
6669 fb11fc  
6670 0af31a  
6671 ea060f  
6672 ff0913  
6673 f014f4  
6674 e2e5f9  
6675 130405  
6676 f6090c  
6677 0b3b20  
6678 18fdfd  
6679 111c20  
6680 0cff12  
6681 0f0f15  
6682 ffeffe  
6683 1bfe17  
6684 eef521  
6685 120407  
6686 1b060b  
6687 22fbff  
6688 020c0b  
6689 f5f512  
6690 0409ed  
6691 f3140a  
6692 0bf9ed  
6693 e319f9  
6694 07f00f  
6695 143918  
6696 ea070a  
6697 e80030  
6698 ffd3ef  
6699 04f6ef  
6700 0000df

6701 061f20  
6702 06e9ff  
6703 03fa14  
6704 f9ede4  
6705 fdf405  
6706 d304d5  
6707 e8e3d6  
6708 e0ebe4  
6709 021dd6  
6710 eb0408  
6711 112212  
6712 e7000b  
6713 f0f6da  
6714 e8c90a  
6715 f003f2  
6716 0fdae1  
6717 2300f4  
6718 1d1c31  
6719 f413fa  
6720 2c1c08  
6721 fbe90f  
6722 211f1d  
6723 24fbfe  
6724 f4eb13  
6725 07fcf9  
6726 280ded  
6727 f63d24  
6728 1b4205  
6729 fb12fa  
6730 f04921  
6731 322200  
6732 19120a  
6733 024838  
6734 0303f2  
6735 260610  
6736 05ecef  
6737 faf8ff  
6738 ff12fe  
6739 0deded  
6740 f2fc00  
6741 0216ff  
6742 f8caef  
6743 e2f5ea  
6744 010bf8  
6745 24270b  
6746 ec0502  
6747 201404  
6748 1c0ef3  
6749 d6df21  
6750 01f2f3  
6751 1bf9f3  
6752 171320  
6753 163c30  
6754 3226f0  
6755 00f009  
6756 00f510  
6757 f6ec02  
6758 14ffcf

6759 f3fbf7  
6760 091a06  
6761 0406f3  
6762 f60cf3  
6763 e90015  
6764 07080d  
6765 cbdbed  
6766 d60020  
6767 16d2db  
6768 161d14  
6769 0e2b19  
6770 fb0a11  
6771 07f5f9  
6772 040df9  
6773 e9dfd3  
6774 dbfcf6  
6775 ee00d2  
6776 0e0a0c  
6777 f81400  
6778 0f06ea  
6779 09fe0e  
6780 f205fc  
6781 f6dde0  
6782 c70018  
6783 e0fdee  
6784 f912e5  
6785 0012f6  
6786 150a01  
6787 160608  
6788 0d00f8  
6789 f7110f  
6790 051708  
6791 cad0b2  
6792 02dff1  
6793 090515  
6794 1111ff  
6795 f8f311  
6796 f2f2f3  
6797 d610ea  
6798 feff00  
6799 f6e8fe  
6800 e8ec05  
6801 07fbf8  
6802 1010f2  
6803 1b0d0d  
6804 011f01  
6805 06ffe0  
6806 fffafc  
6807 bff302  
6808 edcfb7  
6809 1530cb  
6810 06f901  
6811 0b0a01  
6812 d5f11c  
6813 0300f2  
6814 14eeee  
6815 fa02ff  
6816 280afa

6817 f5f115  
6818 f6f9f3  
6819 ff0dfa  
6820 0209fa  
6821 fb080f  
6822 e4f6ed  
6823 001809  
6824 1008fb  
6825 dce309  
6826 11101d  
6827 f70e0d  
6828 10100c  
6829 f9001a  
6830 001508  
6831 f5ffff  
6832 bcea1a  
6833 08e7db  
6834 0000fa  
6835 001107  
6836 f91c0a  
6837 03f1f3  
6838 0bff1a  
6839 f214f0  
6840 071518  
6841 31f6f3  
6842 f92518  
6843 eaf2f7  
6844 230e00  
6845 000900  
6846 f91b0e  
6847 1dfd10  
6848 10f201  
6849 010910  
6850 091a20  
6851 fcedf4  
6852 28fee9  
6853 0ff04b  
6854 130804  
6855 e7ebf5  
6856 f50405  
6857 f5d9f5  
6858 f106fc  
6859 04e9fd  
6860 db0108  
6861 1bc0b5  
6862 dffb13  
6863 10151f  
6864 31141f  
6865 f8e81f  
6866 032de5  
6867 eda6da  
6868 fe0820  
6869 f0ea04  
6870 08130f  
6871 19efe6  
6872 f30002  
6873 0f1520  
6874 1d220e

6875 f307f7  
6876 f50003  
6877 322e01  
6878 2eeb18  
6879 eb1918  
6880 f2e9ff  
6881 f4dddc  
6882 ebe400  
6883 f508fc  
6884 0afff1  
6885 ff261b  
6886 f01c15  
6887 2d07ff  
6888 efe322  
6889 f11c02  
6890 f0f1df  
6891 e5d51c  
6892 0d280e  
6893 231ae7  
6894 020a0a  
6895 e2ec05  
6896 faf2e2  
6897 fc080d  
6898 0b00fa  
6899 011e0f  
6900 3f0cdb  
6901 031b1c  
6902 1700fd  
6903 02e530  
6904 392c0a  
6905 00ffe7  
6906 ed0b05  
6907 ecebe2  
6908 f3e4ce  
6909 f81811  
6910 0cffffb  
6911 041106  
6912 0008f6  
6913 f216fd  
6914 f0e2f5  
6915 0015eb  
6916 2201eb  
6917 180422  
6918 08ea1e  
6919 110d25  
6920 0600f4  
6921 fff20d  
6922 fbf0fd  
6923 f6f2ee  
6924 e7e3fc  
6925 1d1613  
6926 0311f6  
6927 00fe09  
6928 fc110a  
6929 041714  
6930 f41812  
6931 fb050d  
6932 f3160c

6933 0e0505  
6934 f40e04  
6935 1508ed  
6936 f2f405  
6937 10f70e  
6938 f3de17  
6939 ecdaf0  
6940 ebeaf1  
6941 08e3dc  
6942 231425  
6943 08020b  
6944 0f1902  
6945 0003f4  
6946 f9f6fe  
6947 f50d19  
6948 140304  
6949 08ee0c  
6950 fa111e  
6951 1f04f2  
6952 fcf50a  
6953 09fa00  
6954 060b11  
6955 f71313  
6956 f60917  
6957 030dff  
6958 f40410  
6959 0503f8  
6960 2f1bdd  
6961 d3fc3f  
6962 1af122  
6963 00f5f5  
6964 07eb01  
6965 e1fff1  
6966 050303  
6967 eff90e  
6968 0000f5  
6969 0f0018  
6970 fb1202  
6971 f6f2f5  
6972 fbf003  
6973 f1fdfa  
6974 0af2ec  
6975 21f906  
6976 04152a  
6977 fefc00  
6978 150008  
6979 11fef1  
6980 db0df5  
6981 1414e4  
6982 05f107  
6983 0c092b  
6984 0f0013  
6985 02f0f4  
6986 f41b1e  
6987 f617fa  
6988 23fae8  
6989 eeecf0  
6990 e62a0e

6991 01f312  
6992 ee04e5  
6993 0fedf8  
6994 d300e9  
6995 f2fad1  
6996 d8b8fa  
6997 001e09  
6998 c8f1e8  
6999 d9c00e  
7000 ee0af6  
7001 c8abc7  
7002 f4fb0a  
7003 09f703  
7004 08f8ee  
7005 fcfceb  
7006 ede322  
7007 010a0d  
7008 0b0ef9  
7009 fce60d  
7010 003c0a  
7011 24ff08  
7012 060deb  
7013 10120f  
7014 faf5f8  
7015 e40915  
7016 00ed09  
7017 05fa04  
7018 1507f4  
7019 fcfff8  
7020 06f502  
7021 08f903  
7022 fc0aff  
7023 2813e7  
7024 fbe706  
7025 e51704  
7026 090b1f  
7027 11f7ec  
7028 f1f3fa  
7029 e203fc  
7030 26fbf1  
7031 0bfa12  
7032 f6f30d  
7033 f517fd  
7034 3805f9  
7035 0d0f45  
7036 021613  
7037 180810  
7038 223302  
7039 eaff00  
7040 004b34  
7041 18fef1  
7042 23f908  
7043 f5201a  
7044 080514  
7045 0bf405  
7046 f8ee05  
7047 f9f209  
7048 fee911

7049 f905f1  
7050 33f5f9  
7051 07000a  
7052 1a1dff  
7053 121b0a  
7054 08fb26  
7055 0b0d04  
7056 1214f8  
7057 fef80b  
7058 05fbfc  
7059 00f7f5  
7060 1011ee  
7061 fb080e  
7062 0fee1a  
7063 09eb11  
7064 12f8f5  
7065 fc0602  
7066 f709fc  
7067 0d251d  
7068 17f8f6  
7069 0b1f1e  
7070 000509  
7071 001309  
7072 0405f0  
7073 fc04ee  
7074 dde309  
7075 171714  
7076 0d0529  
7077 0aeeff  
7078 fe0816  
7079 0df5f2  
7080 fbf8f2  
7081 ec0dfc  
7082 f405f3  
7083 ece014  
7084 f81d37  
7085 09eee5  
7086 f802f3  
7087 10f813  
7088 12fe0a  
7089 e30115  
7090 1218f3  
7091 16e7fa  
7092 07f614  
7093 04fa09  
7094 e61516  
7095 11f0fe  
7096 f1f4fe  
7097 03f8ee  
7098 f00e27  
7099 1becdf  
7100 0619e9  
7101 02e0e8  
7102 000007  
7103 091113  
7104 0a0eee  
7105 fd230b  
7106 13de0e

7107 0a0102  
7108 d60cff  
7109 dcf1ed  
7110 00c9df  
7111 102d19  
7112 f7fb00  
7113 071f1e  
7114 070611  
7115 f000fa  
7116 f2ff0f  
7117 fce8ec  
7118 ebe6f4  
7119 ff200b  
7120 524828  
7121 01f231  
7122 fe380d  
7123 f505f7  
7124 00e60a  
7125 04dbf7  
7126 fdf6ee  
7127 eaf6fa  
7128 1011fc  
7129 ed0b23  
7130 342f0e  
7131 1010db  
7132 f2402f  
7133 1e0fe1  
7134 e5f70a  
7135 f72119  
7136 0a0501  
7137 1501f8  
7138 f0f10b  
7139 021700  
7140 1802fb  
7141 04fb04  
7142 01fe01  
7143 0cf103  
7144 06f931  
7145 2d1b16  
7146 f2e50a  
7147 000711  
7148 fc0700  
7149 0cf1f8  
7150 f2dcf1  
7151 fcef09  
7152 ea04f7  
7153 fefd05  
7154 120316  
7155 49291c  
7156 fd0b10  
7157 eb1d21  
7158 120e1d  
7159 edd4ed  
7160 d3fae3  
7161 f900ee  
7162 1918f3  
7163 ebf701  
7164 060107

7165 0200f8  
7166 f8181b  
7167 e1f7f6  
7168 f2e812  
7169 24f2d9  
7170 1d0523  
7171 2d241e  
7172 002130  
7173 0aebe2  
7174 10ec0b  
7175 e5c7dd  
7176 cae8dd  
7177 dceccf  
7178 03fc0b  
7179 0807fd  
7180 060d03  
7181 e8fe0f  
7182 06f8e8  
7183 03db00  
7184 e50f1a  
7185 f4eaed  
7186 06f9f5  
7187 ff0f19  
7188 00171b  
7189 180502  
7190 0ce5fa  
7191 f4dc13  
7192 0bfaf9  
7193 f3daea  
7194 23e4df  
7195 f70001  
7196 101708  
7197 fcf1f8  
7198 ebf612  
7199 f005ef  
7200 ffe2e9  
7201 ed040a  
7202 0ff0f4  
7203 fafa0b  
7204 16eeff  
7205 171f17  
7206 d5fb01  
7207 0002fb  
7208 f102df  
7209 f2f5f1  
7210 cee3f9  
7211 2f4bd6  
7212 f00111  
7213 2b1322  
7214 f208f4  
7215 d11124  
7216 0f12ee  
7217 10d6d5  
7218 fafcf0  
7219 fe17fe  
7220 07f316  
7221 040c12  
7222 1af1f4

7223 0cf819  
7224 001a0d  
7225 06e6eb  
7226 ec1b0e  
7227 eb09f1  
7228 2008f8  
7229 1ffa11  
7230 f62213  
7231 f3fee1  
7232 0005f1  
7233 fddfee  
7234 e4150c  
7235 eff4e5  
7236 00000b  
7237 0ffe2a  
7238 f70b0e  
7239 fd0eee  
7240 0ae703  
7241 fd0300  
7242 0f0400  
7243 100fe7  
7244 f0fetc  
7245 0a10fd  
7246 002401  
7247 130005  
7248 d911ff  
7249 10e7cf  
7250 ebf5f0  
7251 1f0a19  
7252 1904ec  
7253 2a04f9  
7254 d81c2d  
7255 032896  
7256 cfd109  
7257 0a030e  
7258 06e6fc  
7259 050706  
7260 ea0900  
7261 f40a04  
7262 08dfde  
7263 e81c30  
7264 0b09fd  
7265 e41cfe  
7266 c1e9e8  
7267 04120b  
7268 f1dff4  
7269 df1109  
7270 e3f6fb  
7271 081710  
7272 0a0de2  
7273 f31d1c  
7274 140312  
7275 f0fdf2  
7276 ff0c00  
7277 0313ec  
7278 37fa11  
7279 cad91c  
7280 0d2fcc

7281 03f900  
7282 0c160a  
7283 0d0318  
7284 1b0501  
7285 e9ff04  
7286 1c0809  
7287 f910fa  
7288 1b050d  
7289 e400fe  
7290 0124fa  
7291 05e3f9  
7292 eef103  
7293 14fae9  
7294 10f1f5  
7295 e7f6db  
7296 fa0815  
7297 070702  
7298 111400  
7299 fc0900  
7300 fb0122  
7301 06ebe7  
7302 100905  
7303 39e10e  
7304 f13349  
7305 2e2fd1  
7306 e7dcfd  
7307 1b0024  
7308 17210a  
7309 28fdfd  
7310 0f1313  
7311 ec0afb  
7312 000b08  
7313 ff0910  
7314 1304f3  
7315 0afc13  
7316 fd0602  
7317 edf50f  
7318 07140f  
7319 f5edfa  
7320 2b36ed  
7321 c8e114  
7322 09032c  
7323 0611f3  
7324 19f608  
7325 fb0f11  
7326 1511fa  
7327 f3e306  
7328 0c01f8  
7329 1b0c0b  
7330 06f80b  
7331 0b03f8  
7332 09ebf0  
7333 fe0a0c  
7334 fcf913  
7335 06f102  
7336 04f30d  
7337 11051d  
7338 170100

7339 e70c18  
7340 dbf9ef  
7341 1814f4  
7342 150a04  
7343 f02125  
7344 f2fb02  
7345 ec10f9  
7346 fcfc0a  
7347 01e3f7  
7348 ee2105  
7349 0f17fc  
7350 f3ff1d  
7351 0703f7  
7352 07f508  
7353 01110a  
7354 2c10fa  
7355 0c0527  
7356 1407fd  
7357 fb05fd  
7358 fbff06  
7359 fb01f1  
7360 f7f917  
7361 0e050b  
7362 e11251  
7363 2ef6e3  
7364 eb1908  
7365 f5f91f  
7366 f40f05  
7367 f71302  
7368 e0110f  
7369 0511fd  
7370 000006  
7371 fc0318  
7372 1ceff7  
7373 02e9f9  
7374 08150d  
7375 04f3f4  
7376 c40c00  
7377 df07ea  
7378 e7f8f1  
7379 1816fc  
7380 e6e805  
7381 191631  
7382 3a0019  
7383 f9f5f1  
7384 050c14  
7385 08e0e4  
7386 d8e1e2  
7387 f0f2e8  
7388 fb110f  
7389 0b0d0d  
7390 d3f710  
7391 0ae7e9  
7392 fee71d  
7393 f8eed  
7394 e2ebe6  
7395 0708ef  
7396 13faea

7397 da0ff4  
7398 fe2e06  
7399 fde0f4  
7400 f41a18  
7401 2d1ae4  
7402 f0f509  
7403 ff390e  
7404 f5fff7  
7405 fd06fc  
7406 f4e505  
7407 18ec00  
7408 eceff7  
7409 eef000  
7410 170ae9  
7411 f600ec  
7412 ef0b0d  
7413 fd03f2  
7414 0f0909  
7415 edf912  
7416 f400ed  
7417 f1d9f7  
7418 f4f7e7  
7419 e9d0f1  
7420 dfddd4  
7421 000af9  
7422 0e0106  
7423 0cf104  
7424 f7ea17  
7425 fbfe00  
7426 01d7e3  
7427 eace03  
7428 eb210a  
7429 1dfacc  
7430 f300fc  
7431 d50b22  
7432 e1fdf7  
7433 03e7ef  
7434 e1fb0d  
7435 f505da  
7436 ece301  
7437 140a0b  
7438 00143a  
7439 f910f7  
7440 0df9e8  
7441 07f6e2  
7442 eae5ef  
7443 dbe2de  
7444 d0fcd1  
7445 e3efda  
7446 f8f215  
7447 11e701  
7448 1d02de  
7449 de0233  
7450 f5fadb  
7451 ebe8f9  
7452 dc14fa  
7453 ec03e1  
7454 131100

7455 fbfa10  
7456 1c00f6  
7457 05f928  
7458 f7dbe7  
7459 ddf206  
7460 04f9e6  
7461 dff1e8  
7462 18dae6  
7463 05f1fd  
7464 171ced  
7465 fe06ff  
7466 030e2c  
7467 230ef4  
7468 e9eb0a  
7469 f3f600  
7470 0000d7  
7471 020911  
7472 fafe06  
7473 090510  
7474 f1e2ef  
7475 f1e1da  
7476 ffe8e6  
7477 edcef6  
7478 d1dcf2  
7479 1a34d9  
7480 08eff8  
7481 e00523  
7482 2806f4  
7483 0ce71e  
7484 350c05  
7485 f5f708  
7486 f51907  
7487 0ff4f1  
7488 f1f5f4  
7489 eb09ee  
7490 03f3ff  
7491 deff22  
7492 1b17ed  
7493 06eb02  
7494 dafa01  
7495 09ffdd  
7496 f70c19  
7497 0206f6  
7498 40302d  
7499 f9ffdc  
7500 0008f8  
7501 0cfbf2  
7502 f41cef  
7503 c9e7ff  
7504 0000e5  
7505 0b0909  
7506 08faf6  
7507 0a141a  
7508 e60cec  
7509 0a0bfe  
7510 f4fbfe  
7511 d21a07  
7512 1ae1f2

7513 fdf800  
7514 ff130c  
7515 fe1cf1  
7516 0de918  
7517 02ff07  
7518 f705f0  
7519 0007de  
7520 eafceb  
7521 061902  
7522 183525  
7523 032ced  
7524 ff072a  
7525 325143  
7526 fcef0b  
7527 292121  
7528 18fd26  
7529 f31b1e  
7530 f5f9fe  
7531 f4dfdd  
7532 21fbd8  
7533 c3ca03  
7534 103e4e  
7535 5eeeed  
7536 03f72f  
7537 0e6a17  
7538 f5fdc9  
7539 e2e3fc  
7540 ea000b  
7541 eee3fa  
7542 06f4e2  
7543 f407d8  
7544 e7f4fc  
7545 fbf4f7  
7546 f5f5ea  
7547 f20810  
7548 1c2207  
7549 e2e9f3  
7550 0e373c  
7551 0ff5f4  
7552 fd1e13  
7553 0c1c07  
7554 07e004  
7555 ffe7ee  
7556 09d4e3  
7557 050f0d  
7558 1c23f3  
7559 0b1125  
7560 0b343c  
7561 53100d  
7562 42004a  
7563 f0086d  
7564 d9e8e5  
7565 f4efe7  
7566 05e1fa  
7567 f90ff9  
7568 1528f1  
7569 1df5fa  
7570 d82120

7571 f222de  
7572 f0f40f  
7573 0425c7  
7574 e8f509  
7575 1b3254  
7576 33e3e0  
7577 f32530  
7578 13f50e  
7579 722412  
7580 fdfae5  
7581 fcf6f4  
7582 f8041b  
7583 010ce6  
7584 0a0e09  
7585 fb0a16  
7586 00160a  
7587 1e06f2  
7588 0af606  
7589 121518  
7590 121c1f  
7591 07ebfb  
7592 08231c  
7593 f300fb  
7594 fef30c  
7595 e30f0b  
7596 fc00f3  
7597 101809  
7598 f600e5  
7599 fe0b19  
7600 fbe911  
7601 ece602  
7602 03f8e4  
7603 0fdeef  
7604 10fee8  
7605 000008  
7606 fae808  
7607 0104f5  
7608 1205e8  
7609 f90900  
7610 09071f  
7611 290af1  
7612 f7fd0c  
7613 f4ff25  
7614 060017  
7615 0d0708  
7616 c7f90f  
7617 0d07dc  
7618 f2c2fa  
7619 0311f2  
7620 e0e6e7  
7621 0cfb06  
7622 f5ee19  
7623 0807e7  
7624 00f7f4  
7625 17effc  
7626 14dfc  
7627 f00ae9  
7628 060bed

7629 090ff9  
7630 13ebf4  
7631 10f112  
7632 f00a02  
7633 1415e5  
7634 1efbf2  
7635 170e19  
7636 150e0c  
7637 11f804  
7638 00000d  
7639 f801e4  
7640 100d08  
7641 00f2fa  
7642 17f8f4  
7643 f0ea14  
7644 1116f9  
7645 f803fe  
7646 0b1d11  
7647 fbfdff  
7648 ef03e3  
7649 04f7fe  
7650 24f801  
7651 fafe0f  
7652 0ee9ef  
7653 fb1201  
7654 1cfee8  
7655 f7fffd  
7656 18e5e0  
7657 10ed3d  
7658 3b2ffc  
7659 ecede6  
7660 f914f5  
7661 0fe8f1  
7662 f2f901  
7663 fc03ee  
7664 1bff10  
7665 0dfce2  
7666 f1eaf8  
7667 0f0706  
7668 15e716  
7669 f30525  
7670 16f7cd  
7671 d7a6eb  
7672 fe18ee  
7673 f9f3e8  
7674 07061a  
7675 0ce9e9  
7676 fefdee  
7677 17151a  
7678 1ff607  
7679 e10206  
7680 0721f3  
7681 1939ff  
7682 14e919  
7683 010825  
7684 e7eee8  
7685 e4f805  
7686 12f807

7687 f60e01  
7688 080012  
7689 e2e80e  
7690 e4f4f8  
7691 290ff6  
7692 17f512  
7693 1b0c2c  
7694 ffded7  
7695 da242a  
7696 0b0ad7  
7697 fb050a  
7698 07f505  
7699 04ee04  
7700 0b0007  
7701 0c0912  
7702 fa0108  
7703 1c0407  
7704 06d7c2  
7705 ee2a16  
7706 10ecd9  
7707 0ee018  
7708 202a16  
7709 0cf9e0  
7710 d4261d  
7711 0afadd  
7712 e2cb1a  
7713 1f1022  
7714 f700e8  
7715 2506ec  
7716 010cf8  
7717 0b0b13  
7718 110f07  
7719 0505ef  
7720 06f100  
7721 05061b  
7722 02ebfc  
7723 17ef10  
7724 07e2ea  
7725 f7fafc  
7726 edfd07  
7727 fefd01  
7728 fceef2  
7729 1d050d  
7730 090006  
7731 00f200  
7732 0209f9  
7733 0b0bf5  
7734 140801  
7735 f40a09  
7736 00fff9  
7737 000ef2  
7738 0b0207  
7739 e5f8fc  
7740 0eeb03  
7741 1b1410  
7742 171e03  
7743 fb0616  
7744 08faec

7745 ec02f3  
7746 180e02  
7747 1df127  
7748 f00402  
7749 00220e  
7750 f0f1e3  
7751 eaed0b  
7752 1fe6fe  
7753 030312  
7754 fcff05  
7755 11f6fa  
7756 02edf9  
7757 fb14f3  
7758 05e9f6  
7759 fdf60c  
7760 180300  
7761 05fd1e  
7762 1700f8  
7763 fa100b  
7764 211fdb  
7765 e70c12  
7766 19f80d  
7767 101a02  
7768 fff3fc  
7769 ffecf9  
7770 0d00fe  
7771 e8000d  
7772 000004  
7773 051501  
7774 ed17fb  
7775 00ef09  
7776 effbf2  
7777 ea00fb  
7778 20f102  
7779 1bf603  
7780 11fd05  
7781 fb0201  
7782 f5fefa  
7783 fcfdfa  
7784 f3e4f9  
7785 100beb  
7786 0a04fb  
7787 1f1d11  
7788 06ed0a  
7789 030cfd  
7790 110b1f  
7791 011f42  
7792 33f0fc  
7793 d3f307  
7794 032b10  
7795 fdf608  
7796 0603fd  
7797 f5f2f9  
7798 d7ea05  
7799 0ce8b2  
7800 afeaf8  
7801 ff1dfa  
7802 ecefef

7803 e6f205  
7804 ef0b07  
7805 f5bdf1  
7806 eded19  
7807 02f108  
7808 05fcf5  
7809 f5f5f3  
7810 f4f60d  
7811 1200f2  
7812 060105  
7813 f60907  
7814 e32914  
7815 2f081e  
7816 1d1c00  
7817 182911  
7818 f2fb06  
7819 01f613  
7820 e3f71d  
7821 f7fdd6  
7822 feff00  
7823 fe0dfb  
7824 ebf1f5  
7825 09051b  
7826 eefdfc  
7827 3409d8  
7828 e1eb01  
7829 051c0a  
7830 290516  
7831 fbfeea  
7832 131801  
7833 1611e4  
7834 fef700  
7835 ebe316  
7836 f3fdf1  
7837 ec04fc  
7838 2c11f4  
7839 ecf83f  
7840 122c13  
7841 23e217  
7842 2f2819  
7843 eb0326  
7844 194017  
7845 1fed02  
7846 261919  
7847 f8fb2a  
7848 010c01  
7849 00f81d  
7850 f4f902  
7851 fe0af3  
7852 00f3f6  
7853 f700f3  
7854 1702f8  
7855 0df6ff  
7856 1d1b0c  
7857 232322  
7858 0bf818  
7859 f905fd  
7860 ee0cec

7861 00f304  
7862 fcec0a  
7863 0f0cf0  
7864 fdf005  
7865 e9e800  
7866 faf2f3  
7867 08ec0b  
7868 f50bea  
7869 fff90f  
7870 1e19fa  
7871 010704  
7872 1d1208  
7873 190aff  
7874 0d0c0e  
7875 19ef06  
7876 00ebf3  
7877 ea06eb  
7878 f3e402  
7879 21f9f0  
7880 ecfe0c  
7881 eaeff1  
7882 ecf8f5  
7883 f4f3e5  
7884 f8ee07  
7885 fd07e8  
7886 daf5ec  
7887 eff913  
7888 fe030e  
7889 f808f5  
7890 f0ffef  
7891 130ffa  
7892 f6f20f  
7893 f20300  
7894 f0f804  
7895 fe09f5  
7896 100604  
7897 1f0911  
7898 0e10fe  
7899 13daf4  
7900 190a06  
7901 e90100  
7902 f7ed09  
7903 1600fe  
7904 090dff  
7905 f4e300  
7906 000005  
7907 e00501  
7908 f8f0f2  
7909 0a0e03  
7910 00f4f4  
7911 00ecf5  
7912 f406fc  
7913 f9f8f4  
7914 f9ddff  
7915 07faee  
7916 eef2eb  
7917 ed0aff  
7918 01e1fe

7919 f8f812  
7920 f1f3e3  
7921 f50b04  
7922 ede9f5  
7923 05e9f3  
7924 f208fa  
7925 eaf5fe  
7926 fc11f7  
7927 fdfdf1  
7928 eaf50e  
7929 e21100  
7930 f9f7f1  
7931 f0f80a  
7932 fef006  
7933 020ce8  
7934 f1fb06  
7935 08f7e7  
7936 f7ec16  
7937 1612f7  
7938 e30305  
7939 fb110a  
7940 faf204  
7941 def0e6  
7942 f9f702  
7943 08ecf2  
7944 00e6f5  
7945 f4e7f2  
7946 e80c05  
7947 13e7d3  
7948 f3e0fc  
7949 f01204  
7950 e6e709  
7951 f60600  
7952 f9fb0b  
7953 faec02  
7954 0006fe  
7955 efe5f4  
7956 f1f5fd  
7957 0bf0f7  
7958 e7f507  
7959 ef02fa  
7960 efdc00  
7961 f4faff  
7962 fd03fc  
7963 0efc09  
7964 f0f512  
7965 0bfc07  
7966 e706ff  
7967 f5eb04  
7968 eff702  
7969 e6f4ec  
7970 e9daf1  
7971 03f6f3  
7972 fbf2ed  
7973 ec0df6  
7974 0fe4f1  
7975 fdebfa  
7976 0c090e

7977 fae4f1  
7978 0d0a0a  
7979 f31600  
7980 0111df  
7981 f3f700  
7982 e4e802  
7983 fcf3f8  
7984 07f0fe  
7985 f0ea02  
7986 e0fd07  
7987 f6f3dd  
7988 fa0004  
7989 faf5e5  
7990 e706f2  
7991 fffceb  
7992 e7e205  
7993 09ebf6  
7994 130307  
7995 04e7ee  
7996 fddf04  
7997 fcedda  
7998 04f400  
7999 f9e106  
8000 feeed  
8001 ecfa0c  
8002 03f5fd  
8003 05e3e8  
8004 fe00e7  
8005 eaf3f9  
8006 e6f5f5  
8007 faf2eb  
8008 05dfee  
8009 eb0801  
8010 f0e5fb  
8011 e8edfb  
8012 f8dbfb  
8013 ebe7fe  
8014 00ecf8  
8015 0df7ef  
8016 e9f3ed  
8017 0cf4fa  
8018 edf7f0  
8019 faef06  
8020 f9f80d  
8021 fceff7  
8022 daf6f5  
8023 f9fdeb  
8024 fee502  
8025 eafbf9  
8026 fff6ee  
8027 fe0401  
8028 fee901  
8029 fc08da  
8030 e4ece5  
8031 eaf608  
8032 10e4f8  
8033 faf10a  
8034 f4fc06

8035 160bfbb  
8036 f6f117  
8037 08ffee  
8038 df01fc  
8039 f1ee05  
8040 0000f3  
8041 e90c0c  
8042 050b01  
8043 05061f  
8044 00f6ff  
8045 f80e0a  
8046 15fcf4  
8047 ebeee7  
8048 f208f4  
8049 f6fbfb  
8050 cddaf7  
8051 fdfe16  
8052 13f401  
8053 080f17  
8054 feeffc  
8055 000317  
8056 faeeff  
8057 0de900  
8058 3020f7  
8059 fe0040  
8060 1dff06  
8061 0c10ff  
8062 fd0721  
8063 f8ebfa  
8064 fafffe  
8065 e70209  
8066 05f7e2  
8067 fef7e0  
8068 cb0603  
8069 0909e4  
8070 dc17ed  
8071 f6ebf1  
8072 e9ea0f  
8073 f2fae3  
8074 000404  
8075 02ede8  
8076 06f1f2  
8077 e202e6  
8078 05e4f7  
8079 f2ece4  
8080 fe00ef  
8081 1003e8  
8082 0cfe03  
8083 0b091a  
8084 faeb1b  
8085 1209e8  
8086 0304ff  
8087 0bf407  
8088 f8ea04  
8089 0405fb  
8090 e2fd0e  
8091 fde1f0  
8092 1510fd

8093 203b27  
8094 09f821  
8095 0f0ef1  
8096 e913f6  
8097 060414  
8098 ff08f7  
8099 f3dff5  
8100 281207  
8101 0508e6  
8102 dd0df8  
8103 fe01ea  
8104 ede700  
8105 fe1eef  
8106 1411f7  
8107 06dc00  
8108 0f0205  
8109 260413  
8110 1c202d  
8111 00fdf3  
8112 052406  
8113 29f815  
8114 0f1bfe  
8115 daf317  
8116 e1e3e9  
8117 08ece8  
8118 110c00  
8119 09f607  
8120 0f130b  
8121 faf2fc  
8122 d50206  
8123 0010eb  
8124 06fb08  
8125 1900f4  
8126 02ee15  
8127 061df8  
8128 fd050e  
8129 14110c  
8130 eefdfb  
8131 0cf602  
8132 06010c  
8133 e6e10f  
8134 f6f5f1  
8135 db400  
8136 f5f8ff  
8137 e403fd  
8138 fdf3fd  
8139 eedd12  
8140 f5eefb  
8141 dcd00  
8142 05e6e4  
8143 1b1f06  
8144 0a1105  
8145 fd0ced  
8146 141525  
8147 f5f405  
8148 0b0a0e  
8149 1731ed  
8150 def00a

8151 011af8  
8152 f0e0f8  
8153 f5f9f5  
8154 ddf4ff  
8155 e6fff8  
8156 03e1f6  
8157 090b03  
8158 faed00  
8159 f0eae2  
8160 fefff2  
8161 f6fde5  
8162 df0afb  
8163 fafcf0  
8164 10fcf7  
8165 f20ef8  
8166 f50d09  
8167 13f916  
8168 1bfb0a  
8169 071120  
8170 000909  
8171 ea04f1  
8172 0a00e9  
8173 fcfbec  
8174 0000ec  
8175 0ff20b  
8176 10fb03  
8177 191004  
8178 07f70e  
8179 0c05f6  
8180 f501f0  
8181 1c06f1  
8182 f315fc  
8183 1701ef  
8184 fc111b  
8185 fe01ec  
8186 f2f005  
8187 fff8e3  
8188 00f306  
8189 f70018  
8190 10ecf9  
8191 0c1324  
8192 dddf11  
8193 ea12e9  
8194 d9edee  
8195 f7faee  
8196 fad3e2  
8197 1c1f09  
8198 110322  
8199 322b15  
8200 6a1805  
8201 f11e6b  
8202 393b00  
8203 26150f  
8204 f20600  
8205 012805  
8206 00dbf0  
8207 f30216  
8208 0000f8

8209 02081e  
8210 17160a  
8211 060a1a  
8212 fa1601  
8213 f9f91b  
8214 16fc1a  
8215 04000d  
8216 fee908  
8217 eacae9  
8218 f905e8  
8219 f500ec  
8220 f70705  
8221 fdedf6  
8222 0cf602  
8223 02081d  
8224 1926fc  
8225 e6fcf7  
8226 0ffeec  
8227 150919  
8228 09102a  
8229 180418  
8230 fff9d6  
8231 ef0202  
8232 05def9  
8233 fafc10  
8234 f6f405  
8235 110431  
8236 0b0902  
8237 0812fb  
8238 fa0f17  
8239 2f1cee  
8240 d8f500  
8241 16faf3  
8242 faf404  
8243 0121e3  
8244 f4ecfd  
8245 00f2f2  
8246 f0f7f0  
8247 07eee5  
8248 faf62e  
8249 350dff  
8250 e5f6f0  
8251 faf1e4  
8252 fd0bf7  
8253 0dfd11  
8254 fb0317  
8255 1801ea  
8256 f01114  
8257 f1f8fd  
8258 dcff02  
8259 d1d3cb  
8260 f1edfb  
8261 00e5f3  
8262 130403  
8263 faf421  
8264 1e020e  
8265 ff070e  
8266 093221

8267	000410
8268	0e05fe
8269	17f503
8270	0cf5ef
8271	0cfd00
8272	09fe10
8273	0b1308
8274	12f70c
8275	e1e5eb
8276	0cddda
8277	fc040f
8278	051907
8279	081911
8280	f0dcf1
8281	e61d23
8282	22fe02
8283	180819
8284	080403
8285	02ff05
8286	ef0e13
8287	f805f3
8288	0513f5
8289	f60913
8290	0102fe
8291	f400eb
8292	130b07
8293	fc0008
8294	ee0cf7
8295	020900
8296	040af1
8297	041817
8298	f4f005
8299	1b0ff6
8300	c8db20
8301	fe2d02
8302	03f0f0
8303	00e9e5
8304	0e1622
8305	23fdea
8306	e91dfa
8307	fd16ff
8308	000011
8309	ea010f
8310	f2200e
8311	05f2eb
8312	1f0606
8313	00dc11
8314	fb05fa
8315	2301e7
8316	010d1c
8317	feef01
8318	3a1707
8319	1603ee
8320	fa230c
8321	e40b1b
8322	1d0af7
8323	d8e7fe
8324	f8140d

8325 19f2f0  
8326 ff2c18  
8327 04f6cd  
8328 c518fa  
8329 1eedf9  
8330 f2060c  
8331 00070a  
8332 f2f821  
8333 fcffef  
8334 e2fffc  
8335 04d111  
8336 24e7dd  
8337 e00a0c  
8338 261313  
8339 fcd710  
8340 0822fe  
8341 10d508  
8342 00f5f4  
8343 e30800  
8344 040d02  
8345 122818  
8346 14f6f3  
8347 fe092d  
8348 0a070d  
8349 fbf000  
8350 16f7e3  
8351 ed2734  
8352 f6f0c5  
8353 d2ed07  
8354 220fee  
8355 e60b00  
8356 1a0003  
8357 eded0d  
8358 fd07f5  
8359 0afe1b  
8360 181004  
8361 e0daff  
8362 f1f5cc  
8363 c30c14  
8364 140eed  
8365 dfd603  
8366 f2090b  
8367 141318  
8368 fff4e1  
8369 09fb11  
8370 f608ee  
8371 140007  
8372 10f9f5  
8373 02ef07  
8374 fc1401  
8375 0e12f0  
8376 f0eb0b  
8377 12fa06  
8378 ceed0e  
8379 1609e7  
8380 ecbff6  
8381 fafc09  
8382 21dcf3

8383 1bf1f7  
8384 060def  
8385 0c0303  
8386 e50c06  
8387 08f402  
8388 0be6fc  
8389 e90a1a  
8390 2d0cef  
8391 11f609  
8392 1ceeef  
8393 17132a  
8394 fc08fd  
8395 f8f3e8  
8396 f507fa  
8397 000e14  
8398 f6f305  
8399 f30b16  
8400 ec01f1  
8401 17ffff  
8402 160a25  
8403 1417ff  
8404 f60013  
8405 1808e6  
8406 eff704  
8407 000bfc  
8408 f10c05  
8409 2113f5  
8410 f20e1c  
8411 d60904  
8412 0dddfa  
8413 f7e104  
8414 f707ed  
8415 f9d9fa  
8416 fdfe11  
8417 e2e3f8  
8418 01f0ec  
8419 ee0fdf  
8420 f1f8f3  
8421 000023  
8422 1e1def  
8423 f30011  
8424 fd302b  
8425 00f012  
8426 11f50e  
8427 0b1013  
8428 281110  
8429 050812  
8430 1d040c  
8431 02f3f5  
8432 0404ec  
8433 f202fd  
8434 1818f5  
8435 f1f720  
8436 f3ecf8  
8437 2ce9fd  
8438 1dfcfd  
8439 d918f5  
8440 fa0ef5

8441 edeef3  
8442 0000dc  
8443 05fe14  
8444 020915  
8445 0ffe06  
8446 080a0e  
8447 fcfde6  
8448 e82719  
8449 f004e6  
8450 07dbff  
8451 00f11e  
8452 ed08fa  
8453 181201  
8454 311713  
8455 0bff25  
8456 282310  
8457 f607ec  
8458 f01700  
8459 dcffe1  
8460 19f2d6  
8461 f2de1a  
8462 242f11  
8463 270bf9  
8464 280910  
8465 f72307  
8466 0708d7  
8467 e6e407  
8468 ab0003  
8469 1fce9d  
8470 e5e2ff  
8471 f0f217  
8472 251b0d  
8473 16e5f1  
8474 112d38  
8475 3a2a06  
8476 061211  
8477 d9fa00  
8478 e5f01a  
8479 16fa06  
8480 0c07e6  
8481 de0307  
8482 ec1103  
8483 faf2fa  
8484 ef1201  
8485 04150f  
8486 09de1d  
8487 f9110a  
8488 1c12fe  
8489 0605fa  
8490 030501  
8491 0606e2  
8492 d4d603  
8493 071310  
8494 dcfe19  
8495 efd8d2  
8496 e0b6f8  
8497 e7fa0c  
8498 1d07e4

8499	25e307
8500	e8331c
8501	020010
8502	f7ea01
8503	fb9fa
8504	c50700
8505	ddffe9
8506	0fe2eb
8507	ddeb19
8508	001606
8509	e80ef2
8510	101f08
8511	d8c227
8512	0820ff
8513	0b04f4
8514	11e9ed
8515	ed3021
8516	1c32c1
8517	e9d301
8518	070603
8519	0b1cfb
8520	fe0909
8521	050a09
8522	f2f3dd
8523	e5030b
8524	f507f2
8525	1d0ae0
8526	fcdbf6
8527	22121f
8528	160aee
8529	f30416
8530	02221c
8531	0f1601
8532	fce5ec
8533	e9e3f3
8534	01efd8
8535	100f0c
8536	1004fb
8537	f41b1b
8538	000923
8539	0a03f9
8540	ecfcfb
8541	f6e80c
8542	e80dfc
8543	0f1815
8544	ff031f
8545	faf2ee
8546	12f417
8547	f6ddf6
8548	f517fa
8549	f3dbe5
8550	f10513
8551	f8f0f4
8552	f100e8
8553	fd0815
8554	2a0400
8555	0c042f
8556	0200fc

8557 fcf7f5  
8558 f9d5f1  
8559 fb0403  
8560 e3fb03  
8561 0407f7  
8562 fc020a  
8563 faf905  
8564 fafe0a  
8565 0ef1eb  
8566 0ae7f1  
8567 f9f903  
8568 1dfdda  
8569 e7f414  
8570 00ff16  
8571 1afa08  
8572 0cfcef  
8573 d4162c  
8574 f0fd02  
8575 fae30c  
8576 0000f3  
8577 00edf9  
8578 0c08f8  
8579 ec08f3  
8580 eb09fb  
8581 0a07eb  
8582 08ec02  
8583 f408fc  
8584 0d0ef5  
8585 0004fe  
8586 f7eb07  
8587 f5f8f9  
8588 f4eef5  
8589 f9fafb  
8590 06faff  
8591 00eefe  
8592 010ef6  
8593 0af7ee  
8594 0ffd0f  
8595 00030a  
8596 fb03ee  
8597 04fb00  
8598 f00d0a  
8599 00ed0d  
8600 040c0a  
8601 fff106  
8602 10fcf0  
8603 f3faf1  
8604 ecf208  
8605 02ecf4  
8606 f5ec0d  
8607 feecee  
8608 0ffbed  
8609 f8ef08  
8610 f20802  
8611 0feff5  
8612 0af9f7  
8613 04eef2  
8614 0d02fa

8615 0003f9  
8616 fb09f9  
8617 07f8f7  
8618 0205f4  
8619 edf00d  
8620 f80d0c  
8621 f3eeff  
8622 fded03  
8623 eb0903  
8624 f4f7fe  
8625 fffcfc4  
8626 05ecee  
8627 0efa0e  
8628 ff0000  
8629 f2ed03  
8630 0000ec  
8631 0beff5  
8632 09f4f0  
8633 04fd09  
8634 f7f4fb  
8635 f503f2  
8636 f9f500  
8637 f70a04  
8638 0df300  
8639 03020f  
8640 f6ee01  
8641 01f6fa  
8642 f2000f  
8643 edf60b  
8644 fa0800  
8645 fceb04  
8646 fffced  
8647 0cf4f7  
8648 09010a  
8649 f4f803  
8650 000bef  
8651 070807  
8652 03fffc  
8653 06ed03  
8654 00f7ee  
8655 f9f104  
8656 fcfef1  
8657 f2000b  
8658 f4ff04  
8659 f70507  
8660 eefb02  
8661 ec0df8  
8662 f8f7f2  
8663 f201fa  
8664 ee04f8  
8665 040400  
8666 feeefa  
8667 0d0cff  
8668 0c0fff  
8669 eef2f7  
8670 06f8f9  
8671 eeed04  
8672 fcfafd

8673 fdf8f6  
8674 0000eb  
8675 fa02f5  
8676 ebf8ed  
8677 ec02f5  
8678 faecf5  
8679 f903f7  
8680 00f3f0  
8681 04050e  
8682 Obfaf7  
8683 f4000b  
8684 ed01fe  
8685 02000b  
8686 fbf605  
8687 fff5f9  
8688 f4030c  
8689 0c040b  
8690 04040c  
8691 ee0300  
8692 f2f4f3  
8693 060900  
8694 010def  
8695 f9f9f3  
8696 ef08ec  
8697 05fd0b  
8698 0ceefe  
8699 f8fa08  
8700 04fb0c  
8701 f5f300  
8702 f7f3ef  
8703 0b01f7  
8704 f501f8  
8705 0a0e01  
8706 fdf2f4  
8707 f70deb  
8708 04f30a  
8709 07f9ff  
8710 000004  
8711 151a0e  
8712 f3f205  
8713 e6ef0a  
8714 f1e7eb  
8715 07fbf6  
8716 17eef3  
8717 fcf9fc  
8718 17f2f8  
8719 151306  
8720 eef2e8  
8721 00040f  
8722 e6def5  
8723 f5fee5  
8724 d9e9ef  
8725 08f90e  
8726 f6e613  
8727 180815  
8728 0b4135  
8729 0d2a21  
8730 2c0d0e

8731 ff0a0b  
8732 002b36  
8733 02ebf9  
8734 0ef21c  
8735 f50406  
8736 23f3f2  
8737 f2191b  
8738 e73ef3  
8739 e404e6  
8740 e42013  
8741 1500e3  
8742 faec04  
8743 e71610  
8744 f4f005  
8745 26f600  
8746 0ae7f1  
8747 00f3e9  
8748 00ee00  
8749 fc05ec  
8750 f4f8f1  
8751 020ef3  
8752 f6041b  
8753 00130b  
8754 13fa0f  
8755 2b2924  
8756 e7fc2f  
8757 00e9f5  
8758 09f4f2  
8759 05f501  
8760 06f8eb  
8761 0ff9ef  
8762 fc0815  
8763 3d3219  
8764 234508  
8765 3218e7  
8766 12042d  
8767 1b2e2c  
8768 2d0810  
8769 fbddf6  
8770 f10611  
8771 e61ce4  
8772 1fd7dd  
8773 e4fcfa  
8774 111208  
8775 0e0af8  
8776 fa0ffd  
8777 06db0b  
8778 0029fb  
8779 271ef9  
8780 12180a  
8781 00ff1b  
8782 382e18  
8783 22eb00  
8784 f91b3c  
8785 1d0904  
8786 fc0117  
8787 eafc04  
8788 1000ee

8789 07e9f5  
8790 fef8fd  
8791 fef20d  
8792 030900  
8793 0e1cf1  
8794 100f03  
8795 091a06  
8796 15091d  
8797 240e27  
8798 0be6f8  
8799 fd0600  
8800 00f60b  
8801 fdeb06  
8802 1e06fb  
8803 f3ff1a  
8804 0505ea  
8805 f8fbf2  
8806 f3f9e1  
8807 f1f506  
8808 01fae9  
8809 ff07f1  
8810 03060c  
8811 fc1714  
8812 160808  
8813 111012  
8814 f800e9  
8815 090aeb  
8816 f70718  
8817 151316  
8818 02030f  
8819 191800  
8820 d9e518  
8821 0ef2f2  
8822 d9e5ea  
8823 06faea  
8824 daf2fd  
8825 e80909  
8826 f1fcfd  
8827 f7e3f1  
8828 0af801  
8829 0d0dfc  
8830 e50ffb  
8831 ecfee9  
8832 f1eef6  
8833 f40efa  
8834 f7f702  
8835 ed0815  
8836 05052d  
8837 0f00ef  
8838 1126fa  
8839 dcfc12  
8840 ea1bfd  
8841 070d05  
8842 fefb17  
8843 04feff  
8844 000000  
8845 1af6fa  
8846 08fdf4

8847 fefd04  
8848 ff08ef  
8849 dc1106  
8850 faf4e5  
8851 d2f810  
8852 0c1810  
8853 f0f2fc  
8854 010701  
8855 f6f4f3  
8856 1ae7fc  
8857 f7f4e7  
8858 0300e6  
8859 fc0700  
8860 06f8f0  
8861 f9f40f  
8862 feea05  
8863 f7f918  
8864 fde706  
8865 05ebe5  
8866 e5270d  
8867 28020b  
8868 0b05ff  
8869 fcf7f0  
8870 1f1dfe  
8871 e30963  
8872 5f4826  
8873 1bfe1b  
8874 135910  
8875 2d1710  
8876 18ef2d  
8877 13361d  
8878 ffe2ee  
8879 281210  
8880 1401e0  
8881 000919  
8882 021e14  
8883 0c0fe9  
8884 f3eaec  
8885 0cf805  
8886 d9e7fc  
8887 fedcee  
8888 f00800  
8889 1317e1  
8890 eccdf5  
8891 04fe17  
8892 0bf222  
8893 e300e4  
8894 e0d4f1  
8895 00eee5  
8896 f812f8  
8897 1a3b1e  
8898 1f1efa  
8899 010408  
8900 e1040a  
8901 000605  
8902 2200f0  
8903 e3fc0d  
8904 0c0c0f

8905 0613f3  
8906 ffca0b  
8907 e6001e  
8908 0bf6dd  
8909 f8efc6  
8910 fae2fe  
8911 ebe9f5  
8912 fddac7  
8913 f2f2eb  
8914 facff0  
8915 bef1f2  
8916 ccd7d7  
8917 e8c2ba  
8918 ddd5e9  
8919 0706f4  
8920 f9f8d0  
8921 f40e07  
8922 0909e6  
8923 041eed  
8924 eafd00  
8925 17ff0f  
8926 ebe9e9  
8927 f5ff13  
8928 e2cbf7  
8929 fcf5f3  
8930 fee8f4  
8931 d80502  
8932 f9fff4  
8933 fcf304  
8934 e3efef  
8935 00f7ff  
8936 f9d4fd  
8937 fb15e9  
8938 f7f10d  
8939 ea0d03  
8940 050cf2  
8941 0001f7  
8942 000e14  
8943 1e08e3  
8944 fcf202  
8945 f0edd4  
8946 ddf410  
8947 0aeae4  
8948 ecf8e8  
8949 db7e4  
8950 e5dcf8  
8951 01dd00  
8952 13fde2  
8953 f9fdf9  
8954 f60c07  
8955 0b0c18  
8956 0af914  
8957 1a0d09  
8958 fcf2f0  
8959 ed1423  
8960 0431f5  
8961 f0e60d  
8962 0818f5

8963	02f70d
8964	f2f91b
8965	1001fd
8966	1ff204
8967	e509f5
8968	0e07e4
8969	f4f2fb
8970	ccd8d4
8971	cd1a14
8972	01e6f2
8973	e2ea08
8974	001e01
8975	1cfaf8
8976	05dc03
8977	e80ffc
8978	0000d7
8979	06f206
8980	fc0500
8981	e4f9fd
8982	00d7ed
8983	1000fd
8984	f9fde3
8985	240910
8986	02ff14
8987	e701f8
8988	161d13
8989	04f6ec
8990	df0f02
8991	e9eceb
8992	daebf8
8993	ed0ef6
8994	11131f
8995	1e1a14
8996	053024
8997	0026e2
8998	e702fa
8999	ed0014
9000	d3bdf
9001	0dfde9
9002	f3e731
9003	31240d
9004	fa0301
9005	f0e0f1
9006	2010ea
9007	d80933
9008	35010c
9009	010315
9010	0d00d3
9011	dcc4ec
9012	fcf4ec
9013	f0eceb
9014	fbe7f7
9015	07fc12
9016	fd02ff
9017	1b00fa
9018	edfff2
9019	ec021e
9020	18fbf9

9021 f90721  
9022 0016db  
9023 eff416  
9024 f10833  
9025 f320e7  
9026 151817  
9027 00f526  
9028 0c000d  
9029 e6f7ff  
9030 12f4f4  
9031 090e04  
9032 1227e6  
9033 e9fd21  
9034 ee0032  
9035 e50d00  
9036 27f1e1  
9037 fff32f  
9038 f7e6f8  
9039 fe0212  
9040 26e6f9  
9041 032206  
9042 021e01  
9043 24daf8  
9044 0af40a  
9045 151100  
9046 eb0e10  
9047 3e1bf5  
9048 d60b11  
9049 09043b  
9050 fcf6f7  
9051 ebeae9  
9052 04c411  
9053 50250f  
9054 14f406  
9055 e40210  
9056 08ef0f  
9057 f10ae9  
9058 f30711  
9059 091110  
9060 2af609  
9061 06fa09  
9062 1512f1  
9063 0f0911  
9064 e9142d  
9065 f303fa  
9066 f4eff5  
9067 f5d9fc  
9068 12f712  
9069 0c0e1d  
9070 fb0310  
9071 1f0609  
9072 f6f81e  
9073 111012  
9074 ffedf9  
9075 01e400  
9076 04ffff  
9077 152508  
9078 03fe03

9079 161cf7  
9080 001a19  
9081 e5fa19  
9082 0500f7  
9083 f10805  
9084 f8f6f6  
9085 23fa12  
9086 1afa1c  
9087 00170e  
9088 f0f7fd  
9089 04e4eb  
9090 df00ef  
9091 eef1cb  
9092 f4efff  
9093 170706  
9094 fa2028  
9095 00e9f8  
9096 1cfb12  
9097 ee0010  
9098 110e0e  
9099 f601fc  
9100 01bf5  
9101 effb0a  
9102 0e06ed  
9103 110002  
9104 0bf000  
9105 16ea17  
9106 e5ffe6  
9107 fd0618  
9108 0300f2  
9109 0505d6  
9110 f20322  
9111 09f9f3  
9112 000003  
9113 ff0d0e  
9114 f7fdff  
9115 081c11  
9116 ebfaf7  
9117 f5ffffb  
9118 03f109  
9119 f2f517  
9120 1e15ec  
9121 2d2df4  
9122 e1ec04  
9123 090d09  
9124 e5e2f7  
9125 f514fb  
9126 e6e8f7  
9127 16ff19  
9128 040224  
9129 0d0126  
9130 0bfd0f  
9131 fe2429  
9132 11eb06  
9133 fcfdff  
9134 f8120f  
9135 09060a  
9136 02ef14

9137 f0ff07  
9138 3c170a  
9139 eb3859  
9140 032118  
9141 2704f5  
9142 cdfaf5  
9143 f02cf1  
9144 eeedfb  
9145 fa0a17  
9146 f30906  
9147 251d00  
9148 ff00f3  
9149 f404f7  
9150 ed14fe  
9151 1dfddc  
9152 f4e310  
9153 010301  
9154 e0f502  
9155 12d4d4  
9156 00f71a  
9157 2b110f  
9158 ec0029  
9159 071d08  
9160 eefbf3  
9161 0b0901  
9162 0d0fed  
9163 fffede  
9164 eef706  
9165 2c2d14  
9166 062e0a  
9167 160acd  
9168 f20419  
9169 171f07  
9170 090d01  
9171 ffecfd  
9172 011afd  
9173 f90ff5  
9174 1ef603  
9175 fce70e  
9176 0e0110  
9177 0421e3  
9178 141101  
9179 05d700  
9180 01f601  
9181 1f1ae4  
9182 f2dd01  
9183 01f236  
9184 341011  
9185 20f602  
9186 0df642  
9187 041209  
9188 06031d  
9189 000106  
9190 10f1e1  
9191 efed0f  
9192 fe1811  
9193 0beb0e  
9194 05f6f9

9195 0c17fb  
9196 ee21fe  
9197 fcf0df  
9198 130514  
9199 030413  
9200 17ea02  
9201 00f8ef  
9202 f0fc00  
9203 090cf9  
9204 11f70a  
9205 050e18  
9206 1d0402  
9207 f6fbf2  
9208 f10b03  
9209 e9070c  
9210 08f805  
9211 0215fb  
9212 1c0a12  
9213 d8f60f  
9214 12fffe  
9215 10180f  
9216 0821fb  
9217 f804f8  
9218 e9ea22  
9219 1c0b15  
9220 1af903  
9221 2e3206  
9222 f70317  
9223 200cfe  
9224 060006  
9225 0e00f8  
9226 f8ec0c  
9227 081900  
9228 1106f1  
9229 09f318  
9230 0f08ff  
9231 f5f80e  
9232 e30001  
9233 f5fbf7  
9234 e9e907  
9235 070fff  
9236 f50cfc  
9237 f21c07  
9238 e0f82a  
9239 0f2ee9  
9240 090ce7  
9241 fbff09  
9242 e1130b  
9243 1efbef  
9244 25efe2  
9245 ff0cff  
9246 0000fc  
9247 0104ff  
9248 09fcf8  
9249 020001  
9250 ff0b05  
9251 0b010f  
9252 f511f7

9253 f5f206  
9254 f0fa14  
9255 feec05  
9256 f9ee09  
9257 f200f9  
9258 0aee01  
9259 fa060f  
9260 0a07f6  
9261 eff801  
9262 070105  
9263 f200f2  
9264 ee0bf0  
9265 06fdf9  
9266 000806  
9267 08fe08  
9268 fd0210  
9269 09f2f5  
9270 ef0508  
9271 f30503  
9272 f5f0f0  
9273 f0ed0c  
9274 f702ee  
9275 f6fd0b  
9276 0902fa  
9277 f20503  
9278 f1f705  
9279 f40b0b  
9280 fcf20f  
9281 fdfd0a  
9282 f30808  
9283 ff0107  
9284 f20ded  
9285 0c00f6  
9286 f10200  
9287 fe0f0b  
9288 0d0809  
9289 09f000  
9290 0e06fc  
9291 00ee0e  
9292 f60710  
9293 06f102  
9294 f3f50f  
9295 04090f  
9296 f105fd  
9297 ed0aed  
9298 fef401  
9299 f0ecfb  
9300 0e00fb  
9301 fe0912  
9302 0610f1  
9303 fb06fc  
9304 08f40a  
9305 eef6f2  
9306 04fe00  
9307 08ef0b  
9308 07f401  
9309 00feec  
9310 f705fe

9311 ed0df2  
9312 eef9ee  
9313 0c0c0e  
9314 00f1ff  
9315 03fd06  
9316 02f1f5  
9317 f3f002  
9318 f5ffffd  
9319 001002  
9320 010206  
9321 f0f3f2  
9322 edef01  
9323 000708  
9324 0af0f6  
9325 03f4fc  
9326 f8eef1  
9327 eeeef8  
9328 f8fbf2  
9329 fc0fee  
9330 09f508  
9331 0708f4  
9332 fcf0f8  
9333 fef2ff  
9334 01f4fd  
9335 0ff4fa  
9336 0806fd  
9337 0a0400  
9338 f8faec  
9339 f7fd0c  
9340 10f9fa  
9341 0c0702  
9342 fc020e  
9343 f0f502  
9344 02ecf4  
9345 f60a00  
9346 0f0400  
9347 edfdf0  
9348 f5ecfe  
9349 fcfdfc  
9350 ef040d  
9351 0600f0  
9352 0808f3  
9353 0d01fe  
9354 faf5f9  
9355 0f06fa  
9356 f804f7  
9357 fe06f2  
9358 fc0cf0  
9359 030c00  
9360 f9fcfc  
9361 edf8fe  
9362 0af509  
9363 f8f4fa  
9364 fa0806  
9365 0ffb0b  
9366 060afa  
9367 f4f5fe  
9368 070f0b

9369 0b0d0d  
9370 f9ed03  
9371 00f9fb  
9372 13fdff  
9373 ecec01  
9374 000b0f  
9375 fb0cff  
9376 fc0906  
9377 0a0afa  
9378 eef600  
9379 00fff8  
9380 00000d  
9381 0509fd  
9382 0deffd  
9383 f20011  
9384 1212f4  
9385 ef0ff3  
9386 f31010  
9387 0901fe  
9388 06f106  
9389 f1040f  
9390 ff0e00  
9391 03fafa  
9392 ed0df0  
9393 02fbee  
9394 040304  
9395 05fff4  
9396 07edf7  
9397 f309f7  
9398 fa07fc  
9399 09f8fa  
9400 08fde7  
9401 0d0702  
9402 f30dee  
9403 ef02fb  
9404 faf204  
9405 05ecf6  
9406 0af00a  
9407 fd05f3  
9408 09f7f0  
9409 08080c  
9410 efe6f9  
9411 050fef  
9412 fbf2f2  
9413 f00b0c  
9414 05f4fe  
9415 ecf803  
9416 f40708  
9417 0f09ee  
9418 0cfefc  
9419 f90ef9  
9420 0cfcfc  
9421 effdfd  
9422 f9fd0b  
9423 f800f2  
9424 f6fb0c  
9425 fd07fd  
9426 090ff2

9427 0b0d03  
9428 fa0008  
9429 f30d01  
9430 0500ec  
9431 ef00ef  
9432 eeee0b  
9433 0af300  
9434 000fee  
9435 f7f90b  
9436 fbf5f8  
9437 ecf4f7  
9438 f5fcf2  
9439 fc5f4  
9440 f40002  
9441 10f8f0  
9442 05eceb  
9443 ff0b08  
9444 fefdf1  
9445 ef0ceb  
9446 efedef  
9447 0609f2  
9448 08e7fe  
9449 0509f6  
9450 03fbf2  
9451 ff020b  
9452 09f5f8  
9453 0a0f02  
9454 fef4ee  
9455 f40700  
9456 0cfd02  
9457 ef090b  
9458 0ef4f9  
9459 fcedf7  
9460 0905f8  
9461 040df2  
9462 f70106  
9463 0aeaf4  
9464 effa01  
9465 f5fd0f  
9466 0d0007  
9467 09fdfd  
9468 03f2ed  
9469 010a04  
9470 fefc10  
9471 09f7fa  
9472 f7fef6  
9473 f4f004  
9474 f60010  
9475 f410f7  
9476 f609f1  
9477 f40ff8  
9478 f9fff1  
9479 ededef  
9480 f304f7  
9481 01f00e  
9482 080afd  
9483 0ff603  
9484 f810fb

9485 08f3ef  
9486 0b0508  
9487 fef709  
9488 fafb0d  
9489 0104f4  
9490 fcedf4  
9491 06f3ef  
9492 f20204  
9493 05f501  
9494 f507f2  
9495 edf5fb  
9496 f2ee08  
9497 0ffdf9  
9498 fef4ec  
9499 000bee  
9500 fff6f7  
9501 050405  
9502 f60ffb  
9503 f4f4ec  
9504 f00af3  
9505 f10bf9  
9506 1105f4  
9507 05f80d  
9508 02f000  
9509 10f8f0  
9510 ef01f1  
9511 f001f8  
9512 01eb0c  
9513 120ef1  
9514 0000f1  
9515 f0fcff  
9516 04f1fa  
9517 f7f90d  
9518 fcfd7  
9519 0ae8f9  
9520 f60bf3  
9521 08feef  
9522 0012ed  
9523 0b0502  
9524 ef00f0  
9525 ed05e5  
9526 f3fdea  
9527 f20bfb  
9528 fdf1fb  
9529 edeb06  
9530 fd0efe  
9531 f2e90b  
9532 0300e9  
9533 030efe  
9534 0af602  
9535 f10df2  
9536 fb0df9  
9537 e807f4  
9538 fb0010  
9539 f5f2fe  
9540 04fafd  
9541 0af7f2  
9542 0cfe0b

9543 fdf600  
9544 0ef6f8  
9545 eceffd  
9546 fefef6  
9547 f9fcff  
9548 03f1f0  
9549 f0f704  
9550 01ecec  
9551 ecfe03  
9552 03ebff  
9553 02fb0b  
9554 fa0109  
9555 000af6  
9556 0b01e6  
9557 0bf003  
9558 fceefc  
9559 f6fa06  
9560 f8ff03  
9561 f500f7  
9562 00fb08  
9563 fd08ed  
9564 03f4f9  
9565 f602f5  
9566 fdff04  
9567 f506ed  
9568 f6eb00  
9569 010bef  
9570 eaedee  
9571 0f01f9  
9572 09f5f8  
9573 f7f504  
9574 02f50c  
9575 ece502  
9576 fdf9eb  
9577 01fbf9  
9578 fbfbf1  
9579 fa0c00  
9580 fbf3f7  
9581 eff509  
9582 fb05e1  
9583 e7e8e7  
9584 fcf3e4  
9585 e5e2ee  
9586 0b07f0  
9587 f707fb  
9588 f9e5f1  
9589 0803f6  
9590 ebebf6  
9591 f6eff5  
9592 fcf8f1  
9593 efe900  
9594 06f6f8  
9595 03eced  
9596 fbed09  
9597 fbf6ec  
9598 f9ea00  
9599 050107  
9600 fa0bf3

9601 01f60e  
9602 f20c06  
9603 05f414  
9604 f5f405  
9605 ebf2f6  
9606 ef09e9  
9607 0c06ed  
9608 0d00fe  
9609 05f9ee  
9610 f8ebe9  
9611 0909ef  
9612 fbfa0b  
9613 f5ef06  
9614 fb0eed  
9615 02f4e7  
9616 f80df9  
9617 fd03f4  
9618 08fd05  
9619 fde900  
9620 0904f5  
9621 f30509  
9622 030cfb  
9623 0306f1  
9624 ff0f08  
9625 f7f401  
9626 020009  
9627 f70cf8  
9628 08f306  
9629 e7fff5  
9630 f5ecfd  
9631 ede8ee  
9632 f1f908  
9633 0f00f6  
9634 020e07  
9635 fdf1ef  
9636 0c0002  
9637 03050e  
9638 feec07  
9639 fdf6ff  
9640 fe000c  
9641 fd0cfd  
9642 0d06fb  
9643 0e0a0c  
9644 f1fa0e  
9645 e5edf7  
9646 0ffcf2  
9647 0af006  
9648 0000ff  
9649 fd1106  
9650 000cfb  
9651 00090f  
9652 00f5fa  
9653 0ae900  
9654 e6f4e7  
9655 fbde02  
9656 0c00df  
9657 1b0705  
9658 0a0012

9659 041e26  
9660 1c130d  
9661 f4fb0e  
9662 f705f8  
9663 ea0a00  
9664 05f5d6  
9665 0ee7fc  
9666 00fdfe  
9667 0900d1  
9668 f3ed0e  
9669 120515  
9670 1e1206  
9671 12151e  
9672 f0e401  
9673 e6f6fb  
9674 0cfc15  
9675 fa012d  
9676 19fcdf  
9677 f60a09  
9678 f014eb  
9679 162003  
9680 091514  
9681 3d4d2c  
9682 fa1314  
9683 00f8fb  
9684 f304fd  
9685 ef1c20  
9686 0800fc  
9687 04e611  
9688 e7f300  
9689 ff21fe  
9690 1001fe  
9691 d50109  
9692 f4fef4  
9693 06f8fa  
9694 1b0a17  
9695 212928  
9696 17211f  
9697 d8e716  
9698 fde2e1  
9699 0f070e  
9700 f70e10  
9701 e100df  
9702 f600eb  
9703 dddcf9  
9704 e4e8fe  
9705 21fe00  
9706 f6fbdc  
9707 0afe1d  
9708 190018  
9709 24fc03  
9710 d71425  
9711 ec06ff  
9712 eafc01  
9713 f2d3d0  
9714 f1212c  
9715 14fcf7  
9716 0c1206

9717 0afe13  
9718 d8ee03  
9719 061e0a  
9720 36e2f9  
9721 15ff16  
9722 0d08ed  
9723 e7e1f2  
9724 fb051d  
9725 031ff9  
9726 fefc0d  
9727 090000  
9728 101606  
9729 f202e9  
9730 f2ee16  
9731 eae8f9  
9732 082306  
9733 f00e01  
9734 f901f7  
9735 0a09f8  
9736 1c1418  
9737 180a14  
9738 f31d10  
9739 e8dcf1  
9740 00e2f0  
9741 f1fc0d  
9742 121e0d  
9743 1b0119  
9744 0210fa  
9745 e809fb  
9746 fb0be9  
9747 fff7e8  
9748 060e00  
9749 09fe06  
9750 100402  
9751 001417  
9752 140d17  
9753 030a21  
9754 0000fd  
9755 f5ecee  
9756 db4f0  
9757 0c13d9  
9758 f4fcf9  
9759 f61108  
9760 070005  
9761 f90603  
9762 e7e0ee  
9763 051bf5  
9764 08ddef  
9765 01f217  
9766 faf104  
9767 130722  
9768 0d0803  
9769 151213  
9770 05f00a  
9771 f40bf1  
9772 01fddd  
9773 00fcf8  
9774 0e132f

9775 111bf6  
9776 000401  
9777 2c2c10  
9778 000be4  
9779 ff0216  
9780 dfec08  
9781 00e5f0  
9782 0000ec  
9783 fe1f14  
9784 13eafc  
9785 101324  
9786 020103  
9787 0bfcfa  
9788 e4e3db  
9789 c8dcf9  
9790 130de6  
9791 341f0d  
9792 d6fa06  
9793 1b321a  
9794 e6f3f8  
9795 030afc  
9796 f3d4f5  
9797 05fdf8  
9798 17f9c3  
9799 0a0111  
9800 f7fe13  
9801 0310e7  
9802 f8e900  
9803 070707  
9804 22e61c  
9805 251f13  
9806 dede2e  
9807 e8eb00  
9808 6a160b  
9809 004454  
9810 354e0a  
9811 0b12e8  
9812 e03218  
9813 2c1b09  
9814 faff12  
9815 ff323a  
9816 01ff06  
9817 19270b  
9818 18f805  
9819 0b0c0f  
9820 f4140f  
9821 0bf6e7  
9822 f1f315  
9823 1d00f0  
9824 fbc006  
9825 bfb9c1  
9826 f1fb02  
9827 0eeeeb  
9828 dd150e  
9829 1bf600  
9830 12020b  
9831 ddd544  
9832 04dae2

9833 1400f1  
9834 f5ebf5  
9835 0f420e  
9836 2c2b07  
9837 03e8f0  
9838 11cffc  
9839 eed8ff  
9840 faf2dd  
9841 f2ea07  
9842 1400f3  
9843 ff2c09  
9844 2e2904  
9845 11f6fe  
9846 f50d04  
9847 c3d6ca  
9848 c91312  
9849 efe6d6  
9850 0200f9  
9851 2148d4  
9852 edcef9  
9853 03eaf9  
9854 0df7f7  
9855 cadae2  
9856 e908df  
9857 21fbe3  
9858 f5120c  
9859 0ff3ee  
9860 1df9f9  
9861 02f0fc  
9862 f810fd  
9863 eefde2  
9864 c51109  
9865 fdece2  
9866 f224ee  
9867 cfdad5  
9868 f3fd04  
9869 0dea0c  
9870 25f4fd  
9871 001124  
9872 24181a  
9873 f0e8e1  
9874 0cdbe4  
9875 f4001a  
9876 0d0d06  
9877 f2fa12  
9878 05ebed  
9879 de17f5  
9880 ebf400  
9881 11e40c  
9882 f3fa17  
9883 00ed00  
9884 10e9f1  
9885 0e2117  
9886 1033f3  
9887 1b38fd  
9888 f400fb  
9889 ce0415  
9890 f6b1bf

9891 294be0  
9892 010026  
9893 100b15  
9894 03170b  
9895 ec0ce7  
9896 f00d11  
9897 f8f6fd  
9898 201300  
9899 030321  
9900 1a04f8  
9901 0cfd8  
9902 f406ee  
9903 0b09f6  
9904 e2fff5  
9905 f91df0  
9906 f0eae4  
9907 dc0811  
9908 cbea2e  
9909 f23dd9  
9910 ea19f8  
9911 ed06ef  
9912 f8140d  
9913 1c18f1  
9914 c4fc04  
9915 00eacf  
9916 0000fe  
9917 0bfbf0  
9918 fb1903  
9919 0c0ffe  
9920 faf0fa  
9921 f9f301  
9922 1ef8f2  
9923 2c0204  
9924 ff2022  
9925 200201  
9926 f7edef  
9927 1e19f6  
9928 09f7f4  
9929 fe1328  
9930 0bf721  
9931 f70719  
9932 f70611  
9933 f301e4  
9934 10e1e1  
9935 15f533  
9936 1ff9d8  
9937 ecdff5  
9938 e50308  
9939 04dae1  
9940 e4040d  
9941 1d03d3  
9942 ee1f11  
9943 1908ee  
9944 e6f606  
9945 180e06  
9946 06e90f  
9947 d3f832  
9948 0106de

9949 ff81e4  
9950 0e1a2b  
9951 210fed  
9952 f50100  
9953 1c13f5  
9954 020eff  
9955 1f1d10  
9956 1c1201  
9957 f40a0f  
9958 030a04  
9959 0d1afe  
9960 f9ea08  
9961 1509ef  
9962 f8fcec  
9963 d912fc  
9964 f5efeb  
9965 e3f411  
9966 180c01  
9967 08fff9  
9968 e7020a  
9969 1b01f2  
9970 fb110f  
9971 fbe9f4  
9972 e4e7db  
9973 ae0c08  
9974 f21dd4  
9975 f700cb  
9976 322313  
9977 02ecff  
9978 221c08  
9979 1b0b18  
9980 010912  
9981 f60c0a  
9982 18ede4  
9983 eefd09  
9984 180be0  
9985 fef821  
9986 23fcfe  
9987 eee4dd  
9988 ca072b  
9989 00dbd9  
9990 dcd304  
9991 01ff11  
9992 0d070a  
9993 0b05ff  
9994 15081c  
9995 070e0d  
9996 fd09f6  
9997 0604ec  
9998 10f1eb  
9999 120d15  
10000 07f40a  
10001 1606fe  
10002 f6e505  
10003 f7fcf9  
10004 0903fa  
10005 e6f312  
10006 0a08ea

10007	150f0c
10008	091907
10009	09fa13
10010	060313
10011	03f811
10012	fa1000
10013	031003
10014	fafa1a
10015	151e12
10016	0ff1f3
10017	efeb04
10018	0011fb
10019	16120a
10020	0cf227
10021	f30704
10022	fae0e5
10023	f10408
10024	151015
10025	1613fc
10026	06ea1a
10027	0a1803
10028	e8f4f3
10029	0911f4
10030	1f09f1
10031	0b0c0d
10032	0d4809
10033	10f40f
10034	f6061d
10035	1303f2
10036	11eff1
10037	0c240c
10038	1120f2
10039	19f2fe
10040	16f201
10041	ee0712
10042	1213e7
10043	f80f15
10044	edfeff
10045	01fafa
10046	0cee0a
10047	ebd2e2
10048	f213e9
10049	fb0211
10050	000007
10051	190810
10052	e2f115
10053	f613f8
10054	01fefb
10055	f708f5
10056	eb0a04
10057	ee0809
10058	01ebe0
10059	011115
10060	ef0004
10061	ef11fa
10062	e3f7f9
10063	06e4ee
10064	fbf6e1

10065	06fde3
10066	f2fbe9
10067	242500
10068	1c2811
10069	161ef8
10070	151414
10071	160b1d
10072	0befee
10073	042222
10074	1d08f1
10075	f50a35
10076	fef9ee
10077	05ff05
10078	0132ea
10079	e3e608
10080	002437
10081	4c12e5
10082	f1fb16
10083	23742a
10084	ef01f0
10085	e1f8fb
10086	02e1fb
10087	e3e30e
10088	090c03
10089	f0e5d4
10090	ecea0c
10091	0410e9
10092	03f5dd
10093	081220
10094	1711ff
10095	09fa0a
10096	ea0518
10097	1ff0f8
10098	fc0101
10099	0a1df3
10100	0af718
10101	060dee
10102	fee40e
10103	0b1f0c
10104	2508f4
10105	2c1129
10106	fd203a
10107	3e0005
10108	12fb35
10109	f8e418
10110	f6eff2
10111	1efbfe
10112	00e713
10113	fa1d09
10114	110e06
10115	f4eefc
10116	d81f24
10117	fd01ca
10118	00f403
10119	001df3
10120	f80e22
10121	121a10
10122	27e9e5

10123 de0d39  
10124 0a37fb  
10125 250be2  
10126 150709  
10127 fde9f9  
10128 faf8f0  
10129 f0e6f8  
10130 f40403  
10131 060b1f  
10132 fb1528  
10133 f109d1  
10134 180af6  
10135 131c0d  
10136 060923  
10137 13fc09  
10138 06fdff  
10139 f6fde5  
10140 0305fb  
10141 ebf2ff  
10142 feee14  
10143 fc090e  
10144 02e7ef  
10145 e40100  
10146 f8f9f0  
10147 eeee07  
10148 02fffe  
10149 eae9ed  
10150 0ff7f7  
10151 0d0e0c  
10152 fc0b02  
10153 e5f7ef  
10154 f20df9  
10155 fc140c  
10156 050915  
10157 1c021b  
10158 f2daef  
10159 e30b10  
10160 f0f9eb  
10161 04de00  
10162 f40c10  
10163 00ece1  
10164 e0f704  
10165 1308fd  
10166 ecdbf0  
10167 02ee13  
10168 fff7ee  
10169 0508f5  
10170 f20303  
10171 0bf9eb  
10172 f7e208  
10173 0012f4  
10174 fc0fdb  
10175 06000c  
10176 0b0e08  
10177 080015  
10178 f205eb  
10179 080018  
10180 04f9f8

10181	fc111d
10182	0afd25
10183	2709f7
10184	0000f8
10185	07f202
10186	ecfefafa
10187	fc0008
10188	f3fff0
10189	e60014
10190	05ea0c
10191	e61013
10192	2500e5
10193	2502fc
10194	d5d801
10195	f522ea
10196	e5d8ef
10197	01130d
10198	0df0fb
10199	f71c10
10200	07f9e5
10201	e6f913
10202	1405f1
10203	0c052f
10204	440620
10205	fd1916
10206	173914
10207	0b212f
10208	1df101
10209	ddf80d
10210	161109
10211	20ff0f
10212	e1e12a
10213	26fd15
10214	2206dc
10215	f9ede6
10216	06261f
10217	3823e2
10218	090114
10219	18f0f6
10220	eedf3
10221	fdd9f5
10222	e4e3f7
10223	faf3b8
10224	e3d707
10225	10ed00
10226	f1fb04
10227	ff0911
10228	ebf81b
10229	262003
10230	f7f1f3
10231	190b1b
10232	f70e08
10233	fc07ca
10234	e7e207
10235	01fcf0
10236	d5051f
10237	fb200b
10238	f2de00

10239	222500
10240	d6072c
10241	423a07
10242	060136
10243	efec24
10244	13190a
10245	17e9e7
10246	d0d6e4
10247	d9fbd6
10248	fc12e9
10249	1bf3fe
10250	0b0516
10251	c1140c
10252	0dfef5
10253	e9dc0d
10254	f70916
10255	0f00ec
10256	362918
10257	150b3a
10258	1e202c
10259	0bfc04
10260	fe0be8
10261	e103ec
10262	050012
10263	0be4fc
10264	ef0cf6
10265	f30ef9
10266	fbf004
10267	0d07f1
10268	131df3
10269	19f70a
10270	03f9fd
10271	142d13
10272	0d0603
10273	1af7ec
10274	d8fe22
10275	edf102
10276	03f5f5
10277	eb050b
10278	1a0ee4
10279	e5f4fe
10280	0f05fc
10281	d7f800
10282	0d12f0
10283	f4eee7
10284	f303f9
10285	ee0dfd
10286	eb000e
10287	1c09ee
10288	0cf3ff
10289	00021b
10290	11f50b
10291	2dfe04
10292	ea0217
10293	15d017
10294	f6f91d
10295	1c22fd
10296	06efed

10297	0c1e02
10298	d0e2e6
10299	ff06f7
10300	f4ddee
10301	fffb09
10302	e3fe0e
10303	f5f3e3
10304	d50612
10305	f7fff5
10306	d4e1e6
10307	fd0efa
10308	f5edda
10309	000202
10310	fcfd11
10311	e918fa
10312	19f2e3
10313	28111b
10314	fb0eee
10315	1b12fd
10316	14cbf3
10317	fa0005
10318	0000ee
10319	ebfcec
10320	f6fafc
10321	f3f50a
10322	0d04f7
10323	f701fc
10324	fe02f3
10325	0806f6
10326	f3fafd
10327	f600f8
10328	f9f201
10329	faff02
10330	f7f3f6
10331	feecfe
10332	08e9e7
10333	fb060b
10334	f700f6
10335	04f4ed
10336	0bf807
10337	ee09ee
10338	0df4fe
10339	0af305
10340	f3ebfb
10341	f40b0d
10342	0ff6f4
10343	fffbff
10344	01fcec
10345	f005ff
10346	fafef9
10347	e9faeb
10348	0f05fc
10349	0001fc
10350	fb07f8
10351	0500ec
10352	f2e6ea
10353	ec06fa
10354	ff0100

10355	0c08ec
10356	0205fb
10357	0a09fe
10358	f90ae9
10359	fbeef7
10360	f3fe0a
10361	0009f2
10362	fae805
10363	ef02f1
10364	11ebe5
10365	e80000
10366	05fbe8
10367	000003
10368	f7ed00
10369	08f708
10370	07f903
10371	ef13f6
10372	000fef
10373	eefbfa
10374	ecfcef
10375	ebf711
10376	f216f2
10377	07e702
10378	f70600
10379	f80602
10380	0a0fec
10381	080702
10382	fffcf6
10383	03fef1
10384	ea03ec
10385	03eff5
10386	ea040f
10387	0afdeb
10388	e8f902
10389	f7edfd
10390	f6fe02
10391	02f3f5
10392	09f1f1
10393	f1ff00
10394	eee7e7
10395	ef0004
10396	ec020b
10397	ea0bf1
10398	0104eb
10399	09edf6
10400	f304ea
10401	08ffe8
10402	00ecf1
10403	04faf9
10404	faeffa
10405	fff0f1
10406	00fae7
10407	0f03f3
10408	ff0307
10409	f20d0a
10410	f3f9f1
10411	f9f205
10412	ef0106

10413 09f7f0  
10414 02f00a  
10415 f6f1ea  
10416 f2f3ea  
10417 04f40b  
10418 f20af5  
10419 fde701  
10420 f6f5e8  
10421 fc00fd  
10422 0400f4  
10423 fff800  
10424 edf7f0  
10425 0709ef  
10426 fff1f4  
10427 f8ebf0  
10428 08e800  
10429 e8000f  
10430 ecdfdd  
10431 f3fefc  
10432 0aef7  
10433 f5ef0a  
10434 f606ea  
10435 060507  
10436 04f4f8  
10437 06eafc  
10438 f2e809  
10439 f7020b  
10440 0404fd  
10441 eae9fb  
10442 eaec0b  
10443 04ecfa  
10444 0aef0  
10445 ebf06  
10446 fc08e3  
10447 f4e902  
10448 02fffa  
10449 12f506  
10450 fafe07  
10451 0bf6fa  
10452 0000f0  
10453 0af308  
10454 04fc03  
10455 101408  
10456 0908f6  
10457 f103f9  
10458 f5fef6  
10459 f9fc0e  
10460 fe00f3  
10461 000908  
10462 f7f60e  
10463 fa0006  
10464 070b0a  
10465 f2f6f4  
10466 f3f7ee  
10467 f3fbfe  
10468 000fff  
10469 0b01ec  
10470 04f7ee

10471	ff0202
10472	06fbee
10473	fd0008
10474	01f30c
10475	ed0801
10476	08f50e
10477	ee0e01
10478	ecfbf7
10479	000afd
10480	f7f3f1
10481	03fd0f
10482	fdf9f7
10483	f7fd0b
10484	fafdee
10485	0df708
10486	eb0bf b
10487	f608fb
10488	0b0805
10489	ed0005
10490	fef0fb
10491	03fc10
10492	fbf8ed
10493	fe03f1
10494	05fd0b
10495	000506
10496	f8f407
10497	f309ec
10498	eed0f
10499	eef8f3
10500	fa0909
10501	07faf6
10502	00090d
10503	f70f02
10504	fd01ef
10505	f9f7ff
10506	08fcfd
10507	f30c01
10508	edf7f2
10509	0f02f6
10510	fdf1fb
10511	07050d
10512	08f1f9
10513	02f604
10514	f30df1
10515	f2eef9
10516	fc070b
10517	ef0002
10518	fdbfba
10519	fd04eb
10520	0b06ff
10521	ff0906
10522	fcfb06
10523	0d0cff
10524	fbfef2
10525	06effc
10526	f6ed0c
10527	f109fe
10528	edee09

10529	02fdfd
10530	f1fcee
10531	0e00fe
10532	010ffe
10533	04fa11
10534	f8f7f0
10535	eb07fa
10536	fe0ef2
10537	03f90d
10538	f4fa0f
10539	070610
10540	f5ef02
10541	05fff2
10542	0509fd
10543	0005f3
10544	f8faf9
10545	f8fafb
10546	f804fe
10547	0704f0
10548	ecf7ea
10549	0c050b
10550	f8ebfc
10551	ff07f6
10552	090ef3
10553	f6fb0b
10554	0a0b0d
10555	02fcf3
10556	02f7ff
10557	04fe02
10558	ebf4fb
10559	eced07
10560	ff0ff2
10561	fdf5f3
10562	fd02f7
10563	0d0cef
10564	000dfd
10565	f0f801
10566	f10aec
10567	0b0bff
10568	fd0308
10569	fbebfb
10570	fff6ff
10571	0cff0e
10572	0b04ed
10573	0701ec
10574	06fdee
10575	ef05fd
10576	ef080b
10577	fc00ec
10578	f6f605
10579	0fff04
10580	02f506
10581	f70bf4
10582	0b09ef
10583	00ef0c
10584	f9eff0
10585	09fafa
10586	000003

10587	fe0afe
10588	f1f809
10589	ff00f3
10590	f2f40c
10591	fa0f08
10592	27fb08
10593	20ffe8
10594	e50c10
10595	0f1803
10596	071d02
10597	e4e5f1
10598	d9220c
10599	00f0e8
10600	070f01
10601	060215
10602	20f40e
10603	291012
10604	ea031b
10605	f827c6
10606	dbecff
10607	09f5eb
10608	0b061c
10609	d8f108
10610	f6fdfc
10611	0702f0
10612	3500e5
10613	e4101c
10614	effe20
10615	e4f823
10616	fef5ef
10617	f7efed
10618	ea0cdd
10619	b5cccf
10620	ec0006
10621	09f501
10622	090ffd
10623	090f06
10624	09fc10
10625	f9141a
10626	291e08
10627	040701
10628	121a10
10629	02effa
10630	162be6
10631	050901
10632	021104
10633	05eae3
10634	e9dbf9
10635	020f04
10636	35010e
10637	ff0ef4
10638	20edf3
10639	f30432
10640	ed12fa
10641	12f9f8
10642	fd0d11
10643	fdec36
10644	110813

10645	02ebf1
10646	e8d9ee
10647	f41c0a
10648	17efe2
10649	0a11ea
10650	071c3e
10651	121b0e
10652	21fb0c
10653	0ef908
10654	071c0f
10655	0f0700
10656	f6f5f1
10657	27f906
10658	0d1427
10659	252e0e
10660	10070c
10661	060a10
10662	00fb03
10663	e50714
10664	0af8e0
10665	0aef01
10666	fb0414
10667	031a06
10668	0702f9
10669	02f00a
10670	001517
10671	effb0c
10672	f6090d
10673	00f409
10674	f3f807
10675	0c0ce5
10676	1c0b05
10677	0318f8
10678	fa130f
10679	0319fa
10680	fbde02
10681	1515fe
10682	090af8
10683	f2f500
10684	170cfc
10685	13feff
10686	14fcf6
10687	fc0102
10688	12040e
10689	e3f8ea
10690	ea01de
10691	0501f6
10692	f60cfa
10693	132029
10694	fe160c
10695	e90a10
10696	13f409
10697	f9e6e7
10698	0e1907
10699	fe02fe
10700	f21910
10701	0be811
10702	e60223

10703	f21bf2
10704	1e11ff
10705	0a041f
10706	f707ee
10707	e2fa04
10708	ff14ec
10709	14f30a
10710	11050f
10711	001408
10712	fdff39
10713	00e7ed
10714	e916f1
10715	f8dbf6
10716	e9db0f
10717	1c0fe3
10718	e715f0
10719	02f2ff
10720	000005
10721	fef505
10722	f50afc
10723	0af6f1
10724	0900fc
10725	f605f3
10726	f5e20a
10727	070015
10728	17f4dc
10729	e8ef15
10730	1c1904
10731	1700e5
10732	010018
10733	f4f5f8
10734	0f07f2
10735	141b0c
10736	0df91c
10737	0efb0c
10738	002413
10739	1a09c1
10740	d5e70b
10741	0c0a13
10742	2622ee
10743	041a29
10744	0306f8
10745	f3eb11
10746	d4dbf2
10747	03f80d
10748	f3c3e8
10749	eef912
10750	03d9c4
10751	e5e6f3
10752	fd1618
10753	0f1516
10754	f5e521
10755	fbff17
10756	05feeb
10757	f1151f
10758	f81902
10759	fafcf8
10760	d80604

10761 fd0609  
10762 2d16f3  
10763 e9fb19  
10764 0b12ec  
10765 11fadf  
10766 24190b  
10767 171e16  
10768 f72c30  
10769 f9f9ea  
10770 02f7fc  
10771 f30a07  
10772 fd07f9  
10773 e2d7e7  
10774 f4f8f0  
10775 f7e2f3  
10776 f00113  
10777 310ad5  
10778 f8ee0e  
10779 f30815  
10780 e60503  
10781 1efef7  
10782 f5fc1f  
10783 fc0515  
10784 f81a05  
10785 f1ebfd  
10786 ff1205  
10787 150409  
10788 ed1e20  
10789 0b18f4  
10790 daec09  
10791 161a3f  
10792 36f9fd  
10793 02123d  
10794 27220e  
10795 d8ce0a  
10796 17fd0e  
10797 ee091d  
10798 080e0f  
10799 0b0d0f  
10800 0c090d  
10801 f708fd  
10802 18eeed  
10803 ef0af0  
10804 29180e  
10805 020725  
10806 040b0e  
10807 021212  
10808 0f1a15  
10809 1b151c  
10810 ea0af8  
10811 e80506  
10812 f7ecea  
10813 011f04  
10814 02ff01  
10815 02180a  
10816 00edfb  
10817 f8fb11  
10818 02f9e8

10819 0909ff  
10820 fd0c10  
10821 0e101f  
10822 0e0721  
10823 fdf3fb  
10824 fff40c  
10825 151500  
10826 051325  
10827 21fbf8  
10828 fde1f8  
10829 e5f5e9  
10830 0e0004  
10831 f4daf5  
10832 101118  
10833 20fff2  
10834 ed1803  
10835 fe05eb  
10836 f3e900  
10837 fe0020  
10838 070902  
10839 02030b  
10840 fd1323  
10841 1406db  
10842 02f5f9  
10843 0d0dee  
10844 f902fd  
10845 fcf0d1  
10846 e0fd25  
10847 2be0ee  
10848 ede700  
10849 002607  
10850 090100  
10851 e9121f  
10852 10fe09  
10853 18fcf3  
10854 0000f9  
10855 34fafb  
10856 d5e51d  
10857 2e17f0  
10858 dcdae  
10859 f6081f  
10860 060005  
10861 f3fa09  
10862 231de6  
10863 0ef7e6  
10864 c8162e  
10865 23fdde  
10866 e0f128  
10867 160fe8  
10868 ecf0e3  
10869 f2f714  
10870 17e5ef  
10871 120d1f  
10872 b0ddf4  
10873 01fc0f  
10874 dad103  
10875 1d0208  
10876 1b06e4

10877	141e26
10878	07e31b
10879	110c1a
10880	02280f
10881	131221
10882	33d018
10883	ed0b12
10884	0d4200
10885	2f03ee
10886	eaff37
10887	0c3216
10888	1decf1
10889	c5d303
10890	05070f
10891	efdade
10892	e4f904
10893	07f2f7
10894	23fa0c
10895	02ede9
10896	1d04fa
10897	d3aae9
10898	0307d3
10899	bccfe5
10900	f30e00
10901	0ffdf7
10902	effc1c
10903	0ee413
10904	0d071a
10905	e000f2
10906	0bf7fd
10907	3adb03
10908	e5cb21
10909	172ac2
10910	0b14ea
10911	121519
10912	200f1e
10913	180535
10914	f0e5df
10915	cf00ff
10916	eb0c0c
10917	e8dae7
10918	15eeff
10919	0409ee
10920	e6280d
10921	0b0f0d
10922	1109fa
10923	cdf00e
10924	340cb9
10925	cef1f1
10926	09e7db
10927	011507
10928	270816
10929	211d04
10930	0effe4
10931	e6c313
10932	f716fd
10933	03ddd2
10934	e00003

10935 00eadb  
10936 fa131b  
10937 0904d9  
10938 001911  
10939 bebbec  
10940 fd06ff  
10941 e2cbe0  
10942 f20205  
10943 fbfd8  
10944 0af3f4  
10945 f8f3eb  
10946 fe0a10  
10947 141b21  
10948 11e3d5  
10949 f31a1c  
10950 faeeef  
10951 e8f5fb  
10952 fe0907  
10953 e2cdea  
10954 f50ff9  
10955 f71402  
10956 dddd00  
10957 18f5d8  
10958 e3e000  
10959 21fae8  
10960 01f4f4  
10961 050a1b  
10962 1eed04  
10963 e8e325  
10964 e33818  
10965 28e0f1  
10966 f2e330  
10967 1928eb  
10968 d7e7e2  
10969 f707f0  
10970 f0e8e8  
10971 fdee18  
10972 ec20ff  
10973 110bbe  
10974 d6f20f  
10975 0c06f7  
10976 e4e4f9  
10977 eb14e1  
10978 07e400  
10979 11f504  
10980 d3f406  
10981 efd5fa  
10982 0fe8eb  
10983 03e8ce  
10984 f61908  
10985 000e09  
10986 05f805  
10987 1b0cee  
10988 0000f2  
10989 0000e8  
10990 fd07fe  
10991 10f100  
10992 0df0e2

10993	01f706
10994	f7f7f3
10995	0b0f03
10996	04f4fb
10997	e4e8f6
10998	f1d8d2
10999	f101e9
11000	f4eee7
11001	010908
11002	1bf604
11003	f20afd
11004	010000
11005	da0005
11006	08d3c7
11007	fcdf2d
11008	4b47eb
11009	0110e9
11010	101815
11011	f6fefe
11012	101be6
11013	f6f712
11014	011915
11015	38e4b8
11016	bad826
11017	0425d4
11018	10faf9
11019	0de015
11020	0e2c17
11021	0b1c04
11022	13eb0d
11023	f9f9fe
11024	f21b06
11025	00e0df
11026	e2e6fa
11027	0bf6d0
11028	e1f10e
11029	00fffa
11030	d3eb03
11031	0ffcbb
11032	fac22c
11033	32421c
11034	020cef
11035	06f0f4
11036	e406e2
11037	1c05e5
11038	de03f9
11039	03fff1
11040	0cfb11
11041	1d0014
11042	e4ec08
11043	1a1714
11044	15dfd
11045	040e21
11046	dafb02
11047	1ee800
11048	fefc0b
11049	f1ee0b
11050	f2faff

11051 ee0ff0  
11052 fdfdf2  
11053 e0ef04  
11054 091105  
11055 cdecdd  
11056 1600fc  
11057 bfd020  
11058 252b20  
11059 32f2eb  
11060 0f3624  
11061 e11d13  
11062 0d0d03  
11063 d5f018  
11064 05e9e4  
11065 ddf9fd  
11066 0a010a  
11067 23dff0  
11068 ec130e  
11069 f514f4  
11070 e2e802  
11071 11f804  
11072 d7c204  
11073 191cf9  
11074 0aeecf  
11075 d81807  
11076 f30910  
11077 0d0edb  
11078 f3f50f  
11079 01fefaf  
11080 f1e4ff  
11081 ecf0f0  
11082 f7eff6  
11083 f5ecf8  
11084 0b01ff  
11085 f70408  
11086 15100b  
11087 edfc01  
11088 f601ff  
11089 efd1d7  
11090 0502f6  
11091 0c0dfb  
11092 f7dbff  
11093 001b2c  
11094 09f412  
11095 e90c08  
11096 eb0900  
11097 edf6ef  
11098 f2ebf8  
11099 110ae9  
11100 0cf50c  
11101 f81107  
11102 f7e4ff  
11103 10fcf2  
11104 04e4f6  
11105 d8fc0c  
11106 f6fafa  
11107 1cedf0  
11108 e5eff4

11109 eb0801  
11110 eeefef6  
11111 120915  
11112 00fcf7  
11113 dbf70c  
11114 11c1bb  
11115 cc1e1a  
11116 2afbf8  
11117 Odd309  
11118 ecf314  
11119 Oa0503  
11120 0406e8  
11121 f2f41d  
11122 0000e4  
11123 fc0a00  
11124 e20bea  
11125 fb1af4  
11126 f206ed  
11127 0107f1  
11128 11ef0b  
11129 eef912  
11130 f410f0  
11131 eefbee  
11132 d9f3f2  
11133 edea04  
11134 03eff8  
11135 ff02de  
11136 e81101  
11137 0512f8  
11138 fdf116  
11139 0bf40b  
11140 1ff507  
11141 0e1d46  
11142 29e400  
11143 fbefb6  
11144 161200  
11145 f4f901  
11146 1115e2  
11147 0207f0  
11148 141dd9  
11149 ee1d20  
11150 f30b14  
11151 2ff7d4  
11152 e021d1  
11153 f60d09  
11154 f4f1e6  
11155 ecd316  
11156 Ofef518  
11157 12100c  
11158 1ef3e7  
11159 dd000d  
11160 e4fc0f  
11161 f9eff7  
11162 fd1d0a  
11163 ff07fd  
11164 00e518  
11165 29fb07  
11166 f5fe35

11167 1219ec  
11168 fdfb11  
11169 f6f621  
11170 e6eafd  
11171 fd1107  
11172 170c02  
11173 f8f4e3  
11174 11f0f0  
11175 283214  
11176 10150c  
11177 2828df  
11178 ff180e  
11179 1a0b08  
11180 080716  
11181 02d1ec  
11182 180617  
11183 0907fc  
11184 fe2121  
11185 13e407  
11186 02f006  
11187 d63108  
11188 0b1804  
11189 fbe5f2  
11190 0000ef  
11191 f810f5  
11192 353905  
11193 eff41a  
11194 141906  
11195 0b0a10  
11196 f61afd  
11197 cafbf9  
11198 ea0608  
11199 f8fbfe  
11200 07faf2  
11201 ddf020  
11202 f41113  
11203 02fc07  
11204 0ff7f2  
11205 091e0b  
11206 051003  
11207 edf5f6  
11208 f2f60b  
11209 120a16  
11210 1af7e8  
11211 e700f0  
11212 fc04fe  
11213 1ff3fc  
11214 17eb0a  
11215 0ae300  
11216 f8f8fe  
11217 f1edf6  
11218 0de2fd  
11219 f41216  
11220 1210f9  
11221 131310  
11222 ff140f  
11223 cbfbf9  
11224 15e2f7

11225	260a0c
11226	e3130d
11227	fc0a08
11228	00ee0c
11229	24fbf6
11230	140e24
11231	1a13e7
11232	07faf0
11233	f7f912
11234	1901fb
11235	12f6fa
11236	eb0bec
11237	ea0015
11238	fdf900
11239	e90b05
11240	fb08ee
11241	05ef00
11242	f2f30e
11243	01efe6
11244	070ae5
11245	1a15f2
11246	000bf6
11247	faf30f
11248	e0020c
11249	050fe9
11250	fdc6f0
11251	cf14f3
11252	e31222
11253	f1d9f6
11254	1812e8
11255	f0ff1d
11256	000002
11257	0eedf0
11258	1313f7
11259	ef1011
11260	08f2ee
11261	e7f0f6
11262	1befe9
11263	0bfe20
11264	1d2124
11265	ebe903
11266	020614
11267	08fef3
11268	e90d0b
11269	f4f6f7
11270	f5ece4
11271	001811
11272	f90824
11273	ff0c17
11274	010702
11275	080009
11276	f5ed01
11277	e3e919
11278	dd1ff5
11279	18f0fe
11280	020214
11281	2109eb
11282	1ff5eb

11283	d6f025
11284	361700
11285	12f500
11286	fdea14
11287	e4071f
11288	05efdc
11289	dcb7f4
11290	f906ef
11291	0b0d0a
11292	110ff8
11293	ea0313
11294	fb0418
11295	0f0301
11296	f5fbed
11297	f304fb
11298	0b01f8
11299	0c0006
11300	0d0c04
11301	0c1dfb
11302	e3f70d
11303	e210fa
11304	03fff9
11305	030a0f
11306	13fc10
11307	05060b
11308	0517e8
11309	122428
11310	271209
11311	f6fe00
11312	f21111
11313	dc201c
11314	22ddfe
11315	05fa0e
11316	f603fd
11317	f60f00
11318	13f217
11319	fd0403
11320	161ef5
11321	1419fe
11322	1a0be6
11323	0f011f
11324	fbfc00
11325	2a2207
11326	f0ee00
11327	ea0a10
11328	eafa00
11329	11c2bd
11330	f0c73f
11331	2c2428
11332	f20ef6
11333	01ff00
11334	eff000
11335	e00903
11336	f5080d
11337	f9f8fa
11338	3febeb
11339	f60630
11340	fcf300

11341 1a0cf6  
11342 0c0c13  
11343 0d130c  
11344 e7f7df  
11345 f3eb11  
11346 ef01f2  
11347 1c0a1b  
11348 fb110b  
11349 0305f4  
11350 f5021b  
11351 00f100  
11352 f00001  
11353 eff5f0  
11354 fb0e0a  
11355 0f21fe  
11356 150b10  
11357 f909f3  
11358 121604  
11359 0dff0c  
11360 1208f9  
11361 fee0fa  
11362 fid9f3  
11363 0cfc0f  
11364 2e0522  
11365 f00025  
11366 ee0f1b  
11367 fbf7f1  
11368 f1f0fa  
11369 0d15e1  
11370 f800fd  
11371 ff1604  
11372 183e0b  
11373 fff50d  
11374 16000c  
11375 000000  
11376 f7fa19  
11377 11fb09  
11378 0ff600  
11379 fa0411  
11380 edfff9  
11381 0203fe  
11382 edfdfa  
11383 012518  
11384 0b13fd  
11385 010105  
11386 fb0100  
11387 0df7ed  
11388 17ffef  
11389 0b0cf8  
11390 00000e  
11391 f7fa00  
11392 fbfc07  
11393 f3fcf4  
11394 f7fff7  
11395 f9f306  
11396 fcf815  
11397 f002f1  
11398 f3fc11

11399	f50ff7
11400	0604fa
11401	04ec0e
11402	f60df7
11403	fef2fd
11404	f30009
11405	0dedfa
11406	fe05fd
11407	000800
11408	0b0802
11409	10f5fb
11410	060700
11411	0af8f2
11412	fd09f4
11413	02fafc
11414	0bf3ec
11415	0200f3
11416	f8f109
11417	eff107
11418	0b00f4
11419	f4fdff
11420	06f910
11421	fbf900
11422	fe02f1
11423	fb08f1
11424	f309ef
11425	fbf4f5
11426	fd1000
11427	0f0eee
11428	04f0ef
11429	06f600
11430	ecf505
11431	fc05f8
11432	0f00fa
11433	f9f7ed
11434	0af206
11435	fb0900
11436	08fc08
11437	f3fcfe
11438	eef002
11439	fbf3fe
11440	0600f0
11441	0d00eb
11442	f20107
11443	0ef607
11444	0aecfa
11445	f60f00
11446	02f1ed
11447	fbf400
11448	ff0905
11449	f9f408
11450	f301f9
11451	04f4fb
11452	f9f5f3
11453	020b00
11454	09effe
11455	ef0a0c
11456	0c09f1

11457 0e06ff  
11458 0ef8fd  
11459 f2f602  
11460 0b01f3  
11461 030aef  
11462 03eff5  
11463 06f101  
11464 fb0ef3  
11465 f00410  
11466 fa0e03  
11467 f506ef  
11468 f6f8ee  
11469 f4f807  
11470 0a06f9  
11471 0303f5  
11472 0c0801  
11473 090ff8  
11474 04030f  
11475 12ffff  
11476 eceff1  
11477 00edfa  
11478 fefd00  
11479 050def  
11480 03f600  
11481 efeff7  
11482 07eef6  
11483 ef040c  
11484 f00806  
11485 fd09ef  
11486 fdfcff  
11487 edf7f1  
11488 020dfb  
11489 ecf7ee  
11490 0406fb  
11491 ed0efa  
11492 ec0cf5  
11493 f403f9  
11494 ec0704  
11495 f5070c  
11496 090406  
11497 f40af1  
11498 030c02  
11499 080b00  
11500 fbf9f5  
11501 f2fdfd  
11502 f609f3  
11503 ee00fa  
11504 f3ecfd  
11505 fef302  
11506 fbf10c  
11507 fefe09  
11508 ff07f4  
11509 eb0200  
11510 f20ffc  
11511 0303fd  
11512 f70e06  
11513 0b0909  
11514 08fbfa

11515 f1f504  
11516 0c1001  
11517 eef500  
11518 f20a0f  
11519 010c06  
11520 0ef1ed  
11521 f3f6f4  
11522 f80af3  
11523 fc00f7  
11524 000005  
11525 02fa10  
11526 00ef0c  
11527 f400f4  
11528 00ff00  
11529 00fef2  
11530 f7f3f3  
11531 09f103  
11532 0401fa  
11533 0c0a0b  
11534 ececfa  
11535 0e0ff2  
11536 f30df0  
11537 00f007  
11538 fa0df4  
11539 fafef6  
11540 fa02f1  
11541 0508f6  
11542 f404fb  
11543 06f0f9  
11544 0c01ff  
11545 00ecf4  
11546 f8eff3  
11547 ed0002  
11548 0eef0  
11549 f5fef9  
11550 f5f7f8  
11551 02f300  
11552 0c0001  
11553 fb0bee  
11554 0cf2f5  
11555 f61110  
11556 fef7f6  
11557 0d07ec  
11558 ed0e0b  
11559 f9070a  
11560 10fb04  
11561 f0fdf8  
11562 f9f1fd  
11563 f6f7fa  
11564 09fb0a  
11565 fbfb08  
11566 040c02  
11567 f7100c  
11568 09ecf5  
11569 08f2fe  
11570 05f60f  
11571 08050a  
11572 fb10f9

11573	080bf7
11574	f1eff2
11575	03fefe
11576	fffbef
11577	fbfcff
11578	fe0203
11579	f2f00b
11580	ef0b08
11581	00ec0d
11582	f8eef6
11583	f6000b
11584	fe0af8
11585	060cf6
11586	0dfdf6
11587	0e0f08
11588	0708f4
11589	f8ed06
11590	0f0400
11591	f307f5
11592	f8f408
11593	0afc07
11594	020505
11595	0705ef
11596	fbfefb
11597	f6fbf7
11598	ec0e10
11599	f2f6fc
11600	f30d0f
11601	f50c0c
11602	f6090b
11603	0608f1
11604	010005
11605	0c02f4
11606	f80004
11607	edf003
11608	f60c03
11609	f8f8e7
11610	ef02f4
11611	0305f1
11612	02fe08
11613	01ecff
11614	f3feed
11615	0bf5f1
11616	050100
11617	09f2f4
11618	f7ee0a
11619	0a090a
11620	ec07ee
11621	00fdf6
11622	08f6ee
11623	0aee04
11624	0afff0
11625	fdf3ed
11626	020bf6
11627	10edf5
11628	0b06f1
11629	03fd0c
11630	ed0307

11631	0600f7
11632	ed0705
11633	f8f8fa
11634	f0ef01
11635	f9eef3
11636	f7fdee
11637	fff9ec
11638	08fdfd
11639	f2ef0f
11640	f5fc08
11641	fcfc05
11642	f9f8fa
11643	ecfdf6
11644	06fc07
11645	ecedf8
11646	f6f103
11647	0efdf2
11648	08fa07
11649	0aecf6
11650	eefcee
11651	f3ef0a
11652	011112
11653	09f40b
11654	fd02f3
11655	080009
11656	f504f5
11657	f3ec04
11658	0000f7
11659	1103f1
11660	e0070a
11661	f600f9
11662	ef0bfa
11663	0c19f9
11664	1a1029
11665	f8f316
11666	f70009
11667	e8fd6
11668	031512
11669	efe9f8
11670	e90007
11671	1805d3
11672	170b04
11673	070100
11674	07010a
11675	170004
11676	040cfb
11677	0a1bf5
11678	06030d
11679	10f609
11680	1d0e0b
11681	03f720
11682	130eec
11683	fc0305
11684	0b19ec
11685	f40823
11686	09fe25
11687	160e16
11688	000acf

11689 efe9e9  
11690 02202c  
11691 fa0cfc  
11692 f0e028  
11693 1c0606  
11694 040de0  
11695 f9000e  
11696 03f70c  
11697 04f900  
11698 0f0bf6  
11699 14fbe9  
11700 07ff25  
11701 09e800  
11702 1724f8  
11703 271cf9  
11704 0c000d  
11705 11ed10  
11706 e2e704  
11707 f6f6d8  
11708 fe130e  
11709 f310fc  
11710 05f90f  
11711 030623  
11712 f4e8eb  
11713 150ebe  
11714 181c10  
11715 4af6f8  
11716 11fc37  
11717 0ff5f7  
11718 03fa12  
11719 ee1de8  
11720 f201f2  
11721 06f8f7  
11722 0efcf3  
11723 ed0c0d  
11724 21162d  
11725 f2c70e  
11726 eff60f  
11727 ef05eb  
11728 0dfbf2  
11729 e5f012  
11730 1a1508  
11731 061e2c  
11732 0f3edf  
11733 bdee01  
11734 16f100  
11735 d81115  
11736 02f6da  
11737 eaebf6  
11738 eb0413  
11739 180c12  
11740 fe0902  
11741 0d0df3  
11742 0528ff  
11743 f5f606  
11744 0f0f0d  
11745 09fbf1  
11746 000cfa

11747 ebf7ee  
11748 e5f1e1  
11749 fef4fa  
11750 faf9f5  
11751 0ff408  
11752 f4e70f  
11753 fc0a0b  
11754 09f3fa  
11755 17050f  
11756 140df9  
11757 04110f  
11758 fbf6f3  
11759 f224ff  
11760 fd1100  
11761 f90b08  
11762 d2e2fa  
11763 f5fbe7  
11764 f40f14  
11765 2802f4  
11766 f70b0c  
11767 e4ede7  
11768 130812  
11769 f5f50e  
11770 27f90b  
11771 1dff01  
11772 ee11fa  
11773 f809f8  
11774 060000  
11775 00140c  
11776 000e00  
11777 f7fc12  
11778 f60b12  
11779 ee07ec  
11780 e5f903  
11781 0107eb  
11782 00f216  
11783 f20ff9  
11784 ee0115  
11785 001105  
11786 06fcdd  
11787 eeda0d  
11788 d9000c  
11789 fde50d  
11790 0112f6  
11791 f912fe  
11792 000005  
11793 f102f8  
11794 e90b0d  
11795 f8e303  
11796 f4ddfe  
11797 f912fa  
11798 13f9fb  
11799 10020c  
11800 001f00  
11801 ee1100  
11802 100a0b  
11803 edf3f2  
11804 e703f0

11805 0e0fd7  
11806 f005f8  
11807 fe1015  
11808 fff31f  
11809 201404  
11810 091639  
11811 ec15f9  
11812 01ed05  
11813 eafcf6  
11814 f61718  
11815 f5fa0a  
11816 f0f309  
11817 05fafb  
11818 26f2ee  
11819 de2c07  
11820 e50808  
11821 f40dfa  
11822 e6f2ef  
11823 ef0ded  
11824 0308db  
11825 d3daf1  
11826 ffd901  
11827 1b0216  
11828 0501e1  
11829 04f212  
11830 eafc1d  
11831 1b08e2  
11832 010ae8  
11833 140d0e  
11834 fd1407  
11835 12f308  
11836 181dea  
11837 05090b  
11838 fefa14  
11839 0903f8  
11840 f5ef0e  
11841 eb0506  
11842 020bfa  
11843 f519ef  
11844 0c0800  
11845 1a0021  
11846 1d2ee9  
11847 1e03e6  
11848 e9f204  
11849 090f0b  
11850 35fa1b  
11851 00eefe  
11852 dbf912  
11853 fb1f0e  
11854 1bfded  
11855 030216  
11856 f31120  
11857 fe0df3  
11858 1d10fa  
11859 05e925  
11860 f92015  
11861 2618ef  
11862 050b04

11863	dae919
11864	2d321b
11865	1207fb
11866	090d34
11867	fa0e05
11868	fef51b
11869	f8040f
11870	1107d6
11871	efe6fc
11872	fb0908
11873	0704f9
11874	19f702
11875	f51107
11876	142c17
11877	ed1409
11878	f9092a
11879	08f8f1
11880	0004e7
11881	e9f4f8
11882	05e8f8
11883	f60dff
11884	060709
11885	05f609
11886	f3dc0f
11887	050de7
11888	f7edde
11889	fc0104
11890	15fb10
11891	0a1402
11892	080906
11893	f31216
11894	1a160c
11895	f1fc1a
11896	e80ef1
11897	1309f4
11898	d7e806
11899	1b0618
11900	000513
11901	09160a
11902	06f0fd
11903	f0c7f4
11904	dd03fd
11905	0811f2
11906	e4f7f1
11907	fe0304
11908	f71621
11909	f4f9ef
11910	1c0c04
11911	fae607
11912	ff010b
11913	f006e6
11914	07e7fb
11915	f70bf5
11916	f50802
11917	0ffe0a
11918	df fd2e
11919	2c0df3
11920	e707e5

11921	edfb09
11922	f91709
11923	02fbd7
11924	010502
11925	0204ea
11926	000022
11927	05edfc
11928	e60d17
11929	f805f2
11930	f5fc0f
11931	0009f9
11932	ffe50e
11933	0d061a
11934	05f807
11935	d3e4ff
11936	161c04
11937	f0f9f6
11938	f20906
11939	180209
11940	e1f9fe
11941	0c040c
11942	01ed1e
11943	180c00
11944	f5000b
11945	f811ed
11946	c9cdf1
11947	fc1718
11948	f704f6
11949	082927
11950	0e0cfa
11951	0beaeb
11952	cefafa
11953	fbdc0c
11954	f6daf2
11955	15dc10
11956	14f4ea
11957	ddf9e6
11958	f1f705
11959	f7dbeb
11960	e8ffee
11961	dfff10
11962	0f04fb
11963	e1041c
11964	ee0300
11965	10ffee
11966	f7fbf0
11967	08f609
11968	1d1b0a
11969	fdfd00
11970	011ded
11971	ebd3da
11972	0bfc19
11973	040813
11974	f7222a
11975	1616dc
11976	f6f3fa
11977	d50404
11978	0d02e9

11979	e6e3fb
11980	12090f
11981	f7e706
11982	e80e0e
11983	1516fd
11984	04f005
11985	f5080a
11986	f7eeef
11987	260ced
11988	f4f121
11989	e3e821
11990	f4f0e3
11991	0b0afb
11992	1718f2
11993	052210
11994	f2f118
11995	f7030f
11996	00f0f8
11997	f60e16
11998	0609f1
11999	08061c
12000	1aff0e
12001	08001d
12002	f1ee00
12003	0af50e
12004	13f20e
12005	f3efe6
12006	0a00f9
12007	f8feee
12008	2a00e4
12009	fbf612
12010	060d08
12011	d2effc
12012	020d11
12013	faf3df
12014	022313
12015	0afc7
12016	0005e9
12017	0e0415
12018	ec1506
12019	1ffce9
12020	eaeffe
12021	0e15f8
12022	1206fa
12023	fefb02
12024	1dfcf5
12025	fdff03
12026	1df306
12027	fe0002
12028	0de302
12029	fffe05
12030	1d050d
12031	f40d0d
12032	d4f60d
12033	2413f9
12034	1b1a15
12035	e0ed1d
12036	000eea

12037	e8dbef
12038	fd0907
12039	17e9ca
12040	e1091b
12041	0c07f3
12042	e80c05
12043	f1fce9
12044	07f7f4
12045	fff900
12046	0b0815
12047	ff0ee4
12048	05edf4
12049	fb0201
12050	04f516
12051	2300f2
12052	d7fdee
12053	16e2df
12054	01d605
12055	1d0c11
12056	090ff5
12057	fe10f9
12058	210200
12059	06090d
12060	000011
12061	ff05f6
12062	faf9fd
12063	fd0bfa
12064	ff0300
12065	0cfe0a
12066	f609f2
12067	f5ff07
12068	010108
12069	f2faf2
12070	ef09ed
12071	01f2fa
12072	fdfff3
12073	efec04
12074	0ffef5
12075	09edf9
12076	f30a0b
12077	f406f5
12078	f91108
12079	f603f1
12080	09f803
12081	f80af2
12082	02fff2
12083	010705
12084	f1fa0d
12085	ed0905
12086	f300f8
12087	09f7f5
12088	0ef4ff
12089	0011f2
12090	f7ef01
12091	000506
12092	f0040e
12093	ef060e
12094	0a0c02

12095	fc02f4
12096	fd01ff
12097	0402f6
12098	fb08f0
12099	0ef900
12100	fdf1f3
12101	f1f6f4
12102	0f03f8
12103	090605
12104	0200f6
12105	f20e09
12106	03ee05
12107	080700
12108	0fecf2
12109	ef08f1
12110	eb0900
12111	0505f8
12112	f4f0ff
12113	ff0307
12114	02eeec
12115	fafeff
12116	f5edeb
12117	08ee08
12118	fcfb0d
12119	0aeffb
12120	effff8
12121	0407ff
12122	fbf9fe
12123	ee05f1
12124	fbef0a
12125	f4efec
12126	f001f2
12127	1110f6
12128	fa07f5
12129	00f40c
12130	ec060c
12131	fbf700
12132	050c0a
12133	f500fb
12134	06f802
12135	08fef6
12136	0dfb06
12137	f40f0f
12138	00faf1
12139	06fdec
12140	effaf3
12141	0707f4
12142	04fc01
12143	f4faef
12144	f1f70a
12145	f7fbf2
12146	0e01fb
12147	f8fff4
12148	f30e0e
12149	f1f8f1
12150	000ff8
12151	f3f30c
12152	f4f40e

12153 f70bf3  
12154 fef7f3  
12155 050aef  
12156 f8080e  
12157 06fcf9  
12158 f607ef  
12159 090d0b  
12160 fc0009  
12161 f1ed05  
12162 fe06f7  
12163 00f6f2  
12164 000703  
12165 ecfeec  
12166 f5080f  
12167 f707f5  
12168 f3f0fb  
12169 f5f1f4  
12170 0d0a0c  
12171 f6f3f1  
12172 fc0ef8  
12173 eefcf1  
12174 fcfd06  
12175 0e07f9  
12176 fa00f9  
12177 0400fc  
12178 eff308  
12179 fe0201  
12180 ed0dfa  
12181 0c06f6  
12182 0100fb  
12183 0cfbf2  
12184 fcfd03  
12185 0f0ff0  
12186 00f5fa  
12187 0101f2  
12188 060009  
12189 fefa07  
12190 fd11f4  
12191 060afb  
12192 f306fc  
12193 01f0ef  
12194 00000b  
12195 e8110a  
12196 ec0df1  
12197 f7ff04  
12198 1003f6  
12199 110f13  
12200 03dfec  
12201 f8100c  
12202 06f208  
12203 faf203  
12204 130401  
12205 08feea  
12206 f10ef7  
12207 021415  
12208 15100b  
12209 1a1331  
12210 0f0508

12211 02eaf0  
12212 dbd819  
12213 fdf91e  
12214 0be7ce  
12215 cef20f  
12216 ef1bef  
12217 f4fc00  
12218 0d1ede  
12219 f6dde0  
12220 df0702  
12221 05ffc9  
12222 d3bc18  
12223 000603  
12224 ebbca4  
12225 b4f20f  
12226 200e02  
12227 179ad7  
12228 0c0c1c  
12229 0503ef  
12230 f904fe  
12231 12160d  
12232 f70e1d  
12233 131a0a  
12234 1002e6  
12235 eff2f6  
12236 1c391e  
12237 08ecf8  
12238 f4f50d  
12239 1f04fa  
12240 f9ea08  
12241 e9161c  
12242 ebfd0f  
12243 f10ae6  
12244 ecf3f8  
12245 09fbfe  
12246 f61700  
12247 effd01  
12248 f9011a  
12249 17e6ed  
12250 def2ef  
12251 f5fae9  
12252 e3f2ea  
12253 0a05cf  
12254 0f0e04  
12255 0ff911  
12256 041bff  
12257 04ed09  
12258 fbe816  
12259 e7000d  
12260 49f0ef  
12261 01f53d  
12262 1a0618  
12263 19fb0a  
12264 0b171b  
12265 2c031e  
12266 0a4457  
12267 1a1903  
12268 1313fc

12269	c5ed11
12270	f8140c
12271	fd0cf8
12272	140d11
12273	f9fe09
12274	17f70a
12275	0c1004
12276	09f4da
12277	02ed16
12278	0402ff
12279	fcfcf5
12280	f4f910
12281	071208
12282	221203
12283	0b0019
12284	f001fb
12285	1b0108
12286	14fbf9
12287	0de604
12288	05ee03
12289	0d11ff
12290	010406
12291	0ffc04
12292	131402
12293	03041a
12294	fb0206
12295	fc08f9
12296	fd050c
12297	120606
12298	06060a
12299	05171b
12300	07ed09
12301	f0130a
12302	e81307
12303	15fcfd
12304	01e0ff
12305	080810
12306	1ff7f3
12307	0803e7
12308	dce502
12309	f6f003
12310	fcf8f1
12311	0419f7
12312	00f30d
12313	0eec0f
12314	14f6f1
12315	0efef1
12316	172b00
12317	22ef03
12318	02ea19
12319	230012
12320	ea0a11
12321	03fce9
12322	0ff8ff
12323	00fc15
12324	eaee1f
12325	02e9f0
12326	fdf0f1

12327 eee205  
12328 00001c  
12329 feef05  
12330 0df5ef  
12331 070c00  
12332 03f2fc  
12333 f2f609  
12334 f00c0b  
12335 ee010a  
12336 0002f4  
12337 0df2fe  
12338 01f7ed  
12339 ec0909  
12340 f6fa0c  
12341 f00903  
12342 ed0a00  
12343 fa10f6  
12344 fbeb00  
12345 0bf40b  
12346 fbf9f7  
12347 03ecfb  
12348 0bf0f2  
12349 000500  
12350 f0ff0a  
12351 02f40e  
12352 000f01  
12353 f002fb  
12354 fdf3f3  
12355 0206f4  
12356 eb0df3  
12357 0cf2fc  
12358 0af2fb  
12359 f20706  
12360 00040c  
12361 f8fcf2  
12362 f601f1  
12363 0a01f8  
12364 0cff01  
12365 f8fdf5  
12366 0204f6  
12367 f2edff  
12368 fdf40c  
12369 f306ff  
12370 0f00f5  
12371 0a050e  
12372 03effb  
12373 f2f00b  
12374 fcf90c  
12375 f70ef0  
12376 f203fc  
12377 06f3f6  
12378 f8f2fd  
12379 ef0a04  
12380 080008  
12381 f4f309  
12382 fa08f3  
12383 f6f80d  
12384 ec00ef

12385	f50ff8
12386	fdf20a
12387	ffef00
12388	fbffef
12389	f4ffed
12390	0c00fe
12391	0307fe
12392	f6fc02
12393	000e09
12394	f8f205
12395	fb0ef6
12396	03fc04
12397	eafa0a
12398	eef000
12399	080904
12400	060ef6
12401	f00c0c
12402	0ef707
12403	f308f8
12404	ffeff3
12405	f5ec00
12406	03f7f5
12407	010508
12408	f5f3ed
12409	0e08fb
12410	02f402
12411	000f03
12412	fbf709
12413	0700fd
12414	f1fb08
12415	0aeef1
12416	fef7fa
12417	02000c
12418	02fe02
12419	fc0400
12420	fef6f6
12421	00ec0a
12422	00f7f6
12423	f1f700
12424	fdfa10
12425	eff4fb
12426	ec0900
12427	000bfb
12428	0cf00c
12429	f40b00
12430	fafef3
12431	0af1ff
12432	0ff4f4
12433	0e09ec
12434	f7000b
12435	f5020b
12436	0d0ded
12437	0df9f1
12438	f1f0ec
12439	0d01fb
12440	10edfe
12441	00efed
12442	f4040a

12443	0708f7
12444	07f306
12445	f20505
12446	0bf610
12447	fa0004
12448	09eefb
12449	f1f7f9
12450	ee0910
12451	f1fcfa
12452	f907f2
12453	f8f1ec
12454	f10907
12455	f4080a
12456	f6fcf6
12457	f000ff
12458	07fd00
12459	f8f5fb
12460	06f205
12461	0a10f2
12462	0000ff
12463	fb0cfa
12464	011609
12465	1a18fd
12466	f2140a
12467	0205e1
12468	fbfc24
12469	0bd8eb
12470	fef7f1
12471	fd00d4
12472	fef2f9
12473	130205
12474	1916fc
12475	0cf1fa
12476	011e24
12477	edf30f
12478	130e18
12479	f4cadb
12480	16fa03
12481	01f32a
12482	2305f6
12483	fc01df
12484	deebfd
12485	05f600
12486	3b2634
12487	11263b
12488	00e3ff
12489	d9c0d4
12490	e10b1c
12491	fc0cf0
12492	28eef5
12493	b3de02
12494	0d29d2
12495	e09cd0
12496	ed09e5
12497	02eff9
12498	eeec04
12499	f10eeb
12500	f1f6f7

12501 ecec0b  
12502 00f0fc  
12503 cacfe8  
12504 f5fec4  
12505 0aec03  
12506 101b05  
12507 fcff05  
12508 eaaffe  
12509 ecf2ed  
12510 1e02ed  
12511 220e1f  
12512 032114  
12513 000eef  
12514 140c06  
12515 381e0c  
12516 fe0538  
12517 0f06f9  
12518 f112ee  
12519 d9d000  
12520 151402  
12521 09ecf4  
12522 190bff  
12523 03f701  
12524 021d00  
12525 001607  
12526 e8d6fe  
12527 eefb00  
12528 f5dda4  
12529 0b0c0e  
12530 292001  
12531 0bfe24  
12532 2e3afa  
12533 fc0efe  
12534 f109e8  
12535 18fdfc  
12536 121af3  
12537 081334  
12538 0607ee  
12539 0c0af1  
12540 14fa08  
12541 f0110b  
12542 f5e6ec  
12543 f21012  
12544 f3e2f2  
12545 0604f1  
12546 fce8eb  
12547 05f3d4  
12548 f41606  
12549 dbefff  
12550 13f0e4  
12551 151318  
12552 fa0c03  
12553 210927  
12554 0aeb09  
12555 d7eefe  
12556 fe6f8  
12557 f71500  
12558 f1ff1d

12559	0f161a
12560	f20109
12561	f517fa
12562	d8d907
12563	faedf8
12564	08ea06
12565	38180a
12566	fcfc2d
12567	110dfe
12568	ea040b
12569	0acbd3
12570	f50a0b
12571	0714cf
12572	e80122
12573	e01800
12574	15eaf3
12575	d6f300
12576	0a27e5
12577	eeee0c
12578	fbe7fc
12579	f8e3e4
12580	e9e0f6
12581	eceff5
12582	f4ea0b
12583	fa05ff
12584	f51004
12585	fe0302
12586	feecf9
12587	d3e3f0
12588	f6eef2
12589	f8df04
12590	f7e51d
12591	d5f0ef
12592	220df9
12593	fa0b20
12594	361bfe
12595	d50617
12596	0000dc
12597	ebfbf3
12598	3df0ee
12599	fb011d
12600	1a17ef
12601	f50912
12602	111d00
12603	e4f1f5
12604	1115fe
12605	201a02
12606	e1dee2
12607	13271d
12608	55f80c
12609	f5214b
12610	0b0910
12611	f1f500
12612	070601
12613	c40e0a
12614	110bd7
12615	fcda32
12616	492cfa

12617	161e1d
12618	1d2404
12619	0a13f7
12620	e20020
12621	f4f7ec
12622	0a012a
12623	18f3e3
12624	feedfa
12625	1ae404
12626	fdf21c
12627	231001
12628	0d1b26
12629	5004fd
12630	fe1e13
12631	081405
12632	02fb2c
12633	0d09f4
12634	f6f2f6
12635	08030f
12636	ece10c
12637	00061c
12638	d1ff02
12639	140715
12640	f5d51b
12641	2e1a00
12642	031a16
12643	0b1b18
12644	fef605
12645	f6ffffd
12646	d5eee1
12647	110f0d
12648	da0607
12649	dc1de2
12650	edd0f8
12651	df0122
12652	f5fcf7
12653	ef09ec
12654	f60907
12655	f91af8
12656	121b0a
12657	e8ebe8
12658	cff80a
12659	f0fef8
12660	0b16e2
12661	e6fff2
12662	edfcfb
12663	d30dff
12664	0ffbe6
12665	gcd40d
12666	041416
12667	2c25fc
12668	0fe309
12669	00fd10
12670	fffd00
12671	f40c0d
12672	00f8fb
12673	2afef9
12674	f5070e

12675	061af8
12676	e5f9ed
12677	01f205
12678	febfff
12679	0311fd
12680	fb8fe
12681	25fbfa
12682	1afdd7
12683	122008
12684	0f040c
12685	0e0e0d
12686	f1ef08
12687	f50005
12688	fcfbcd
12689	f602f7
12690	0848e6
12691	fbef9
12692	1a2c36
12693	0e10ef
12694	0e0cf6
12695	f002e6
12696	f205fd
12697	fff607
12698	0b0f0d
12699	fe14fd
12700	2bf80e
12701	ea0401
12702	13f6eb
12703	fefef0
12704	0113f3
12705	29fefb
12706	f4f213
12707	0a4527
12708	0b01ea
12709	0b0707
12710	0eeafd
12711	091b05
12712	1d0ae7
12713	2004fb
12714	f6e9fe
12715	fa1fe5
12716	fbcd03
12717	fe051d
12718	f6090d
12719	fe0c14
12720	fdf704
12721	0000f1
12722	03f1e1
12723	ed11fe
12724	0d2104
12725	2f3d0d
12726	1206f9
12727	040b10
12728	11dae9
12729	f5faf6
12730	0000e7
12731	fdfe09
12732	0cef09

12733	0afef3
12734	0efcf8
12735	0af411
12736	fceaec
12737	fe02f2
12738	f90812
12739	fbf30d
12740	f40102
12741	0608f3
12742	ff00fa
12743	0e08fa
12744	02f2ee
12745	fbfef2
12746	0ff309
12747	fff3f6
12748	0cf3f8
12749	e9f5ff
12750	f5f9e8
12751	f205f3
12752	fbfb01
12753	f807f7
12754	0dfd06
12755	0e0aff
12756	f5f1f5
12757	0cfe00
12758	ede9ec
12759	0cf307
12760	09fc00
12761	0501fa
12762	fd0cfb
12763	fa05ef
12764	eff00b
12765	ebf30e
12766	ef09f4
12767	ed0c07
12768	050b03
12769	ff0afd
12770	08030b
12771	f30100
12772	fe02ef
12773	00f1fb
12774	070efa
12775	0a0bee
12776	f00407
12777	fa01fc
12778	040302
12779	fdf9fc
12780	ed03f9
12781	ec0dfa
12782	f10af7
12783	f6ebfe
12784	f3f500
12785	0103ed
12786	f20b0b
12787	02ec03
12788	f6fcef
12789	030200
12790	f8ebee

12791	09feee
12792	ee0cf2
12793	02ffec
12794	06f1fa
12795	ffeb0b
12796	0106fb
12797	f9120f
12798	f1ed02
12799	ef05f9
12800	f90708
12801	ff030c
12802	090b07
12803	0000fc
12804	fdf801
12805	f8f1fe
12806	fc010a
12807	fe0def
12808	fbf9f1
12809	f30dff
12810	ec070f
12811	f90aea
12812	0707f3
12813	f10e00
12814	03e9fe
12815	f1f7fb
12816	f1eff6
12817	02efec
12818	03f810
12819	fbf5f2
12820	02090c
12821	0d02ed
12822	f302f8
12823	f2ec04
12824	ed08fd
12825	ff0eff
12826	fe0a06
12827	f9f5ef
12828	fcfc00
12829	fefbee
12830	efedee
12831	00050e
12832	060df9
12833	08050b
12834	edf2f4
12835	f508f4
12836	edfdfa
12837	f1f602
12838	05fcf5
12839	f2040f
12840	f006fc
12841	fa05f8
12842	eef5f5
12843	f3fafe
12844	fbf005
12845	07f6fe
12846	0ff606
12847	f6eb00
12848	ecf7ff

12849	fb0800
12850	0109f8
12851	0cef0f
12852	fdfcf9
12853	01edfe
12854	f9080a
12855	0bf203
12856	06fcf9
12857	f2fdf7
12858	eeff05
12859	f1f012
12860	09f401
12861	05f20b
12862	08f805
12863	03eb03
12864	000002
12865	02eb07
12866	f501e6
12867	fe01f6
12868	0deffc
12869	1515ef
12870	e703e8
12871	fb0b04
12872	f30e0c
12873	060006
12874	ef0109
12875	f30c06
12876	e4ed00
12877	f9f503
12878	fbfc0e
12879	fa08e0
12880	daf6fb
12881	02feeb
12882	fb060a
12883	000909
12884	0c04f5
12885	00e1fb
12886	fae801
12887	fc170b
12888	0df7e8
12889	f1f801
12890	e1db0a
12891	fc02e6
12892	00e7f2
12893	d2f0f7
12894	fa1703
12895	0aeef8
12896	fadefe
12897	da06e2
12898	ebf6fb
12899	d904e0
12900	f0f50b
12901	fadb6
12902	d9f6df
12903	d1f80c
12904	00fbea
12905	00d9f1
12906	e40a0a

12907	f810fe
12908	1000f7
12909	f0e609
12910	fdf2eb
12911	f4dcf1
12912	e602f6
12913	dbf5ee
12914	e3eddb
12915	efe808
12916	dcead9
12917	ebece2
12918	f2f5fc
12919	11f603
12920	fc0dd3
12921	fd1519
12922	fffe07
12923	0104cf
12924	efe2ed
12925	e7f7e2
12926	0b00d9
12927	f0eaf6
12928	ebf519
12929	f304f1
12930	1108eb
12931	ece510
12932	eee3f7
12933	00ebf5
12934	fdfc00
12935	feffdf
12936	fb15fd
12937	f900f0
12938	f8fbef
12939	e5fa0d
12940	e4fdfb
12941	eedcf0
12942	f3d4ea
12943	e9fafa
12944	ee19e8
12945	f0eae6
12946	fe06fa
12947	120704
12948	001314
12949	fbedf2
12950	f7ecf2
12951	f5e706
12952	ede90b
12953	f602cc
12954	fe000b
12955	e2f0fd
12956	02f700
12957	ece500
12958	f0e6fb
12959	d7edfe
12960	d8f1f8
12961	01f8f2
12962	f9d1fe
12963	ee01f9
12964	03f4ca

12965 f5fb0b  
12966 ddf4f5  
12967 e800e0  
12968 faeff6  
12969 e4050c  
12970 f2ee03  
12971 ec13e5  
12972 f8f6fb  
12973 08f10f  
12974 f8f7ef  
12975 030df3  
12976 0002f7  
12977 ffe6f7  
12978 ee01f1  
12979 f2d5dc  
12980 d6fae5  
12981 030dea  
12982 01e912  
12983 f702dd  
12984 e2f6f5  
12985 d4000d  
12986 f8efe2  
12987 f7e5dc  
12988 ece60f  
12989 0614e6  
12990 040a12  
12991 fe01f6  
12992 f5ef02  
12993 02f2d9  
12994 e1fced  
12995 0613fb  
12996 00ea03  
12997 f2e3f4  
12998 000017  
12999 f5fc0b  
13000 040c0b  
13001 f0fc07  
13002 f201ec  
13003 f3eef8  
13004 07fbf9  
13005 08ecfe  
13006 0311f8  
13007 0bf104  
13008 f910ee  
13009 f6f0ff  
13010 f2ed01  
13011 f7060a  
13012 01fef1  
13013 f3f5f9  
13014 08f60b  
13015 fbf8fa  
13016 eafcf5  
13017 e90ef9  
13018 f311fa  
13019 100bff  
13020 ff0dfd  
13021 0d0cf4  
13022 f4f9fb

13023	f40204
13024	eb06f0
13025	050908
13026	0d0907
13027	f3ebfb
13028	1204ed
13029	0b04fb
13030	0f0d04
13031	00f208
13032	f00dff
13033	f6f3f8
13034	0fed03
13035	f5020c
13036	08f00b
13037	f2f7ee
13038	10f3f4
13039	ee01fe
13040	0000ef
13041	e9f4ed
13042	e9fdf3
13043	f9f9fb
13044	0d0b07
13045	fcf301
13046	eeee05
13047	eef3f3
13048	eff0e7
13049	f607f8
13050	ff0706
13051	05ecfb
13052	f50aff
13053	fa0af6
13054	030ffc
13055	fd0bf3
13056	fd0f03
13057	09fc10
13058	f7f10a
13059	f9f303
13060	0802f8
13061	f9f7eb
13062	0df2ed
13063	0101ff
13064	0802fa
13065	f50afc
13066	0807f7
13067	030afb
13068	0507f7
13069	fbfa06
13070	090ff0
13071	f8fdfa
13072	0dee04
13073	040af2
13074	f6f8f5
13075	01f1f1
13076	fcfe0b
13077	0ef5ee
13078	0200f1
13079	fd0407
13080	030703

13081	f700fd
13082	03f706
13083	04f0fd
13084	00f702
13085	ee06fe
13086	060ff9
13087	f6fefb
13088	0001ff
13089	ffefee
13090	0df808
13091	eefd05
13092	fa0f04
13093	04f4ec
13094	00fb0a
13095	fefcf3
13096	ec0e10
13097	0e00f8
13098	eff5ef
13099	00f9fb
13100	f0faf6
13101	0501f1
13102	fa0d00
13103	0fec03
13104	e8f602
13105	0206fa
13106	f6ff10
13107	f9f10e
13108	0d0212
13109	fc01ed
13110	ec0cfc
13111	08f30b
13112	090e08
13113	fa0d11
13114	08f800
13115	ff0108
13116	fc0200
13117	f2f5ef
13118	00fefc
13119	fcf70d
13120	0b0af5
13121	f3f800
13122	f5f00a
13123	08fcf5
13124	fd0105
13125	02ffff
13126	08f10e
13127	0c090a
13128	0b0905
13129	020ffb
13130	edf800
13131	02f5ec
13132	00000c
13133	07ec02
13134	f506f2
13135	0cf9fe
13136	02f526
13137	f81210
13138	051c2a

13139 fefae7  
13140 ed0012  
13141 fd09e2  
13142 06f9f1  
13143 def8ea  
13144 0e21f8  
13145 0404ec  
13146 13161e  
13147 fcfb1b  
13148 0d14ff  
13149 09f6e5  
13150 09f6f6  
13151 e4f80a  
13152 1d1b04  
13153 f702f7  
13154 2a0015  
13155 f7f608  
13156 001100  
13157 0d020b  
13158 4021f6  
13159 1923fd  
13160 fb122a  
13161 0a1efc  
13162 162316  
13163 11e5e7  
13164 1333ff  
13165 f4bbfb  
13166 f9030e  
13167 000414  
13168 08fd0e  
13169 140302  
13170 16e30a  
13171 0afc1d  
13172 1a0f09  
13173 1af4d8  
13174 f404fe  
13175 191b0a  
13176 f0f6f6  
13177 fbf904  
13178 e30603  
13179 11c9d0  
13180 e7d9ee  
13181 f606fe  
13182 080618  
13183 0705f1  
13184 10ea0a  
13185 151338  
13186 ebe32c  
13187 1516ee  
13188 0531eb  
13189 38d910  
13190 f5172d  
13191 fefcd1  
13192 fff7f5  
13193 ed1cfc  
13194 f217eb  
13195 fa03eb  
13196 03e407

13197	da0626
13198	131700
13199	d5f100
13200	08f8e9
13201	deee05
13202	351ee9
13203	0cd6c3
13204	06f418
13205	f51818
13206	fb1cc8
13207	d00a1e
13208	f20109
13209	fd0ef2
13210	120200
13211	02fb0c
13212	f30114
13213	f4220c
13214	ee0e0a
13215	020903
13216	060b06
13217	ea08fe
13218	eae7ce
13219	090109
13220	e50d0b
13221	041fdf
13222	050afb
13223	1901f7
13224	f3fafd
13225	07000f
13226	1602f7
13227	fbf6fa
13228	0c0a0e
13229	1b08fa
13230	040c0a
13231	01fc04
13232	fef008
13233	01ffee
13234	f0fc01
13235	05f300
13236	e5e5fe
13237	15f2ef
13238	131118
13239	0feaf2
13240	0e1709
13241	fcf3e6
13242	fff20f
13243	120b05
13244	24f414
13245	012110
13246	eb27fa
13247	fd040a
13248	e8cd16
13249	f9120a
13250	fdf5fc
13251	19faf3
13252	0a0605
13253	00fffe
13254	ef0e0b

13255 17fbff  
13256 1b0717  
13257 e3f603  
13258 03fa05  
13259 d6f60a  
13260 f7f6ef  
13261 faebee  
13262 f4e4fb  
13263 fd04fc  
13264 fa2809  
13265 f5fd00  
13266 0000fc  
13267 dfebf8  
13268 0af4f4  
13269 1b0616  
13270 012503  
13271 fef506  
13272 011012  
13273 ef03fc  
13274 08ecec  
13275 1f0dfc  
13276 c1ecdd  
13277 e10606  
13278 01e0e6  
13279 f2180a  
13280 050d00  
13281 fefb1e  
13282 01fbf5  
13283 fc0700  
13284 f9e9fe  
13285 d0dd21  
13286 14f0ce  
13287 e2fcff  
13288 20fceb  
13289 010312  
13290 1a140e  
13291 0c1b1f  
13292 482d18  
13293 0e2737  
13294 d61e1b  
13295 27f1d0  
13296 e5c9f5  
13297 f808ef  
13298 e61dde  
13299 ecd710  
13300 01f113  
13301 0003f8  
13302 fb18f0  
13303 fc512  
13304 f30b15  
13305 f6f6f4  
13306 f21b0f  
13307 0cf0f6  
13308 01f902  
13309 e8d1d7  
13310 d1dc1c  
13311 0700e8  
13312 eb14f3

13313	10040b
13314	fdfff4
13315	0f0c22
13316	142523
13317	f0efdc
13318	14f2fc
13319	21140e
13320	11042d
13321	101d0a
13322	f126f0
13323	01eaf8
13324	f9010b
13325	f2e7ef
13326	130e01
13327	001e24
13328	0006fb
13329	ff00ee
13330	150015
13331	e10e15
13332	eafc6
13333	f1daf7
13334	f7dee4
13335	f90bf9
13336	25fff9
13337	12f2dc
13338	0b0a1e
13339	07091f
13340	ef0f04
13341	08e200
13342	f1df00
13343	0de3f6
13344	0518fa
13345	110710
13346	050900
13347	0d0c02
13348	cb0c06
13349	fefedc
13350	d60205
13351	f2b6ab
13352	eccbda
13353	0206f3
13354	fa080d
13355	00190a
13356	fb160c
13357	1d17fd
13358	fa110f
13359	e800f5
13360	1814da
13361	fce9e7
13362	f900f8
13363	0e0dfc
13364	0efd12
13365	e90115
13366	06000c
13367	cacbeb
13368	f1f5df
13369	050808
13370	1b0c0b

13371	103012
13372	1a1322
13373	f81018
13374	e4feec
13375	211cea
13376	f6f61a
13377	0e0b1a
13378	1f04f8
13379	f1fff8
13380	fefb08
13381	ed01e3
13382	e5e1e1
13383	081906
13384	fa00f1
13385	05ece8
13386	d3f2f9
13387	040700
13388	fa1a00
13389	1e16fe
13390	100009
13391	041806
13392	d2db0d
13393	e013e0
13394	fdf9ea
13395	1c0600
13396	eefa03
13397	250b07
13398	1a11f4
13399	03f81a
13400	000008
13401	01dbe7
13402	e2f905
13403	101518
13404	de04fe
13405	ef19ee
13406	eaf81a
13407	07fe0e
13408	fd0afb
13409	02fa0a
13410	02feff
13411	1506fe
13412	000108
13413	022a20
13414	e0f904
13415	ef2009
13416	ecfd10
13417	f2ee10
13418	e8e8ff
13419	020f08
13420	12ce01
13421	271a1e
13422	f302e0
13423	f6350c
13424	e4f2de
13425	05f3ea
13426	ec17f9
13427	020840
13428	f7cb19

13429 49fa0a  
13430 2609d9  
13431 f60907  
13432 0d1e2a  
13433 0c13fc  
13434 f3f01d  
13435 020914  
13436 26f7db  
13437 e4f91b  
13438 eb000a  
13439 1c02f6  
13440 f8faf4  
13441 140af7  
13442 fae5ed  
13443 0cedea  
13444 e21313  
13445 17e6d0  
13446 140df7  
13447 ec2536  
13448 f61025  
13449 0706f5  
13450 05fbf1  
13451 eef7ff  
13452 f41801  
13453 c3e5f5  
13454 eaffe4  
13455 09f3df  
13456 f70020  
13457 2308e5  
13458 e80a1c  
13459 fbf4fa  
13460 f9fe10  
13461 1609d8  
13462 e5f01c  
13463 f1c911  
13464 0922e1  
13465 fb17e5  
13466 fb0903  
13467 e4e10b  
13468 ebddd1  
13469 fce301  
13470 e3020e  
13471 d5fa13  
13472 15f0d4  
13473 e70e2b  
13474 22f3e4  
13475 e8e91d  
13476 15f1f4  
13477 e2fe01  
13478 051707  
13479 edf7fc  
13480 e305fb  
13481 02eafb  
13482 2ef702  
13483 0c161b  
13484 f90ff8  
13485 02f600  
13486 f0f5fc

13487	e801ea
13488	1c2a1f
13489	ecdc06
13490	f70d07
13491	f5fbfd
13492	fbfdf1
13493	0308f3
13494	12fc06
13495	0f0c22
13496	15eee9
13497	f1fb02
13498	0213ea
13499	fb0500
13500	e70004
13501	f80915
13502	f50400
13503	fd00eb
13504	08040d
13505	dd0ef7
13506	cee510
13507	340df2
13508	00f409
13509	09f0f6
13510	0d130c
13511	0f0cf1
13512	f50220
13513	15e9c7
13514	040c0c
13515	0a1bfc
13516	03200c
13517	01f514
13518	f601f4
13519	e7d905
13520	17021c
13521	1302e2
13522	0fdff3
13523	130604
13524	ea1606
13525	e5f60b
13526	e7d2fc
13527	ff080f
13528	0fb2e0
13529	40031b
13530	e10619
13531	ecf1c3
13532	0cf008
13533	0206f3
13534	0000e3
13535	f31313
13536	0cf903
13537	f7f4ee
13538	ef03f9
13539	07f402
13540	fe22fd
13541	f303de
13542	e9f8fb
13543	fdef0f
13544	09ebf9

13545 1ef908  
13546 f90f08  
13547 0efa00  
13548 0bf305  
13549 0ce606  
13550 0410e5  
13551 f7fd04  
13552 d4ecf6  
13553 0ae1e6  
13554 120df7  
13555 0efaf6  
13556 051223  
13557 0406fb  
13558 141703  
13559 03f419  
13560 efff05  
13561 31f6ed  
13562 101ff8  
13563 e406ff  
13564 101b27  
13565 40f1de  
13566 f91416  
13567 2b5928  
13568 05eff6  
13569 05040f  
13570 020b03  
13571 0eedef  
13572 11ed03  
13573 e0fbf4  
13574 d4ee0d  
13575 00f3fc  
13576 07d4ef  
13577 e1dee3  
13578 06f900  
13579 0f0407  
13580 0108e9  
13581 23000b  
13582 0a01f0  
13583 02fcfe  
13584 eff0ee  
13585 241019  
13586 001521  
13587 f3fdf2  
13588 eaf300  
13589 f4f315  
13590 1ae4ef  
13591 10d8f0  
13592 fce7dd  
13593 111409  
13594 0a0cfa  
13595 000d09  
13596 fc01f5  
13597 0014e9  
13598 f4f1e6  
13599 ddcefe  
13600 d50b00  
13601 ee07c7  
13602 1e230c

13603	f10500
13604	00fafc
13605	1411fc
13606	0204fa
13607	deff08
13608	0c1fd0
13609	05ced6
13610	fe1700
13611	02fb08
13612	0afa09
13613	0506ff
13614	0d00ef
13615	e3fc17
13616	dcfc05
13617	04fbee
13618	03e9e0
13619	ec03ea
13620	0008f0
13621	051900
13622	fbf900
13623	0b0817
13624	0cfd02
13625	d4e1dd
13626	04eef8
13627	0a0a11
13628	fe0800
13629	fe0709
13630	f6f6fa
13631	f50af1
13632	04fb00
13633	faf507
13634	ed0bf7
13635	1ff100
13636	f91608
13637	f51e03
13638	f0f419
13639	fd0011
13640	070d10
13641	e700fe
13642	d1d6db
13643	08ffde
13644	f7fffc
13645	160aee
13646	f0130c
13647	f7030d
13648	28f707
13649	0b00fc
13650	02e500
13651	050702
13652	fdfb19
13653	0604f4
13654	0600f8
13655	f3000c
13656	f4ee05
13657	0110f0
13658	0a01f1
13659	e4f012
13660	00fb13

13661 edf1fa  
13662 0d2f1b  
13663 ebde0a  
13664 0b15f9  
13665 eb080c  
13666 d41a18  
13667 0de4f8  
13668 0000f9  
13669 efeef1  
13670 02fc0e  
13671 fcf4e8  
13672 02040c  
13673 fffe1a  
13674 0d03ec  
13675 0418f1  
13676 ead4fb  
13677 e7f40d  
13678 0507f5  
13679 f7ed02  
13680 001bf0  
13681 f7f6e7  
13682 0c2406  
13683 fef915  
13684 f80b09  
13685 1f14f6  
13686 fbfe31  
13687 fdea0d  
13688 f822ff  
13689 ddc9c8  
13690 ded0e5  
13691 fe1804  
13692 261c08  
13693 261c10  
13694 0604fb  
13695 feeee0  
13696 110006  
13697 020bf6  
13698 2f0412  
13699 15ff01  
13700 e70609  
13701 04ffc3  
13702 eaf2fe  
13703 e6edfa  
13704 0c1507  
13705 1c11f3  
13706 180816  
13707 0d0528  
13708 230fef  
13709 f5e8ef  
13710 1509f0  
13711 18f517  
13712 15000a  
13713 daef01  
13714 cbd3c9  
13715 12dedb  
13716 00170c  
13717 241e21  
13718 011720

13719	ee08f3
13720	2204f1
13721	141831
13722	09f738
13723	0b221a
13724	14d0c1
13725	d80624
13726	dceced
13727	1bfff4
13728	0a0017
13729	0a1e17
13730	0dfd00
13731	231503
13732	ffe01b
13733	22e10c
13734	ded6c8
13735	2006ce
13736	270a1d
13737	fd061f
13738	320e01
13739	01f0d0
13740	95131f
13741	e8dbc8
13742	150809
13743	1dfe27
13744	e1e0e4
13745	15fef3
13746	ed0c0d
13747	1a0bf6
13748	24fcfe
13749	0d1c08
13750	0313f5
13751	f2dc0a
13752	f2e8f9
13753	edf6f3
13754	feeacf
13755	e7f7e4
13756	eff5f3
13757	0a220b
13758	040317
13759	0022fe
13760	f7240c
13761	eef902
13762	08f6e9
13763	040606
13764	000b15
13765	120c00
13766	090d11
13767	131efb
13768	ffeff1
13769	e8ddd7
13770	f4e2e8
13771	fef5f3
13772	1f1f0e
13773	21032e
13774	0401fd
13775	edf50f
13776	f8f3fe

13777	f7ef14
13778	fb00ee
13779	eafef7
13780	090804
13781	020d14
13782	1813fa
13783	f40814
13784	f20706
13785	f606f7
13786	f6eff8
13787	ec0b02
13788	06faee
13789	040a08
13790	1c0904
13791	fe0214
13792	0aff1c
13793	18fde8
13794	fdf3ee
13795	daf71f
13796	0f0211
13797	efe3e2
13798	fc06f7
13799	1708f8
13800	fd0a08
13801	fbed13
13802	000010
13803	fefef2
13804	090907
13805	0908ee
13806	f10e05
13807	f004ef
13808	edf801
13809	ff0809
13810	f0080c
13811	edfcf4
13812	f90df5
13813	02e7ff
13814	0c01f7
13815	0c02f3
13816	f9fdfb
13817	04f603
13818	00eff2
13819	0607eb
13820	03e90d
13821	f0f1fd
13822	02ecf0
13823	0602f3
13824	06ee08
13825	0b03ed
13826	0705fd
13827	0505fd
13828	fdee02
13829	f0ff08
13830	fdf7fe
13831	f5fbf0
13832	09ec00
13833	eefa10
13834	ed0afe

13835 10f609  
13836 0bf2ea  
13837 edf5fb  
13838 07fced  
13839 0a0500  
13840 fef0fb  
13841 efff00  
13842 0a0b05  
13843 ee0ff2  
13844 fdf80c  
13845 0bff04  
13846 f30402  
13847 f0f5fc  
13848 000a09  
13849 eaf1ec  
13850 06edfd  
13851 1000f8  
13852 f00de9  
13853 00e905  
13854 08ed0b  
13855 f8f9f9  
13856 f2f0ff  
13857 f90f00  
13858 f2f302  
13859 03fc0d  
13860 f6040e  
13861 04eaf3  
13862 06ff06  
13863 00effb  
13864 f30af4  
13865 f0f5fd  
13866 fffdf6  
13867 09ec03  
13868 f2f8f0  
13869 fdeff6  
13870 fbfa01  
13871 ff0b08  
13872 f5f9f8  
13873 ecf310  
13874 ed07f3  
13875 0cec06  
13876 fc06ee  
13877 f81402  
13878 00ebea  
13879 f8f602  
13880 fdfcfa  
13881 f2fced  
13882 07ecf4  
13883 f1f9f2  
13884 130cf2  
13885 0cf7f1  
13886 f5fc0d  
13887 0206f8  
13888 fe08eb  
13889 feecf1  
13890 0502fc  
13891 0c01ef  
13892 04ea0e

13893	05faf2
13894	f506ed
13895	05eb05
13896	0aed09
13897	0bfdee
13898	f1eef6
13899	f603fc
13900	f1090b
13901	f8f2f7
13902	fb030e
13903	f0fe0b
13904	f80a09
13905	fdefee
13906	04000a
13907	f202f7
13908	fbfcfd
13909	f1f8f4
13910	f50f06
13911	edf9fd
13912	f7f6f1
13913	04f2ea
13914	fe0bec
13915	fb0200
13916	f10a08
13917	fe0400
13918	0cf8eb
13919	f2ee06
13920	f8f10e
13921	0cf8fc
13922	00f1fb
13923	f5ecf5
13924	f6eb0a
13925	0dec0e
13926	010b08
13927	08eded
13928	05f901
13929	ea0eff
13930	ea0703
13931	f30afa
13932	f203fc
13933	0cf700
13934	ed020e
13935	fcf8f3
13936	0000f0
13937	f305fa
13938	f40211
13939	020207
13940	fcef00
13941	070ded
13942	070000
13943	f0f9fb
13944	ed02f0
13945	f3ff0f
13946	f2ebfc
13947	f20d05
13948	070dee
13949	07fdf3
13950	080eed

13951	fafc09
13952	00f2fc
13953	f5fa05
13954	ecf500
13955	09fcf1
13956	0af0f2
13957	fdec07
13958	0feb0c
13959	060505
13960	05fff6
13961	010fec
13962	fafa05
13963	f3fef6
13964	ee1005
13965	f8fff6
13966	0eeaf6
13967	0ef407
13968	ef020b
13969	f00b09
13970	0bf4f9
13971	070d0a
13972	f9f206
13973	01f803
13974	f60dfc
13975	02f5f2
13976	020bfd
13977	0c0fef
13978	05ed08
13979	08040d
13980	eb0004
13981	01f1f1
13982	0df2f6
13983	0df5f9
13984	0ffd01
13985	f7fa01
13986	f802ef
13987	ebf706
13988	f8ef00
13989	070805
13990	0c09ff
13991	f10c00
13992	ec000b
13993	f3f802
13994	0dffe0
13995	f5f1f5
13996	ed030c
13997	ebfe0e
13998	f1fdfa
13999	f7f2f5
14000	0f00f7
14001	eff6fc
14002	ff0409
14003	0407ef
14004	ff080e
14005	f900fe
14006	0605f3
14007	000ffe
14008	0df90b

14009 eefe0a  
14010 fef9fe  
14011 0404f6  
14012 f206f2  
14013 fb0c0c  
14014 0e02f3  
14015 f30a00  
14016 f70ef5  
14017 fffa0a  
14018 f70108  
14019 0b0efb  
14020 f60cf4  
14021 f0eefe  
14022 00100d  
14023 ff0507  
14024 0eee00  
14025 fa0400  
14026 ffeef8  
14027 ecfdf8  
14028 0df808  
14029 f00cf6  
14030 00fa0b  
14031 f109fa  
14032 f8f40d  
14033 01f00d  
14034 030efe  
14035 02ee00  
14036 06eff7  
14037 02faec  
14038 f8edf1  
14039 05f1fb  
14040 ef08eb  
14041 ecffef  
14042 0bf4f9  
14043 010409  
14044 04eef0  
14045 fbf000  
14046 f6fc00  
14047 f6f903  
14048 08ecf2  
14049 0cfef3  
14050 02f1f2  
14051 fb0105  
14052 fcfc02  
14053 eef9eb  
14054 f3030b  
14055 0bf6f0  
14056 f3f7ee  
14057 ebf5fd  
14058 faf506  
14059 0602fd  
14060 f9effa  
14061 f40800  
14062 f3edee  
14063 03fd06  
14064 090605  
14065 f60008  
14066 01f50a

14067 0cec05  
14068 f5fbec  
14069 01f0ef  
14070 0000f2  
14071 07f107  
14072 02fe08  
14073 e8fd07  
14074 fae00b  
14075 0611f8  
14076 070e16  
14077 28f6fc  
14078 f0fb08  
14079 04ee06  
14080 1c0807  
14081 ffedf0  
14082 ed110c  
14083 0009de  
14084 f819f9  
14085 fe1323  
14086 fafe22  
14087 24161b  
14088 060b23  
14089 ea0ebc  
14090 e50e03  
14091 fa09cf  
14092 02e304  
14093 fc1c0e  
14094 00040d  
14095 140b06  
14096 1419df  
14097 f1fe1e  
14098 1cecf9  
14099 e6031b  
14100 0c08fa  
14101 1fefe3  
14102 d0f7e8  
14103 c602ee  
14104 e9e1f0  
14105 0efb04  
14106 0613f4  
14107 fbf9f3  
14108 0fe310  
14109 eeef0e  
14110 08130a  
14111 e5e902  
14112 0c0500  
14113 d4f007  
14114 f8f0dc  
14115 d9000f  
14116 f90ef7  
14117 13d9ee  
14118 00eb13  
14119 ff0011  
14120 282008  
14121 0deefd  
14122 3bf2e3  
14123 fb0124  
14124 f02203

14125	e3f600
14126	180ef6
14127	08d610
14128	04f513
14129	ff0100
14130	e6e604
14131	ea162c
14132	2afdd0
14133	1bf9f9
14134	071837
14135	f5f404
14136	08e8ff
14137	2400f3
14138	062734
14139	2f2aea
14140	d8e0fe
14141	2902e7
14142	0af622
14143	f00e08
14144	0c19fe
14145	3af800
14146	06eae
14147	fcf303
14148	ecf7f6
14149	1300ea
14150	f90a20
14151	f8fa17
14152	1d0e14
14153	f6effe
14154	ec02ef
14155	efee03
14156	0ffae0
14157	05ec0d
14158	d90503
14159	1010fd
14160	162618
14161	fa1a04
14162	08081b
14163	0112ff
14164	120204
14165	06f806
14166	060a05
14167	0ff709
14168	03ec0e
14169	0414f9
14170	ed04f2
14171	01f1e2
14172	17f5f3
14173	e2000d
14174	0621ec
14175	040509
14176	021800
14177	191f1b
14178	f001fc
14179	f9ff08
14180	0bf803
14181	edf306
14182	f500f1

14183	fd07f7
14184	1713f3
14185	effd00
14186	dcf923
14187	ec0000
14188	07f6f7
14189	14f824
14190	f50500
14191	ee08fc
14192	02f90f
14193	120dfc
14194	040202
14195	0e0de9
14196	e5f42b
14197	090b00
14198	eefbfe
14199	1b00ea
14200	07f1f4
14201	3134f4
14202	0532fe
14203	0e1506
14204	000009
14205	03ef02
14206	ededf0
14207	f3f70a
14208	000605
14209	09fef3
14210	f0ef03
14211	0012f5
14212	f30a09
14213	04fdfe
14214	050b0d
14215	0505fc
14216	020a0c
14217	0cfbfb
14218	f2fbff
14219	0bef05
14220	f307ef
14221	f1f9ef
14222	0d10f6
14223	f2eefc
14224	f8f6fc
14225	0e04fd
14226	f7f108
14227	faf90a
14228	fdf902
14229	0c0010
14230	f2f30d
14231	f0f7f1
14232	03eb00
14233	0eff0a
14234	01ec05
14235	faf4fc
14236	0704ee
14237	fdf4f2
14238	f50705
14239	f0f2fa
14240	ff0df2

14241	06f5fa
14242	ff0ef7
14243	0ef009
14244	06f7f5
14245	fdf0f9
14246	f90dfd
14247	fa0203
14248	f8f8fa
14249	f60a0e
14250	f9f6f3
14251	0d08fd
14252	0506fb
14253	0210fd
14254	f0f5fe
14255	0af400
14256	fdf801
14257	04fbf6
14258	02fef7
14259	09f711
14260	fa01f0
14261	f7fc07
14262	fdec0d
14263	0a11ff
14264	edeefd
14265	fe0605
14266	10aff
14267	f8faf2
14268	040bfd
14269	07f700
14270	0ded0b
14271	0a0d0e
14272	ff07f2
14273	fa09fb
14274	eef2f4
14275	f8fbfc
14276	0c04f9
14277	10ffeb
14278	0cf801
14279	09f511
14280	06ec04
14281	edf603
14282	0611f4
14283	f80cf0
14284	0b0df8
14285	ef06ed
14286	07030a
14287	080506
14288	f10dfe
14289	040204
14290	ecf7e7
14291	f400ea
14292	f30703
14293	f8ef11
14294	06f80c
14295	0ff304
14296	f3f0f3
14297	ed060f
14298	0a00f7

14299	0bfbfc
14300	06ffed
14301	fbf40d
14302	fe0d04
14303	0df1fe
14304	0c0bf9
14305	f4ee0b
14306	100dfb
14307	f604fd
14308	ec00fb
14309	02f0ee
14310	01f1f8
14311	ecfc00
14312	f1fdfc
14313	f007ee
14314	fdfff2
14315	f1ee02
14316	040100
14317	ec07f4
14318	f5f7fe
14319	eef005
14320	f800f2
14321	f408f8
14322	edfb03
14323	f50009
14324	f6f9f6
14325	0cf201
14326	090bf4
14327	04f502
14328	fdec05
14329	02ec0a
14330	f3efec
14331	0df303
14332	fdeefc
14333	06fb09
14334	fefef2
14335	fcfbe8
14336	e500fc
14337	f90700
14338	000002
14339	03060a
14340	09eff4
14341	fbf818
14342	08f104
14343	1df1e5
14344	e713f9
14345	ef0dea
14346	080e0b
14347	1e11ef
14348	f20b12
14349	0b120f
14350	f50120
14351	f8edf6
14352	f8f6fc
14353	ebe7d7
14354	f107f2
14355	170107
14356	2a1215

14357 0e111d  
14358 f9211e  
14359 180cfa  
14360 00fb29  
14361 04f2e6  
14362 fdfc36  
14363 2d311c  
14364 23ff16  
14365 0befdf  
14366 1f3fef  
14367 dbdd07  
14368 092f6b  
14369 560900  
14370 12dcf3  
14371 e1310b  
14372 070ae0  
14373 0a00fd  
14374 dbfc19  
14375 0ae8df  
14376 f5fde8  
14377 e90c09  
14378 03f5f9  
14379 f802ff  
14380 e9e1eb  
14381 0e23fe  
14382 130cf4  
14383 ff0915  
14384 e80f2d  
14385 09f4cd  
14386 12e5fa  
14387 000119  
14388 0df3fc  
14389 0ffde9  
14390 1608fa  
14391 1e2d0f  
14392 282119  
14393 010824  
14394 0f0d1a  
14395 ea2b32  
14396 231606  
14397 fb0d24  
14398 f1f5fa  
14399 e8f91c  
14400 26efd0  
14401 df21f1  
14402 1f1d28  
14403 2a0018  
14404 e5f7f5  
14405 e11ad7  
14406 120bfe  
14407 1f330c  
14408 f602ee  
14409 0cff17  
14410 1010fd  
14411 0805f6  
14412 d8fb15  
14413 623713  
14414 edfa07

14415 0ff209  
14416 e b d f 1 4  
14417 1 0 1 1 0 9  
14418 f 4 0 6 e 2  
14419 0 3 0 d 0 9  
14420 0 3 1 b 1 d  
14421 1 e 0 0 f f  
14422 f f f 9 0 9  
14423 0 1 0 2 f 1  
14424 f 9 0 d 0 d  
14425 1 5 0 8 f a  
14426 e d d 3 1 1  
14427 0 4 0 6 e 8  
14428 1 c f 4 f e  
14429 f 3 f d 0 9  
14430 0 c 2 4 2 1  
14431 f 2 0 3 0 e  
14432 1 5 f f e b  
14433 0 5 e e f 8  
14434 f f e b f f  
14435 e 4 0 c e 3  
14436 0 3 e 2 e d  
14437 0 b f 7 f d  
14438 f b f 3 f 9  
14439 1 2 e e e b  
14440 0 0 0 d f 9  
14441 f e 0 b 0 7  
14442 f b 2 4 1 0  
14443 1 8 f c f 4  
14444 0 5 0 3 e e  
14445 e e 2 3 1 8  
14446 0 f 0 0 1 3  
14447 2 3 2 9 3 8  
14448 f 5 f d 1 6  
14449 0 d 1 0 f 4  
14450 d 8 0 8 e 0  
14451 e 8 f a 0 1  
14452 0 0 d 9 e f  
14453 f 7 0 8 f f  
14454 0 9 1 6 0 8  
14455 0 0 e 6 f c  
14456 0 5 0 1 1 9  
14457 1 4 f 8 f 3  
14458 f e 0 1 f c  
14459 0 6 0 2 2 7  
14460 f d 0 1 f 8  
14461 f 2 0 0 f d  
14462 f 4 f c f 5  
14463 f 4 f f f c  
14464 1 c 0 6 e 6  
14465 f 2 f d f 7  
14466 0 0 2 d 1 1  
14467 f c f d 0 1  
14468 f e f 5 f 7  
14469 1 7 f 0 f 6  
14470 f d 0 0 3 5  
14471 1 b 0 f 0 d  
14472 0 0 0 0 1 9

14473	ecf9f6
14474	171be3
14475	040a1b
14476	002919
14477	f5110a
14478	1eed12
14479	1e15f7
14480	ebe913
14481	f8e500
14482	0df601
14483	fc02e1
14484	1dfcf6
14485	ff280b
14486	1e1514
14487	200c2a
14488	031c14
14489	f30bf b
14490	eed6f c
14491	f50d21
14492	1b07e9
14493	e4e8e3
14494	f625f9
14495	0405ea
14496	061aeb
14497	f613f4
14498	fd0f1f
14499	1ff4f6
14500	f6c71e
14501	f210df
14502	00d3df
14503	f2cd19
14504	171400
14505	02abcb
14506	09052a
14507	fe06f7
14508	ef1ffc
14509	131400
14510	09041c
14511	032015
14512	011501
14513	01fa03
14514	eb1516
14515	02f7fa
14516	f7f710
14517	240ff3
14518	06f6f5
14519	f505fb
14520	06f903
14521	0816f7
14522	070300
14523	ff00f5
14524	1000f c
14525	13f80b
14526	fd0a0a
14527	1a0103
14528	de04f7
14529	f8050a
14530	ef1600

14531 0c05d4  
14532 1911fd  
14533 fbfb10  
14534 13fc1a  
14535 15f607  
14536 fe1c02  
14537 f5f121  
14538 2a09df  
14539 f8ec0e  
14540 f600f9  
14541 0c0904  
14542 1d17fc  
14543 2206f8  
14544 fc2822  
14545 1a130a  
14546 11ed0b  
14547 d5fa25  
14548 faf2fc  
14549 f30707  
14550 09020a  
14551 17090f  
14552 110d12  
14553 ef0bfd  
14554 1ff9e3  
14555 07fb20  
14556 e4f119  
14557 17fc00  
14558 ffedda  
14559 f00cef  
14560 0f04f8  
14561 1005f7  
14562 00e80c  
14563 ff2a26  
14564 ecf301  
14565 01fcff  
14566 0ef908  
14567 0007fe  
14568 021dfd  
14569 0c1311  
14570 031a0f  
14571 121814  
14572 f80ef7  
14573 ebdded  
14574 f60305  
14575 09080e  
14576 160000  
14577 0a0421  
14578 01fc22  
14579 1f0004  
14580 f1050b  
14581 ffe6f3  
14582 f8ff05  
14583 170f1f  
14584 1c0108  
14585 051f12  
14586 0e0b09  
14587 f5e704  
14588 f5fb0d

14589 fc0601  
14590 00060b  
14591 11fc06  
14592 04e1f2  
14593 09170f  
14594 fd0305  
14595 1e0501  
14596 1afa16  
14597 1bfdf0  
14598 edfbd7  
14599 f20a01  
14600 01efe6  
14601 11f50b  
14602 04f717  
14603 f8f0fd  
14604 31f7d9  
14605 ec021d  
14606 0000fd  
14607 090af4  
14608 f5e3e8  
14609 f30800  
14610 f01ef3  
14611 0e2015  
14612 16f3ff  
14613 f01815  
14614 f705f6  
14615 fe0615  
14616 d4f6f2  
14617 ffe7f1  
14618 10db07  
14619 2a1c02  
14620 14f612  
14621 0d1200  
14622 fef6ff  
14623 fbfaec  
14624 fde203  
14625 eb0520  
14626 2bf1db  
14627 062224  
14628 080f18  
14629 01fa07  
14630 ff03bc  
14631 d5dcfc  
14632 d60100  
14633 112ed0  
14634 cbc213  
14635 210cf0  
14636 ddf4cf  
14637 f8d302  
14638 11080c  
14639 fad2f5  
14640 00ec1c  
14641 f00726  
14642 2006e4  
14643 f3cefa  
14644 dde31d  
14645 0800c5  
14646 05fef6

14647 f90ef4  
14648 ec1f16  
14649 220207  
14650 e7fc34  
14651 3a2af6  
14652 250a0e  
14653 ffe00  
14654 d3f81e  
14655 ff15da  
14656 fc170f  
14657 070cec  
14658 f7f0ee  
14659 f6e809  
14660 0119fb  
14661 3119c8  
14662 fc143e  
14663 2611f9  
14664 0c1710  
14665 f3f4cd  
14666 fb1010  
14667 f60cfa  
14668 0505f3  
14669 04df0e  
14670 000500  
14671 f022fb  
14672 2904f9  
14673 e4fd0d  
14674 ff04f9  
14675 efe0de  
14676 26061b  
14677 020325  
14678 244514  
14679 09201f  
14680 0e04fe  
14681 c3e608  
14682 faedf9  
14683 080012  
14684 0208e8  
14685 d8010a  
14686 e5f0fe  
14687 fef819  
14688 0104ef  
14689 fd1818  
14690 0406fc  
14691 18f50d  
14692 00f21a  
14693 020011  
14694 0f19ff  
14695 0015ee  
14696 e8f6fc  
14697 17f90c  
14698 0dfaee  
14699 f400ed  
14700 fef8f8  
14701 d8ef03  
14702 1306ff  
14703 ee0e11  
14704 16f80c

14705	f3fd1c
14706	0ff9f6
14707	fa1003
14708	00f8f2
14709	0c13f0
14710	e901f7
14711	031bf7
14712	e8031c
14713	270e06
14714	f0080e
14715	f5e8ea
14716	fff5ff
14717	0aee0a
14718	13070b
14719	0b0af4
14720	cbf517
14721	f20909
14722	fc08eb
14723	f303f4
14724	04fcea
14725	08f9e2
14726	f1e31f
14727	0c0cff
14728	eb0de4
14729	0b0bf1
14730	02031d
14731	0a1812
14732	ddf3f6
14733	0613ec
14734	1cf6e4
14735	e72112
14736	c002f8
14737	fadf0e
14738	3601f0
14739	f7f6fb
14740	000017
14741	05fcf2
14742	0002f6
14743	f8090f
14744	f6f403
14745	02f4f7
14746	eed10
14747	020a09
14748	0df5fb
14749	eefff8
14750	f801fb
14751	0df001
14752	f4f4fe
14753	f6f4fc
14754	fff300
14755	fff5fc
14756	f1f5ec
14757	0df50e
14758	140f0a
14759	fa0ffc
14760	ec0cf2
14761	080bee
14762	020e08

14763	08f10a
14764	f8f400
14765	09fa0e
14766	fdfa03
14767	01fe03
14768	0ef1f2
14769	05030e
14770	f208fa
14771	0fed06
14772	06f502
14773	00f2ed
14774	f00a0b
14775	0906f7
14776	09fc0c
14777	f1f8f5
14778	faed03
14779	f20ff8
14780	0c0cf9
14781	04edf9
14782	0b0ef7
14783	fb01f4
14784	0100f5
14785	fa0f07
14786	ed0cee
14787	02f105
14788	ef0cf0
14789	0410ee
14790	041004
14791	ee0008
14792	f2f6f0
14793	00eff6
14794	fd0ff0
14795	eef8f8
14796	070cfb
14797	f0080e
14798	050f04
14799	09faff
14800	08f903
14801	05fcf5
14802	0cfa0a
14803	01f207
14804	fafe0a
14805	0e02fc
14806	00fa07
14807	020605
14808	0d0500
14809	06fff0
14810	fdf007
14811	08f9f3
14812	f3f2f8
14813	fb0409
14814	ef09f0
14815	00f3ed
14816	02f4f0
14817	07fe04
14818	fd0209
14819	fbf703
14820	00f107

14821	f00709
14822	ee0bf1
14823	09f20a
14824	01f7f2
14825	02f310
14826	efed0b
14827	02fcf8
14828	03f80e
14829	ef0906
14830	faf507
14831	0a0ef8
14832	fdecfc
14833	0dfa0a
14834	07f8f0
14835	fc03fd
14836	0ff8f4
14837	0eecfe
14838	010500
14839	010108
14840	040008
14841	fefb01
14842	02f7fb
14843	07eefa
14844	ed0d03
14845	03f6f7
14846	ff08ef
14847	fffff1
14848	f0ec06
14849	f1f300
14850	ed01f5
14851	ec0502
14852	eef408
14853	f00f02
14854	ef00f3
14855	fdf401
14856	0df4f9
14857	f300fb
14858	eeefc
14859	0cfb09
14860	ec10ef
14861	0ef40d
14862	ef0e0a
14863	f2f5fc
14864	000af4
14865	ed050d
14866	f91403
14867	fd07f4
14868	fc0101
14869	f60000
14870	f908f9
14871	07f2f1
14872	0e03ee
14873	0df200
14874	00000b
14875	f6fdfd
14876	0e0d06
14877	160b17
14878	21fa0b

14879	100103
14880	f104f8
14881	eb150d
14882	1808f4
14883	ff0711
14884	070704
14885	effb09
14886	f607f3
14887	fcf901
14888	fd0008
14889	07130d
14890	0906f8
14891	08fe18
14892	e4d9ea
14893	fafcd4
14894	c1d8e8
14895	ebfaee
14896	00d3dc
14897	2122ff
14898	f10718
14899	27131b
14900	2b1413
14901	020b4d
14902	2b03f6
14903	11f6fd
14904	16ece6
14905	001e14
14906	03e7fc
14907	0b100b
14908	000f0b
14909	0504f5
14910	1a1209
14911	012d20
14912	ff040e
14913	f6040a
14914	f4e518
14915	0a0411
14916	33ffff
14917	c1d8e3
14918	e2f6fc
14919	ddd1da
14920	e7f5e5
14921	0e12fe
14922	2c1914
14923	f6f915
14924	000800
14925	fff014
14926	fd0104
14927	cddfdd
14928	fefaf3
14929	e8ef13
14930	07f3de
14931	cf0bf7
14932	e3deee
14933	000222
14934	f9f2fe
14935	15fa1e
14936	0d0624

14937 19241b  
14938 0cfa0c  
14939 1ef8df  
14940 cfee07  
14941 0b13fb  
14942 f5ea05  
14943 e911e9  
14944 dad6ee  
14945 1904f7  
14946 edd8de  
14947 f0ece8  
14948 07faff  
14949 1c10ee  
14950 ef0406  
14951 1eede6  
14952 ee010a  
14953 1702f2  
14954 200714  
14955 f60900  
14956 fe0b12  
14957 080cee  
14958 e407f3  
14959 f3e6df  
14960 dddfde  
14961 06dff0  
14962 1f1d18  
14963 151509  
14964 0d1727  
14965 fafff9  
14966 090c14  
14967 f9080e  
14968 0e100f  
14969 1a0010  
14970 170dfb  
14971 071116  
14972 041105  
14973 f4f60a  
14974 0afc7  
14975 f0f8f5  
14976 01ffe5  
14977 d602ff  
14978 2c07eb  
14979 1c0d2d  
14980 1a05fa  
14981 d31a07  
14982 04fcf9  
14983 0aef1e  
14984 19000e  
14985 0b0411  
14986 0118fc  
14987 f9000d  
14988 fc0213  
14989 1a160f  
14990 1412f3  
14991 080012  
14992 261600  
14993 000f0c  
14994 1406f4

14995	071300
14996	08f102
14997	f6001b
14998	0b1207
14999	f9f011
15000	f7eafb
15001	040af5
15002	e7e9db
15003	2c020c
15004	12fdfc
15005	fd1b0b
15006	e40107
15007	13fffe
15008	0000f6
15009	031106
15010	2a03f0
15011	fd210c
15012	031eed
15013	0c040c
15014	00eee4
15015	1aff14
15016	133017
15017	17f900
15018	100f04
15019	121213
15020	04f1fb
15021	10fc15
15022	e1f402
15023	1c1816
15024	071a21
15025	061817
15026	fff1e0
15027	f317df
15028	f6ecfc
15029	04fef0
15030	e010e1
15031	2416f8
15032	fb0431
15033	0b1110
15034	28142a
15035	011064
15036	430d0a
15037	28e812
15038	040506
15039	114018
15040	06d11d
15041	0e1b0f
15042	12fafb
15043	0b1010
15044	fc050b
15045	0f0a0c
15046	f9041f
15047	150c09
15048	f7fbec
15049	f2fb0d
15050	f007fc
15051	dfe8ed
15052	dbe6f9

15053	f6ece7
15054	ed0a12
15055	fa380e
15056	062217
15057	f0fc02
15058	270ee2
15059	edf3f2
15060	f40d02
15061	f0fbf4
15062	283de4
15063	fee912
15064	f7d8f0
15065	c11207
15066	1febb5
15067	e2fd2b
15068	100113
15069	1f1804
15070	330309
15071	f2f513
15072	fb2318
15073	36feec
15074	fefef1
15075	091600
15076	e2f5fb
15077	103404
15078	ced1fc
15079	f80f0a
15080	dd0cfb
15081	04dfe1
15082	ecdb40
15083	46140c
15084	fc14fc
15085	090af0
15086	0d070f
15087	031402
15088	fcfaf6
15089	fbf9fe
15090	09f912
15091	010808
15092	f5e907
15093	ebdee0
15094	fce8ee
15095	00f1f1
15096	131717
15097	06f111
15098	122410
15099	0f151b
15100	000b15
15101	fd1212
15102	101701
15103	f20513
15104	161007
15105	0c060f
15106	fc1009
15107	1825f0
15108	1704fc
15109	e0e7f6
15110	f4f7f0

15111	000a18
15112	181b02
15113	ea1106
15114	02f3fe
15115	f724fe
15116	0e08f5
15117	391f39
15118	fc121e
15119	0d1408
15120	ee0af2
15121	0400fe
15122	2afdff
15123	101921
15124	2f430a
15125	0d00f9
15126	2018fd
15127	02ed09
15128	0205f2
15129	16141c
15130	2adfee
15131	fff616
15132	ef0f04
15133	040e06
15134	cbead7
15135	f82ffa
15136	02defd
15137	28040d
15138	091b28
15139	0207e0
15140	fde702
15141	22ffe1
15142	000027
15143	f5fcf0
15144	05ffec
15145	f21519
15146	020201
15147	01f900
15148	21fafe
15149	fcf401
15150	021c29
15151	16f0fb
15152	ebeaf9
15153	161700
15154	03f5f0
15155	140915
15156	020f0f
15157	0afd1d
15158	160e11
15159	ddf10b
15160	03dbc2
15161	16ef48
15162	221afb
15163	eefcf2
15164	f226f7
15165	16f2f3
15166	ef1208
15167	1b09ef
15168	231e11

15169	001920
15170	051a1a
15171	2d16f6
15172	dae118
15173	e70b16
15174	18f7e6
15175	eeaaef
15176	030920
15177	201800
15178	fef916
15179	2212f1
15180	0efa09
15181	221506
15182	0e00fb
15183	f50af9
15184	ce0aff
15185	0a00c1
15186	eeee1c
15187	221614
15188	f6e5ec
15189	ee10ef
15190	0c00e4
15191	000c0f
15192	1d1714
15193	f3ec08
15194	000107
15195	101df7
15196	042112
15197	12ffff
15198	f2fae1
15199	cd0606
15200	f1fee5
15201	1100e1
15202	130e0e
15203	f0ed0a
15204	1c0d02
15205	0ff212
15206	0005f8
15207	f81306
15208	1400da
15209	ebf8f8
15210	fde5dc
15211	0efd09
15212	1b1b01
15213	18feed
15214	da120f
15215	03e7d0
15216	f1ef12
15217	1dff0d
15218	f3ef02
15219	0c00e8
15220	f80b16
15221	07090b
15222	03fdf8
15223	fafaf2
15224	0306f6
15225	080c15
15226	eef800

15227	21f0f3
15228	09e7d8
15229	f8000b
15230	f2060a
15231	edf80a
15232	0304f1
15233	2422ff
15234	060f1b
15235	e20a0d
15236	1819fd
15237	ee01f1
15238	150a0b
15239	151705
15240	130914
15241	0c14f9
15242	f81802
15243	e9ed08
15244	e9f4ff
15245	170cfe
15246	021f02
15247	180311
15248	fae400
15249	f50716
15250	001810
15251	131815
15252	f40f24
15253	030f02
15254	f2f401
15255	050af8
15256	16f2f2
15257	050302
15258	1a2bfd
15259	f50f04
15260	f1f217
15261	0f1206
15262	03edeb
15263	050c00
15264	1c1706
15265	fc0a1f
15266	071801
15267	fd07ff
15268	e7f4d1
15269	e318fb
15270	f60bf5
15271	170809
15272	fcff06
15273	14eadd
15274	151604
15275	06000b
15276	00000d
15277	f90016
15278	05fd09
15279	0af708
15280	0ef6ff
15281	f50800
15282	11f2d6
15283	d90cf3
15284	042d14

15285	190710
15286	02110d
15287	171e06
15288	08ee10
15289	f1f30b
15290	12f6e5
15291	eef618
15292	08eae2
15293	e710fa
15294	1404ff
15295	ed070c
15296	15f0e2
15297	d7f42d
15298	e814e2
15299	f1d5ca
15300	eefb1c
15301	16f1f5
15302	08d90d
15303	ff0af0
15304	1b18e7
15305	28000b
15306	04bc1a
15307	e40c2c
15308	29f2b8
15309	d9a1fa
15310	fa2b07
15311	25efe2
15312	e00011
15313	031b12
15314	0d1ff1
15315	11ed00
15316	080ff3
15317	ea1f07
15318	0df311
15319	fb0afe
15320	06f716
15321	fb0f1c
15322	d2fc14
15323	dd04e3
15324	f1dfc3
15325	03de06
15326	fef9ff
15327	02fd12
15328	dd100c
15329	2cf5f9
15330	0efe12
15331	fafb06
15332	f70516
15333	bf1c01
15334	19f7a8
15335	f61cff
15336	071606
15337	f9ccf7
15338	f605fd
15339	112929
15340	fe18fc
15341	1911ff
15342	13e9f5

15343	162e20
15344	0e1c09
15345	e7e105
15346	f9ff03
15347	160b04
15348	061020
15349	05e9d0
15350	b1ae17
15351	2132ff
15352	ff181b
15353	00ecfd
15354	080813
15355	ea090c
15356	fd16f6
15357	0006ef
15358	010300
15359	18ff0d
15360	06040d
15361	10000a
15362	f91328
15363	0ceaff
15364	e7e3da
15365	f7eafe
15366	0af7f7
15367	08fc0d
15368	1d0110
15369	060d14
15370	140a09
15371	fcfbfc
15372	f81a0a
15373	e9f102
15374	020c13
15375	110403
15376	051d15
15377	001103
15378	22eafd
15379	050bf8
15380	eaea09
15381	04f3f1
15382	08efe7
15383	eafc01
15384	2812eb
15385	230e1a
15386	e20404
15387	1d2bf7
15388	d00600
15389	fe0f00
15390	d9d312
15391	fc101e
15392	1704ef
15393	0bfc15
15394	11e717
15395	f60000
15396	0910ed
15397	101100
15398	09e80d
15399	eff505
15400	0e09fb

15401	180b07
15402	2a22db
15403	1616fb
15404	eb2c0f
15405	de12f4
15406	0be2db
15407	ed00e2
15408	f3f9e7
15409	fef4e4
15410	000005
15411	f6edeb
15412	071914
15413	faefe3
15414	0cf626
15415	fde6fa
15416	13110d
15417	33f8fe
15418	fcfc28
15419	e2eff6
15420	2d141a
15421	0dd5dc
15422	ea270f
15423	0b000f
15424	1600f8
15425	f00000
15426	fe1329
15427	1efaff
15428	f50900
15429	fef5d8
15430	e6fcf2
15431	010de7
15432	d3f500
15433	fa04f8
15434	13fb17
15435	211d12
15436	fe100f
15437	f9cc0b
15438	28dffa
15439	06012f
15440	540e00
15441	e7d912
15442	0a01e4
15443	05b7c9
15444	fcf8e4
15445	e0fb0e
15446	03fd10
15447	0a2007
15448	1d00f3
15449	101819
15450	1201f8
15451	fbe8fd
15452	1510fc
15453	0a1912
15454	0b08f3
15455	b5e20e
15456	0008f1
15457	ddec02
15458	0c250b

15459	050914
15460	230910
15461	02151a
15462	17faf5
15463	e5e420
15464	f1e510
15465	e6fe0b
15466	0201ed
15467	dcf812
15468	e20413
15469	0f311a
15470	eaf2f8
15471	0d1006
15472	07120d
15473	01140f
15474	16f100
15475	19061f
15476	1a08ff
15477	2c17ed
15478	0f0d10
15479	fcf602
15480	e0e606
15481	f9f2d8
15482	d8e2fc
15483	e500f5
15484	17dcf4
15485	34161b
15486	faf1eb
15487	0d1205
15488	fe0d11
15489	0af3fc
15490	190714
15491	080ffd
15492	200eee
15493	14030d
15494	f5d700
15495	07effa
15496	0df3ea
15497	cc01f6
15498	eb0b00
15499	09f204
15500	15feef
15501	072414
15502	ff1c00
15503	180101
15504	f0000a
15505	1c1512
15506	fff811
15507	1b07ef
15508	01fc12
15509	ff1206
15510	fcf109
15511	f9fff3
15512	03ff09
15513	e7ee0c
15514	0de614
15515	00f204
15516	ff0eda

15517	03eff9
15518	131007
15519	e20015
15520	1a17fc
15521	e1eadc
15522	fdfcf5
15523	eaf505
15524	271ffb
15525	01011a
15526	002308
15527	02f0f2
15528	190602
15529	091022
15530	270e17
15531	f6f519
15532	101716
15533	13f810
15534	010601
15535	0504f8
15536	07f5d9
15537	ecdd1b
15538	fb0002
15539	26e306
15540	150100
15541	ea11f5
15542	1108fc
15543	f00409
15544	000006
15545	f512fc
15546	050cf3
15547	05080c
15548	11fde9
15549	ee0513
15550	08fafc
15551	060212
15552	04030c
15553	f8f8fb
15554	e0f8fc
15555	ff060c
15556	2aeefc
15557	23241c
15558	13f211
15559	05210d
15560	0312f8
15561	da09f5
15562	fac6d4
15563	1dee51
15564	3b1ce7
15565	fd06fa
15566	001709
15567	ed0005
15568	0005e5
15569	edf8e9
15570	d70222
15571	17f7b4
15572	c6d019
15573	2305e1
15574	ec000c

15575 ece217  
15576 1b20fc  
15577 ffb8f1  
15578 1e0f0c  
15579 0cf713  
15580 fffc16  
15581 0fe6e1  
15582 ed08fa  
15583 0412f0  
15584 00f6f8  
15585 09f503  
15586 dd1c04  
15587 1a2003  
15588 fbe82b  
15589 2327fe  
15590 1602ea  
15591 f20812  
15592 f503f8  
15593 0b03dd  
15594 fe0002  
15595 fd17f9  
15596 d00511  
15597 0af4d0  
15598 e8f413  
15599 161be5  
15600 e4cef  
15601 f1fb05  
15602 f305e7  
15603 fff9d9  
15604 0e03fd  
15605 fcf3fb  
15606 e212fe  
15607 f30001  
15608 0800fe  
15609 0210fd  
15610 28dde7  
15611 cf1b13  
15612 1bddcd  
15613 f6df26  
15614 281912  
15615 1518f1  
15616 e52803  
15617 fa04e1  
15618 f9e206  
15619 d2fc07  
15620 17f9f4  
15621 26fe08  
15622 11f60c  
15623 0f03fa  
15624 e5fbf1  
15625 1b0cf0  
15626 efee03  
15627 101c13  
15628 00e00d  
15629 0f0b0e  
15630 0ffcf1  
15631 f6fd17  
15632 1009fa

15633	eff100
15634	e1f0f6
15635	fa0f09
15636	15030b
15637	e8eb06
15638	001df9
15639	ea01fc
15640	100e0f
15641	050a01
15642	05fa08
15643	0c0312
15644	0a0b00
15645	f6f70c
15646	dfd0b
15647	250efc
15648	0a021a
15649	06f518
15650	fcfb07
15651	f902f7
15652	031503
15653	12efee
15654	fff5ff
15655	fd320b
15656	02e811
15657	01f30f
15658	f5e9ff
15659	0df812
15660	0206f4
15661	091902
15662	f8f60e
15663	030dea
15664	f1fc02
15665	0b0cfa
15666	f519ff
15667	f5f900
15668	fcf905
15669	021b12
15670	1a03c1
15671	d8fa24
15672	1a0107
15673	1003f7
15674	fe0304
15675	010317
15676	0bfddb
15677	e3fef3
15678	0000eb
15679	2209fd
15680	2304f2
15681	10fe04
15682	1bff02
15683	1f0804
15684	0c0300
15685	1007e7
15686	162424
15687	291c04
15688	180228
15689	fb0105
15690	0b081e

15691 f5f41b  
15692 fc0404  
15693 0ef205  
15694 150f03  
15695 f4f3fa  
15696 d6bdc1  
15697 d0e5b5  
15698 ed0ffc  
15699 14edea  
15700 1d02f5  
15701 1c18fb  
15702 00ef24  
15703 f9f001  
15704 50430d  
15705 151f53  
15706 393611  
15707 04282a  
15708 1f251c  
15709 481715  
15710 152e46  
15711 0f5516  
15712 1008ff  
15713 241509  
15714 f90817  
15715 20ff0d  
15716 07040d  
15717 0f1a0a  
15718 f30303  
15719 070e07  
15720 f3fdfa  
15721 b3b8e0  
15722 d1bcd4  
15723 05d1ca  
15724 0b0dfe  
15725 1e05d1  
15726 181610  
15727 fdfdf9  
15728 09fee8  
15729 201307  
15730 1d001e  
15731 cc0723  
15732 f40ee3  
15733 d8f808  
15734 0eebea  
15735 f0d9f5  
15736 0a1438  
15737 fefd22  
15738 ef12fc  
15739 ed122a  
15740 08f2d7  
15741 0002e1  
15742 f6fc00  
15743 e7e6eb  
15744 000e10  
15745 0903fe  
15746 ebe5f4  
15747 dcece8  
15748 c1acc6

15749 f2d8be  
15750 01d3ce  
15751 cff405  
15752 0716df  
15753 16fe12  
15754 f805e5  
15755 111af6  
15756 f70df9  
15757 0a0f06  
15758 09ec0b  
15759 f21602  
15760 f6f2f9  
15761 fd0e04  
15762 bff3eb  
15763 efddb2  
15764 ffd7b3  
15765 17e3f1  
15766 f31413  
15767 f90b0b  
15768 091b1e  
15769 0df500  
15770 00e505  
15771 1f2307  
15772 092315  
15773 1302fa  
15774 fa0f0a  
15775 fc06fa  
15776 fb0501  
15777 f6190c  
15778 f5f509  
15779 09ea04  
15780 06ffee  
15781 f5f904  
15782 03f2e7  
15783 eee6f8  
15784 f8f3ee  
15785 0ffd02  
15786 0700f0  
15787 fefe05  
15788 081319  
15789 0f160a  
15790 07060b  
15791 f61f10  
15792 1b1417  
15793 0ef717  
15794 181a25  
15795 fff51b  
15796 1f00f2  
15797 0bf826  
15798 fb0713  
15799 f80e18  
15800 080306  
15801 f1faf9  
15802 0d06f8  
15803 f912fb  
15804 c3cdf4  
15805 c016ea  
15806 ddfbe1

15807	1dfd02
15808	2dfc08
15809	2529f2
15810	f1f212
15811	09fd01
15812	000006
15813	f6ffef
15814	0bf506
15815	fff9fc
15816	0bf301
15817	fb0b07
15818	fe04f3
15819	0bf6f1
15820	05eeec
15821	f2f9fc
15822	f104ec
15823	08ed00
15824	0d050c
15825	00f305
15826	02ffed
15827	0bff00
15828	f6fdf5
15829	070000
15830	f1fb04
15831	fbf1ef
15832	f8f500
15833	faf5fe
15834	f40009
15835	e91004
15836	f2f302
15837	f908ea
15838	0ceaf7
15839	e7f9e6
15840	e514f4
15841	f7f409
15842	ea0509
15843	fb0be3
15844	000a03
15845	ecf3ff
15846	eefef6
15847	fdedf1
15848	fbed00
15849	0305f2
15850	fcebf5
15851	0002fc
15852	fe07fb
15853	f3f7fd
15854	fdfaf1
15855	efe707
15856	00060c
15857	ed03f3
15858	f3ef00
15859	f40004
15860	fa090d
15861	000af7
15862	0bf905
15863	02ee04
15864	06feeb

15865 eff5f1  
15866 f6ef07  
15867 08eff8  
15868 f4edf4  
15869 fc0afb  
15870 e80d0a  
15871 08f0fc  
15872 01ff04  
15873 eff1ec  
15874 f6edf3  
15875 00ef08  
15876 06f909  
15877 ffec00  
15878 f50eef  
15879 f506e4  
15880 ff0a03  
15881 ed02ff  
15882 f0f507  
15883 f400fd  
15884 f6fae5  
15885 fcf5f4  
15886 f50605  
15887 08f703  
15888 f9eeee  
15889 08f903  
15890 f0edfd  
15891 0d01f1  
15892 0d0c03  
15893 e90208  
15894 f0fdfc  
15895 00ecf3  
15896 07f2ed  
15897 06ea06  
15898 02e307  
15899 ff0202  
15900 010f0b  
15901 faff08  
15902 ebf8f2  
15903 0006f3  
15904 0a0601  
15905 06ee06  
15906 fcf6f3  
15907 01f2fe  
15908 0e0bf4  
15909 f600f6  
15910 06f201  
15911 fd07f4  
15912 06f4f6  
15913 f5ed00  
15914 f2fa0c  
15915 00100c  
15916 fa01f1  
15917 f302ff  
15918 09f2fc  
15919 f9eef9  
15920 000603  
15921 0df1f7  
15922 0aeef1

15923 f80af2  
15924 00faf7  
15925 e90400  
15926 fcec07  
15927 0805e6  
15928 07ecf7  
15929 f2f805  
15930 f30500  
15931 ec07ee  
15932 f4ef0a  
15933 f5fcfd  
15934 ff05ef  
15935 f4fa0a  
15936 0001f8  
15937 00f3e9  
15938 f3f8f8  
15939 05f2ec  
15940 eaef12  
15941 f102f0  
15942 f6f003  
15943 02fe00  
15944 010df4  
15945 020bea  
15946 0000e5  
15947 ebdb05  
15948 d7121c  
15949 00fafb  
15950 ede60a  
15951 ecf6ee  
15952 180022  
15953 0ff2e6  
15954 ff0105  
15955 e6fdff  
15956 221405  
15957 03e2cb  
15958 cf0aed  
15959 f5fde9  
15960 172008  
15961 ecf622  
15962 1e070c  
15963 260cfe  
15964 cf0a35  
15965 ea12a3  
15966 de06fd  
15967 fb02e2  
15968 f9f2fc  
15969 d201f6  
15970 fef2f0  
15971 fbe9f0  
15972 1ff3da  
15973 e8e91c  
15974 03d1ed  
15975 e4f923  
15976 0aeecf  
15977 c5deed  
15978 12fbed  
15979 e7b7e2  
15980 eafaff

15981	f603fa
15982	000a03
15983	e82512
15984	07e700
15985	03f51c
15986	191e0b
15987	fe0009
15988	420d01
15989	d10930
15990	eb20c4
15991	e0ed00
15992	090bf5
15993	e9101a
15994	f3ef03
15995	faf50a
15996	f5000c
15997	e7e2fe
15998	27fee8
15999	f3d81e
16000	03f1ea
16001	02fef6
16002	fd00f9
16003	0be90e
16004	110317
16005	edf900
16006	e7e9ff
16007	012909
16008	fbfdfa
16009	270d04
16010	ee150a
16011	0b1712
16012	041200
16013	26080b
16014	f21910
16015	1d30f1
16016	dfe006
16017	05050a
16018	080c1c
16019	241a16
16020	060df9
16021	0508f7
16022	f002ec
16023	e70cfb
16024	0ef4ea
16025	f5e8f2
16026	02e810
16027	e801fc
16028	1d04f4
16029	010a13
16030	3a390f
16031	dfec17
16032	fb00fc
16033	0006f5
16034	e9f6fa
16035	e60907
16036	05e3f7
16037	fb17fe
16038	f800f4

16039	1406ff
16040	e4ee28
16041	1c00fb
16042	0503ff
16043	19f406
16044	000509
16045	071211
16046	ff0818
16047	1301f8
16048	ea0e03
16049	e9fde4
16050	e9f8fc
16051	f2f2f7
16052	0e170e
16053	03edf0
16054	07150c
16055	d4e30e
16056	0bef06
16057	e7caf4
16058	0613ed
16059	e50ef4
16060	ea07f2
16061	0a06fa
16062	eaf720
16063	0101f3
16064	04fff5
16065	10ef1c
16066	060ff7
16067	e2f8e1
16068	030506
16069	1707fb
16070	f8fc0e
16071	15fef5
16072	fe2e36
16073	02efe3
16074	08ebee
16075	f2e506
16076	03ebe2
16077	f7f2fd
16078	ee0700
16079	f504eb
16080	0000f1
16081	000015
16082	0000e8
16083	0000f1
16084	000007
16085	0000fe
16086	000010
16087	0000e4
16088	000034
16089	00002d
16090	00002d
16091	000007
16092	000012
16093	00007f
16094	000044
16095	0000ff
16096	00000f

16097	000012
16098	0000df
16099	00000b
16100	0000f6
16101	000009
16102	000050
16103	0000d1
16104	000067
16105	0000e0
16106	000028
16107	0000e3
16108	0000cb
16109	000072
16110	0000de
16111	0000be
16112	000055
16113	000006
16114	00001a
16115	000029
16116	000051
16117	000030
16118	000053
16119	0000dd
16120	0000d5
16121	000014
16122	0000b4
16123	000047
16124	00003f
16125	0000ca
16126	0000dd
16127	000035
16128	0000ac
16129	000039
16130	000072
16131	0000cd
16132	000056
16133	000029
16134	0000b8
16135	000022
16136	0000bc
16137	000002
16138	00001f
16139	000036
16140	0000d9
16141	000013
16142	000050
16143	000072
16144	0000cb
16145	000020
16146	0000ec
16147	000001
16148	0000dd
16149	0000ca
16150	0000df
16151	0000fa
16152	0000e1
16153	000040
16154	000015

16155	00001a
16156	0000e1
16157	000003
16158	0000a5
16159	0000db
16160	000028
16161	000035
16162	0000d6
16163	000021
16164	00001c
16165	000039
16166	0000f5
16167	0000e1
16168	0000da
16169	000015
16170	00001a
16171	0000cc
16172	000020
16173	000039
16174	000023
16175	00000f
16176	0000f2
16177	000098
16178	000024
16179	0000eb
16180	00002a
16181	00001d
16182	00003b
16183	000012
16184	0000c4
16185	000028
16186	0000dd
16187	0000cf
16188	0000cc
16189	00001e
16190	00000b
16191	000011
16192	00005b
16193	00000c
16194	000057
16195	000009
16196	00003d
16197	00003a
16198	000049
16199	0000e4
16200	0000d7

Listing 24: Packed FC2 weights and biases for three-lane FPGA ROM loading.

```
1 e9f9e3
2 fc152d
3 121bd6
4 f0f500
5 fa1a07
6 fd4de7
7 280806
8 2d1304
9 00d918
10 293b28
11 e9d9ec
12 db30eb
13 dff9f5
14 0f2a0a
15 36ec00
16 c703c9
17 d7cd1b
18 fafbf3
19 dc14e6
20 f31c0d
21 c21d03
22 0521de
23 07ed00
24 ede2da
25 2e1fd5
26 24f7d5
27 dfde08
28 f85aa0
29 e52fe8
30 0d09f5
31 e530d7
32 d9f93e
33 e4d11d
34 e01f26
35 23ef30
36 0bdacf
37 14f01f
38 3910fa
39 00eef5
40 e6e095
41 1630d4
42 e7e3d4
43 1c22ea
44 051819
45 f3d3dc
46 fcd2f0
47 1e25b9
48 13de14
49 0ddb02
50 e42413
51 2fd4dd
52 100bfa
53 3e1b02
54 26f6fd
55 e92b05
56 071dfe
```

57 f2e412  
58 e027e3  
59 e91304  
60 e50012  
61 1ddbd2  
62 3df7fa  
63 0235e1  
64 0ffbfd  
65 faefe8  
66 fe0e27  
67 33ee0a  
68 eef7e3  
69 0f0efd  
70 141006  
71 edf10d  
72 05b724  
73 f80b01  
74 f7b21d  
75 2cd603  
76 0fd52e  
77 e2efe9  
78 ef1a04  
79 e2ea22  
80 321604  
81 def9ea  
82 e5fa02  
83 020d07  
84 e4dd0c  
85 f6ef14  
86 040d17  
87 271c17  
88 e52fe3  
89 0f1747  
90 eb0cf0  
91 e9f422  
92 f329e7  
93 fcd17  
94 fb19da  
95 122b4e  
96 24e538  
97 fb351f  
98 c2ec2a  
99 e3f618  
100 e60935  
101 0acdda  
102 da12fa  
103 0fe9ee  
104 e1fc15  
105 fdd0f2  
106 213c17  
107 efe3ed  
108 0f71cb  
109 ecda02  
110 1610f8  
111 fe2af1  
112 0723e8  
113 ead805  
114 dd48e9

115 010a00  
116 d0fea0  
117 e24b17  
118 dfc90a  
119 020ad6  
120 d517d8  
121 eb1c1f  
122 ea24f1  
123 18c134  
124 f8dbfb  
125 021d12  
126 f1cdfb  
127 f9f804  
128 fa0d01  
129 ee28f1  
130 e1eb13  
131 f21619  
132 08222a  
133 fa2e25  
134 e817e3  
135 0cf3da  
136 1a22ed  
137 fad5d1  
138 fd2122  
139 1537ff  
140 fac8e2  
141 253202  
142 f0ea0f  
143 ff40de  
144 ee12f5  
145 183819  
146 eec6fd  
147 10ef1c  
148 d10ee4  
149 e7e1fe  
150 cc19f4  
151 10f0e1  
152 f5f3de  
153 2be705  
154 f6e7fe  
155 19ec44  
156 42e93c  
157 05b326  
158 121c43  
159 d01b24  
160 0af328  
161 27dc08  
162 0fe534  
163 fd1407  
164 19db00  
165 2a1431  
166 0f21fb  
167 f6efd2  
168 05f4d8  
169 26d5e7  
170 d9fffc  
171 e210da  
172 f11ddf

173 df3b05  
174 e23cd1  
175 26130c  
176 fdd705  
177 da0918  
178 f62537  
179 1717fa  
180 e6262e  
181 2b18f3  
182 fb0a32  
183 f9fe06  
184 02f9d3  
185 18a6eb  
186 e636eb  
187 0b1cd5  
188 f35194  
189 092510  
190 f5f6fe  
191 0be80a  
192 f03048  
193 d00000  
194 e7e721  
195 0f1d0c  
196 fd1cd0  
197 0b302d  
198 4521eb  
199 431b17  
200 e2e31b  
201 d2ff22  
202 e3ebec  
203 fb1ae9  
204 0c04cb  
205 c62714  
206 393838  
207 21f41e  
208 07003c  
209 002602  
210 001c0b  
211 e207e2  
212 12e6f7  
213 2339ec  
214 ea20ef  
215 d4f520  
216 1bcce3  
217 fefee5  
218 d13724  
219 0e24c5  
220 d93130  
221 21dd1f  
222 fdd9ee  
223 2a2810  
224 d8291a  
225 47c7f4  
226 d609e4  
227 dfe128  
228 1f3784  
229 ffdc1d  
230 fa1ef4

231 1d11f7  
232 dc0b1c  
233 e01031  
234 e71cf5  
235 df14f0  
236 e9f805  
237 d8ed15  
238 493ff2  
239 420924  
240 e7fc24  
241 00baf1  
242 e6e61d  
243 00e9ec  
244 06021d  
245 15c82b  
246 0e44d9  
247 1ffe1d  
248 df0f1f  
249 0debf7  
250 e61301  
251 e8b014  
252 2c2af4  
253 3de41e  
254 020009  
255 e2d39e  
256 f4da0c  
257 d31a08  
258 c03a3f  
259 0ed9c5  
260 fe4832  
261 ccd01f  
262 3f0932  
263 1cd605  
264 13ddde  
265 1ac2cf  
266 f60a04  
267 0b39f3  
268 2960d4  
269 1024ed  
270 f20101  
271 171204  
272 110952  
273 44f6cc  
274 12e34d  
275 15ea35  
276 0130f0  
277 ed0819  
278 34c4dc  
279 1ff5ef  
280 2318d1  
281 2ad726  
282 07e830  
283 e00435  
284 1f2bde  
285 221d3e  
286 25d623  
287 fc293a  
288 0800ff

289 f02910  
290 f4e400  
291 ffd21  
292 dfef23  
293 c82ef0  
294 17cc04  
295 f32d39  
296 f3ebd7  
297 f0081a  
298 e01230  
299 dfc725  
300 07e5e5  
301 ef3600  
302 c5e5da  
303 0be147  
304 180923  
305 1a06e0  
306 fb0ed6  
307 f83805  
308 36e25c  
309 e703ef  
310 012c31  
311 0fbec  
312 15e233  
313 1fed06  
314 db2832  
315 0b21e8  
316 b71240  
317 12092c  
318 3b0f31  
319 3b36ce  
320 161459  
321 e1f0f1  
322 1b0dfa  
323 1513dd  
324 0600f4  
325 e3f01e  
326 e11cc2  
327 d63e3b  
328 c82ad3  
329 ed1312  
330 220a14  
331 d63724  
332 b5f413  
333 e20a05  
334 02e222  
335 3f270a  
336 1ce71d  
337 4635eb  
338 0b2df4  
339 f5c205  
340 ca09de  
341 e5000a  
342 2ceae2  
343 3de1ff  
344 09f9ed  
345 f43417  
346 06fc1e

347 0c16dd  
348 3ec105  
349 2be9f0  
350 2c2438  
351 1efb24  
352 f5e4c5  
353 0119f8  
354 ee3820  
355 ededc9  
356 def2e2  
357 e43bdb  
358 d6b9cd  
359 dc09d5  
360 d2fae9  
361 181613  
362 eaec05  
363 3d11db  
364 21dee6  
365 f2d0f9  
366 e6f531  
367 4831e8  
368 3228da  
369 fadbfe  
370 f821e0  
371 fcef0c  
372 d5cb30  
373 ef1419  
374 ec3001  
375 f809e3  
376 0bee3a  
377 f7dd00  
378 ce3d50  
379 05cfe5  
380 c30ae9  
381 19caca  
382 14f132  
383 101c59  
384 ea0f01  
385 edbdf0  
386 20e9fe  
387 33d1ea  
388 fde010  
389 05dc2d  
390 421a02  
391 033df1  
392 1ce539  
393 fc211f  
394 e92bd8  
395 e7ee24  
396 fc0027  
397 e52f22  
398 15cf13  
399 1731f9  
400 1ded16  
401 160ee4  
402 06391e  
403 41e003  
404 0be2d1

405 f9e5cc  
406 cbe604  
407 ea0638  
408 080701  
409 ee1cbb  
410 1b03e3  
411 eb1bf8  
412 c03036  
413 f6e6e1  
414 0bc923  
415 c4ec10  
416 d618ea  
417 08ddff  
418 fdef26  
419 eef300  
420 f4250c  
421 0706fa  
422 3816cd  
423 2f4402  
424 fd230d  
425 d70009  
426 1efd37  
427 1f2cde  
428 1ff324  
429 0ddc04  
430 dc17ff  
431 2ac8ff  
432 2514cc  
433 0feff1  
434 fbfad5  
435 0e1ae4  
436 1429db  
437 db0ade  
438 211704  
439 f9e10e  
440 f00527  
441 07331a  
442 202ffb  
443 1fe410  
444 fb0108  
445 310af6  
446 16ca0f  
447 00fde4  
448 120d33  
449 f7e2d7  
450 212d1b  
451 f7dd1c  
452 dc181f  
453 07ddf3  
454 f1d747  
455 0329fb  
456 1c16f7  
457 1ecd10  
458 f526db  
459 250c46  
460 e4f0e4  
461 30052a  
462 15e726

463 e233b1  
464 fdde24  
465 0730d3  
466 1ef834  
467 f1431d  
468 ebb62d  
469 f6e6f6  
470 f73b01  
471 d9231d  
472 f3edcc  
473 17efed  
474 06f6f4  
475 0525f8  
476 1ded29  
477 eedc2a  
478 d1f72f  
479 d4100d  
480 35141f  
481 ce140d  
482 14fb19  
483 f3222b  
484 e41241  
485 03394e  
486 1b00da  
487 e7090a  
488 2ae9e4  
489 1d1723  
490 2714f1  
491 ddfcf3  
492 4b0ae0  
493 c33fdb  
494 17cdd5  
495 0adf13  
496 2f0008  
497 2b3821  
498 edeb11  
499 0117e0  
500 32e30e  
501 25f6c6  
502 a1fc2c  
503 dde33a  
504 ea22df  
505 052e12  
506 fb3400  
507 ddd8e4  
508 f61ef6  
509 0feb1c  
510 03ce0c  
511 f1d1df  
512 e654e1  
513 1815e5  
514 e35a2b  
515 e0dd51  
516 00d611  
517 ec0614  
518 013123  
519 3733e7  
520 e21525

521 00e2eb  
522 fbd6fe  
523 e5f615  
524 0f0f14  
525 f93f36  
526 003fd7  
527 1e19e5  
528 e92929  
529 e2e919  
530 17e9e9  
531 19ceeb  
532 3800db  
533 e02ad0  
534 30e6f1  
535 e4d520  
536 11c0ec  
537 d649e9  
538 d1e3d7  
539 f314ea  
540 071c30  
541 f3eb10  
542 0df517  
543 13cf0b  
544 e8e6f2  
545 1500fe  
546 f04705  
547 0cfcea  
548 2240ac  
549 19f9f7  
550 001f03  
551 f224e9  
552 ea3412  
553 e4172c  
554 12ed45  
555 08dbeb  
556 da0900  
557 0f2f35  
558 3c14fa  
559 140009  
560 cc002f  
561 161c10  
562 1d06e9  
563 ff1b38  
564 f10ef2  
565 e83d3d  
566 dcd11f  
567 fad81a  
568 35112c  
569 1b2309  
570 f42708  
571 d32424  
572 fdd7f8  
573 e40bce  
574 240317  
575 fff207  
576 0af2e4  
577 e81a07  
578 08df15

579 cb3332  
580 dddbdc  
581 37f8d6  
582 e610e3  
583 2adbe9  
584 06fd0d  
585 2a430b  
586 ebfbc  
587 11000d  
588 db0c6  
589 27151d  
590 d7ecff  
591 e60b27  
592 e01c2e  
593 00e900  
594 254c2f  
595 1be110  
596 df221d  
597 d8060f  
598 39fc11  
599 e71bfe  
600 2ceb41  
601 202a1a  
602 fafb1d  
603 05004a  
604 e7f5e6  
605 d32238  
606 f5ef2c  
607 00f04b  
608 282361  
609 e1f0cb  
610 09f6ed  
611 122c24  
612 42385c  
613 463e2c  
614 1fff12  
615 d224ec  
616 0b4400  
617 4b1319  
618 51e635  
619 b21cec  
620 eaf900  
621 190128  
622 d60721  
623 28e8de  
624 e1e12d  
625 0f1d0a  
626 e6c917  
627 d225f8  
628 ec03b2  
629 20fbff  
630 ae00e4  
631 f017f0  
632 e02025  
633 490411  
634 1fd90b  
635 d4e915  
636 fbfbdc

637 27cbeb  
638 ede824  
639 fd0bf2  
640 e5da01  
641 2c3704  
642 2403fd  
643 f2e4e2  
644 15e0ff  
645 f002e4  
646 e31fe8  
647 f334f1  
648 de15ef  
649 02d1f4  
650 1b2205  
651 0843e8  
652 06c727  
653 2dfcd7  
654 1fe3f1  
655 042c0d  
656 e228c8  
657 f4fede  
658 441c02  
659 f7ea40  
660 2318cd  
661 2e3211  
662 392c06  
663 13461c  
664 16dfeb  
665 f4fc04  
666 eac913  
667 0aebf1  
668 09f16b  
669 e2ee00  
670 001627  
671 e617db  
672 e3d419  
673 e2dee4  
674 cf0adf  
675 dbe3e5  
676 de162b  
677 fb0f1e  
678 f4f0e7  
679 d1f834  
680 e9db10  
681 1233de  
682 39eb09  
683 daef35  
684 e91125  
685 24df2a  
686 1623fd  
687 e2f1e1  
688 d2102b  
689 f2ea19  
690 f7e4f7  
691 e34cfe  
692 13270e  
693 3dfdf7  
694 0c28f9

695 2bfa17  
696 0e48ed  
697 f0cb27  
698 fafae0  
699 3125e9  
700 1dd107  
701 28e9fa  
702 2e11fe  
703 d209db  
704 d323d8  
705 dd64cb  
706 0ce9fa  
707 ebdb1a  
708 d5f7cc  
709 ef1cf6  
710 fef7d3  
711 19f2f4  
712 ed1ecc  
713 d81e34  
714 28b7e3  
715 08f30f  
716 570326  
717 f0efe3  
718 06201e  
719 f8f1ec  
720 e6fce9  
721 2d1403  
722 d4faef  
723 0bf4cd  
724 0d200f  
725 23d8b7  
726 fddf4b  
727 2c09f9  
728 0de501  
729 19e416  
730 2428ed  
731 00d203  
732 00e90c  
733 18fa1f  
734 00ffcb  
735 dc151d  
736 e513fb  
737 99b6ff  
738 1d1820  
739 fdf61f  
740 e50824  
741 4b15fe  
742 c6edf6  
743 1b180b  
744 e209f9  
745 ec0c6  
746 ddb3f8  
747 db461c  
748 fd9d5c  
749 e0e721  
750 2d292a  
751 e1f722  
752 13de24

753 30e8fc  
754 c7fad3  
755 180cf9  
756 26dc08  
757 07d026  
758 2e15e6  
759 e73209  
760 200d21  
761 ed23e6  
762 1c1c10  
763 332d08  
764 d61003  
765 e2fd17  
766 f2d8d9  
767 f21bf6  
768 193c14  
769 2afe37  
770 31e61a  
771 0914f0  
772 fbe436  
773 effff9  
774 19e0fc  
775 f609de  
776 343933  
777 f91005  
778 31f422  
779 2a311d  
780 0309ca  
781 1af3f4  
782 1ef233  
783 ecf9c6  
784 f9dce7  
785 e31531  
786 df0904  
787 2bea1a  
788 0d2d06  
789 fcfde0  
790 16f618  
791 1405e8  
792 f400ec  
793 e720d3  
794 2f12d0  
795 e30003  
796 212bf6  
797 e80ed3  
798 ccb308  
799 13f2bb  
800 0bd2c4  
801 37fb07  
802 240209  
803 3af819  
804 f0e706  
805 d5ffde  
806 e61e22  
807 0fd023  
808 e4e02b  
809 e3f0bf  
810 242f1d

811 0003e4  
812 e00001  
813 552adf  
814 0bf118  
815 2023f1  
816 e2fbe3  
817 0906f0  
818 30ee27  
819 f60c38  
820 2b11ef  
821 0b1925  
822 0404fd  
823 0e20df  
824 ffd8e4  
825 00f012  
826 dbceee  
827 e302d8  
828 17addd  
829 18ff0f  
830 cff002  
831 0eee26  
832 00060e  
833 fb09fc  
834 39e0bd  
835 faeabd  
836 13dbfe  
837 2adacc  
838 c9e021  
839 edf258  
840 1c02c3  
841 172217  
842 e62e00  
843 eae607  
844 1a0000  
845 2fe9e9  
846 e04ab0  
847 d034f4  
848 e305db  
849 1b08e4  
850 ff2501  
851 f4d2fc  
852 17f4a9  
853 072112  
854 251748  
855 421dc7  
856 e6220f  
857 42c7ec  
858 13311b  
859 feed14  
860 f223f6  
861 d80e1b  
862 3821d8  
863 112a2e  
864 e9070a  
865 dc39cb  
866 f2ec32  
867 e1da1f  
868 4d00fd

869 1adf14  
870 1a140d  
871 0b24ff  
872 0aca26  
873 27ea2a  
874 bfc735  
875 13081d  
876 f3ea14  
877 ed4bfd  
878 2fe304  
879 f5e626  
880 f91da2  
881 e3eafa  
882 0df812  
883 08301c  
884 eaedf1  
885 c90eeb  
886 001a04  
887 d83218  
888 e30cf6  
889 e6dbcf  
890 f61114  
891 fd4000  
892 cbd352  
893 af41f8  
894 fcc604  
895 13fc00  
896 0fbbc4  
897 1b0916  
898 26e2e2  
899 f0f0cf  
900 2220f3  
901 d3fa0e  
902 2cf6d0  
903 f9f844  
904 1a071a  
905 de2bd4  
906 e40ee0  
907 e50bdc  
908 1a2d3a  
909 19fc1b  
910 040527  
911 00f023  
912 d20db9  
913 d9fddd  
914 ff37e8  
915 ef03ef  
916 f8261a  
917 f1350e  
918 281c1e  
919 18fb1f  
920 d2e75b  
921 0401d7  
922 2ef3f3  
923 343bfe  
924 141ed8  
925 fdc8ff  
926 0f0426

927 1df628  
928 d1232f  
929 fce115  
930 e9f11a  
931 ec1601  
932 c4280d  
933 290903  
934 f832f0  
935 3a00a0  
936 030847  
937 09cdf9  
938 e60603  
939 363145  
940 ef1ad2  
941 f11128  
942 4f0e21  
943 0b3be8  
944 dbdeeb  
945 00ec04  
946 fb1a32  
947 09f7e1  
948 ecf000  
949 0ff1e3  
950 f3092a  
951 12fa14  
952 db85b  
953 1c01ec  
954 0ab6ec  
955 f127c9  
956 fc2908  
957 ef0605  
958 ffa6ed  
959 1b0e04  
960 f20feb  
961 2afa12  
962 e5042b  
963 2f10dd  
964 1319bd  
965 0d04f9  
966 f1280d  
967 12f4b8  
968 18dbf1  
969 0ad6ed  
970 eb2416  
971 e32f25  
972 0e08f1  
973 1a33ce  
974 f10a1e  
975 d8f4fd  
976 030bc8  
977 f92110  
978 1b2639  
979 09ffd9  
980 faf008  
981 f334dd  
982 0b1aec  
983 ee4f4f  
984 0edf00

985 2eece8  
986 ddae0e  
987 031012  
988 16c632  
989 dee81d  
990 10022a  
991 25f9f0  
992 d6f83f  
993 d81404  
994 e2f4d4  
995 e7d831  
996 23e4e0  
997 f6e4ef  
998 276208  
999 122661  
1000 e5de0a  
1001 effc1c  
1002 ee1aeb  
1003 0d0f04  
1004 e8e9cf  
1005 fedef5  
1006 d906cf  
1007 f01fe7  
1008 f207ea  
1009 1a1bdc  
1010 d4f4e7  
1011 e7e3e9  
1012 09060a  
1013 fbf3f0  
1014 eaf91c  
1015 e70f21  
1016 e30c0a  
1017 f5e0ec  
1018 e9e901  
1019 06f9e8  
1020 001211  
1021 05edd8  
1022 d1e3ef  
1023 ff10f0  
1024 f0081d  
1025 e31600  
1026 2b18ea  
1027 12d4f3  
1028 05fecd  
1029 ef26e8  
1030 e21613  
1031 d9d4f7  
1032 f6e9f9  
1033 0feb2b  
1034 ddd9cc  
1035 f211e1  
1036 19e4c9  
1037 dd11d7  
1038 c8eae8  
1039 fd1ddd  
1040 cef5ef  
1041 0006d9  
1042 21d70f

1043 46020b  
1044 0b13c5  
1045 ed3512  
1046 050f0e  
1047 2624fa  
1048 d302c9  
1049 e818e0  
1050 0fdf00  
1051 183014  
1052 0209de  
1053 1f31d2  
1054 e0121c  
1055 fc003a  
1056 f0d7ff  
1057 2d150e  
1058 2f1b0b  
1059 183728  
1060 db190c  
1061 08e81b  
1062 3710be  
1063 012d35  
1064 ebf0d3  
1065 1916ea  
1066 d9f21c  
1067 f90ae5  
1068 3a9ae8  
1069 12e443  
1070 f216fc  
1071 ebdae6  
1072 da02f0  
1073 f90711  
1074 031805  
1075 f51bc3  
1076 17ffe7  
1077 240fff  
1078 0835e4  
1079 d7fb39  
1080 ce2d26  
1081 2adc07  
1082 f70d07  
1083 1025fe  
1084 e0f0c8  
1085 d9110b  
1086 e304fb  
1087 1dfef2  
1088 fede27  
1089 15ebe9  
1090 e1f8e9  
1091 fc5fdb  
1092 f80a10  
1093 10210b  
1094 da03d2  
1095 db1eb3  
1096 e6f90f  
1097 1709e7  
1098 de1c1d  
1099 083114  
1100 090820

1101 ed1515  
1102 32f222  
1103 19f042  
1104 f5d804  
1105 2fffc  
1106 e8f7eb  
1107 db1213  
1108 3b28e5  
1109 ea273b  
1110 d01ffe  
1111 e8150d  
1112 e01c19  
1113 3aec01  
1114 dedc22  
1115 272c07  
1116 3713f7  
1117 08012b  
1118 26f9d1  
1119 1a2142  
1120 dd0e0b  
1121 32310c  
1122 294214  
1123 01f8e6  
1124 e1e2fb  
1125 3bf304  
1126 ebe9ce  
1127 ed2d2a  
1128 fbeec2  
1129 0b1a08  
1130 ee1ad3  
1131 2a00f1  
1132 ccd701  
1133 e607f7  
1134 22000b  
1135 251aee  
1136 3907c5  
1137 2c0502  
1138 df0ef6  
1139 24df04  
1140 05f9f4  
1141 0723d2  
1142 3824d4  
1143 093b2d  
1144 f21cd7  
1145 d026fc  
1146 0b062e  
1147 11f508  
1148 18cd37  
1149 24fb0d  
1150 183c10  
1151 1feff2  
1152 dab715  
1153 131fea  
1154 bd2801  
1155 e621aa  
1156 f60107  
1157 0728db  
1158 c81b11

1159 fe06df  
1160 401bd6  
1161 cbc609  
1162 10dbd9  
1163 10db0d  
1164 dfdedf  
1165 e6100e  
1166 d4e614  
1167 fcd717  
1168 ebeadb  
1169 dc05f7  
1170 06e11f  
1171 ead9e5  
1172 d22cdb  
1173 0aecf9  
1174 f7e71a  
1175 ebf00f  
1176 f206e8  
1177 15db2b  
1178 02d10f  
1179 ffd8fe  
1180 0009fc  
1181 f01b1c  
1182 f90fd7  
1183 df10e2  
1184 def2f4  
1185 f2f21a  
1186 28d7f9  
1187 fde4f0  
1188 d20307  
1189 e417db  
1190 e40ef2  
1191 e1101e  
1192 0d1cdd  
1193 d90ce3  
1194 19f7f7  
1195 1821f4  
1196 d01512  
1197 1ce914  
1198 190cdd  
1199 03c4ea  
1200 1d23c5  
1201 1af11a  
1202 dd3134  
1203 133201  
1204 f5f6e4  
1205 035815  
1206 1c0700  
1207 d9e6ed  
1208 f932e3  
1209 f7e3e6  
1210 0b0801  
1211 2c2e13  
1212 16c529  
1213 f0f620  
1214 032913  
1215 112557  
1216 e43300

1217 1d1d2f  
1218 54f00b  
1219 e6e91c  
1220 d4e3d8  
1221 0ce905  
1222 f0090a  
1223 14183e  
1224 f0eaef  
1225 db1e53  
1226 ea2402  
1227 1602fb  
1228 2cc711  
1229 10d92a  
1230 030315  
1231 18ed19  
1232 2323cd  
1233 f521f7  
1234 14eef0  
1235 ead8c4  
1236 07d813  
1237 faef01  
1238 0724fe  
1239 f106e4  
1240 e7da35  
1241 1beef0  
1242 dd3536  
1243 ef3e12  
1244 d5d60f  
1245 08f2e6  
1246 2f00f6  
1247 d82758  
1248 1b1e0e  
1249 0409e4  
1250 31250f  
1251 0d11ed  
1252 e22560  
1253 ebdffe  
1254 29f6f8  
1255 073dd6  
1256 e319de  
1257 4bef15  
1258 2d06e5  
1259 f3dee7  
1260 11130e  
1261 1cfe1f  
1262 0bf9fe  
1263 fb010b  
1264 de2000  
1265 dae407  
1266 d90b36  
1267 27d6ef  
1268 342025  
1269 0ce9ef  
1270 fc1cee  
1271 e6c911  
1272 ebd208  
1273 ee06f6  
1274 09f6fb

1275 fce5df  
1276 ea1dca  
1277 efe709  
1278 eebff3  
1279 fa0dc0  
1280 00f80f  
1281 f9ede9  
1282 0ff217  
1283 12d3f0  
1284 e6d8de  
1285 06f9e9  
1286 fb0bdd  
1287 1fdc22  
1288 e800d4  
1289 e804db  
1290 e2e42b  
1291 23ed1d  
1292 de2308  
1293 0bcee7  
1294 24e61e  
1295 edc612  
1296 0e14da  
1297 0d17d5  
1298 f726ee  
1299 0e25d0  
1300 171917  
1301 11dbe9  
1302 e1daf3  
1303 e3f1f2  
1304 290afc  
1305 14f808  
1306 2708e3  
1307 faede3  
1308 10e2e9  
1309 0c23e4  
1310 fafbf6  
1311 d4e2f8  
1312 1b071f  
1313 e3282b  
1314 0ad620  
1315 0711d3  
1316 e924e7  
1317 110c09  
1318 01100c  
1319 d3dc1a  
1320 07e919  
1321 18f2db  
1322 0beffc  
1323 2603e3  
1324 e4f8e0  
1325 22eae5  
1326 0ef0d1  
1327 0025f5  
1328 1b34fb  
1329 090309  
1330 e813fa  
1331 1e3700  
1332 06e334

1333 02d0e8  
1334 eb0e1e  
1335 e02203  
1336 103132  
1337 370a28  
1338 47fe19  
1339 45fb08  
1340 1ee4dd  
1341 dfd7eb  
1342 2a1425  
1343 1fe408  
1344 09dd11  
1345 da2c1d  
1346 ca2e2a  
1347 ebe925  
1348 2cf71f  
1349 2014ef  
1350 ed06e1  
1351 1ecbf5  
1352 de17eb  
1353 0431f8  
1354 2fc801  
1355 04f1cf  
1356 1717c2  
1357 0f292b  
1358 d8d3c4  
1359 2bddbe  
1360 00020a  
1361 031526  
1362 07f511  
1363 0529fc  
1364 fad8f5  
1365 0a07e5  
1366 f724dc  
1367 1121d2  
1368 9cdee4  
1369 dd00db  
1370 f501fa  
1371 2936ee  
1372 d1c514  
1373 21e5ee  
1374 fbe909  
1375 061a1f  
1376 fb201f  
1377 20d722  
1378 4fd700  
1379 2054fe  
1380 d120e0  
1381 d5da2c  
1382 3116f3  
1383 e0e00f  
1384 1311e6  
1385 bb1326  
1386 1ffc01  
1387 2717ef  
1388 093a18  
1389 21df20  
1390 e0efda

1391 2c0e1b  
1392 2015c0  
1393 c5f8ff  
1394 2714dc  
1395 0327e9  
1396 f503f7  
1397 191eb3  
1398 d1d6da  
1399 f7020a  
1400 070010  
1401 f4e606  
1402 07c4f6  
1403 fc24d0  
1404 28e2e2  
1405 bf00f5  
1406 dbf220  
1407 d2d71b  
1408 1310e7  
1409 fd25de  
1410 ff0527  
1411 0fe4fa  
1412 e2c20e  
1413 e41ad7  
1414 f71ee7  
1415 e809d9  
1416 02c5fe  
1417 fc14e2  
1418 f917f4  
1419 b7fcec  
1420 f61cc0  
1421 1808de  
1422 ed00d5  
1423 f2e0e3  
1424 0127f0  
1425 1ac7dd  
1426 ee13e7  
1427 f203f7  
1428 e61311  
1429 fbe9f3  
1430 f4e514  
1431 ebfde9  
1432 12131f  
1433 e029db  
1434 06ebff  
1435 e4dada  
1436 cbeccd  
1437 0bd21a  
1438 d5d804  
1439 f9e0ca  
1440 0cd7d5  
1441 122417  
1442 ea0722  
1443 d8082c  
1444 051637  
1445 f50ddf  
1446 280639  
1447 2ddb0e  
1448 262ae9

1449 16e417  
1450 2e27d7  
1451 f011e8  
1452 1b0b46  
1453 f205f5  
1454 2808fb  
1455 d7f9f5  
1456 2b14fa  
1457 252121  
1458 19d915  
1459 3a17fc  
1460 3215e8  
1461 1bc41f  
1462 dc254c  
1463 dcd0ea  
1464 02fb0b  
1465 000436  
1466 dd32f4  
1467 0f0f1b  
1468 ca0b14  
1469 e2ebe4  
1470 decb11  
1471 1ed8ec  
1472 e1ee20  
1473 140b04  
1474 01fdd5  
1475 f7d238  
1476 f01210  
1477 dad1f0  
1478 e1f3f2  
1479 1133aa  
1480 fd0c06  
1481 c1df00  
1482 f60f0a  
1483 1218dd  
1484 ed1317  
1485 d2d70f  
1486 2705f4  
1487 d3d600  
1488 07f414  
1489 0edae2  
1490 eae7f1  
1491 27f6f9  
1492 2602e4  
1493 f7eef6  
1494 d7e3e4  
1495 12d8e1  
1496 edf7e5  
1497 cf1518  
1498 e8eff1  
1499 e723e2  
1500 d50828  
1501 15f4e6  
1502 f02ae5  
1503 de000e  
1504 27e6fe  
1505 db14ae  
1506 04e0f7

1507 d4f424  
1508 1d0ec9  
1509 2a0001  
1510 e2dd11  
1511 d61626  
1512 f4232a  
1513 fdd4d8  
1514 d6c7de  
1515 f8e208  
1516 08221e  
1517 2adb00  
1518 f2db02  
1519 fd0215  
1520 bc07b7  
1521 0f23fd  
1522 253201  
1523 2c00f9  
1524 20ec19  
1525 17cced  
1526 0102bc  
1527 2a0120  
1528 e5eacf  
1529 edde1e  
1530 23fded  
1531 d70107  
1532 1a09b8  
1533 e6dafd  
1534 0518fb  
1535 16f2e7  
1536 09faf2  
1537 16d725  
1538 be0ee9  
1539 fae92a  
1540 e820e5  
1541 ef2ae8  
1542 eb0ed4  
1543 040231  
1544 d8f81e  
1545 cb0cd6  
1546 e312e0  
1547 2b1c27  
1548 41e159  
1549 e40233  
1550 fb340f  
1551 f72625  
1552 01ccea  
1553 29e70d  
1554 02ea4c  
1555 1705e9  
1556 e1d5ed  
1557 e52e09  
1558 1ec614  
1559 dd25ed  
1560 f4f7e8  
1561 fb3ad4  
1562 142e30  
1563 26f901  
1564 d6fb05

1565 330dd2  
1566 ff1233  
1567 0907eb  
1568 fe1800  
1569 f9edc0  
1570 2af600  
1571 fc34ed  
1572 e0fb20  
1573 3c0028  
1574 e62ef4  
1575 d91dea  
1576 1f2b19  
1577 f10503  
1578 12f51b  
1579 3c2cdb  
1580 070a09  
1581 1afd23  
1582 e2f003  
1583 fe423d  
1584 0bfafc  
1585 2417c2  
1586 04f4ef  
1587 ddfd1c  
1588 05e847  
1589 1cfd22  
1590 e92106  
1591 e4f9e9  
1592 f22d3f  
1593 f9f2f6  
1594 e4e8de  
1595 19ec00  
1596 22e0fb  
1597 e4f1e7  
1598 f929d6  
1599 e824fd  
1600 05001f  
1601 0a051a  
1602 114efa  
1603 271134  
1604 e312f8  
1605 160ce3  
1606 ee2cd9  
1607 ff0d02  
1608 d7fdf3  
1609 22dbf9  
1610 fb15d9  
1611 e746db  
1612 f30e4f  
1613 db3600  
1614 eac601  
1615 df1500  
1616 cf3bb3  
1617 2aec05  
1618 2ed1f7  
1619 e6eeee  
1620 00feea  
1621 e10602  
1622 12f2cd

1623 14ef1c  
1624 eb1ee7  
1625 1344dd  
1626 22ec3a  
1627 020f08  
1628 1aee30  
1629 fff000  
1630 02f40e  
1631 ecb8db  
1632 e4f01c  
1633 33f9da  
1634 db1844  
1635 f6dd08  
1636 3b14f2  
1637 1907ef  
1638 f40728  
1639 11f5ea  
1640 f3d61a  
1641 0ff406  
1642 dd17e5  
1643 221822  
1644 1ce6f9  
1645 03ebdc  
1646 e0e700  
1647 0800d9  
1648 ce1832  
1649 08e8d8  
1650 ea08f8  
1651 1fecf6  
1652 c311e8  
1653 26ee0b  
1654 fd3400  
1655 ec24dd  
1656 df040b  
1657 1516e1  
1658 ddfd09  
1659 cb19f7  
1660 c12bd9  
1661 23111d  
1662 3be421  
1663 f82804  
1664 faee20  
1665 2ef004  
1666 2925d3  
1667 fdf2e5  
1668 1d0fcd  
1669 d6f535  
1670 e810d8  
1671 d82e2a  
1672 12e70e  
1673 ebec19  
1674 2f0011  
1675 152fc0  
1676 2211cc  
1677 0030e3  
1678 f826fd  
1679 f7d1f2  
1680 f01ae4

1681 ddbee4  
1682 ed1af6  
1683 00010b  
1684 22ee08  
1685 d2d423  
1686 1d13f0  
1687 b93526  
1688 e02b21  
1689 e9e4f7  
1690 f70d13  
1691 ff392a  
1692 ecf44f  
1693 b72627  
1694 05f9ef  
1695 bdf836  
1696 dee6ed  
1697 30ec26  
1698 0602eb  
1699 ec01f4  
1700 0b00e0  
1701 c42f0c  
1702 e9e31b  
1703 16e414  
1704 01e6e4  
1705 0d3c18  
1706 153d1e  
1707 ecf41a  
1708 e42f4e  
1709 09212f  
1710 db1727  
1711 14ecd7  
1712 2414d1  
1713 e8e500  
1714 fb31e2  
1715 1a01d4  
1716 d2fc0c  
1717 d4fdf9  
1718 0e3eff  
1719 d7d10e  
1720 abce23  
1721 1ffb17  
1722 113814  
1723 20eaf0  
1724 ece423  
1725 300eea  
1726 fe07cf  
1727 20f2dc  
1728 23fada  
1729 0fff38  
1730 3805f2  
1731 0920f3  
1732 fc0fdc  
1733 311119  
1734 040d0c  
1735 fb1826  
1736 0352fd  
1737 34fb14  
1738 011ee2

1739 f1152d  
1740 28010b  
1741 19dbf9  
1742 cef21d  
1743 fa17b9  
1744 29e5f0  
1745 132f2b  
1746 14ebf6  
1747 051ffe  
1748 d82d2f  
1749 e7ddeb  
1750 01d8ed  
1751 17261a  
1752 e2e5d8  
1753 00eed  
1754 2ce7ec  
1755 210b39  
1756 0004fe  
1757 04f30c  
1758 d4ffe8  
1759 22fb26  
1760 17f4fe  
1761 f31803  
1762 f21b31  
1763 3646e2  
1764 ede5f2  
1765 f009f8  
1766 fa3cd7  
1767 f62efd  
1768 eaedfd  
1769 0613ed  
1770 16fde0  
1771 ee2af9  
1772 000418  
1773 1632e0  
1774 071218  
1775 01351c  
1776 ea1043  
1777 fd2e11  
1778 1d27e5  
1779 04200d  
1780 0f32ea  
1781 f5c627  
1782 f5ec20  
1783 2e0408  
1784 fbec17  
1785 23fafb  
1786 ec1d22  
1787 44ffd7  
1788 18e5e9  
1789 e2f52c  
1790 34ed0c  
1791 e6fd00  
1792 0213f5  
1793 e91308  
1794 f92dd5  
1795 1adcc4  
1796 cfd8cd

1797 d92fe1  
1798 e9d810  
1799 08fcfd  
1800 fbe2e5  
1801 f50b14  
1802 fid53a  
1803 ef2312  
1804 040944  
1805 26fecf  
1806 1729cd  
1807 2329fa  
1808 da46e8  
1809 00f917  
1810 3ff812  
1811 271915  
1812 1ac5ea  
1813 3edfe7  
1814 163709  
1815 1b47d5  
1816 060f50  
1817 202626  
1818 1207d8  
1819 340e1c  
1820 f718d4  
1821 03a924  
1822 0df9fb  
1823 1a0d08  
1824 fe1c00  
1825 bafc23  
1826 00163e  
1827 360113  
1828 382115  
1829 f90808  
1830 00222f  
1831 2117e8  
1832 fe0fd3  
1833 17fcdf  
1834 f631ea  
1835 f7dcec  
1836 180bc4  
1837 fc0e1f  
1838 cddaea  
1839 fiebe0  
1840 f918f3  
1841 dd09f2  
1842 f34dd2  
1843 040c06  
1844 ff13fe  
1845 08d7e3  
1846 0401c0  
1847 1426d4  
1848 32efd0  
1849 e72cef  
1850 0bfc01  
1851 cf2a1c  
1852 ea40bc  
1853 0f30e0  
1854 0a1111

1855 04021e  
1856 113aba  
1857 20e20c  
1858 0dfad2  
1859 2700fe  
1860 d3e724  
1861 fe34f9  
1862 01f918  
1863 ea2008  
1864 c92711  
1865 ed0aff  
1866 daec16  
1867 0206ea  
1868 f5251b  
1869 f000f4  
1870 2726e9  
1871 fefbd4  
1872 0cee09  
1873 ee1e2f  
1874 ead001  
1875 0b0032  
1876 2d06fb  
1877 15fbf4  
1878 351f2b  
1879 ce1204  
1880 ece5bf  
1881 43281b  
1882 063912  
1883 1421d4  
1884 ed0fe2  
1885 0ccce5  
1886 f00e36  
1887 2a21d0  
1888 19e4f1  
1889 e7d3dd  
1890 fc01e0  
1891 15d515  
1892 b71c36  
1893 e9060c  
1894 e0d5e9  
1895 0c18ee  
1896 162ece  
1897 c2d912  
1898 b72019  
1899 10c32c  
1900 f433f3  
1901 f9171f  
1902 f1e9de  
1903 19411f  
1904 e2db17  
1905 e70019  
1906 0812fd  
1907 3039e3  
1908 ed9b2b  
1909 db1a46  
1910 ff3f0b  
1911 f7fae5  
1912 dbb823

1913 25dd12  
1914 1b03e1  
1915 00fa1f  
1916 fedd41  
1917 ee152a  
1918 dbebe8  
1919 eb35f7  
1920 2000f2  
1921 01cd19  
1922 ff0ae3  
1923 db14f7  
1924 11ed65  
1925 fdea26  
1926 09c8fd  
1927 183c1e  
1928 132d38  
1929 df1e08  
1930 d4f402  
1931 fb30f5  
1932 3be83c  
1933 cb2df0  
1934 09d233  
1935 da152d  
1936 281b02  
1937 673d0c  
1938 0fc1d9  
1939 0a0deb  
1940 34e9e7  
1941 f31c30  
1942 19dc1e  
1943 21de3f  
1944 e6f4f8  
1945 c91b1d  
1946 004f1a  
1947 2fc808  
1948 004f4a  
1949 15dbed  
1950 faaceb  
1951 22f6d6  
1952 1248cc  
1953 03250d  
1954 3c3f04  
1955 ebe00c  
1956 042b1d  
1957 db07db  
1958 d64720  
1959 20e7e1  
1960 af2d55  
1961 d7dd27  
1962 1d1fe1  
1963 eb1600  
1964 f6ece4  
1965 e4ddd6  
1966 d811d7  
1967 eaf2d9  
1968 0e07ef  
1969 edece0  
1970 e50fef

1971 edf80c  
1972 e9f525  
1973 de19d9  
1974 10dbf2  
1975 e7fa01  
1976 d81700  
1977 16da26  
1978 16f6db  
1979 22f3d3  
1980 0dfbea  
1981 de23d5  
1982 f2f1dd  
1983 1a2217  
1984 dadbe5  
1985 d11efc  
1986 d6d6f6  
1987 14000a  
1988 d8e010  
1989 1405ea  
1990 0de1d9  
1991 080005  
1992 e31df3  
1993 1203d7  
1994 130af5  
1995 e10df7  
1996 18ea04  
1997 1fedd8  
1998 dd0ce6  
1999 22e0e1  
2000 181ee4  
2001 fbe0e9  
2002 05ef1a  
2003 dde3da  
2004 f9ded9  
2005 03d207  
2006 001be6  
2007 1604ff  
2008 23eddc  
2009 25d6fa  
2010 fce816  
2011 01e5ea  
2012 d1ebf9  
2013 f8f011  
2014 07d203  
2015 e80bf0  
2016 21dbe1  
2017 e60b19  
2018 2cd905  
2019 efed00  
2020 22e7e7  
2021 e402f5  
2022 2208fa  
2023 0a061a  
2024 f80223  
2025 eb00dd  
2026 1422ef  
2027 23ec06  
2028 0aebfe

2029 2712e9  
2030 def6f1  
2031 221f18  
2032 07ff08  
2033 e62403  
2034 0b0810  
2035 fa05ed  
2036 e92514  
2037 05f6e9  
2038 030d26  
2039 01eae5  
2040 202617  
2041 d1301c  
2042 d4fa11  
2043 b8f719  
2044 d4e142  
2045 fc2a45  
2046 27e8eb  
2047 ee30e9  
2048 22d2d1  
2049 11e927  
2050 25382b  
2051 1b0ae3  
2052 4cdea3  
2053 e138e7  
2054 17370a  
2055 fce222  
2056 04ea1b  
2057 1f34ec  
2058 f0f4d7  
2059 e6f3f1  
2060 1de71f  
2061 0c37f3  
2062 ee121b  
2063 22ddd7  
2064 f5fffd  
2065 1af04e  
2066 fa1a1f  
2067 f30edb  
2068 140bbd  
2069 e1e2f1  
2070 Odd1e3  
2071 2b2827  
2072 0f0ddc  
2073 f71fee  
2074 f20b00  
2075 f1224e  
2076 2ee51c  
2077 022422  
2078 311c46  
2079 341727  
2080 ea05f8  
2081 f141f0  
2082 2831f5  
2083 361003  
2084 ff1bd8  
2085 23470c  
2086 2318de

2087 fd2dee  
2088 f1e1af  
2089 0a1b05  
2090 0a2ade  
2091 fbe901  
2092 d1269e  
2093 f5220a  
2094 ed3ded  
2095 41f543  
2096 c8d9d1  
2097 d9ef19  
2098 11010e  
2099 060d42  
2100 1d120c  
2101 13fde2  
2102 d41cd5  
2103 0b2be8  
2104 f21500  
2105 171216  
2106 2efbff  
2107 301fea  
2108 21ced2  
2109 e6293a  
2110 421500  
2111 0d0e15  
2112 1f2606  
2113 c41409  
2114 db637  
2115 efde19  
2116 001a2f  
2117 f8e1ed  
2118 fa5524  
2119 f22757  
2120 2714d3  
2121 2eb0df  
2122 23ccec  
2123 0e09e4  
2124 e0ddcc  
2125 1abbf2  
2126 f7ff09  
2127 40ea1c  
2128 cbfdff  
2129 1c04ff  
2130 f1ec14  
2131 e600f2  
2132 d4e722  
2133 1ed811  
2134 110bdf  
2135 ee0fbd  
2136 2a1118  
2137 13dee3  
2138 fb1b1a  
2139 00e8ea  
2140 f73de6  
2141 06c23d  
2142 2ff52f  
2143 0feb20  
2144 f1dade

2145 cad61a  
2146 e14b26  
2147 0b2a25  
2148 ed21d7  
2149 17ea1b  
2150 092112  
2151 db2a00  
2152 24e8f6  
2153 2bf330  
2154 2bc72e  
2155 08e604  
2156 1efb10  
2157 f43113  
2158 05f7b1  
2159 1fdd1d  
2160 ff1923  
2161 250cf9  
2162 263603  
2163 0cf923  
2164 fd0406  
2165 efdbd9  
2166 19fff8  
2167 d2d624  
2168 18ea2b  
2169 00f4f7  
2170 f91bfd  
2171 00f8e0  
2172 123220  
2173 37de25  
2174 e0ec00  
2175 fcf5be  
2176 2b24cb  
2177 bcecec  
2178 c20b37  
2179 c11c04  
2180 0b1123  
2181 e61220  
2182 e1e810  
2183 e5e7f7  
2184 f700e2  
2185 1cf5aa  
2186 03f8f1  
2187 db110c  
2188 273f09  
2189 f5150b  
2190 e708ff  
2191 272826  
2192 23d22f  
2193 122e0d  
2194 cf011c  
2195 152e37  
2196 1cf2f6  
2197 00cf16  
2198 e9f1f1  
2199 323115  
2200 1c0008  
2201 31132f  
2202 d4c000

2203 c62cf2  
2204 25d238  
2205 e7e609  
2206 04fde2  
2207 130732  
2208 262d29  
2209 12e74d  
2210 0716f6  
2211 dfddd7  
2212 df1dee  
2213 e60efb  
2214 f129d4  
2215 13d5db  
2216 ee1c4a  
2217 e5130f  
2218 cbfe2a  
2219 07e5fe  
2220 e405e5  
2221 19ca0a  
2222 f4020e  
2223 d3e4f9  
2224 d32423  
2225 1aca1f  
2226 f249de  
2227 151517  
2228 f944e7  
2229 1a08d9  
2230 000f36  
2231 251b03  
2232 e92331  
2233 3aeed8  
2234 2e131d  
2235 df04ed  
2236 f7e9e1  
2237 da2a31  
2238 20d6e1  
2239 00f1db  
2240 f71f2b  
2241 f0ab29  
2242 faed07  
2243 19e617  
2244 ff1dc5  
2245 030a3a  
2246 ddee1d  
2247 ea2326  
2248 e72062  
2249 0aefe2  
2250 dacee6  
2251 02e120  
2252 172b12  
2253 d1d220  
2254 0620f9  
2255 d0dac0  
2256 d8df39  
2257 0d1824  
2258 580d33  
2259 0b00de  
2260 0af830

2261 d7e456  
2262 fcff12  
2263 25a8e1  
2264 e4e0ee  
2265 371e00  
2266 e6f6d3  
2267 c10021  
2268 1a27dc  
2269 08210c  
2270 c6152d  
2271 f611ff  
2272 ff2a01  
2273 2be621  
2274 ec0022  
2275 de020c  
2276 3b12dd  
2277 f6e703  
2278 270bf6  
2279 200f1e  
2280 e1ff05  
2281 113712  
2282 1d2e1e  
2283 0cbe0e  
2284 f6e75c  
2285 293333  
2286 22fbeb  
2287 d2e9ee  
2288 e4e104  
2289 e514c5  
2290 0728ec  
2291 f946e6  
2292 33e41d  
2293 0d3927  
2294 1d062c  
2295 1901f4  
2296 f230f9  
2297 3332e6  
2298 1aca20  
2299 463508  
2300 0ac117  
2301 1cfbdd  
2302 f2e831  
2303 0c11d0  
2304 07e3df  
2305 d308fc  
2306 03f127  
2307 2a08d8  
2308 c10d5f  
2309 e4dbbd  
2310 ded2eb  
2311 d1e0da  
2312 2b04c4  
2313 0affe0  
2314 440ace  
2315 e0032c  
2316 4827fe  
2317 10e3cf  
2318 e8352e

2319 dc35e2  
2320 f51930  
2321 bac41e  
2322 0beb0c  
2323 ccf12b  
2324 05250b  
2325 35faec  
2326 0f00f5  
2327 2e092e  
2328 3204f2  
2329 09f516  
2330 1533d8  
2331 dde6ff  
2332 0034cc  
2333 e4ead8  
2334 e92ce3  
2335 15acf5  
2336 f931f5  
2337 dfjee  
2338 b2cbcc  
2339 12f2ef  
2340 0bf6eb  
2341 f70916  
2342 f7daf3  
2343 01f5d7  
2344 d9f808  
2345 0113cd  
2346 20463a  
2347 ed33e6  
2348 e113f3  
2349 fc1cf6  
2350 dccef9  
2351 e51409  
2352 241205  
2353 23eaf6  
2354 fe0044  
2355 f2db32  
2356 fde043  
2357 123436  
2358 f00329  
2359 f028c2  
2360 2a0509  
2361 e6100a  
2362 0ff207  
2363 d00918  
2364 db0507  
2365 f7e61d  
2366 e0e30e  
2367 0cc4e0  
2368 e8dcf5  
2369 def414  
2370 0af5d5  
2371 d70ce2  
2372 0e0afd  
2373 effe24  
2374 27da09  
2375 14d8ec  
2376 2bd500

2377	01f6e9
2378	251cd7
2379	0208f3
2380	f6fff5
2381	07e2cf
2382	d90af5
2383	fef30c
2384	10152a
2385	dbf2cf
2386	de1d05
2387	c50dd5
2388	f015db
2389	1bf3dc
2390	00d3f1
2391	efe90a
2392	14c815
2393	e8f31b
2394	06fe17
2395	f319d5
2396	111f15
2397	1101d8
2398	f6dfe3
2399	101603
2400	de1f1d
2401	242ef4
2402	231be3
2403	3be7ed
2404	f325fd
2405	e9e2d0
2406	072501
2407	2438ee
2408	ebcd06
2409	22e8d2
2410	2c1a29
2411	1125db
2412	0700dd
2413	f9e00f
2414	e0df04
2415	f316c2
2416	e1fcd9
2417	ead000
2418	07f428
2419	2bfcf2
2420	18df26
2421	1102f2
2422	330000
2423	28f7fc
2424	0414f5
2425	10120a
2426	eec1ec
2427	f82efd
2428	12e96d
2429	00fe2d
2430	0e16e8
2431	09eb18
2432	d60c32
2433	e91ddc
2434	d6efec

2435 2cf21d  
2436 11f332  
2437 29ee14  
2438 00ef21  
2439 f02801  
2440 f813f0  
2441 e7de01  
2442 0be027  
2443 f20c07  
2444 2efbdb  
2445 1944fc  
2446 fa19df  
2447 0bdef2  
2448 08d81a  
2449 2718f8  
2450 f60025  
2451 0aceda  
2452 1311f2  
2453 fa1c0f  
2454 fb16f5  
2455 16f7ea  
2456 cec42e  
2457 0b1c00  
2458 2415e5  
2459 160de7  
2460 d20035  
2461 e61dd9  
2462 23d7f5  
2463 300a0f  
2464 1e0ffc  
2465 3b030a  
2466 2b0408  
2467 d61120  
2468 2f139e  
2469 e20cea  
2470 df01de  
2471 1c391f  
2472 f903fd  
2473 e52e08  
2474 07002b  
2475 e9f0ec  
2476 28ef27  
2477 07121e  
2478 18170e  
2479 2ace28  
2480 bdf6f3  
2481 f3112d  
2482 03f1f4  
2483 ca2d2e  
2484 ef05d6  
2485 171f33  
2486 35f514  
2487 ddf121  
2488 ef27fc  
2489 060e30  
2490 f40506  
2491 03f222  
2492 0c00e4

2493 0719fa  
2494 0a04c3  
2495 0c002a  
2496 0ddcf8  
2497 f537e7  
2498 f52c00  
2499 badfe8  
2500 d9f03e  
2501 261e0a  
2502 ddfe31  
2503 06e1d4  
2504 000421  
2505 32c90a  
2506 e40e10  
2507 dcc4d1  
2508 0f00c4  
2509 10e41a  
2510 280c1f  
2511 d12cff  
2512 f71f3a  
2513 23ecee  
2514 e700fd  
2515 142e2c  
2516 062900  
2517 f6403c  
2518 1ee914  
2519 33e314  
2520 181be4  
2521 1bfee1  
2522 24e61d  
2523 d2f3dd  
2524 f21715  
2525 0421f2  
2526 e3d628  
2527 1bc508  
2528 cbebe5  
2529 08231e  
2530 efc000  
2531 d71cea  
2532 04d314  
2533 e6eaf3  
2534 ddfb08  
2535 f10bf7  
2536 2a06d7  
2537 ebc9ff  
2538 24de21  
2539 00dfec  
2540 12d6fe  
2541 e2f224  
2542 1edcd1  
2543 12d4ee  
2544 2a0008  
2545 e10adb  
2546 d6e21a  
2547 0016ed  
2548 290a14  
2549 d511d4  
2550 d501f2

2551 fa11ff  
2552 26d4ce  
2553 c7072d  
2554 cad8da  
2555 f9fd07  
2556 cbfb00  
2557 19dfffd  
2558 f3ef12  
2559 f503d3  
2560 e0fed6  
2561 232a23  
2562 13060e  
2563 28f2e3  
2564 0e0eda  
2565 fbf2ee  
2566 1d18f6  
2567 01ffd0  
2568 2de0a9  
2569 131a30  
2570 25271d  
2571 0d0207  
2572 c8e497  
2573 052522  
2574 e3fd2f  
2575 222b14  
2576 03d70d  
2577 0c1f23  
2578 0600fa  
2579 d9d21c  
2580 180acc  
2581 09ec12  
2582 24def2  
2583 203521  
2584 1e1001  
2585 0cdbc7  
2586 1efcf5  
2587 2a2a22  
2588 0ab7bc  
2589 eeda0b  
2590 283ef2  
2591 e9360d  
2592 29e32a  
2593 c6e612  
2594 fc191e  
2595 1705f6  
2596 d817e6  
2597 022e11  
2598 0b3704  
2599 0eff1a  
2600 30deb4  
2601 05cc03  
2602 05f3c9  
2603 d5da00  
2604 e01000  
2605 2a02f3  
2606 f005d1  
2607 f12c3a  
2608 ec1929

2609 132834  
2610 e8d0f7  
2611 d8d9ee  
2612 0eca06  
2613 1403f8  
2614 f8f228  
2615 25b8f0  
2616 070e26  
2617 fb15f5  
2618 fe10d7  
2619 351ae9  
2620 0608f1  
2621 ebdefe  
2622 27dfffa  
2623 fd01f4  
2624 28ed11  
2625 c70500  
2626 004b04  
2627 102af5  
2628 31f217  
2629 e40d09  
2630 eb0f26  
2631 eed60f  
2632 e4df05  
2633 faff36  
2634 ec123b  
2635 dadbf4  
2636 e327ff  
2637 1f0e22  
2638 ebfae9  
2639 d4f2da  
2640 110c26  
2641 c3ffdd  
2642 0914e7  
2643 c10005  
2644 27ff2e  
2645 2c482e  
2646 f0fee7  
2647 d915c4  
2648 f417d0  
2649 02ed0e  
2650 2d3407  
2651 db2cde  
2652 f4038b  
2653 c30000  
2654 262a0b  
2655 32001e  
2656 001ce3  
2657 1cebf3  
2658 15d0be  
2659 33e105  
2660 260ab7  
2661 dd0718  
2662 fb01e7  
2663 0dd2b5  
2664 ece81c  
2665 ed7f15  
2666 0be5e5

2667 2bc91b  
2668 ddd90f  
2669 1415c6  
2670 fa11c4  
2671 dcf805  
2672 220930  
2673 d42313  
2674 fc2b12  
2675 fb0e3a  
2676 0e1129  
2677 23e6dc  
2678 d9f609  
2679 aec6c8  
2680 d00322  
2681 c1d40a  
2682 1b15fc  
2683 d1b914  
2684 d61d2c  
2685 e02610  
2686 0ccb7  
2687 ac1e13  
2688 030ee7  
2689 22e908  
2690 f1f7ee  
2691 ebdded  
2692 ee2ec8  
2693 a3e32e  
2694 0f0d2f  
2695 2bcd1c  
2696 0301f1  
2697 08e9da  
2698 ebdccd  
2699 11220b  
2700 d6bef0  
2701 ff30ef  
2702 ce2402  
2703 26db2a  
2704 d6dd19  
2705 1f3c1e  
2706 ed00f4  
2707 ea1815  
2708 f22332  
2709 efff0d  
2710 19e0ed  
2711 0b02e8  
2712 fa2c22  
2713 0122ff  
2714 261948  
2715 eb231b  
2716 ea000d  
2717 fbc82c  
2718 293522  
2719 f3e8da  
2720 fcf16  
2721 d105da  
2722 ece01b  
2723 e7d422  
2724 ec271c

2725 2922e5  
2726 150d28  
2727 272804  
2728 dc04f0  
2729 fe1239  
2730 e3e4e5  
2731 19d6f7  
2732 0fd6ea  
2733 07dae2  
2734 051406  
2735 36f0f7  
2736 140900  
2737 1cfa04  
2738 e9240d  
2739 e71f46  
2740 d804ef  
2741 db00f5  
2742 3729f4  
2743 2f0009  
2744 0cd7f6  
2745 e53916  
2746 ebf30c  
2747 252eda  
2748 f1ced0  
2749 03f016  
2750 002632  
2751 122fff  
2752 14f6dc  
2753 f6f23a  
2754 fie117  
2755 011cde  
2756 23f1f7  
2757 00ec07  
2758 e01b31  
2759 16dc3e  
2760 01dde9  
2761 ffe3e1  
2762 2dd014  
2763 34fb19  
2764 eb1ff9  
2765 be15f5  
2766 f41041  
2767 f20cf9  
2768 1efa2a  
2769 09e6d0  
2770 d31b1c  
2771 0012e3  
2772 e30740  
2773 d6f602  
2774 062eed  
2775 fb1b3d  
2776 e30fed  
2777 072b27  
2778 4209ea  
2779 cf3e2e  
2780 211af4  
2781 e8e634  
2782 181bf8

2783 001d25  
2784 00e31e  
2785 21393e  
2786 ecf7f0  
2787 02bffb  
2788 24feaa  
2789 0eebff  
2790 dc0df3  
2791 1d2fde  
2792 202e26  
2793 d8ebfa  
2794 1b35f7  
2795 2209af  
2796 d12418  
2797 133826  
2798 461fdc  
2799 15f335  
2800 f109d8  
2801 fff1fd  
2802 0e33ed  
2803 b5dc29  
2804 1e1145  
2805 e23640  
2806 e500c3  
2807 cf25f0  
2808 2ef2e2  
2809 18f9f7  
2810 fle11e  
2811 14281e  
2812 2e00f3  
2813 e0f6fe  
2814 07f00f  
2815 0d26ee  
2816 154538  
2817 3f2bde  
2818 24c6d7  
2819 2af618  
2820 f9e0e6  
2821 fde1e2  
2822 c4021d  
2823 f404c5  
2824 170a30  
2825 ea1b20  
2826 e21934  
2827 0be6f9  
2828 d34921  
2829 e7dba8  
2830 2deadc  
2831 d7d617  
2832 052ed3  
2833 e3dfd3  
2834 443411  
2835 d71b32  
2836 0200e7  
2837 d8e611  
2838 0023f0  
2839 f322aa  
2840 cce6e0

2841 c519e6  
2842 05f1c8  
2843 c2e23c  
2844 e113e9  
2845 002100  
2846 d1c5d5  
2847 d023f6  
2848 e3dcf3  
2849 0cf410  
2850 18d71c  
2851 1ade04  
2852 1a09ad  
2853 bee221  
2854 0acbf2  
2855 02d942  
2856 1ffa12  
2857 320308  
2858 4997c7  
2859 15eb1f  
2860 21c3df  
2861 d61ffb  
2862 eff1fe  
2863 28f8db  
2864 e2f51d  
2865 f9132a  
2866 0a1f1f  
2867 233a19  
2868 fdeae9  
2869 d505cd  
2870 0edf08  
2871 fd212b  
2872 1018ce  
2873 10f3f8  
2874 e8db50  
2875 d6fbe5  
2876 1b0062  
2877 f2acf8  
2878 3f2e0b  
2879 0904da  
2880 20e738  
2881 30e1d8  
2882 16e60b  
2883 dbf2f7  
2884 231411  
2885 1cfded  
2886 060af2  
2887 e6e31b  
2888 e9d207  
2889 1becd0  
2890 091dfc  
2891 1fea28  
2892 d6f3d7  
2893 d6db1f  
2894 ffcce4  
2895 cbef12  
2896 db13e7  
2897 170b1d  
2898 0ce7eb

2899 c9e3d7  
2900 11f7ec  
2901 fcf3da  
2902 e10d08  
2903 17d1d0  
2904 29042b  
2905 eb1cd0  
2906 1f0adf  
2907 ff0d17  
2908 e00b1c  
2909 d12fd7  
2910 dc08eb  
2911 04f2e3  
2912 08edfd  
2913 161d12  
2914 fefdf2  
2915 21e6fe  
2916 04100c  
2917 1406fa  
2918 f9f71f  
2919 fceadf  
2920 f1efdf  
2921 363711  
2922 2e4ef4  
2923 3b110a  
2924 e32b1b  
2925 1f2a18  
2926 14fabe  
2927 ce36dd  
2928 082a0f  
2929 1709cb  
2930 1f090f  
2931 ff35e0  
2932 dbd118  
2933 f1360e  
2934 00f123  
2935 efff27  
2936 f820e1  
2937 4ae916  
2938 33d7d2  
2939 460b04  
2940 ecf2e6  
2941 fb0f32  
2942 241c0a  
2943 24ec3a  
2944 0100fe  
2945 dd6b18  
2946 d50438  
2947 040617  
2948 f0cf76  
2949 18d831  
2950 0df32f  
2951 1fc4d9  
2952 f7d2c7  
2953 e2db22  
2954 f6fec9  
2955 2d20cb  
2956 26ead9

2957 eee6dd  
2958 e03dfb  
2959 0405f7  
2960 d6f9f2  
2961 e0c81d  
2962 e847d5  
2963 f3e6f5  
2964 ecd225  
2965 0615e8  
2966 ed22dc  
2967 e01f1f  
2968 ff28f2  
2969 0d12ef  
2970 ecf627  
2971 17ebe3  
2972 ee3b37  
2973 bc36f5  
2974 d9dede  
2975 d9d736  
2976 e5edf3  
2977 f2ffe0  
2978 e4f01a  
2979 4fd8ed  
2980 1afd1e  
2981 12071d  
2982 341d1b  
2983 322954  
2984 fd09f4  
2985 cef1e7  
2986 173d30  
2987 fde410  
2988 09fc4a  
2989 fd1110  
2990 31e72d  
2991 d7ef12  
2992 0b1ef9  
2993 ddf238  
2994 ce41e6  
2995 281308  
2996 360617  
2997 fadfe6  
2998 ee03e5  
2999 0720d5  
3000 0b0b57  
3001 cb13e2  
3002 d6e1f7  
3003 ded600  
3004 e4d8d9  
3005 d3cd16  
3006 00d9f2  
3007 00ced2  
3008 f0dadb  
3009 d8e9d1  
3010 e9df26  
3011 20fd06  
3012 d801e3  
3013 fe2029  
3014 101c06

3015 cf07e8  
3016 e21f0a  
3017 cb0ff6  
3018 e9efe6  
3019 0ce518  
3020 e90c01  
3021 f7ddd9  
3022 e42102  
3023 16cce1  
3024 23e81c  
3025 0015e7  
3026 ea1cf3  
3027 ee04f8  
3028 d7e921  
3029 e5eafe  
3030 e7cbe5  
3031 d6031e  
3032 120f27  
3033 f116fd  
3034 051b19  
3035 2efbe9  
3036 1af2ef  
3037 ff03d0  
3038 14f900  
3039 fae5cc  
3040 d4dafe  
3041 112ef1  
3042 d8da36  
3043 2a2de3  
3044 04d4e7  
3045 04ee0e  
3046 32e54a  
3047 2200e8  
3048 3bdbed  
3049 14fd07  
3050 1f31da  
3051 0afed5  
3052 3036fd  
3053 060cf7  
3054 0b1e18  
3055 e4000e  
3056 f5dd38  
3057 af221e  
3058 e63933  
3059 16dcee  
3060 f93d3e  
3061 37dc18  
3062 c9e3f9  
3063 f4feda  
3064 fdd7f0  
3065 1fa8fa  
3066 11e0f9  
3067 1910d8  
3068 121add  
3069 eff412  
3070 0ae814  
3071 22fa00  
3072 d83930

3073 e120e8  
3074 26fec8  
3075 fcea0f  
3076 f7ecd9  
3077 fc152c  
3078 ebfe19  
3079 2b3939  
3080 e22fdc  
3081 f9ede0  
3082 2721f4  
3083 2d1317  
3084 fcd2dd  
3085 f1ed24  
3086 1921f6  
3087 1bfc06  
3088 d8dd06  
3089 1afefd  
3090 0a19f3  
3091 2a3a05  
3092 10d212  
3093 f11601  
3094 dde9fe  
3095 40c827  
3096 0008f0  
3097 3a22f7  
3098 55f5db  
3099 380036  
3100 ef05fe  
3101 de19f6  
3102 572be1  
3103 45e5fc  
3104 ee2312  
3105 ff3929  
3106 d7f328  
3107 2e00da  
3108 051af2  
3109 1b0507  
3110 0c2015  
3111 f2d921  
3112 1630b8  
3113 cbf623  
3114 fbc8f8  
3115 dfd3e6  
3116 25faff  
3117 22032c  
3118 1205e7  
3119 fdca48  
3120 1e190e  
3121 eb05d2  
3122 f301cf  
3123 160616  
3124 f40fdd  
3125 05ed07  
3126 12f34e  
3127 0eea0d  
3128 ce2106  
3129 15edb8  
3130 d4b6f1

3131 220d0b  
3132 d5134e  
3133 39e914  
3134 2017e4  
3135 c627fc  
3136 112ff5  
3137 200600  
3138 11142c  
3139 ce57ec  
3140 2c201d  
3141 0fcff7  
3142 0cda34  
3143 e71715  
3144 eb002b  
3145 1af7dc  
3146 29eff6  
3147 e139df  
3148 e4dbff  
3149 f1f2f7  
3150 afeb30  
3151 f50b2b  
3152 080c07  
3153 42ec22  
3154 ffedd1  
3155 f60cd0  
3156 3edd2e  
3157 00ebe7  
3158 10ffe1  
3159 152bf6  
3160 24de2f  
3161 1cef1c  
3162 defcdc  
3163 ea41de  
3164 20e516  
3165 efd8f0  
3166 30e9c0  
3167 db2bd7  
3168 1e1af8  
3169 0b2402  
3170 3fef26  
3171 dfe60d  
3172 cfb8d4  
3173 dde2f9  
3174 262500  
3175 270047  
3176 0b001a  
3177 ed2f10  
3178 02fafa  
3179 15021c  
3180 de35e0  
3181 021dd0  
3182 eadd19  
3183 1a0617  
3184 ef06e3  
3185 25c120  
3186 f24de1  
3187 3e26e2  
3188 3a0144

3189 1906fd  
3190 1a031c  
3191 fff728  
3192 1cd2fb  
3193 0f0310  
3194 c547e9  
3195 d523ae  
3196 c1dd27  
3197 1e26e1  
3198 11ecfc  
3199 150cf4  
3200 ffed0e  
3201 1e00fd  
3202 eed9e6  
3203 1d1ae4  
3204 f8edd5  
3205 e300f3  
3206 0d0ffe  
3207 d8dbd3  
3208 fff0de  
3209 12ecd6  
3210 20e41d  
3211 1ee214  
3212 e7de1d  
3213 d9df0c  
3214 25f6d5  
3215 fceef5  
3216 08ed05  
3217 0c1411  
3218 fd0ad1  
3219 e601e2  
3220 dde5d6  
3221 e400de  
3222 00e30c  
3223 ef1105  
3224 d8e820  
3225 e3e7d8  
3226 f8f7f2  
3227 fa000c  
3228 0301f3  
3229 01d2f0  
3230 0b29e7  
3231 1904fa  
3232 d8e4f5  
3233 eb13f7  
3234 ee2200  
3235 d51a17  
3236 e42b19  
3237 03f013  
3238 f2def5  
3239 e00b29  
3240 210803  
3241 3910dc  
3242 28db39  
3243 17232d  
3244 f40745  
3245 2a2631  
3246 03f2f5

3247 0dd21b  
3248 f23216  
3249 d81119  
3250 00eb1c  
3251 dd2e20  
3252 17cc17  
3253 421725  
3254 f8def8  
3255 2f330f  
3256 fc083b  
3257 2320eb  
3258 2a1d17  
3259 fa160c  
3260 fc04da  
3261 48d701  
3262 08d640  
3263 fbe7a9  
3264 efea28  
3265 f2e534  
3266 eb2a06  
3267 46231c  
3268 0529fc  
3269 d7d0bf  
3270 e5fe21  
3271 210d07  
3272 e32ff6  
3273 01d613  
3274 3323ce  
3275 ecf609  
3276 d5dd0f  
3277 1a0cf8  
3278 e1d511  
3279 fef8eb  
3280 f4113c  
3281 282507  
3282 ccd334  
3283 1e22ff  
3284 14e528  
3285 200a06  
3286 153d0b  
3287 2946ce  
3288 0417b0  
3289 f52c42  
3290 2e13fb  
3291 df1b1b  
3292 30b7d7  
3293 e10df5  
3294 1a2101  
3295 42306d  
3296 27e70f  
3297 e548e4  
3298 f4ecdb  
3299 260bfd  
3300 1f1bd9  
3301 f41d00  
3302 da2429  
3303 1ad634  
3304 1be008

3305 d6c940  
3306 cb0030  
3307 31e50d  
3308 e7dc35  
3309 00f1e9  
3310 0ffefb  
3311 f82601  
3312 13f5ff  
3313 e002e6  
3314 d25a00  
3315 1104ea  
3316 e804d8  
3317 f3fd1d  
3318 ee4334  
3319 ea0fe9  
3320 0edc05  
3321 fb321d  
3322 21d939  
3323 dafa12  
3324 eaf5fc  
3325 cb031b  
3326 1a17f2  
3327 26e90a  
3328 133017  
3329 de0f06  
3330 0930e1  
3331 e2f91a  
3332 2a16f2  
3333 eee4f0  
3334 101c21  
3335 0b26e8  
3336 e6feea  
3337 d7dcdc  
3338 091839  
3339 e4d3e4  
3340 fb3d1f  
3341 e7080c  
3342 08edf1  
3343 fce3e6  
3344 fad200  
3345 f8cfd6  
3346 e6db1c  
3347 0d0e09  
3348 1712f7  
3349 0df40a  
3350 270817  
3351 1b3227  
3352 28f332  
3353 f0dd0c  
3354 000a11  
3355 e90d0e  
3356 c20b16  
3357 eae535  
3358 00fff6  
3359 e91601  
3360 27e4d8  
3361 000000  
3362 0000a5

3363	00003a
3364	000045
3365	000007
3366	000005
3367	0000fe
3368	00007e
3369	000032
3370	00005f
3371	0000ca
3372	000011
3373	000077
3374	000001
3375	0000e2
3376	000042
3377	00002c
3378	0000d4
3379	000019
3380	00000f
3381	0000c9
3382	000051
3383	00007f
3384	0000c7
3385	000067
3386	0000a5
3387	0000e1
3388	000057
3389	00001d
3390	0000af
3391	000049
3392	000047
3393	0000ef
3394	0000f7
3395	000096
3396	0000af
3397	000059
3398	000098
3399	0000c9
3400	00005b
3401	000039
3402	0000c5
3403	0000c5
3404	00002b
3405	000012
3406	0000cc
3407	0000c8
3408	0000f4
3409	000008
3410	0000ed
3411	0000f3
3412	0000ee
3413	0000e0
3414	000006
3415	000002
3416	000003
3417	00001a
3418	000053
3419	000056
3420	00009e

```
3421 0000ad
3422 000044
3423 000038
3424 0000fa
3425 00001a
3426 00001a
3427 0000c2
3428 0000fd
3429 0000fa
3430 000065
3431 000005
3432 0000b7
3433 000002
3434 00001e
3435 000018
3436 00000b
3437 00001a
3438 000030
3439 000002
3440 000036
3441 000001
3442 000046
3443 00004a
3444 0000d7
```

Listing 25: Packed FC3 weights and biases for three-lane FPGA ROM loading.

```
1 e0229a
2 2b033a
3 a2422a
4 391d2f
5 49c001
6 ee0524
7 05cf48
8 340f9d
9 d21e1b
10 0fc837
11 da17d0
12 18afb4
13 d21616
14 e5383f
15 d7ea00
16 47b893
17 f037ac
18 21beb1
19 e71f21
20 d51800
21 12d92b
22 21b3ed
23 f6070f
24 d5cac9
25 1ee4e9
26 993bc7
27 ce9c39
28 de9b10
29 d2f59b
30 f7ded9
31 424cf6
```

32 e74b1d  
33 e4f00f  
34 f5220a  
35 a7bbfa  
36 dc6519  
37 44f515  
38 f24616  
39 ec3d17  
40 101313  
41 27bfba  
42 bb6329  
43 29a26e  
44 1fe8da  
45 d8f248  
46 1be4ba  
47 e38da8  
48 16d3e2  
49 d3ce2e  
50 36dfd0  
51 0818be  
52 f39506  
53 5947c3  
54 3d97f6  
55 e7e3c6  
56 ad05ba  
57 ca4326  
58 fdf41d  
59 9eaacc  
60 442a1a  
61 b1afe5  
62 26ffc1  
63 31da58  
64 0bc241  
65 101658  
66 19f30f  
67 d6fad3  
68 f7d9f3  
69 090adc  
70 e70d4f  
71 f228c5  
72 3332de  
73 fc1703  
74 f64c13  
75 a6b4f1  
76 f82114  
77 ddf551  
78 f61110  
79 023552  
80 d72ab5  
81 f21add  
82 1a1e33  
83 fcac0e  
84 effcbf  
85 042cc5  
86 b1ccaf  
87 3c290e  
88 30fa1d  
89 e3c4ec

90 ac1ced  
91 0e23fb  
92 368b23  
93 df0fb6  
94 c90de4  
95 fb27f3  
96 fdf9ef  
97 62d6e3  
98 f689e4  
99 281a8c  
100 319e3e  
101 2ac5b9  
102 fdf9f6  
103 ce0cf9  
104 f610ab  
105 08cd17  
106 4140e4  
107 26f399  
108 e1b8a1  
109 b4e300  
110 04f3fe  
111 f82fa7  
112 1c1a16  
113 f6f3c2  
114 d0c53b  
115 fd22c1  
116 3be381  
117 3b223f  
118 1ed209  
119 bf000a  
120 8ceb8e  
121 0ad8ad  
122 f1e9d1  
123 fde501  
124 dce7d6  
125 ed0608  
126 0003aa  
127 f60f02  
128 dd02cf  
129 323338  
130 bb0553  
131 e3eb00  
132 0a4535  
133 00fded  
134 3ead0e  
135 356c40  
136 5b26f1  
137 1307f3  
138 36cb1c  
139 ebecca  
140 c81018  
141 27181d  
142 253cc4  
143 02ca54  
144 c7182b  
145 bf3ffe  
146 13a9bf  
147 bf05a8

148 34fa1c  
149 0328a5  
150 24912a  
151 0d12da  
152 bf1c14  
153 dc0aed  
154 0ff0f8  
155 10ea24  
156 b70023  
157 e91133  
158 53b6f0  
159 1f3eb8  
160 1b3daa  
161 1116fd  
162 3cd1cb  
163 4c01d1  
164 17d6df  
165 29a1f2  
166 2b1104  
167 Oddafb  
168 d4f3fd  
169 15e53c  
170 553df7  
171 e24361  
172 b1cebc  
173 0b313f  
174 f2aa3a  
175 2ce1b3  
176 b31c4a  
177 b2bdef  
178 ebbc4f  
179 ea1a85  
180 e205a8  
181 1335de  
182 d25ad6  
183 d9e12f  
184 c91093  
185 2ac6b7  
186 dfd714  
187 40d742  
188 e417c6  
189 4c0af7  
190 dfd601  
191 b64fe2  
192 dd0005  
193 cfb21f  
194 ef14d2  
195 c39bfb  
196 18bbc5  
197 b713dc  
198 f792f9  
199 3199b5  
200 4525af  
201 fabbc0  
202 24174f  
203 544aae  
204 37ac3f  
205 20f2d0

206 e62304  
207 e34436  
208 c02d0d  
209 00c1f1  
210 1d390f  
211 2813b1  
212 dc1321  
213 fe1bf6  
214 ebe6d1  
215 4baac1  
216 c2991b  
217 d5fc15  
218 080bd2  
219 2f0c1f  
220 30f427  
221 8447f7  
222 3fb3f2  
223 24b84b  
224 cec128  
225 28060c  
226 4b4cdc  
227 f211bd  
228 dbb82c  
229 31251c  
230 2dfcf0  
231 00d9e1  
232 dc08d8  
233 51c52d  
234 ead903  
235 35c512  
236 2bf0d4  
237 dbc5db  
238 05a70d  
239 3fdeb3  
240 d8c5f7  
241 25dcca  
242 b74a15  
243 cbd1a1  
244 0cc8dd  
245 431fb9  
246 b953dd  
247 2ae7af  
248 d5aa44  
249 c4e1e9  
250 2f3abd  
251 2bfadc  
252 f43d1d  
253 39e3e7  
254 d40ce4  
255 fbec28  
256 f7ce2c  
257 d8dd33  
258 1eea1a  
259 d34231  
260 ee1e87  
261 befcae  
262 20c0ac  
263 ea3cfe

264	e7280a
265	b5053c
266	d0c003
267	342526
268	a7a949
269	d9283f
270	b3a214
271	bc0031
272	f8eb45
273	26b3d0
274	eac0fd
275	d78dca
276	9240ba
277	0f1111
278	f53512
279	23efc6
280	143c46
281	000021
282	0000b6
283	000098
284	000095
285	0000d4
286	000081
287	00006a
288	0000b2
289	000053
290	00000a