ES Final Project: BlackjackCounter
Design Document

Lennart Schulze
ls3932

Joseph Han
jh4632

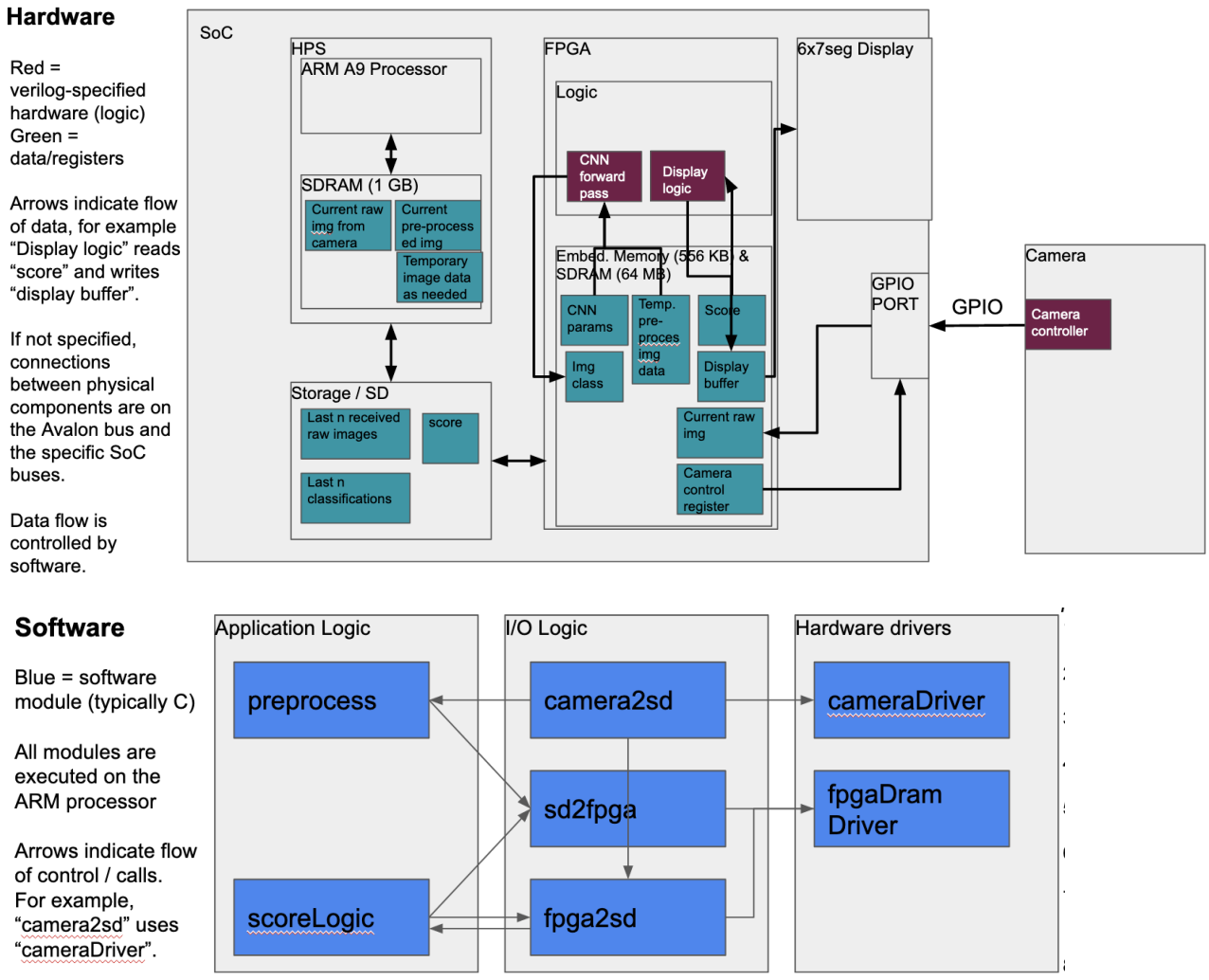
Michael Ozymy
mjo2156

1 Contents

2 Project objective

We propose to use the combined software-hardware nature and peripheral connectivity of FPGAs to develop a computer-vision-based card counting mechanism for the game blackjack. To this end, we connect the FPGA to a camera that observes a fixed limited playing field, in which one player and the dealer place their cards. We then implement a computer vision model, that is a convolutional neural network, on the FPGA chip to recognize the state of the game by detecting the type of cards. This prediction is hardware-accelerated on the FPGA via SystemVerilog. We produce and output these computations in pseudo-realtime to onboard hex displays.

3 Block diagram



Software module description

- fpgaDramDriver - controls FPGA's DRAM and provides interfaces to use in other programs.
- cameraDriver - controls camera and provides interfaces to use it in other programs; for this purpose uses fpga2sd since camera is FPGA peripheral.
- camera2sd - loads data from camera to SD card via FPGA; controls logic how often this happens (regular photo VS video stream, etc.).
- sd2fpga - loads data from SD card into FPGA memory, for subsequent classification by FPGA circuitry and for score display; controls logic of this process; uses fpga driver.
- fpga2sd - loads data from FPGA memory into SD card; uses fpga driver.

- preprocess - manipulates camera data (i.e. crop, downsample, etc.) to produce data that can be classified; sends it to FPGA.
- scoreLogic - computes blackjack card count score based on newest classification result and score from previous classifications; sends score for display to FPGA.

Software flow

1. Data arrives at FPGA.
2. Camera2d stores input on sd card
3. .. by calling cameraDriver
4. .. and using fpga2d logic to transfer file
5. .. which uses fpgaDramDriver.
6. Preprocess prepares input,
7. .. sends it to FPGA for classification
8. using sd2fpga
9. .. which uses fpgaDRAMdriver.
10. (Classification.)
11. After classification, result gets used to compute score on ARM in scoreLogic.
12. .. which requires to have classification result on sd card using fpga2sd. Score gets send to FPGA for display via sd2fpga.

4 Algorithms

The algorithms we plan to use are a convolutional neural network to classify images from the camera are preprocessed. From the classification result, we compute the blackjack card count. Note that only the prediction of new datapoints on the trained CNN model (forward pass) will be performed in an accelerated manner on the FPGA. The training itself (backward pass and gradient descent) of the model happens elsewhere before that.

Image preprocessing

We will apply simple algorithms such as cropping, down-sizing, and filter-based methods to prepare a single image for classification by the CNN on the FPGA.

Convolution

Description:

A discrete 2d-, that is matrix, convolution describes the operation of transforming a given input matrix into an output matrix by sliding a smaller matrix with fixed components, referred to as kernel, over the image and applying a matrix operation between the window of the input and the small matrix, with both have the same size. At each step, an elementwise multiplication is performed and the resulting matrix is reduced to a single value via adding all the values, which is assigned as the value to the component of the output matrix at that position. This is repeated over the entire input matrix by beginning in the top left most window, computing the value for the top left element in the output matrix, and then moving down in an Z shape to the bottom right most window, that is for each row right until the end of the row followed by moving down one row. The size of the output matrix is thus determined by the kernel. Padding around the input matrix can be applied to enforce desired output sizes. The stride determines how fast the window moves, that is by how many columns it moves right or rows down per iteration.

Formal definition:

The element of the output matrix O with indices x, y is defined as:

$$O[x, y] = (I * K)[x, y] = \sum_{-N \leq i \leq N} \sum_{-M \leq j \leq M} I[x - i, y - j]K[i, j] \tag{1}$$

Where I is the input matrix and K is the convolution kernel with shape (N, M), per convention (note however the symmetry of the convolution operation).

Algorithm:

A simplified algorithm disregarding padding and stride looks like:

```
Def convolve(I, K):
    s = size(K)
    O = [][]
    For x in I:
        For y in I[x]:
            O[x,y] = sum_elements(multiply_elementwise(I[x:x+s, y:y+s], K))
    return O
```

Convolutional neural network (forward pass)

Description:

Broadly, a CNN is a special type of an artificial neural network, which is a universal function approximator. As opposed to regular ANNs such as multilayer-perceptrons, three types of layers can usually be found in a CNN: fully-connected layers, where each unit in the current layer is connected to every unit in the next layer via a weighted sum of the current units with trainable weights; convolutional layers, where the units of the current layer are convolved with a trainable kernel matrix; and pooling layers, where the dimensionality of the vector of the current layer is reduced in the next layer, for instance via taking the mean of the pixel values in a given window that slides over the image. The layers are connected via non-linear activation functions, such as Relu. The output is generated by applying the layers and activations in order starting from the input. Depending on the task, the last activation function is chosen.

In our case, the input is a flattened vector corresponding to pixel color values and the output is one of the 52 distinct cards in blackjack. This represents a multiclass classification problem, for which reason the last activation function is chosen to produce a probability distribution over the 52 cards, from which

Formal definition:

A CNN F, acting on the input vector x can be described as:

$$F : X \to Y \tag{2}$$

$$F(x) = (f_n \circ ... \circ f_2 \circ f_1)(x) \tag{3}$$

Where $Y = 1, ..., 52$ the integer representing the class of the card; $X = 0, ..., 255^{(W*H*3)}$, the flattened vector of pixel of dimension H*W*3 one pixel is a RGB value represented as triplet (R,G,B) in which each channel can assume a value in the range $[0, 255]$; and $f_i \in \{$linear, convolution, pool, activation$\}$.

A linear activation is:

$$f(x) = Wx + b \tag{4}$$

A typical activation for intermediate layers is Relu:

$$f(x) = max(0, x) \tag{5}$$

A typical activation for the output layer for multiclass classifier CNNs is softmax:

$$f(x)[i] = \frac{exp(x[i])}{\sum_{j \in [|x|]} exp(x[j])} \tag{6}$$

Algorithm:

A simplified, forward-only CNN with given weights w can be described as

```
Def cnn(x,w):
    Layer_defs = {"linear":..., "cnn":..., "avg_pool":..., "relu":..., "softmax":...}
    Layers = [linear, relu, avg_pool, cnn, avg_pool,
        linear, relu, ..., softmax] # definition of cnn architecture
    y = x
    For i in {0,...,length(layers)}:
```

```
        y = layer_defs[layers[i]](y, w[i])
    return y
```

Compare to [1].

[1] Albawi, S., Mohammed, T. A., Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET) (pp. 1-6). Ieee.

Card counting algorithm

Using the CNN to classify snapshots of the playing field in a fixed interval such as every 10 seconds, we keep record of which cards have been played. We then count the score of the cards, using the "Hi-Lo" card counting algorithm laid out in [2].

[2] https://www.blackjackapprenticeship.com/how-to-count-cards/

5 Resource Budgets

Memory to be used on FPGA

Available embedded memory: 512 kb.
w = width, h = height, c = channels, l=layers, o=output classes

What	size/#bits	total
score	6*4bits	= 3 bytes
Image pixels to classify	400w*400h*3c*8bit	= 270 000 bytes
CNN weights: fully connected layers	Avg. 10w*10h*3c*52o*1l*4bytes	= 62 400 bytes
CNN weights: convolution and pooling	7w*7h*3c*6l*4bytes	= 3528 bytes
	total	*== 335 931 bytes* *== 329kb*
	available	*512kb*
	remaining	*183kb*

Camera-to-SoC connection bandwidth

We connect 8 pins via GPIO, and one byte of RGB data output is sent per pixel clock cycle (pclk) over the 8 pins. The input clock to the camera module has a typical clock rate of 24 Mhz. The pixel clock generated by the camera module has a maximum clock rate of 27.648Mhz.

Data	size/rate	Total size/rate
size of one frame	640x480 pixels x 3 RGB colors/pixel x 1B per RGB	= 0.9216MB
Framerate	Up to 30 frame per second	= 30 fps
Pixel clock frequency	Up to 30 x 640x480 x 3Hz	= Up to 27.648MHz
Bandwidth	1 Byte/t_pclk	= Up to 27.648 MB/s
FPGA image processing framerate	4 frames per second	= 4 fps
FPGA image processing clock frequency	Up to 4 x 640x480 x 3Hz	= Up to 3.6864 Mhz
FPGA default clock frequency	50Mhz	= 50Mhz
		=> feasible

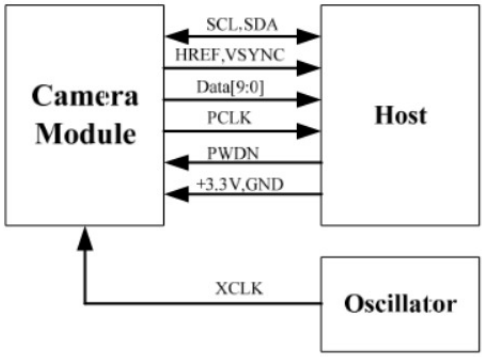
6 Hardware-Software Interfaces

Camera to FPGA 2x20 GPIO Interface

OV7670 Camera Breakout Board

Connection between FPGA (Host, Oscillator) and OV7670

With female flyer wires, connect OV7670 pins one-to-one to GPIO pins on the DE1-SOC board in the order outlined in their respective printouts below.



Pinout of Camera Module:

Pin No.	PIN NAME	TYPE	DESCRIPTION
1	VCC	POWER	3.3v Power supply
2	GND	Ground	Power ground
3	SCL	Input	Two-Wire Serial Interface Clock
4	SDATA	Bi-directional	Two-Wire Serial Interface Data I/O
5	VSYNC	Output	Active High: Frame Valid; indicates active frame
6	HREF	Output	Active High: Line/Data Valid; indicates active pixels
7	PCLK	Output	Pixel Clock output from sensor
8	XCLK	Input	Master Clock into Sensor
9	DOUT9	Output	Pixel Data Output 9 (MSB)
10	DOUT8	Output	Pixel Data Output 8
11	DOUT7	Output	Pixel Data Output 7
12	DOUT6	Output	Pixel Data Output 6
13	DOUT5	Output	Pixel Data Output 5
14	DOUT4	Output	Pixel Data Output 4
15	DOUT3	Output	Pixel Data Output 3
16	DOUT2	Output	Pixel Data Output 2 (LSB)

DE1-SOC 2x20 GPIO Pinout:

Table 3-11 Pin Assignments for Expansion Headers

Signal Name	FPGA Pin No.	Description	I/O Standard
GPIO_0[0]	PIN_AC18	GPIO Connection 0[0]	3.3V
GPIO_0 [1]	PIN_Y17	GPIO Connection 0[1]	3.3V
GPIO_0 [2]	PIN_AD17	GPIO Connection 0[2]	3.3V
GPIO_0 [3]	PIN_Y18	GPIO Connection 0[3]	3.3V
GPIO_0 [4]	PIN_AK16	GPIO Connection 0[4]	3.3V
GPIO_0 [5]	PIN_AK18	GPIO Connection 0[5]	3.3V
GPIO_0 [6]	PIN_AK19	GPIO Connection 0[6]	3.3V
GPIO_0 [7]	PIN_AJ19	GPIO Connection 0[7]	3.3V
GPIO_0 [8]	PIN_AJ17	GPIO Connection 0[8]	3.3V
GPIO_0 [9]	PIN_AJ16	GPIO Connection 0[9]	3.3V
GPIO_0 [10]	PIN_AH18	GPIO Connection 0[10]	3.3V
GPIO_0 [11]	PIN_AH17	GPIO Connection 0[11]	3.3V
GPIO_0 [12]	PIN_AG16	GPIO Connection 0[12]	3.3V
GPIO_0 [13]	PIN_AE16	GPIO Connection 0[13]	3.3V
GPIO_0 [14]	PIN_AF16	GPIO Connection 0[14]	3.3V
GPIO_0 [15]	PIN_AG17	GPIO Connection 0[15]	3.3V

Configuration of camera module

Output format configuration:

Using GPIO0 [3] (PINY18) on the FPGA, select address 12 and set its value to 04 to select RGB output format

Address (Hex)	Register Name	Default (Hex)	R/W	Description
11	CLKRC	80	RW	Internal Clock Bit[7]: Reserved Bit[6]: Use external clock directly (no clock pre-scale available) Bit[5:0]: Internal clock pre-scalar F(internal clock) = F(input clock)/(Bit[5:0]+1) • Range: [0 0000] to [1 1111]
12	COM7	00	RW	Common Control 7 Bit[7]: SCCB Register Reset 0: No change 1: Resets all registers to default values Bit[6]: Reserved Bit[5]: Output format - CIF selection Bit[4]: Output format - QVGA selection Bit[3]: Output format - QCIF selection Bit[2]: Output format - RGB selection (see below) Bit[1]: Color bar 0: Disable 1: Enable Bit[0]: Output format - Raw RGB (see below) YUV: COM7[2]=0, COM7[0]=0 RGB: COM7[2]=1, COM7[0]=0 Bayer RAW: COM7[2]=0, COM7[0]=1 Processed Bayer RAW: COM7[2]=1, COM7[0]=1

pclk Frequency Configuration:

Using GPIO0 [3] (PINY18) on the FPGA, select address 3E and set its value to 14 to select the lowest pclk frequency

3E	COM14	00	RW	Common Control 14 Bit[7:5]: Reserved Bit[4]: DCW and scaling PCLK enable 0: Normal PCLK 1: DCW and scaling PCLK, controlled by register COM14[2:0] and SCALING_PCLK_DIV[3:0] (0x73)) Bit[3]: Manual scaling enable for pre-defined resolution modes such as CIF, QCIF, and QVGA 0: Scaling parameter cannot be adjusted manually 1: Scaling parameter can be adjusted manually Bit[2:0]: PCLK divider (only when COM14[4] = 1) 000: Divided by 1 001: Divided by 2 010: Divided by 4 011: Divided by 8 100: Divided by 16 101~111: Not allowed

Sources [3],[4],[5],[6],[7]

[3] https://www.openhacks.com/uploadsproductos/ov7670_cmos_camera_module_revc_ds.pdf
[4] https://www.openhacks.com/uploadsproductos/ov7670_cmos_camera_module_revc_ds.pdf
[5] https://www.openhacks.com/uploadsproductos/ov7670_cmos_camera_module_revc_ds.pdf
[6] http://www.ee.ic.ac.uk/pcheung/teaching/ee2_digital/de1-soc_user_manual.pdf
[7] http://web.mit.edu/6.111/www/f2016/tools/OV7670_2006.pdf

FPGA to 7-segment display Interface

FPGA memory buffer, total size: 3 byte. Available hex digits: 6.

7-seg 0	7-seg 1	7-seg 2	7-seg 3	7-seg 4	7-seg 5
4bit	4bit	4bit	4bit	4bit	4bit
Instruction to play/wait letter 1	Instruction to play/wait letter 2	Instruction to play/wait letter 3	Score decimal* digit 1	Score decimal* digit 2	Score decimal* digit 3

* for instance via double dabble algorithm.