PLazeR: a planar laser rangefinder

Robert Ying[*], "Derek" Xingzhou He[†], Peiqian Li[‡], Minh Trang Nguyen[§]

May 2015

[*]ry2242@columbia.edu
[†]x.he@columbia.edu
[‡]pl2521@columbia.edu
[§]mnn2108@columbia.edu

Contents

1. Introduction

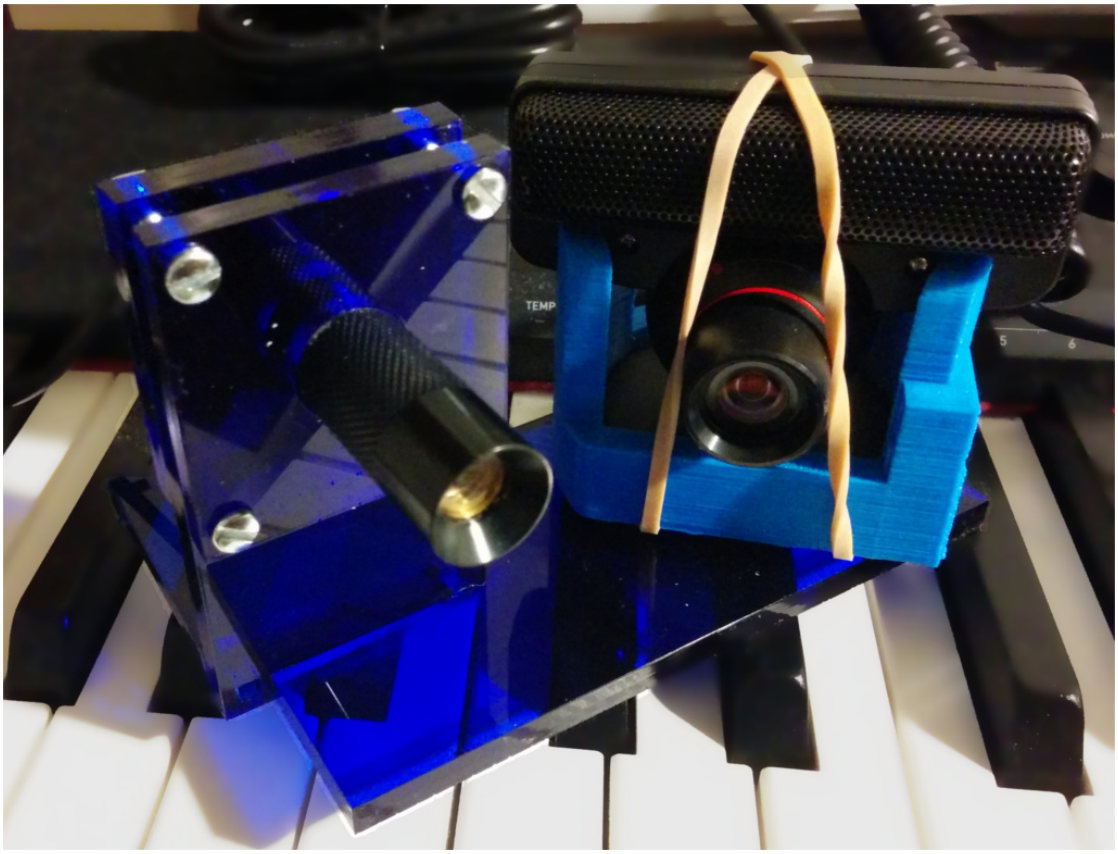

Figure 1.1.: The PLazeR device. On the left, there is a planar laser; on the right, a PS3 Eye camera

PLazeR is a low-cost planar laser rangefinder created by coupling together a PS3 Eye camera and a line laser diode. As these two components are offset from one another, the position of the planar laser in the camera's field of view is related to the distance of the object that the laser is shining upon. This is distinct from a traditional scanning laser rangefinder, which uses time-of-flight characteristics of a rapidly spinning laser module to recover distance.

Time-of-flight (TOF) systems are expensive, due to the extremely precise timing necessary to measure useful distances accurately. As we do not have access to high-speed

analog electronics, we chose not to implement a TOF sensor.

Many systems have taken advantage of this triangulation effect, including devices like the Sharp GP2Y0A21YK0F analog distance sensor and the Neato XV-11 cleaning robot's obstacle detection sensor. However, most of these systems use a physical rotation method to generate the planar scan, which is mechanically unreliable and leads to tearing effects in the data.

In order to successfully reconstruct the distance, the laser's location in the camera image needs to be detected across the entire plane. However, as the data tends to be very noisy, it is important to filter and pre-process the data. In section 3.1, we describe our filtering pipeline and how the FPGA, CPU, laser, and camera interact. Afterwards, in section 3.2, we reconstruct object distance values from the collected data.

2. System Overview

Our system is made up of three main parts: a physical device, which collects the data (Fig. 1.1), the software on the ARM CPU that interacts with the camera and serves as an interconnect, and the image processing algorithm on the FPGA.

2.1. Algorithm

The general algorithm that we use in PLazeR is as follows:

Let the space $\mathbb{Z}_{8+}$ denote the space of 8-bit unsigned integers. Then, an image from the camera can be represented as a vector $I_{rgb} \in \mathbb{Z}_{8+}^{640\times480\times3}$, for a 640 by 480 RGB image. Since the laser is sufficiently powerful that it actually saturates all three channels, we desaturate the image and use its grayscale form $I \in \mathbb{Z}_{8+}^{640\times480}$.

The first step is to try and eliminate noise from the image data. Since we treat each row of the image as a separate distance value to be computed, it suffices to filter within the space of a particular row. In this case, we use a discrete 16-byte-wide Gaussian convolution, with some variation σ^2.

$$C_i = \sum_{i=1}^{8} I[i]G[i] + \sum_{i=8}^{16} I[i]G[16-i] \tag{2.1}$$

where C_i is the convolved value, I is an image buffer, and G is the pre-computed Gaussian filter.

Then, the system computes the index which has the maximum convolved peak, e.g.

$$i^* = \arg\max_i C_i \tag{2.2}$$

From the example data, this i^* for each row can be mapped to a distance. We make the simplifying assumption that the camera is a perspective camera, which is a fairly close approximation to reality. Error from this assumption will lead to inconsistency in the distance values, but can be easily corrected by collecting more data.

The distance for each row follows the relation

$$D = \hat{w}_0 x^{\hat{w}_1} + b \tag{2.3}$$

We determine $\hat{w}$ and b from calibration data.

2.2. Software

2.2.1. Kernel driver

In our kernel driver, we use `ioread32`/`iowrite32` to transfer data between software and hardware. The following commands through `ioctl()` calls from userland are supported:

1. `PLAZER_CONV_MAX`: sets input data to the convolution, and returns the convolved peak as well as the index where the peak happens
2. `PLAZER_SET_CONV`: sets the convolution kernel
3. `PLAZER_READ_MEMORY`: returns all values in the memory, useful for debugging purposes
4. `PLAZER_RESET`: sets the entire memory space to zeros
5. `PLAZER_GET_BUFFER`: returns the most recent buffer

2.2.2. Userland application

At a high level, the userland application reads in a 640 by 480 image, converts it to grayscale using OpenCV, and then finds the index corresponding to the maximum convolved peak within each row. Using our calibrated rangefinder settings, we finally find the distance for each row following Eqn. 2.3.

A one-dimensional Gaussian kernel of length 16 is first computed. Since the Gaussian kernel is symmetrical, only half of the values are necessary. These values are sent to the hardware as the convolution kernel. Note that we could have used any filtering here, not just Gaussian, since the hardware implements a general convolution process.

Since the hardware only takes in 32 bytes and returns their convolved peak within one clock cycle, the user program splits each row of 640 pixels into 20 groups of 32-pixel segments, and sends each individual group per loop iteration to the hardware to find the convolved peak from each particular group. Since the convolution requires 8 extra pixels to the left and to the right of a group, those pixels are sent to the hardware together with each group of 32 pixels. We simply pad zeros at the edges when there are not enough number of surrounding pixels. The global maximum of convolved peaks from all 20 groups is the maximum for the entire row.

The index of each of these peaks is used as an input into Eqn. 2.3 to compute the final distance. The list of distances is then printed out to the console.

2.2.3. Calibration

We calibrate the system by taking pictures with the camera at known distances from the wall. By virtue of our simplifying assumption of the camera being perspective, we find the relationship between the laser distance and the pixel location representing by a power regression of a set of ordered pairs $(x, D) \in [0, 640] \times \mathbb{R}_+$ to generate the parameters $\theta = \hat{w}, b$ in Eqn. 2.3. We store θ as our calibrated rangefinder settings.

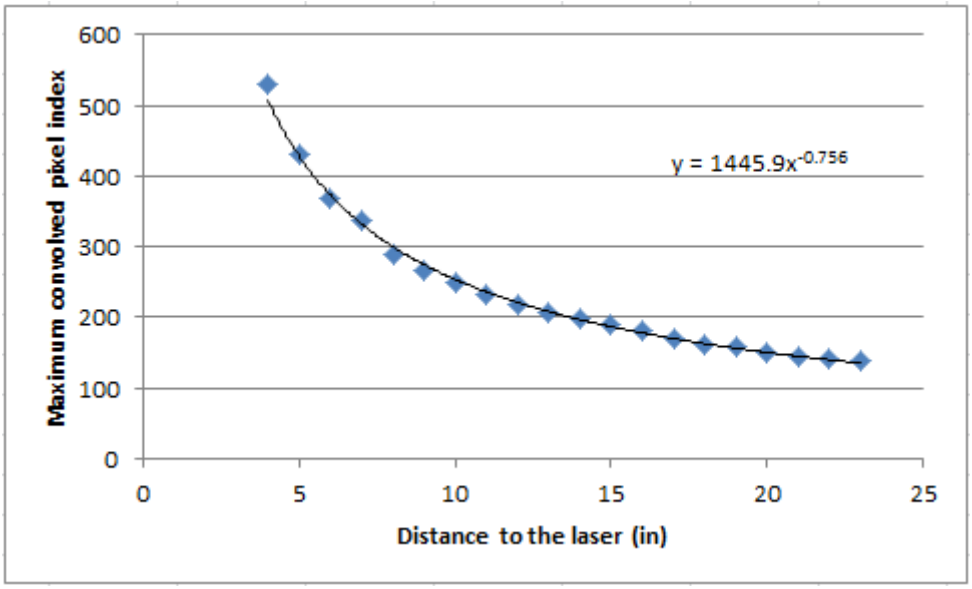

Figure 2.1.: Distance against max convolution index

As we can see in Fig. 2.1, there is a distinct power relationship between the distance and the maximum convolution index:

$$D = 15138.3 \times (i^*)^{-1.32275} \tag{2.4}$$

According the relationship above, the distance of any point on the object can be calculate using the image of the laser beam scans on the object.

2.3. Hardware

2.3.1. Addressable memory

The hardware represents itself as an addressable memory device with a single 32-bit word that cannot be written to, which it uses to return results. This memory device is implemented using a large unpacked byte array in `process_device.sv`, which holds the entire 60-byte address space.

The original implementation of the hardware used a shift register to shift in the data. However, as there is only an 8-byte overlap between buffered windows (the right fill of the i-th buffer overlaps with the data of the $i+1$-th buffer), it would only save two 4-byte writes to shift the data in, as opposed to writing it in at random addresses. Moreover, the addressable memory abstraction allows us to write the convolution vector in only once at the beginning of the program.

The primary working module is in `convmax.sv`, which aggregates the individual points to find the point with the highest convolved value. It contains 32 separate convolution blocks, as well as 32 comparators.

2.3.2. Single-clock convolution

As the total time of the algorithm is strongly dominated by the time it takes to transfer in the full 48-byte vector for each operation, we opted to maximize the amount of data that could be handled within a single clock cycle. In general, we wanted to avoid having the master block or poll the slave when possible, as those clock cycles would be better used transmitting data.

We constructed a sixteen-value discrete convolution module in `conv.sv`, which uses a Altera IP-based 16-way parallel adder and hardware multiplier units where available. As there were a limited number of floating-point multipliers, we chose to use a fixed-point representation for the convolution.

2.3.3. Physical setup

The physical setup of the device is designed to hold the camera and the laser pointer in fixed relation to one another. To do this, we designed and laser-cut an enclosure for the laser module, and then 3D-printed a mount for the PS3 Eye camera. Unlike in other triangulation-based rangefinders, we do not use direct measurements off of the experimental setup to serve as the parameters for the distance equation. This allowed us to be relatively imprecise with our mechanical design, and greatly simplified the process of construction.

2.4. Interface

2.4.1. Memory layout

The FPGA presents itself to the CPU as an Avalon memory-mapped slave on a 32-bit bus, running at 500 MHz. As the bus is only 4 bytes wide, the space is represented as a large contiguous shared memory with the layout as shown in Fig. 2.2.

initial fill	8 bytes
data	32 bytes
end fill	8 bytes
convolution vector	8 bytes
max value	2 bytes
max position	1 byte

Figure 2.2.: Memory layout for PLazeR FPGA co-processor

The initial and end fill blocks of 8 bytes each are necessary to make the convolutions well-defined for the first 8 bytes and the last 8 bytes of the data, as the convolution operates on a 16-wide buffer and returns the filtered value for its center.

Since each computation is completed within a single clock cycle, the FPGA does not need to assert `waitrequest` to wait for computation. Moreover, with the memory-like interface, we do not need to rewrite the convolution vector after properly initializing the

device. To avoid contention, the first 56 addresses are written to only by the CPU / supervising program, and the last four bytes are written to only by the FPGA.

The sequence of commands is therefore

1. write convolution vector
2. write left fill
3. write data
4. write right fill
5. read result
6. return to 2.

3. Design Decisions

3.1. Filtering

We experimented with a number of different filtering options in our software prototype. Our initial discussion focused on filters that could be implemented with convolutions, since a discrete convolution can be implemented as a finite impulse response filter with the FPGA's digital signal processing blocks.

Our initial results showed that the 2D Gaussian filter performed particularly well on our captured image data. This filter takes the following form:

$$C[i,j] = \sum_{i=1}^{16}\sum_{j=1}^{16} I[i,j]\frac{1}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}}e^{-\frac{i^2+j^2}{\sigma_1^2+\sigma_2^2}} \tag{3.1}$$

However, as we look at each row separately, this can be replaced by a smaller 1D Gaussian filter:

$$C[i] = \sum_{i=1}^{16} I[i]\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{i^2}{\sigma^2}} \tag{3.2}$$

Furthermore, the Gaussian filter is symmetric, so we only need to transfer the first half of the filter, and then mirror it for the other side. This results in Eqn. 2.1, which we have implemented in hardware.

3.2. Physics

The general design of the system is shown in Fig. 3.1. The PS3 Eye camera and the laser line are fixed with respect to one another. The laser is positioned to project a bright green vertical laser line across the object, which can be easily detected after processing.

We make the assumption throughout the processing that there is a power relationship between the detected x-value of the laser in the image and the distance to the object. This assumption holds if the camera is a perspective camera, which can capture the perpendicular image to the camera vector. Unfortunately, most cameras – including the PS3 Eye camera – are not true perspective cameras, and will introduce distortion into the measurement.

In theory, we can derive the relationship between the line position and the object distance from trigonometric principles. However, there is sufficient error in the measurements and the construction of the device that we found it more helpful to compute the distance coefficients from a set of calibration images with known distances.

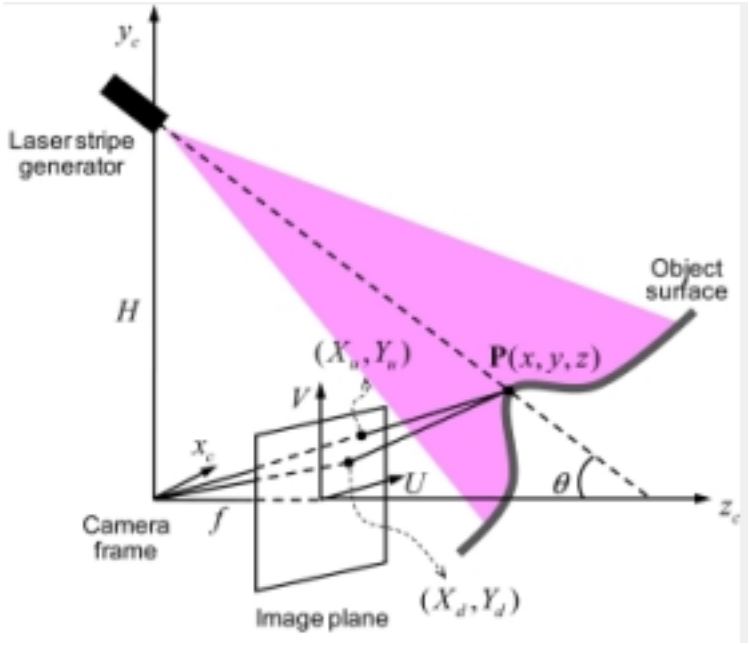

Figure 3.1.: Diagram of the laser setup, with the camera and laser line [**?**]

3.3. Memory and timing

One of the more involved decisions we made in the hardware and software design was the amount of data to store on the FPGA for processing. It was immediately obvious that we would not be able to store the entire image, its post-convolution result, and the other associated metadata in registers in hardware. However, because of the need to transmit an extra sixteen bytes of padding (eight bytes leading and eight bytes trailing), the overhead per computation was nontrivial.

We experimented with using M10K memory and DDR3 blocks. However, due to the fact that these communications also run over the Avalon bus and therefore have a strictly limited bus width, we decided that it would be more useful to be able to compute convolutions on the fly rather than to load more of an image in before computing the actual output result.

The max-convolution algorithm is embarrassingly parallel – each individual 16-byte convolution could be computed separately, and the n-way max could be implemented easily enough with binary comparators. Thus, we decided to use a purely combinatorial circuit to compute max-convolution, allowing us to return a result for a sub-buffer within a single clock cycle. The downside of this approach is that each convolution computation requires sixteen multipliers and a sixteen-way adder, so increasing the number of simultaneous computations was very expensive in terms of space.

Our final solution was to compute the max-convolution for a 32-byte buffer, which evenly divides both 640 (the width) and 480 (the height) of the image. This means that,

even if we decided to compute the full 2D Gaussian, we would not have to spend an extra transaction to catch the remainder of the image buffer.

An auxiliary benefit of a single-clock-cycle output was that we did not need to include signalling logic for starting and finishing the computation – instead, we simply compute the max-convolution on every clock cycle. We only read the result register after we have written all of the data that is necessary to have a valid output. This considerably simplifies the timing of the circuit, which effectively operates as an addressable memory object. In fact, when we remove the convolution logic, Quartus compiles the entire system as an M10K block!

4. Development Process

4.1. Software prototyping

We developed a software prototype for this system early on using OpenCV for image capture and processing. The prototype is built ground-up from the OpenCV libary functions, and for each part we did verification on the feasibility of the algorithm and the actual result it gives in our test setting. It allowed us to evaluate different methods for smoothing out the random noise in the image, and to gradually build up the various parts of the hardware and hardware-software interface.

Our prototype was designed to be modular, so in each stage we could replace a part of our prototype with a hardware implementation – the application actually uses the same functions in both the software- and hardware-backed versions, so that we could verify that our hardware worked as expected. In particular, this allows us to do integration tests on the hardware parts: the software part can be pulled from the system with the hardware swapped in, and we should expect to see the same result in the end. This ensures that the hardware will integrate with the existing system.

In addition, we wrote a system console test script (`syscon-test.tcl`) that could test the hardware independently of our software implementation. This proved especially useful when the software design was in flux, and helped verify the state of the hardware design.

4.2. Hardware Implementation

As discussed above, we decided to use hardware to gradually swapping out software, so we did Gaussian and finding row max separately; when they are combined, however, we ran into problem that the board did not have enough block components. This forced us to scale down and do less Gaussian for each data write (we started with 640*1 for the entire row), but did not change the overall design.

4.3. Testing

We did both testing of hardware using JTAG and testing of the system using the software prototype. In addition, we created some image test cases that the system can run to verify the result.

4.4. DE2i-150 vs. SoCKit

One of the problems we ran into early on with the software prototype was the fact that the SoCKit has insufficient USB bandwidth to properly stream video from the PS3 Eye camera. As a result, we spent a significant amount of time trying to work around this issue.

Among the many things we tried was investigating an FPGA development board with more onboard processing power, the Terasic DE2i-150. While it has a less capable FPGA (a Cyclone IV), it has a full Atom-based computer on it, and more importantly a high-speed USB controller hooked up to the PCI Express bus. Although we did not eventually use this (we ran into problem connecting the CPU and the FPGA - detail to be discussed in the "Challenges" section), we spent some amount of time trying to implement the algorithm on Cyclone IV and the x86 software environment based on the official and community tutorials.

5. Conclusion

5.1. Challenges faced

- USB bus bandwidth

  Our initial plan included a software-based program that streams live feed from the PS3 Eye camera and do live calculation of the distance using FPGA. However, the SoCKit board we are using seems to lack enough USB bandwidth to complete the action. After recompiling the ARM kernel to incorporate the USB camera driver, we ran into problems where even a single-frame image capture results in select() timeout in camera library and / or CPU stall, and can freeze the entire system. Changing the video library results in the same.

- PCI-E enumeration

  When we were testing on the DE2i-150 board with Linux configuration, the FPGA board, connected though a PCI Express interface implemented on it, cannot be found by the CPU-side operating system. Several attempts were made to fix this, including re-installing the kernel to the default version, clearing the CMOS chip, and replaying the steps listed on the official documentation, but they were not successful in the end. Besides the possibility that the board is defective, we suspect that the problem lies in that we are using new Altera version that ships an newer PCI-E interface module that is incompatible with the older version BIOS, but as we did not have BIOS flashing resources, we stopped our attempt on the board. Also, we sought resources online for the problem, very little could be found.

- Avalon bus width

  In our implementation we are strongly constrained in the bus width of the Avalon Memory-Mapped Interfaces, in which only 32 bits can be written / read from the FPGA board at the same time. As we are dealing with images with size at level of kilobytes, this creates large input and output overhead between the CPU and the FPGA.

5.2. Lessons learned

The biggest lesson we learned throughout the process of designing and developing PLazeR is that connecting hardware and software together is really hard, even when it's on a device which is intended to facilitate that connection. We spent an extraordinary amount

of time – much more than expected – trying to get our FPGA to communicate properly with the userland code that interfaced with the camera.

Beyond even communicating between hardware and software, though, was the difficulty in integrating the various parallel strands of work. Since all of the parts of the project were closely integrated, changes in one aspect of the project would require project-wide refactoring. Often, this meant that a significant amount of work had to be redone between iterations of the design.

If we were to do this project again, we would start by investigating the interfaces available between the project components, and ensure that we had working communication before attempting to implement the algorithms. If we had realized the limitations of the SoCKit board early on, we could have saved ourselves a lot of blood, sweat, and tears.

5.3. Future work

The SoCKit board has additional functions that can be explored related to this. project. For example, with more resources on the USB Camera interface, it is possible that the USB connection to the camera be handled directly by the FPGA board.

5.4. Distribution of work

Robert Ying	algorithm, physical design, software prototype, kernel driver, FPGA design
"Derek" Xingzhou He	software prototype, kernel build, kernel driver, integration,
Peiqian Li	userland software, FPGA design, test image generation script
Minh Trang Nguyen	calibration, data collection,

A. Source listings

A.1. process_device.sv

```
module process_device(
    input logic                 reset,
    input logic                 clk,
    input logic                 write,
    input logic                 read,
    input logic [10:0]          address,
    input logic [3:0]           byteenable,
    input logic [31:0]          writedata,
    output logic [31:0]         readdata,
    output logic                waitrequest
);

// memory layout
// address = 0x00
// byteenable = 1111
// [image data ][gaussian ]
// [144 x 8 bit][8 x 8 bit  ]
// [0.....1151][1152...1216]
// note that this is only the left half of the gaussian, since
// the gaussian kernel is symmetric about 0

// readdata layout
// address = 0x00
// byteenable = 1110
// [1 x 16 bit][1 x 8 bit]
// [0.......15][16.....23]

logic [15:0]        maxval;
logic [7:0]         maxpos;
logic               ready;
logic [7:0]         memory [0:59];
logic [7:0]         convval [0:31];

convmax cm1(.clk        (clk),
```

```
            .indata     (memory[0:47]),
            .gauss      (memory[48:55]),
                                .val        (convval),
            .maxval     (maxval),
            .maxpos     (maxpos),
            .ready      (ready)
            );

always_ff @(posedge clk)
begin
    //if (reset) memory <= 0;
    if (write) begin
             if (address < 56) begin
            if (byteenable[0]) memory[4 * address] <= writedata[7:0];
            if (byteenable[1]) memory[4 * address + 1] <= writedata[15:8];
            if (byteenable[2]) memory[4 * address + 2] <= writedata[23:16];
            if (byteenable[3]) memory[4 * address + 3] <= writedata[31:24];
                end
                if (address == 56) memory[8:39] <= convval[0:31];
    end
        memory[56] <= maxval[7:0];
    memory[57] <= maxval[15:8];
    memory[58] <= maxpos[7:0];
    memory[59] <= ready;
end

assign waitrequest = 0;

assign readdata[7:0] = memory[4 * address];
assign readdata[15:8] = memory[4 * address + 1];
assign readdata[23:16] = memory[4 * address + 2];
assign readdata[31:24] = memory[4 * address + 3];

endmodule
```

A.2. conv.sv

```
module conv(
    input logic                 clk,
    input logic [7:0]           data [0:15],
    input logic [7:0]           gauss [0:7],

    output logic [16:0]         convvalue
);
```

```
logic [15:0] marr [0:15];
logic [19:0] val;

mult m0(data[0], gauss[0], marr[0]);
//assign marr[0] = 0;
mult m1(data[1], gauss[1], marr[1]);
//assign marr[1] = 0;
mult m2(data[2], gauss[2], marr[2]);
mult m3(data[3], gauss[3], marr[3]);
mult m4(data[4], gauss[4], marr[4]);
mult m5(data[5], gauss[5], marr[5]);
mult m6(data[6], gauss[6], marr[6]);
mult m7(data[7], gauss[7], marr[7]);

mult m15(data[15], gauss[0], marr[15]);
//assign marr[15] = 0;
mult m14(data[14], gauss[1], marr[14]);
//assign marr[14] = 0;
mult m13(data[13], gauss[2], marr[13]);
mult m12(data[12], gauss[3], marr[12]);
mult m11(data[11], gauss[4], marr[11]);
mult m10(data[10], gauss[5], marr[10]);
mult m9(data[9], gauss[6], marr[9]);
mult m8(data[8], gauss[7], marr[8]);

p_add p0(marr[0], marr[1], marr[2], marr[3],
         marr[4], marr[5], marr[6], marr[7],
         marr[8], marr[9], marr[10], marr[11],
         marr[12], marr[13], marr[14], marr[15],
         val);

assign convvalue = val[19:3];

endmodule
```

A.3. convmax.sv

```
module convmax(
    input logic                 clk,
    input logic [7:0]           indata[0:47],
    input logic [7:0]           gauss[0:7],

        output logic [15:0]         val[0:31],
```

```
    output logic [15:0]          maxval,
    output logic [7:0]           maxpos,
    output logic                 ready
);

logic[15:0] maxval_so_far[0:31];
logic[7:0]  maxpos_so_far[0:31];

assign maxval_so_far[0] = 16'b0;
assign maxpos_so_far[0] = 8'b0;

genvar i;
generate
begin
    for( i = 0; i < 32; i++ ) begin: for_i
        conv(.clk         (clk),        //input
             .data        (indata[i:i+15]),     // input
             .gauss       (gauss),              //input
             .convvalue   (val[i])      // output [16:0]  convvalue
        );
        if (i>0) begin
            assign maxval_so_far[i] = (maxval_so_far[i-1] > val[i]) ? maxval_
            assign maxpos_so_far[i] = (maxval_so_far[i-1] > val[i]) ? maxpos_
        end
    end
end
endgenerate

assign maxval = maxval_so_far[31];
assign maxpos = maxpos_so_far[31];
assign ready = 1;

endmodule
```

A.4. mult.v

```
// megafunction wizard: %LPM_MULT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: lpm_mult

// ============================================================
// File Name: mult.v
// Megafunction Name(s):
```

```
//                          lpm_mult
//
// Simulation Library File(s):
//                          lpm
// ================================================================
// ****************************************************************
// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
//
// 14.0.0 Build 200 06/17/2014 SJ Web Edition
// ****************************************************************


//Copyright (C) 1991-2014 Altera Corporation. All rights reserved.
//Your use of Altera Corporation's design tools, logic functions
//and other software and tools, and its AMPP partner logic
//functions, and any output files from any of the foregoing
//(including device programming or simulation files), and any
//associated documentation or information are expressly subject
//to the terms and conditions of the Altera Program License
//Subscription Agreement, the Altera Quartus II License Agreement,
//the Altera MegaCore Function License Agreement, or other
//applicable license agreement, including, without limitation,
//that your use is for the sole purpose of programming logic
//devices manufactured by Altera and sold by Altera or its
//authorized distributors.  Please refer to the applicable
//agreement for further details.


// synopsys translate_off
`timescale 1 ps / 1 ps
// synopsys translate_on
module mult (
        dataa,
        datab,
        result);

        input   [7:0]  dataa;
        input   [7:0]  datab;
        output  [15:0]  result;

        wire [15:0] sub_wire0;
        wire [15:0] result = sub_wire0[15:0];

        lpm_mult        lpm_mult_component (
```

```
                                .dataa (dataa),
                                .datab (datab),
                                .result (sub_wire0),
                                .aclr (1'b0),
                                .clken (1'b1),
                                .clock (1'b0),
                                .sum (1'b0));
        defparam
                lpm_mult_component.lpm_hint = "DEDICATED_MULTIPLIER_CIRCUITRY=
                lpm_mult_component.lpm_representation = "UNSIGNED",
                lpm_mult_component.lpm_type = "LPM_MULT",
                lpm_mult_component.lpm_widtha = 8,
                lpm_mult_component.lpm_widthb = 8,
                lpm_mult_component.lpm_widthp = 16;


endmodule

// ============================================================
// CNX file retrieval info
// ============================================================
// Retrieval info: PRIVATE: AutoSizeResult NUMERIC "0"
// Retrieval info: PRIVATE: B_isConstant NUMERIC "0"
// Retrieval info: PRIVATE: ConstantB NUMERIC "0"
// Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "Cyclone V"
// Retrieval info: PRIVATE: LPM_PIPELINE NUMERIC "0"
// Retrieval info: PRIVATE: Latency NUMERIC "0"
// Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"
// Retrieval info: PRIVATE: SignedMult NUMERIC "0"
// Retrieval info: PRIVATE: USE_MULT NUMERIC "1"
// Retrieval info: PRIVATE: ValidConstant NUMERIC "0"
// Retrieval info: PRIVATE: WidthA NUMERIC "8"
// Retrieval info: PRIVATE: WidthB NUMERIC "8"
// Retrieval info: PRIVATE: WidthP NUMERIC "16"
// Retrieval info: PRIVATE: aclr NUMERIC "0"
// Retrieval info: PRIVATE: clken NUMERIC "0"
// Retrieval info: PRIVATE: new_diagram STRING "1"
// Retrieval info: PRIVATE: optimize NUMERIC "0"
// Retrieval info: LIBRARY: lpm lpm.lpm_components.all
// Retrieval info: CONSTANT: LPM_HINT STRING "DEDICATED_MULTIPLIER_CIRCUITRY=
// Retrieval info: CONSTANT: LPM_REPRESENTATION STRING "UNSIGNED"
// Retrieval info: CONSTANT: LPM_TYPE STRING "LPM_MULT"
// Retrieval info: CONSTANT: LPM_WIDTHA NUMERIC "8"
// Retrieval info: CONSTANT: LPM_WIDTHB NUMERIC "8"
```

```
// Retrieval info: CONSTANT: LPM_WIDTHP NUMERIC "16"
// Retrieval info: USED_PORT: dataa 0 0 8 0 INPUT NODEFVAL "dataa[7..0]"
// Retrieval info: USED_PORT: datab 0 0 8 0 INPUT NODEFVAL "datab[7..0]"
// Retrieval info: USED_PORT: result 0 0 16 0 OUTPUT NODEFVAL "result[15..0]"
// Retrieval info: CONNECT: @dataa 0 0 8 0 dataa 0 0 8 0
// Retrieval info: CONNECT: @datab 0 0 8 0 datab 0 0 8 0
// Retrieval info: CONNECT: result 0 0 16 0 @result 0 0 16 0
// Retrieval info: GEN_FILE: TYPE_NORMAL mult.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL mult.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL mult.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL mult.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL mult_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL mult_bb.v TRUE
// Retrieval info: LIB_FILE: lpm
```

A.5. p_add.v

```
// megafunction wizard: %PARALLEL_ADD%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: parallel_add

// ============================================================
// File Name: p_add.v
// Megafunction Name(s):
//                      parallel_add
//
// Simulation Library Files(s):
//                      altera_mf
// ============================================================
// ************************************************************
// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
//
// 14.0.0 Build 200 06/17/2014 SJ Web Edition
// ************************************************************


//Copyright (C) 1991-2014 Altera Corporation. All rights reserved.
//Your use of Altera Corporation's design tools, logic functions
//and other software and tools, and its AMPP partner logic
//functions, and any output files from any of the foregoing
//(including device programming or simulation files), and any
//associated documentation or information are expressly subject
//to the terms and conditions of the Altera Program License
```

```
//Subscription Agreement, the Altera Quartus II License Agreement,
//the Altera MegaCore Function License Agreement, or other
//applicable license agreement, including, without limitation,
//that your use is for the sole purpose of programming logic
//devices manufactured by Altera and sold by Altera or its
//authorized distributors.  Please refer to the applicable
//agreement for further details.


// synopsys translate_off
`timescale 1 ps / 1 ps
// synopsys translate_on
module p_add (
        data0x,
        data10x,
        data11x,
        data12x,
        data13x,
        data14x,
        data15x,
        data1x,
        data2x,
        data3x,
        data4x,
        data5x,
        data6x,
        data7x,
        data8x,
        data9x,
        result);

        input   [15:0]  data0x;
        input   [15:0]  data10x;
        input   [15:0]  data11x;
        input   [15:0]  data12x;
        input   [15:0]  data13x;
        input   [15:0]  data14x;
        input   [15:0]  data15x;
        input   [15:0]  data1x;
        input   [15:0]  data2x;
        input   [15:0]  data3x;
        input   [15:0]  data4x;
        input   [15:0]  data5x;
        input   [15:0]  data6x;
```

```
input    [15:0]  data7x;
input    [15:0]  data8x;
input    [15:0]  data9x;
output   [19:0]  result;

wire [19:0] sub_wire17;
wire [15:0] sub_wire16 = data15x[15:0];
wire [15:0] sub_wire15 = data14x[15:0];
wire [15:0] sub_wire14 = data13x[15:0];
wire [15:0] sub_wire13 = data12x[15:0];
wire [15:0] sub_wire12 = data11x[15:0];
wire [15:0] sub_wire11 = data10x[15:0];
wire [15:0] sub_wire10 = data9x[15:0];
wire [15:0] sub_wire9 = data8x[15:0];
wire [15:0] sub_wire8 = data7x[15:0];
wire [15:0] sub_wire7 = data6x[15:0];
wire [15:0] sub_wire6 = data5x[15:0];
wire [15:0] sub_wire5 = data4x[15:0];
wire [15:0] sub_wire4 = data3x[15:0];
wire [15:0] sub_wire3 = data2x[15:0];
wire [15:0] sub_wire2 = data1x[15:0];
wire [15:0] sub_wire0 = data0x[15:0];
wire [255:0] sub_wire1 = {sub_wire16, sub_wire15, sub_wire14, sub_wire
wire [19:0] result = sub_wire17[19:0];

parallel_add     parallel_add_component (
                        .data (sub_wire1),
                        .result (sub_wire17)
                        // synopsys translate_off
                        ,
                        .aclr (),
                        .clken (),
                        .clock ()
                        // synopsys translate_on
                        );
defparam
        parallel_add_component.msw_subtract = "NO",
        parallel_add_component.pipeline = 0,
        parallel_add_component.representation = "UNSIGNED",
        parallel_add_component.result_alignment = "LSB",
        parallel_add_component.shift = 0,
        parallel_add_component.size = 16,
        parallel_add_component.width = 16,
        parallel_add_component.widthr = 20;
```

endmodule

```
// ============================================================
// CNX file retrieval info
// ============================================================
// Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "Cyclone V"
// Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"
// Retrieval info: LIBRARY: altera_mf altera_mf.altera_mf_components.all
// Retrieval info: CONSTANT: MSW_SUBTRACT STRING "NO"
// Retrieval info: CONSTANT: PIPELINE NUMERIC "0"
// Retrieval info: CONSTANT: REPRESENTATION STRING "UNSIGNED"
// Retrieval info: CONSTANT: RESULT_ALIGNMENT STRING "LSB"
// Retrieval info: CONSTANT: SHIFT NUMERIC "0"
// Retrieval info: CONSTANT: SIZE NUMERIC "16"
// Retrieval info: CONSTANT: WIDTH NUMERIC "16"
// Retrieval info: CONSTANT: WIDTHR NUMERIC "20"
// Retrieval info: USED_PORT: data0x 0 0 16 0 INPUT NODEFVAL "data0x[15..0]"
// Retrieval info: USED_PORT: data10x 0 0 16 0 INPUT NODEFVAL "data10x[15..0]"
// Retrieval info: USED_PORT: data11x 0 0 16 0 INPUT NODEFVAL "data11x[15..0]"
// Retrieval info: USED_PORT: data12x 0 0 16 0 INPUT NODEFVAL "data12x[15..0]"
// Retrieval info: USED_PORT: data13x 0 0 16 0 INPUT NODEFVAL "data13x[15..0]"
// Retrieval info: USED_PORT: data14x 0 0 16 0 INPUT NODEFVAL "data14x[15..0]"
// Retrieval info: USED_PORT: data15x 0 0 16 0 INPUT NODEFVAL "data15x[15..0]"
// Retrieval info: USED_PORT: data1x 0 0 16 0 INPUT NODEFVAL "data1x[15..0]"
// Retrieval info: USED_PORT: data2x 0 0 16 0 INPUT NODEFVAL "data2x[15..0]"
// Retrieval info: USED_PORT: data3x 0 0 16 0 INPUT NODEFVAL "data3x[15..0]"
// Retrieval info: USED_PORT: data4x 0 0 16 0 INPUT NODEFVAL "data4x[15..0]"
// Retrieval info: USED_PORT: data5x 0 0 16 0 INPUT NODEFVAL "data5x[15..0]"
// Retrieval info: USED_PORT: data6x 0 0 16 0 INPUT NODEFVAL "data6x[15..0]"
// Retrieval info: USED_PORT: data7x 0 0 16 0 INPUT NODEFVAL "data7x[15..0]"
// Retrieval info: USED_PORT: data8x 0 0 16 0 INPUT NODEFVAL "data8x[15..0]"
// Retrieval info: USED_PORT: data9x 0 0 16 0 INPUT NODEFVAL "data9x[15..0]"
// Retrieval info: USED_PORT: result 0 0 20 0 OUTPUT NODEFVAL "result[19..0]"
// Retrieval info: CONNECT: @data 0 0 16 0 data0x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 160 data10x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 176 data11x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 192 data12x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 208 data13x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 224 data14x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 240 data15x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 16 data1x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 32 data2x 0 0 16 0
```

```
// Retrieval info: CONNECT: @data 0 0 16 48 data3x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 64 data4x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 80 data5x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 96 data6x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 112 data7x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 128 data8x 0 0 16 0
// Retrieval info: CONNECT: @data 0 0 16 144 data9x 0 0 16 0
// Retrieval info: CONNECT: result 0 0 20 0 @result 0 0 20 0
// Retrieval info: GEN_FILE: TYPE_NORMAL p_add.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL p_add.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL p_add.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL p_add.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL p_add_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL p_add_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf
```

A.6. SoCKit_top.v

```
// ============================================================================
// Copyright (c) 2013 by Terasic Technologies Inc.
// ============================================================================
//
// Permission:
//
//   Terasic grants permission to use and modify this code for use
//   in synthesis for all Terasic Development Boards and Altera Development
//   Kits made by Terasic.  Other use of this code, including the selling
//   ,duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
//   This VHDL/Verilog or C/C++ source code is intended as a design reference
//   which illustrates how these types of functions can be implemented.
//   It is the user's responsibility to verify their design for
//   consistency and functionality through the use of formal
//   verification methods.  Terasic provides no warranty regarding the use
//   or functionality of this code.
//
// ============================================================================
//
//  Terasic Technologies Inc
//  9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu City, 30070. Taiwa
//
//
```

```
//                          web: http://www.terasic.com/
//                          email: support@terasic.com
//
// =============================================================================
// =============================================================================
//
// Major Functions:      SoCKit_Default
//
// =============================================================================
// Revision History :
// =============================================================================
//   Ver  :| Author              :| Mod. Date :| Changes Made:
//   V1.0 :| xinxian             :| 04/02/13  :| Initial Revision
// =============================================================================

//`define ENABLE_DDR3
//`define ENABLE_HPS
//`define ENABLE_HSMC_XCVR

module SoCKit_Top(

                        ///////////AUD/////////////
                        AUD_ADCDAT,
                        AUD_ADCLRCK,
                        AUD_BCLK,
                        AUD_DACDAT,
                        AUD_DACLRCK,
                        AUD_I2C_SCLK,
                        AUD_I2C_SDAT,
                        AUD_MUTE,
                        AUD_XCK,

`ifdef ENABLE_DDR3
                        ////////DDR3////////
                        DDR3_A,
                        DDR3_BA,
                        DDR3_CAS_n,
                        DDR3_CKE,
                        DDR3_CK_n,
                        DDR3_CK_p,
                        DDR3_CS_n,
                        DDR3_DM,
                        DDR3_DQ,
                        DDR3_DQS_n,
```

```
                        DDR3_DQS_p,
                        DDR3_ODT,
                        DDR3_RAS_n,
                        DDR3_RESET_n,
                        DDR3_RZQ,
                        DDR3_WE_n,
`endif /*ENABLE_DDR3*/

                        ////////FAN////////
                        FAN_CTRL,

`ifdef ENABLE_HPS
                        ////////HPS////////
                        HPS_CLOCK_25,
                        HPS_CLOCK_50,
                        HPS_CONV_USB_n,
                        HPS_DDR3_A,
                        HPS_DDR3_BA,
                        HPS_DDR3_CAS_n,
                        HPS_DDR3_CKE,
                        HPS_DDR3_CK_n,
                        HPS_DDR3_CK_p,
                        HPS_DDR3_CS_n,
                        HPS_DDR3_DM,
                        HPS_DDR3_DQ,
                        HPS_DDR3_DQS_n,
                        HPS_DDR3_DQS_p,
                        HPS_DDR3_ODT,
                        HPS_DDR3_RAS_n,
                        HPS_DDR3_RESET_n,
                        HPS_DDR3_RZQ,
                        HPS_DDR3_WE_n,
                        HPS_ENET_GTX_CLK,
                        HPS_ENET_INT_n,
                        HPS_ENET_MDC,
                        HPS_ENET_MDIO,
                        HPS_ENET_RESET_n,
                        HPS_ENET_RX_CLK,
                        HPS_ENET_RX_DATA,
                        HPS_ENET_RX_DV,
                        HPS_ENET_TX_DATA,
                        HPS_ENET_TX_EN,
                        HPS_FLASH_DATA,
                        HPS_FLASH_DCLK,
```

```
                        HPS_FLASH_NCSO,
                        HPS_GSENSOR_INT,
                        HPS_I2C_CLK,
                        HPS_I2C_SDA,
                        HPS_KEY,
                        HPS_LCM_D_C,
                        HPS_LCM_RST_N,
                        HPS_LCM_SPIM_CLK,
                        HPS_LCM_SPIM_MISO,
                        HPS_LCM_SPIM_MOSI,
                        HPS_LCM_SPIM_SS,
                        HPS_LED,
                        HPS_LTC_GPIO,
                        HPS_RESET_n,
                        HPS_SD_CLK,
                        HPS_SD_CMD,
                        HPS_SD_DATA,
                        HPS_SPIM_CLK,
                        HPS_SPIM_MISO,
                        HPS_SPIM_MOSI,
                        HPS_SPIM_SS,
                        HPS_SW,
                        HPS_UART_RX,
                        HPS_UART_TX,
                        HPS_USB_CLKOUT,
                        HPS_USB_DATA,
                        HPS_USB_DIR,
                        HPS_USB_NXT,
                        HPS_USB_RESET_PHY,
                        HPS_USB_STP,
                        HPS_WARM_RST_n,
`endif /*ENABLE_HPS*/

                        ////////HSMC////////
                        HSMC_CLKIN_n,
                        HSMC_CLKIN_p,
                        HSMC_CLKOUT_n,
                        HSMC_CLKOUT_p,
                        HSMC_CLK_IN0,
                        HSMC_CLK_OUT0,
                        HSMC_D,

`ifdef ENABLE_HSMC_XCVR
```

```
                    HSMC_GXB_RX_p,
                    HSMC_GXB_TX_p,
                    HSMC_REF_CLK_p,
`endif
                    HSMC_RX_n,
                    HSMC_RX_p,
                    HSMC_SCL,
                    HSMC_SDA,
                    HSMC_TX_n,
                    HSMC_TX_p,

                    ////////IRDA////////
                    IRDA_RXD,

                    ////////KEY////////
                    KEY,

                    ////////LED////////
                    LED,

                    ////////OSC////////
                    OSC_50_B3B,
                    OSC_50_B4A,
                    OSC_50_B5B,
                    OSC_50_B8A,

                    ////////PCIE////////
                    PCIE_PERST_n,
                    PCIE_WAKE_n,

                    ////////RESET////////
                    RESET_n,

                    ////////SI5338////////
                    SI5338_SCL,
                    SI5338_SDA,

                    ////////SW////////
                    SW,

                    ////////TEMP////////
                    TEMP_CS_n,
                    TEMP_DIN,
                    TEMP_DOUT,
```

```
TEMP_SCLK,

//////////USB//////////
USB_B2_CLK,
USB_B2_DATA,
USB_EMPTY,
USB_FULL,
USB_OE_n,
USB_RD_n,
USB_RESET_n,
USB_SCL,
USB_SDA,
USB_WR_n,

//////////VGA//////////
VGA_B,
VGA_BLANK_n,
VGA_CLK,
VGA_G,
VGA_HS,
VGA_R,
VGA_SYNC_n,
VGA_VS,
////////////hps//////////
memory_mem_a,
memory_mem_ba,
memory_mem_ck,
memory_mem_ck_n,
memory_mem_cke,
memory_mem_cs_n,
memory_mem_ras_n,
memory_mem_cas_n,
memory_mem_we_n,
memory_mem_reset_n,
memory_mem_dq,
memory_mem_dqs,
memory_mem_dqs_n,
memory_mem_odt,
memory_mem_dm,
memory_oct_rzqin,
hps_io_hps_io_emac1_inst_TX_CLK,
hps_io_hps_io_emac1_inst_TXD0,
hps_io_hps_io_emac1_inst_TXD1,
hps_io_hps_io_emac1_inst_TXD2,
```

```
hps_io_hps_io_emac1_inst_TXD3,
hps_io_hps_io_emac1_inst_RXD0,
hps_io_hps_io_emac1_inst_MDIO,
hps_io_hps_io_emac1_inst_MDC,
hps_io_hps_io_emac1_inst_RX_CTL,
hps_io_hps_io_emac1_inst_TX_CTL,
hps_io_hps_io_emac1_inst_RX_CLK,
hps_io_hps_io_emac1_inst_RXD1,
hps_io_hps_io_emac1_inst_RXD2,
hps_io_hps_io_emac1_inst_RXD3,
hps_io_hps_io_qspi_inst_IO0,
hps_io_hps_io_qspi_inst_IO1,
hps_io_hps_io_qspi_inst_IO2,
hps_io_hps_io_qspi_inst_IO3,
hps_io_hps_io_qspi_inst_SS0,
hps_io_hps_io_qspi_inst_CLK,
hps_io_hps_io_sdio_inst_CMD,
hps_io_hps_io_sdio_inst_D0,
hps_io_hps_io_sdio_inst_D1,
hps_io_hps_io_sdio_inst_CLK,
hps_io_hps_io_sdio_inst_D2,
hps_io_hps_io_sdio_inst_D3,
hps_io_hps_io_usb1_inst_D0,
hps_io_hps_io_usb1_inst_D1,
hps_io_hps_io_usb1_inst_D2,
hps_io_hps_io_usb1_inst_D3,
hps_io_hps_io_usb1_inst_D4,
hps_io_hps_io_usb1_inst_D5,
hps_io_hps_io_usb1_inst_D6,
hps_io_hps_io_usb1_inst_D7,
hps_io_hps_io_usb1_inst_CLK,
hps_io_hps_io_usb1_inst_STP,
hps_io_hps_io_usb1_inst_DIR,
hps_io_hps_io_usb1_inst_NXT,
hps_io_hps_io_spim0_inst_CLK,
hps_io_hps_io_spim0_inst_MOSI,
hps_io_hps_io_spim0_inst_MISO,
hps_io_hps_io_spim0_inst_SS0,
hps_io_hps_io_spim1_inst_CLK,
hps_io_hps_io_spim1_inst_MOSI,
hps_io_hps_io_spim1_inst_MISO,
hps_io_hps_io_spim1_inst_SS0,
hps_io_hps_io_uart0_inst_RX,
hps_io_hps_io_uart0_inst_TX,
```

```
                    hps_io_hps_io_i2c1_inst_SDA,
                    hps_io_hps_io_i2c1_inst_SCL,
                    hps_io_hps_io_gpio_inst_GPIO00
                    );

    //=======================================================
    //  PORT declarations
    //=======================================================

    ///////// AUD /////////
    input                                               AUD_ADCDAT;
    inout                                               AUD_ADCLRCK;
    inout                                               AUD_BCLK;
    output                                              AUD_DACDAT;
    inout                                               AUD_DACLRCK;
    output                                              AUD_I2C_SCLK;
    inout                                               AUD_I2C_SDAT;
    output                                              AUD_MUTE;
    output                                              AUD_XCK;

`ifdef ENABLE_DDR3
    ///////// DDR3 /////////
    output [14:0]                                       DDR3_A;
    output [2:0]                                        DDR3_BA;
    output                                              DDR3_CAS_n;
    output                                              DDR3_CKE;
    output                                              DDR3_CK_n;
    output                                              DDR3_CK_p;
    output                                              DDR3_CS_n;
    output [3:0]                                        DDR3_DM;
    inout [31:0]                                        DDR3_DQ;
    inout [3:0]                                         DDR3_DQS_n;
    inout [3:0]                                         DDR3_DQS_p;
    output                                              DDR3_ODT;
    output                                              DDR3_RAS_n;
    output                                              DDR3_RESET_n;
    input                                               DDR3_RZQ;
    output                                              DDR3_WE_n;
`endif /*ENABLE_DDR3*/

    ///////// FAN /////////
    output                                              FAN_CTRL;

`ifdef ENABLE_HPS
```

```
//////// HPS ////////
input                                HPS_CLOCK_25;
input                                HPS_CLOCK_50;
input                                HPS_CONV_USB_n;
output [14:0]                        HPS_DDR3_A;
output [2:0]                         HPS_DDR3_BA;
output                               HPS_DDR3_CAS_n;
output                               HPS_DDR3_CKE;
output                               HPS_DDR3_CK_n;
output                               HPS_DDR3_CK_p;
output                               HPS_DDR3_CS_n;
output [3:0]                         HPS_DDR3_DM;
inout  [31:0]                        HPS_DDR3_DQ;
inout  [3:0]                         HPS_DDR3_DQS_n;
inout  [3:0]                         HPS_DDR3_DQS_p;
output                               HPS_DDR3_ODT;
output                               HPS_DDR3_RAS_n;
output                               HPS_DDR3_RESET_n;
input                                HPS_DDR3_RZQ;
output                               HPS_DDR3_WE_n;
input                                HPS_ENET_GTX_CLK;
input                                HPS_ENET_INT_n;
output                               HPS_ENET_MDC;
inout                                HPS_ENET_MDIO;
output                               HPS_ENET_RESET_n;
input                                HPS_ENET_RX_CLK;
input  [3:0]                         HPS_ENET_RX_DATA;
input                                HPS_ENET_RX_DV;
output [3:0]                         HPS_ENET_TX_DATA;
output                               HPS_ENET_TX_EN;
inout  [3:0]                         HPS_FLASH_DATA;
output                               HPS_FLASH_DCLK;
output                               HPS_FLASH_NCSO;
input                                HPS_GSENSOR_INT;
inout                                HPS_I2C_CLK;
inout                                HPS_I2C_SDA;
inout  [3:0]                         HPS_KEY;
output                               HPS_LCM_D_C;
output                               HPS_LCM_RST_N;
input                                HPS_LCM_SPIM_CLK;
inout                                HPS_LCM_SPIM_MISO;
output                               HPS_LCM_SPIM_MOSI;
output                               HPS_LCM_SPIM_SS;
output [3:0]                         HPS_LED;
```

```
    inout                                       HPS_LTC_GPIO;
    input                                       HPS_RESET_n;
    output                                      HPS_SD_CLK;
    inout                                       HPS_SD_CMD;
    inout  [3:0]                                HPS_SD_DATA;
    output                                      HPS_SPIM_CLK;
    input                                       HPS_SPIM_MISO;
    output                                      HPS_SPIM_MOSI;
    output                                      HPS_SPIM_SS;
    input  [3:0]                                HPS_SW;
    input                                       HPS_UART_RX;
    output                                      HPS_UART_TX;
    input                                       HPS_USB_CLKOUT;
    inout  [7:0]                                HPS_USB_DATA;
    input                                       HPS_USB_DIR;
    input                                       HPS_USB_NXT;
    output                                      HPS_USB_RESET_PHY;
    output                                      HPS_USB_STP;
    input                                       HPS_WARM_RST_n;
`endif /*ENABLE_HPS*/

    //////// HSMC ////////
    input  [2:1]                                HSMC_CLKIN_n;
    input  [2:1]                                HSMC_CLKIN_p;
    output [2:1]                                HSMC_CLKOUT_n;
    output [2:1]                                HSMC_CLKOUT_p;
    input                                       HSMC_CLK_IN0;
    output                                      HSMC_CLK_OUT0;
    inout  [3:0]                                HSMC_D;
`ifdef ENABLE_HSMC_XCVR
    input  [7:0]                                HSMC_GXB_RX_p;
    output [7:0]                                HSMC_GXB_TX_p;
    input                                       HSMC_REF_CLK_p;
`endif
    inout  [16:0]                               HSMC_RX_n;
    inout  [16:0]                               HSMC_RX_p;
    output                                      HSMC_SCL;
    inout                                       HSMC_SDA;
    inout  [16:0]                               HSMC_TX_n;
    inout  [16:0]                               HSMC_TX_p;

    //////// IRDA ////////
    input                                       IRDA_RXD;
```

```
//////// KEY ////////
input  [3:0]                    KEY;

//////// LED ////////
output [3:0]                    LED;

//////// OSC ////////
input                           OSC_50_B3B;
input                           OSC_50_B4A;
input                           OSC_50_B5B;
input                           OSC_50_B8A;

//////// PCIE ////////
input                           PCIE_PERST_n;
input                           PCIE_WAKE_n;

//////// RESET ////////
input                           RESET_n;

//////// SI5338 ////////
inout                           SI5338_SCL;
inout                           SI5338_SDA;

//////// SW ////////
input  [3:0]                    SW;

//////// TEMP ////////
output                          TEMP_CS_n;
output                          TEMP_DIN;
input                           TEMP_DOUT;
output                          TEMP_SCLK;

//////// USB ////////
input                           USB_B2_CLK;
inout  [7:0]                    USB_B2_DATA;
output                          USB_EMPTY;
output                          USB_FULL;
input                           USB_OE_n;
input                           USB_RD_n;
input                           USB_RESET_n;
inout                           USB_SCL;
inout                           USB_SDA;
input                           USB_WR_n;
```

```
//////// VGA ////////
output [7:0]                                VGA_B;
output                                      VGA_BLANK_n;
output                                      VGA_CLK;
output [7:0]                                VGA_G;
output                                      VGA_HS;
output [7:0]                                VGA_R;
output                                      VGA_SYNC_n;
output                                      VGA_VS;

////////hps pin//////
output wire [14:0]                          memory_mem_a;
output wire [2:0]                           memory_mem_ba;
output wire                                 memory_mem_ck;
output wire                                 memory_mem_ck_n;
output wire                                 memory_mem_cke;
output wire                                 memory_mem_cs_n;
output wire                                 memory_mem_ras_n;
output wire                                 memory_mem_cas_n;
output wire                                 memory_mem_we_n;
output wire                                 memory_mem_reset_n;
inout  wire [31:0]                          memory_mem_dq;
inout  wire [3:0]                           memory_mem_dqs;
inout  wire [3:0]                           memory_mem_dqs_n;
output wire                                 memory_mem_odt;
output wire [3:0]                           memory_mem_dm;
input  wire                                 memory_oct_rzqin;
output wire                                 hps_io_hps_io_emac1_inst_
output wire                                 hps_io_hps_io_emac1_inst_
output wire                                 hps_io_hps_io_emac1_inst_
output wire                                 hps_io_hps_io_emac1_inst_
output wire                                 hps_io_hps_io_emac1_inst_
input  wire                                 hps_io_hps_io_emac1_inst_
inout  wire                                 hps_io_hps_io_emac1_inst_
output wire                                 hps_io_hps_io_emac1_inst_
input  wire                                 hps_io_hps_io_emac1_inst_
output wire                                 hps_io_hps_io_emac1_inst_
input  wire                                 hps_io_hps_io_emac1_inst_
input  wire                                 hps_io_hps_io_emac1_inst_
input  wire                                 hps_io_hps_io_emac1_inst_
input  wire                                 hps_io_hps_io_emac1_inst_
inout  wire                                 hps_io_hps_io_qspi_inst_
inout  wire                                 hps_io_hps_io_qspi_inst_
inout  wire                                 hps_io_hps_io_qspi_inst_
```

```
inout   wire                                    hps_io_hps_io_qspi_inst_
output  wire                                    hps_io_hps_io_qspi_inst_
output  wire                                    hps_io_hps_io_qspi_inst_C
inout   wire                                    hps_io_hps_io_sdio_inst_C
inout   wire                                    hps_io_hps_io_sdio_inst_
inout   wire                                    hps_io_hps_io_sdio_inst_
output  wire                                    hps_io_hps_io_sdio_inst_C
inout   wire                                    hps_io_hps_io_sdio_inst_
inout   wire                                    hps_io_hps_io_sdio_inst_
inout   wire                                    hps_io_hps_io_usb1_inst_
inout   wire                                    hps_io_hps_io_usb1_inst_
inout   wire                                    hps_io_hps_io_usb1_inst_
inout   wire                                    hps_io_hps_io_usb1_inst_
inout   wire                                    hps_io_hps_io_usb1_inst_
inout   wire                                    hps_io_hps_io_usb1_inst_
inout   wire                                    hps_io_hps_io_usb1_inst_
inout   wire                                    hps_io_hps_io_usb1_inst_
input   wire                                    hps_io_hps_io_usb1_inst_C
output  wire                                    hps_io_hps_io_usb1_inst_S
input   wire                                    hps_io_hps_io_usb1_inst_
input   wire                                    hps_io_hps_io_usb1_inst_N
output  wire                                    hps_io_hps_io_spim0_inst_
output  wire                                    hps_io_hps_io_spim0_inst_
input   wire                                    hps_io_hps_io_spim0_inst_
output  wire                                    hps_io_hps_io_spim0_inst
output  wire                                    hps_io_hps_io_spim1_inst_
output  wire                                    hps_io_hps_io_spim1_inst_
input   wire                                    hps_io_hps_io_spim1_inst_
output  wire                                    hps_io_hps_io_spim1_inst
input   wire                                    hps_io_hps_io_uart0_inst
output  wire                                    hps_io_hps_io_uart0_inst
inout   wire                                    hps_io_hps_io_i2c1_inst_S
inout   wire                                    hps_io_hps_io_i2c1_inst_S
inout   wire                                    hps_io_hps_io_gpio_inst_G
//=======================================================
//  REG/WIRE declarations
//=======================================================

//    For Audio CODEC
wire                                            AUD_CTRL_CLK;       //

reg [31:0]                                      Cont;
wire                                            VGA_CTRL_CLK;
wire [9:0]                                      mVGA_R;
```

```
    wire [9:0]                                  mVGA_G;
    wire [9:0]                                  mVGA_B;
    wire [19:0]                                 mVGA_ADDR;
    wire                                        DLY_RST;

    //    For VGA Controller
    wire                                        mVGA_CLK;
    wire [9:0]                                  mRed;
    wire [9:0]                                  mGreen;
    wire [9:0]                                  mBlue;
    wire                                        VGA_Read; //       VGA d

    wire [9:0]                                  recon_VGA_R;
    wire [9:0]                                  recon_VGA_G;
    wire [9:0]                                  recon_VGA_B;

    //    For Down Sample
    wire [3:0]                                  Remain;
    wire [9:0]                                  Quotient;

    wire                                        AUD_MUTE;

    // Drive the LEDs with the switches
    assign LED = SW;

    // Make the FPGA reset cause an HPS reset
    reg [19:0]                                  hps_reset_counter = 20'
    reg                                         hps_fpga_reset_n = 0;

    always @(posedge OSC_50_B4A) begin
       if (hps_reset_counter == 20'h ffffff) hps_fpga_reset_n <= 1;
       hps_reset_counter <= hps_reset_counter + 1;
    end

    plazer u0 (
            .clk_clk                            (OSC_50_B4A),
//                  clk.clk
            .reset_reset_n                      (hps_fpga_reset_n),
//                  reset.reset_n
            .memory_mem_a                       (memory_mem_a),
//                  memory.mem_a
            .memory_mem_ba                      (memory_mem_ba),
//                  .mem_ba
```

```
                .memory_mem_ck                                  (memory_mem_ck),
//                      .mem_ck
                .memory_mem_ck_n                                (memory_mem_ck_n),
//                      .mem_ck_n
                .memory_mem_cke                                 (memory_mem_cke),
//                      .mem_cke
                .memory_mem_cs_n                                (memory_mem_cs_n),
//                      .mem_cs_n
                .memory_mem_ras_n                               (memory_mem_ras_n),
//                      .mem_ras_n
                .memory_mem_cas_n                               (memory_mem_cas_n),
//                      .mem_cas_n
                .memory_mem_we_n                                (memory_mem_we_n),
//                      .mem_we_n
                .memory_mem_reset_n                             (memory_mem_reset_n),
//                      .mem_reset_n
                .memory_mem_dq                                  (memory_mem_dq),
//                      .mem_dq
                .memory_mem_dqs                                 (memory_mem_dqs),
//                      .mem_dqs
                .memory_mem_dqs_n                               (memory_mem_dqs_n),
//                      .mem_dqs_n
                .memory_mem_odt                                 (memory_mem_odt),
//                      .mem_odt
                .memory_mem_dm                                  (memory_mem_dm),
//                      .mem_dm
                .memory_oct_rzqin                               (memory_oct_rzqin),
//                      .oct_rzqin
                .hps_io_hps_io_emac1_inst_TX_CLK                        (hps_io_hps_io_emac1
.hps_0_hps_io.hps_io_emac1_inst_TX_CLK
                .hps_io_hps_io_emac1_inst_TXD0                  (hps_io_hps_io_emac1
//                      .hps_io_emac1_inst_TXD0
                .hps_io_hps_io_emac1_inst_TXD1                  (hps_io_hps_io_emac1
//                      .hps_io_emac1_inst_TXD1
                .hps_io_hps_io_emac1_inst_TXD2                  (hps_io_hps_io_emac1
//                      .hps_io_emac1_inst_TXD2
                .hps_io_hps_io_emac1_inst_TXD3                  (hps_io_hps_io_emac1
//                      .hps_io_emac1_inst_TXD3
                .hps_io_hps_io_emac1_inst_RXD0                  (hps_io_hps_io_emac1
//                      .hps_io_emac1_inst_RXD0
                .hps_io_hps_io_emac1_inst_MDIO                  (hps_io_hps_io_emac1
//                      .hps_io_emac1_inst_MDIO
                .hps_io_hps_io_emac1_inst_MDC                   (hps_io_hps_io_emac1
//                      .hps_io_emac1_inst_MDC
```

```
                .hps_io_hps_io_emac1_inst_RX_CTL                (hps_io_hps_io_emac1
.hps_io_emac1_inst_RX_CTL
                .hps_io_hps_io_emac1_inst_TX_CTL                (hps_io_hps_io_emac1
.hps_io_emac1_inst_TX_CTL
                .hps_io_hps_io_emac1_inst_RX_CLK                (hps_io_hps_io_emac1
.hps_io_emac1_inst_RX_CLK
                .hps_io_hps_io_emac1_inst_RXD1                  (hps_io_hps_io_emac1
//                      .hps_io_emac1_inst_RXD1
                .hps_io_hps_io_emac1_inst_RXD2                  (hps_io_hps_io_emac1
//                      .hps_io_emac1_inst_RXD2
                .hps_io_hps_io_emac1_inst_RXD3                  (hps_io_hps_io_emac1
//                      .hps_io_emac1_inst_RXD3
                .hps_io_hps_io_qspi_inst_IO0                    (hps_io_hps_io_qspi_
//                      .hps_io_qspi_inst_IO0
                .hps_io_hps_io_qspi_inst_IO1                    (hps_io_hps_io_qspi_
//                      .hps_io_qspi_inst_IO1
                .hps_io_hps_io_qspi_inst_IO2                    (hps_io_hps_io_qspi_
//                      .hps_io_qspi_inst_IO2
                .hps_io_hps_io_qspi_inst_IO3                    (hps_io_hps_io_qspi_
//                      .hps_io_qspi_inst_IO3
                .hps_io_hps_io_qspi_inst_SS0                    (hps_io_hps_io_qspi_
//                      .hps_io_qspi_inst_SS0
                .hps_io_hps_io_qspi_inst_CLK                    (hps_io_hps_io_qspi_
//                      .hps_io_qspi_inst_CLK
                .hps_io_hps_io_sdio_inst_CMD                    (hps_io_hps_io_sdio_i
//                      .hps_io_sdio_inst_CMD
                .hps_io_hps_io_sdio_inst_D0                     (hps_io_hps_io_sdio_
//                      .hps_io_sdio_inst_D0
                .hps_io_hps_io_sdio_inst_D1                     (hps_io_hps_io_sdio_
//                      .hps_io_sdio_inst_D1
                .hps_io_hps_io_sdio_inst_CLK                    (hps_io_hps_io_sdio_
//                      .hps_io_sdio_inst_CLK
                .hps_io_hps_io_sdio_inst_D2                     (hps_io_hps_io_sdio_
//                      .hps_io_sdio_inst_D2
                .hps_io_hps_io_sdio_inst_D3                     (hps_io_hps_io_sdio_
//                      .hps_io_sdio_inst_D3
                .hps_io_hps_io_usb1_inst_D0                     (hps_io_hps_io_usb1_
//                      .hps_io_usb1_inst_D0
                .hps_io_hps_io_usb1_inst_D1                     (hps_io_hps_io_usb1_
//                      .hps_io_usb1_inst_D1
                .hps_io_hps_io_usb1_inst_D2                     (hps_io_hps_io_usb1_
//                      .hps_io_usb1_inst_D2
                .hps_io_hps_io_usb1_inst_D3                     (hps_io_hps_io_usb1_
//                      .hps_io_usb1_inst_D3
```

```
        .hps_io_hps_io_usb1_inst_D4                (hps_io_hps_io_usb1
//              .hps_io_usb1_inst_D4
        .hps_io_hps_io_usb1_inst_D5                (hps_io_hps_io_usb1
//              .hps_io_usb1_inst_D5
        .hps_io_hps_io_usb1_inst_D6                (hps_io_hps_io_usb1
//              .hps_io_usb1_inst_D6
        .hps_io_hps_io_usb1_inst_D7                (hps_io_hps_io_usb1
//              .hps_io_usb1_inst_D7
        .hps_io_hps_io_usb1_inst_CLK               (hps_io_hps_io_usb1_
//              .hps_io_usb1_inst_CLK
        .hps_io_hps_io_usb1_inst_STP               (hps_io_hps_io_usb1_
//              .hps_io_usb1_inst_STP
        .hps_io_hps_io_usb1_inst_DIR               (hps_io_hps_io_usb1_
//              .hps_io_usb1_inst_DIR
        .hps_io_hps_io_usb1_inst_NXT               (hps_io_hps_io_usb1_
//              .hps_io_usb1_inst_NXT
        .hps_io_hps_io_spim0_inst_CLK              (hps_io_hps_io_spim0
//              .hps_io_spim0_inst_CLK
        .hps_io_hps_io_spim0_inst_MOSI             (hps_io_hps_io_spim0
//              .hps_io_spim0_inst_MOSI
        .hps_io_hps_io_spim0_inst_MISO             (hps_io_hps_io_spim0
//              .hps_io_spim0_inst_MISO
        .hps_io_hps_io_spim0_inst_SS0              (hps_io_hps_io_spim
//              .hps_io_spim0_inst_SS0
        .hps_io_hps_io_spim1_inst_CLK              (hps_io_hps_io_spim1
//              .hps_io_spim1_inst_CLK
        .hps_io_hps_io_spim1_inst_MOSI             (hps_io_hps_io_spim1
//              .hps_io_spim1_inst_MOSI
        .hps_io_hps_io_spim1_inst_MISO             (hps_io_hps_io_spim1
//              .hps_io_spim1_inst_MISO
        .hps_io_hps_io_spim1_inst_SS0              (hps_io_hps_io_spim
//              .hps_io_spim1_inst_SS0
        .hps_io_hps_io_uart0_inst_RX               (hps_io_hps_io_uart0
//              .hps_io_uart0_inst_RX
        .hps_io_hps_io_uart0_inst_TX               (hps_io_hps_io_uart0
//              .hps_io_uart0_inst_TX
        .hps_io_hps_io_i2c1_inst_SDA               (hps_io_hps_io_i2c1_
//              .hps_io_i2c1_inst_SDA
        .hps_io_hps_io_i2c1_inst_SCL               (hps_io_hps_io_i2c1_
//              .hps_io_i2c1_inst_SCL
    );

endmodule
```

A.7. plazer.qsys

```
<?xml version="1.0" encoding="UTF-8"?>
<system name="$${FILENAME}">
 <component
   name="$${FILENAME}"
   displayName="$${FILENAME}"
   version="1.0"
   description=""
   tags=""
   categories="System" />
 <parameter name="bonusData"><![CDATA[bonusData
{
   element $${FILENAME}
   {
   }
   element processor_module_0.avalon_slave_0
   {
      datum baseAddress
      {
         value = "0";
         type = "String";
      }
   }
   element clk_0
   {
      datum _sortIndex
      {
         value = "0";
         type = "int";
      }
   }
   element hps_0
   {
      datum _sortIndex
      {
         value = "1";
         type = "int";
      }
   }
   element master_0
   {
      datum _sortIndex
```

```
        {
            value = "2";
            type = "int";
        }
    }
    element processor_module_0
    {
        datum _sortIndex
        {
            value = "3";
            type = "int";
        }
    }
}
]]></parameter>
 <parameter name="clockCrossingAdapter" value="HANDSHAKE" />
 <parameter name="device" value="5CSXFC6D6F31C8ES" />
 <parameter name="deviceFamily" value="Cyclone_V" />
 <parameter name="deviceSpeedGrade" value="8_H6" />
 <parameter name="fabricMode" value="QSYS" />
 <parameter name="generateLegacySim" value="false" />
 <parameter name="generationId" value="0" />
 <parameter name="globalResetBus" value="false" />
 <parameter name="hdlLanguage" value="VERILOG" />
 <parameter name="hideFromIPCatalog" value="false" />
 <parameter name="maxAdditionalLatency" value="1" />
 <parameter name="projectName" value="plazer.qpf" />
 <parameter name="sopcBorderPoints" value="false" />
 <parameter name="systemHash" value="0" />
 <parameter name="testBenchDutName" value="" />
 <parameter name="timeStamp" value="0" />
 <parameter name="useTestBenchNamingPattern" value="false" />
 <instanceScript></instanceScript>
 <interface name="clk" internal="clk_0.clk_in" type="clock" dir="end" />
 <interface name="reset" internal="clk_0.clk_in_reset" type="reset" dir="end"
 <interface name="memory" internal="hps_0.memory" type="conduit" dir="end" />
 <interface name="hps_io" internal="hps_0.hps_io" type="conduit" dir="end" />
 <module kind="clock_source" version="14.0" enabled="1" name="clk_0">
  <parameter name="clockFrequency" value="500000000" />
  <parameter name="clockFrequencyKnown" value="true" />
  <parameter name="inputClockFrequency" value="0" />
  <parameter name="resetSynchronousEdges" value="NONE" />
 </module>
 <module kind="altera_hps" version="14.0" enabled="1" name="hps_0">
```

```
<parameter name="MEM_VENDOR" value="Micron" />
<parameter name="MEM_FORMAT" value="DISCRETE" />
<parameter name="RDIMM_CONFIG" value="0000000000000000" />
<parameter name="LRDIMM_EXTENDED_CONFIG">0x0000000000000000</parameter>
<parameter name="DISCRETE_FLY_BY" value="true" />
<parameter name="DEVICE_DEPTH" value="1" />
<parameter name="DEVICE_WIDTH" value="1" />
<parameter name="MEM_MIRROR_ADDRESSING" value="0" />
<parameter name="MEM_CLK_FREQ_MAX" value="800.0" />
<parameter name="MEM_ROW_ADDR_WIDTH" value="15" />
<parameter name="MEM_COL_ADDR_WIDTH" value="10" />
<parameter name="MEM_DQ_WIDTH" value="32" />
<parameter name="MEM_DQ_PER_DQS" value="8" />
<parameter name="MEM_BANKADDR_WIDTH" value="3" />
<parameter name="MEM_IF_DM_PINS_EN" value="true" />
<parameter name="MEM_IF_DQSN_EN" value="true" />
<parameter name="MEM_NUMBER_OF_DIMMS" value="1" />
<parameter name="MEM_NUMBER_OF_RANKS_PER_DIMM" value="1" />
<parameter name="MEM_NUMBER_OF_RANKS_PER_DEVICE" value="1" />
<parameter name="MEM_RANK_MULTIPLICATION_FACTOR" value="1" />
<parameter name="MEM_CK_WIDTH" value="1" />
<parameter name="MEM_CS_WIDTH" value="1" />
<parameter name="MEM_CLK_EN_WIDTH" value="1" />
<parameter name="ALTMEMPHY_COMPATIBLE_MODE" value="false" />
<parameter name="NEXTGEN" value="true" />
<parameter name="MEM_IF_BOARD_BASE_DELAY" value="10" />
<parameter name="MEM_IF_SIM_VALID_WINDOW" value="0" />
<parameter name="MEM_GUARANTEED_WRITE_INIT" value="false" />
<parameter name="MEM_VERBOSE" value="true" />
<parameter name="PINGPONGPHY_EN" value="false" />
<parameter name="DUPLICATE_AC" value="false" />
<parameter name="REFRESH_BURST_VALIDATION" value="false" />
<parameter name="MEM_BL" value="OTF" />
<parameter name="MEM_BT" value="Sequential" />
<parameter name="MEM_ASR" value="Manual" />
<parameter name="MEM_SRT" value="Normal" />
<parameter name="MEM_PD" value="DLL␣off" />
<parameter name="MEM_DRV_STR" value="RZQ/7" />
<parameter name="MEM_DLL_EN" value="true" />
<parameter name="MEM_RTT_NOM" value="RZQ/4" />
<parameter name="MEM_RTT_WR" value="RZQ/4" />
<parameter name="MEM_WTCL" value="8" />
<parameter name="MEM_ATCL" value="Disabled" />
<parameter name="MEM_TCL" value="11" />
```

```
<parameter name="MEM_AUTO_LEVELING_MODE" value="true" />
<parameter name="MEM_USER_LEVELING_MODE" value="Leveling" />
<parameter name="MEM_INIT_EN" value="false" />
<parameter name="MEM_INIT_FILE" value="" />
<parameter name="DAT_DATA_WIDTH" value="32" />
<parameter name="TIMING_TIS" value="180" />
<parameter name="TIMING_TIH" value="140" />
<parameter name="TIMING_TDS" value="30" />
<parameter name="TIMING_TDH" value="65" />
<parameter name="TIMING_TDQSQ" value="125" />
<parameter name="TIMING_TQHS" value="300" />
<parameter name="TIMING_TQH" value="0.38" />
<parameter name="TIMING_TDQSCK" value="255" />
<parameter name="TIMING_TDQSCKDS" value="450" />
<parameter name="TIMING_TDQSCKDM" value="900" />
<parameter name="TIMING_TDQSCKDL" value="1200" />
<parameter name="TIMING_TDQSS" value="0.25" />
<parameter name="TIMING_TDQSH" value="0.35" />
<parameter name="TIMING_TQSH" value="0.4" />
<parameter name="TIMING_TDSH" value="0.2" />
<parameter name="TIMING_TDSS" value="0.2" />
<parameter name="MEM_TINIT_US" value="500" />
<parameter name="MEM_TMRD_CK" value="4" />
<parameter name="MEM_TRAS_NS" value="35.0" />
<parameter name="MEM_TRCD_NS" value="13.75" />
<parameter name="MEM_TRP_NS" value="13.75" />
<parameter name="MEM_TREFI_US" value="7.8" />
<parameter name="MEM_TRFC_NS" value="260.0" />
<parameter name="CFG_TCCD_NS" value="2.5" />
<parameter name="MEM_TWR_NS" value="15.0" />
<parameter name="MEM_TWTR" value="4" />
<parameter name="MEM_TFAW_NS" value="30.0" />
<parameter name="MEM_TRRD_NS" value="7.5" />
<parameter name="MEM_TRTP_NS" value="7.5" />
<parameter name="POWER_OF_TWO_BUS" value="false" />
<parameter name="SOPC_COMPAT_RESET" value="false" />
<parameter name="AVL_MAX_SIZE" value="4" />
<parameter name="BYTE_ENABLE" value="true" />
<parameter name="ENABLE_CTRL_AVALON_INTERFACE" value="true" />
<parameter name="CTL_DEEP_POWERDN_EN" value="false" />
<parameter name="CTL_SELF_REFRESH_EN" value="false" />
<parameter name="AUTO_POWERDN_EN" value="false" />
<parameter name="AUTO_PD_CYCLES" value="0" />
<parameter name="CTL_USR_REFRESH_EN" value="false" />
```

```
<parameter name="CTL_AUTOPCH_EN" value="false" />
<parameter name="CTL_ZQCAL_EN" value="false" />
<parameter name="ADDR_ORDER" value="0" />
<parameter name="CTL_LOOK_AHEAD_DEPTH" value="4" />
<parameter name="CONTROLLER_LATENCY" value="5" />
<parameter name="CFG_REORDER_DATA" value="true" />
<parameter name="STARVE_LIMIT" value="10" />
<parameter name="CTL_CSR_ENABLED" value="false" />
<parameter name="CTL_CSR_CONNECTION" value="INTERNAL_JTAG" />
<parameter name="CTL_ECC_ENABLED" value="false" />
<parameter name="CTL_HRB_ENABLED" value="false" />
<parameter name="CTL_ECC_AUTO_CORRECTION_ENABLED" value="false" />
<parameter name="MULTICAST_EN" value="false" />
<parameter name="CTL_DYNAMIC_BANK_ALLOCATION" value="false" />
<parameter name="CTL_DYNAMIC_BANK_NUM" value="4" />
<parameter name="DEBUG_MODE" value="false" />
<parameter name="ENABLE_BURST_MERGE" value="false" />
<parameter name="CTL_ENABLE_BURST_INTERRUPT" value="false" />
<parameter name="CTL_ENABLE_BURST_TERMINATE" value="false" />
<parameter name="LOCAL_ID_WIDTH" value="8" />
<parameter name="WRBUFFER_ADDR_WIDTH" value="6" />
<parameter name="MAX_PENDING_WR_CMD" value="16" />
<parameter name="MAX_PENDING_RD_CMD" value="32" />
<parameter name="USE_MM_ADAPTOR" value="true" />
<parameter name="USE_AXI_ADAPTOR" value="false" />
<parameter name="HCX_COMPAT_MODE" value="false" />
<parameter name="CTL_CMD_QUEUE_DEPTH" value="8" />
<parameter name="CTL_CSR_READ_ONLY" value="1" />
<parameter name="CFG_DATA_REORDERING_TYPE" value="INTER_BANK" />
<parameter name="NUM_OF_PORTS" value="1" />
<parameter name="ENABLE_BONDING" value="false" />
<parameter name="ENABLE_USER_ECC" value="false" />
<parameter name="AVL_DATA_WIDTH_PORT" value="32,32,32,32,32,32" />
<parameter name="PRIORITY_PORT" value="1,1,1,1,1,1" />
<parameter name="WEIGHT_PORT" value="0,0,0,0,0,0" />
<parameter name="CPORT_TYPE_PORT">Bidirectional, Bidirectional, Bidirectional
<parameter name="ENABLE_EMIT_BFM_MASTER" value="false" />
<parameter name="FORCE_SEQUENCER_TCL_DEBUG_MODE" value="false" />
<parameter name="ENABLE_SEQUENCER_MARGINING_ON_BY_DEFAULT" value="false" />
<parameter name="REF_CLK_FREQ" value="25.0" />
<parameter name="REF_CLK_FREQ_PARAM_VALID" value="false" />
<parameter name="REF_CLK_FREQ_MIN_PARAM" value="0.0" />
<parameter name="REF_CLK_FREQ_MAX_PARAM" value="0.0" />
<parameter name="PLL_DR_CLK_FREQ_PARAM" value="0.0" />
```

```
<parameter name="PLL_DR_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_DR_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_DR_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_DR_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_DR_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_MEM_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_MEM_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_MEM_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_MEM_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_MEM_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_MEM_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_AFI_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_AFI_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_AFI_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_AFI_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_AFI_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_AFI_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_WRITE_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_WRITE_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_WRITE_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_WRITE_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_WRITE_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_WRITE_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_ADDR_CMD_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_ADDR_CMD_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_ADDR_CMD_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_ADDR_CMD_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_ADDR_CMD_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_ADDR_CMD_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_AFI_HALF_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_AFI_HALF_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_AFI_HALF_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_AFI_HALF_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_AFI_HALF_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_AFI_HALF_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_NIOS_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_NIOS_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_NIOS_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_NIOS_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_NIOS_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_NIOS_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_CONFIG_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_CONFIG_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_CONFIG_CLK_PHASE_PS_PARAM" value="0" />
```

```
<parameter name="PLL_CONFIG_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_CONFIG_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_CONFIG_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_P2C_READ_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_P2C_READ_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_P2C_READ_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_P2C_READ_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_P2C_READ_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_P2C_READ_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_C2P_WRITE_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_C2P_WRITE_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_C2P_WRITE_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_C2P_WRITE_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_C2P_WRITE_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_C2P_WRITE_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_HR_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_HR_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_HR_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_HR_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_HR_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_HR_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_AFI_PHY_CLK_FREQ_PARAM" value="0.0" />
<parameter name="PLL_AFI_PHY_CLK_FREQ_SIM_STR_PARAM" value="" />
<parameter name="PLL_AFI_PHY_CLK_PHASE_PS_PARAM" value="0" />
<parameter name="PLL_AFI_PHY_CLK_PHASE_PS_SIM_STR_PARAM" value="" />
<parameter name="PLL_AFI_PHY_CLK_MULT_PARAM" value="0" />
<parameter name="PLL_AFI_PHY_CLK_DIV_PARAM" value="0" />
<parameter name="PLL_CLK_PARAM_VALID" value="false" />
<parameter name="ENABLE_EXTRA_REPORTING" value="false" />
<parameter name="NUM_EXTRA_REPORT_PATH" value="10" />
<parameter name="ENABLE_ISS_PROBES" value="false" />
<parameter name="CALIB_REG_WIDTH" value="8" />
<parameter name="USE_SEQUENCER_BFM" value="false" />
<parameter name="PLL_SHARING_MODE" value="None" />
<parameter name="NUM_PLL_SHARING_INTERFACES" value="1" />
<parameter name="EXPORT_AFI_HALF_CLK" value="false" />
<parameter name="ABSTRACT_REAL_COMPARE_TEST" value="false" />
<parameter name="INCLUDE_BOARD_DELAY_MODEL" value="false" />
<parameter name="INCLUDE_MULTIRANK_BOARD_DELAY_MODEL" value="false" />
<parameter name="USE_FAKE_PHY" value="false" />
<parameter name="FORCE_MAX_LATENCY_COUNT_WIDTH" value="0" />
<parameter name="ENABLE_NON_DESTRUCTIVE_CALIB" value="false" />
<parameter name="ENABLE_DELAY_CHAIN_WRITE" value="false" />
<parameter name="TRACKING_ERROR_TEST" value="false" />
```

```
<parameter name="TRACKING_WATCH_TEST" value="false" />
<parameter name="MARGIN_VARIATION_TEST" value="false" />
<parameter name="EXTRA_SETTINGS" value="" />
<parameter name="MEM_DEVICE" value="MISSING_MODEL" />
<parameter name="FORCE_SYNTHESIS_LANGUAGE" value="" />
<parameter name="FORCED_NUM_WRITE_FR_CYCLE_SHIFTS" value="0" />
<parameter name="SEQUENCER_TYPE" value="NIOS" />
<parameter name="ADVERTIZE_SEQUENCER_SW_BUILD_FILES" value="false" />
<parameter name="FORCED_NON_LDC_ADDR_CMD_MEM_CK_INVERT" value="false" />
<parameter name="PHY_ONLY" value="false" />
<parameter name="SEQ_MODE" value="0" />
<parameter name="ADVANCED_CK_PHASES" value="false" />
<parameter name="COMMAND_PHASE" value="0.0" />
<parameter name="MEM_CK_PHASE" value="0.0" />
<parameter name="P2C_READ_CLOCK_ADD_PHASE" value="0.0" />
<parameter name="C2P_WRITE_CLOCK_ADD_PHASE" value="0.0" />
<parameter name="ACV_PHY_CLK_ADD_FR_PHASE" value="0.0" />
<parameter name="MEM_VOLTAGE" value="1.5V_DDR3" />
<parameter name="PLL_LOCATION" value="Top_Bottom" />
<parameter name="SKIP_MEM_INIT" value="true" />
<parameter name="READ_DQ_DQS_CLOCK_SOURCE" value="INVERTED_DQS_BUS" />
<parameter name="DQ_INPUT_REG_USE_CLKN" value="false" />
<parameter name="DQS_DQSN_MODE" value="DIFFERENTIAL" />
<parameter name="AFI_DEBUG_INFO_WIDTH" value="32" />
<parameter name="CALIBRATION_MODE" value="Skip" />
<parameter name="NIOS_ROM_DATA_WIDTH" value="32" />
<parameter name="READ_FIFO_SIZE" value="8" />
<parameter name="PHY_CSR_ENABLED" value="false" />
<parameter name="PHY_CSR_CONNECTION" value="INTERNAL_JTAG" />
<parameter name="USER_DEBUG_LEVEL" value="1" />
<parameter name="TIMING_BOARD_DERATE_METHOD" value="AUTO" />
<parameter name="TIMING_BOARD_CK_CKN_SLEW_RATE" value="2.0" />
<parameter name="TIMING_BOARD_AC_SLEW_RATE" value="1.0" />
<parameter name="TIMING_BOARD_DQS_DQSN_SLEW_RATE" value="2.0" />
<parameter name="TIMING_BOARD_DQ_SLEW_RATE" value="1.0" />
<parameter name="TIMING_BOARD_TIS" value="0.0" />
<parameter name="TIMING_BOARD_TIH" value="0.0" />
<parameter name="TIMING_BOARD_TDS" value="0.0" />
<parameter name="TIMING_BOARD_TDH" value="0.0" />
<parameter name="TIMING_BOARD_ISI_METHOD" value="AUTO" />
<parameter name="TIMING_BOARD_AC_EYE_REDUCTION_SU" value="0.0" />
<parameter name="TIMING_BOARD_AC_EYE_REDUCTION_H" value="0.0" />
<parameter name="TIMING_BOARD_DQ_EYE_REDUCTION" value="0.0" />
<parameter name="TIMING_BOARD_DELTA_DQS_ARRIVAL_TIME" value="0.0" />
```

```
<parameter name="TIMING_BOARD_READ_DQ_EYE_REDUCTION" value="0.0" />
<parameter name="TIMING_BOARD_DELTA_READ_DQS_ARRIVAL_TIME" value="0.0" />
<parameter name="PACKAGE_DESKEW" value="false" />
<parameter name="AC_PACKAGE_DESKEW" value="false" />
<parameter name="TIMING_BOARD_MAX_CK_DELAY" value="0.03" />
<parameter name="TIMING_BOARD_MAX_DQS_DELAY" value="0.02" />
<parameter name="TIMING_BOARD_SKEW_CKDQS_DIMM_MIN" value="0.09" />
<parameter name="TIMING_BOARD_SKEW_CKDQS_DIMM_MAX" value="0.16" />
<parameter name="TIMING_BOARD_SKEW_BETWEEN_DIMMS" value="0.05" />
<parameter name="TIMING_BOARD_SKEW_WITHIN_DQS" value="0.01" />
<parameter name="TIMING_BOARD_SKEW_BETWEEN_DQS" value="0.08" />
<parameter name="TIMING_BOARD_DQ_TO_DQS_SKEW" value="0.0" />
<parameter name="TIMING_BOARD_AC_SKEW" value="0.03" />
<parameter name="TIMING_BOARD_AC_TO_CK_SKEW" value="0.0" />
<parameter name="RATE" value="Full" />
<parameter name="MEM_CLK_FREQ" value="400.0" />
<parameter name="USE_MEM_CLK_FREQ" value="false" />
<parameter name="FORCE_DQS_TRACKING" value="AUTO" />
<parameter name="FORCE_SHADOW_REGS" value="AUTO" />
<parameter name="MRS_MIRROR_PING_PONG_ATSO" value="false" />
<parameter name="SYS_INFO_DEVICE_FAMILY" value="Cyclone␣V" />
<parameter name="PARSE_FRIENDLY_DEVICE_FAMILY_PARAM_VALID" value="false" />
<parameter name="PARSE_FRIENDLY_DEVICE_FAMILY_PARAM" value="" />
<parameter name="DEVICE_FAMILY_PARAM" value="" />
<parameter name="SPEED_GRADE" value="7" />
<parameter name="IS_ES_DEVICE" value="false" />
<parameter name="DISABLE_CHILD_MESSAGING" value="false" />
<parameter name="HARD_EMIF" value="true" />
<parameter name="HHP_HPS" value="true" />
<parameter name="HHP_HPS_VERIFICATION" value="false" />
<parameter name="HHP_HPS_SIMULATION" value="false" />
<parameter name="HPS_PROTOCOL" value="DDR3" />
<parameter name="CUT_NEW_FAMILY_TIMING" value="true" />
<parameter name="ENABLE_EXPORT_SEQ_DEBUG_BRIDGE" value="false" />
<parameter name="CORE_DEBUG_CONNECTION" value="EXPORT" />
<parameter name="ADD_EXTERNAL_SEQ_DEBUG_NIOS" value="false" />
<parameter name="ED_EXPORT_SEQ_DEBUG" value="false" />
<parameter name="ADD_EFFICIENCY_MONITOR" value="false" />
<parameter name="ENABLE_ABS_RAM_MEM_INIT" value="false" />
<parameter name="ABS_RAM_MEM_INIT_FILENAME" value="meminit" />
<parameter name="DLL_SHARING_MODE" value="None" />
<parameter name="NUM_DLL_SHARING_INTERFACES" value="1" />
<parameter name="OCT_SHARING_MODE" value="None" />
<parameter name="NUM_OCT_SHARING_INTERFACES" value="1" />
```

```
<parameter name="show_advanced_parameters" value="false" />
<parameter name="configure_advanced_parameters" value="false" />
<parameter name="customize_device_pll_info" value="false" />
<parameter name="device_pll_info_manual">{320000000 1600000000} {320000000
<parameter name="show_debug_info_as_warning_msg" value="false" />
<parameter name="show_warning_as_error_msg" value="false" />
<parameter name="eosc1_clk_mhz" value="25.0" />
<parameter name="eosc2_clk_mhz" value="25.0" />
<parameter name="F2SCLK_SDRAMCLK_Enable" value="false" />
<parameter name="F2SCLK_PERIPHCLK_Enable" value="false" />
<parameter name="F2SCLK_SDRAMCLK_FREQ" value="0" />
<parameter name="F2SCLK_PERIPHCLK_FREQ" value="0" />
<parameter name="periph_pll_source" value="0" />
<parameter name="sdmmc_clk_source" value="2" />
<parameter name="nand_clk_source" value="2" />
<parameter name="qspi_clk_source" value="1" />
<parameter name="l4_mp_clk_source" value="1" />
<parameter name="l4_sp_clk_source" value="1" />
<parameter name="use_default_mpu_clk" value="true" />
<parameter name="desired_mpu_clk_mhz" value="800.0" />
<parameter name="l3_mp_clk_div" value="1" />
<parameter name="l3_sp_clk_div" value="1" />
<parameter name="dbctrl_stayosc1" value="true" />
<parameter name="dbg_at_clk_div" value="0" />
<parameter name="dbg_clk_div" value="1" />
<parameter name="dbg_trace_clk_div" value="0" />
<parameter name="desired_l4_mp_clk_mhz" value="100.0" />
<parameter name="desired_l4_sp_clk_mhz" value="100.0" />
<parameter name="desired_cfg_clk_mhz" value="100.0" />
<parameter name="desired_sdmmc_clk_mhz" value="200.0" />
<parameter name="desired_nand_clk_mhz" value="12.5" />
<parameter name="desired_qspi_clk_mhz" value="400.0" />
<parameter name="desired_emac0_clk_mhz" value="250.0" />
<parameter name="desired_emac1_clk_mhz" value="250.0" />
<parameter name="desired_usb_mp_clk_mhz" value="200.0" />
<parameter name="desired_spi_m_clk_mhz" value="200.0" />
<parameter name="desired_can0_clk_mhz" value="100.0" />
<parameter name="desired_can1_clk_mhz" value="100.0" />
<parameter name="desired_gpio_db_clk_hz" value="32000" />
<parameter name="S2FCLK_USER0CLK_Enable" value="false" />
<parameter name="S2FCLK_USER1CLK_Enable" value="false" />
<parameter name="S2FCLK_USER2CLK_Enable" value="false" />
<parameter name="S2FCLK_USER1CLK_FREQ" value="100.0" />
<parameter name="S2FCLK_USER2CLK_FREQ" value="100.0" />
```

```
<parameter name="main_pll_m" value="63" />
<parameter name="main_pll_n" value="0" />
<parameter name="main_pll_c3" value="3" />
<parameter name="main_pll_c4" value="3" />
<parameter name="main_pll_c5" value="15" />
<parameter name="periph_pll_m" value="79" />
<parameter name="periph_pll_n" value="1" />
<parameter name="periph_pll_c0" value="3" />
<parameter name="periph_pll_c1" value="3" />
<parameter name="periph_pll_c2" value="1" />
<parameter name="periph_pll_c3" value="19" />
<parameter name="periph_pll_c4" value="4" />
<parameter name="periph_pll_c5" value="9" />
<parameter name="usb_mp_clk_div" value="0" />
<parameter name="spi_m_clk_div" value="0" />
<parameter name="can0_clk_div" value="1" />
<parameter name="can1_clk_div" value="1" />
<parameter name="gpio_db_clk_div" value="6249" />
<parameter name="l4_mp_clk_div" value="1" />
<parameter name="l4_sp_clk_div" value="1" />
<parameter name="MPU_EVENTS_Enable" value="false" />
<parameter name="GP_Enable" value="false" />
<parameter name="DEBUGAPB_Enable" value="false" />
<parameter name="STM_Enable" value="false" />
<parameter name="CTI_Enable" value="false" />
<parameter name="TPIUFPGA_Enable" value="false" />
<parameter name="BOOTFROMFPGA_Enable" value="false" />
<parameter name="TEST_Enable" value="false" />
<parameter name="HLGPI_Enable" value="false" />
<parameter name="BSEL_EN" value="false" />
<parameter name="BSEL" value="1" />
<parameter name="CSEL_EN" value="false" />
<parameter name="CSEL" value="0" />
<parameter name="F2S_Width" value="2" />
<parameter name="S2F_Width" value="3" />
<parameter name="LWH2F_Enable" value="true" />
<parameter name="F2SDRAM_Type" value="" />
<parameter name="F2SDRAM_Width" value="" />
<parameter name="BONDING_OUT_ENABLED" value="false" />
<parameter name="S2FCLK_COLDRST_Enable" value="false" />
<parameter name="S2FCLK_PENDINGRST_Enable" value="false" />
<parameter name="F2SCLK_DBGRST_Enable" value="false" />
<parameter name="F2SCLK_WARMRST_Enable" value="false" />
<parameter name="F2SCLK_COLDRST_Enable" value="false" />
```

```
<parameter name="DMA_Enable">No,No,No,No,No,No,No,No</parameter>
<parameter name="F2SINTERRUPT_Enable" value="true" />
<parameter name="S2FINTERRUPT_CAN_Enable" value="false" />
<parameter name="S2FINTERRUPT_CLOCKPERIPHERAL_Enable" value="false" />
<parameter name="S2FINTERRUPT_CTI_Enable" value="false" />
<parameter name="S2FINTERRUPT_DMA_Enable" value="false" />
<parameter name="S2FINTERRUPT_EMAC_Enable" value="false" />
<parameter name="S2FINTERRUPT_FPGAMANAGER_Enable" value="false" />
<parameter name="S2FINTERRUPT_GPIO_Enable" value="false" />
<parameter name="S2FINTERRUPT_I2CEMAC_Enable" value="false" />
<parameter name="S2FINTERRUPT_I2CPERIPHERAL_Enable" value="false" />
<parameter name="S2FINTERRUPT_L4TIMER_Enable" value="false" />
<parameter name="S2FINTERRUPT_NAND_Enable" value="false" />
<parameter name="S2FINTERRUPT_OSCTIMER_Enable" value="false" />
<parameter name="S2FINTERRUPT_QSPI_Enable" value="false" />
<parameter name="S2FINTERRUPT_SDMMC_Enable" value="false" />
<parameter name="S2FINTERRUPT_SPIMASTER_Enable" value="false" />
<parameter name="S2FINTERRUPT_SPISLAVE_Enable" value="false" />
<parameter name="S2FINTERRUPT_UART_Enable" value="false" />
<parameter name="S2FINTERRUPT_USB_Enable" value="false" />
<parameter name="S2FINTERRUPT_WATCHDOG_Enable" value="false" />
<parameter name="EMAC0_PinMuxing" value="Unused" />
<parameter name="EMAC0_Mode" value="N/A" />
<parameter name="EMAC1_PinMuxing" value="HPS␣I/O␣Set␣0" />
<parameter name="EMAC1_Mode" value="RGMII" />
<parameter name="NAND_PinMuxing" value="Unused" />
<parameter name="NAND_Mode" value="N/A" />
<parameter name="QSPI_PinMuxing" value="HPS␣I/O␣Set␣0" />
<parameter name="QSPI_Mode" value="1␣SS" />
<parameter name="SDIO_PinMuxing" value="HPS␣I/O␣Set␣0" />
<parameter name="SDIO_Mode" value="4-bit␣Data" />
<parameter name="USB0_PinMuxing" value="Unused" />
<parameter name="USB0_Mode" value="N/A" />
<parameter name="USB1_PinMuxing" value="HPS␣I/O␣Set␣0" />
<parameter name="USB1_Mode" value="SDR" />
<parameter name="SPIM0_PinMuxing" value="HPS␣I/O␣Set␣0" />
<parameter name="SPIM0_Mode" value="Single␣Slave␣Select" />
<parameter name="SPIM1_PinMuxing" value="HPS␣I/O␣Set␣0" />
<parameter name="SPIM1_Mode" value="Single␣Slave␣Select" />
<parameter name="SPIS0_PinMuxing" value="Unused" />
<parameter name="SPIS0_Mode" value="N/A" />
<parameter name="SPIS1_PinMuxing" value="Unused" />
<parameter name="SPIS1_Mode" value="N/A" />
<parameter name="UART0_PinMuxing" value="HPS␣I/O␣Set␣0" />
```

```
<parameter name="UART0_Mode" value="No Flow Control" />
<parameter name="UART1_PinMuxing" value="Unused" />
<parameter name="UART1_Mode" value="N/A" />
<parameter name="I2C0_PinMuxing" value="Unused" />
<parameter name="I2C0_Mode" value="N/A" />
<parameter name="I2C1_PinMuxing" value="HPS I/O Set 0" />
<parameter name="I2C1_Mode" value="I2C" />
<parameter name="I2C2_PinMuxing" value="Unused" />
<parameter name="I2C2_Mode" value="N/A" />
<parameter name="I2C3_PinMuxing" value="Unused" />
<parameter name="I2C3_Mode" value="N/A" />
<parameter name="CAN0_PinMuxing" value="Unused" />
<parameter name="CAN0_Mode" value="N/A" />
<parameter name="CAN1_PinMuxing" value="Unused" />
<parameter name="CAN1_Mode" value="N/A" />
<parameter name="TRACE_PinMuxing" value="Unused" />
<parameter name="TRACE_Mode" value="N/A" />
<parameter name="GPIO_Enable">No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,N
<parameter name="LOANIO_Enable">No,No,No,No,No,No,No,No,No,No,No,No,No,No,No,N
<parameter name="F2H_AXI_CLOCK_FREQ" value="500000000" />
<parameter name="H2F_AXI_CLOCK_FREQ" value="500000000" />
<parameter name="H2F_LW_AXI_CLOCK_FREQ" value="500000000" />
<parameter name="F2H_SDRAM0_CLOCK_FREQ" value="100" />
<parameter name="F2H_SDRAM1_CLOCK_FREQ" value="100" />
<parameter name="F2H_SDRAM2_CLOCK_FREQ" value="100" />
<parameter name="F2H_SDRAM3_CLOCK_FREQ" value="100" />
<parameter name="F2H_SDRAM4_CLOCK_FREQ" value="100" />
<parameter name="F2H_SDRAM5_CLOCK_FREQ" value="100" />
<parameter name="H2F_CTI_CLOCK_FREQ" value="100" />
<parameter name="H2F_TPIU_CLOCK_IN_FREQ" value="100" />
<parameter name="H2F_DEBUG_APB_CLOCK_FREQ" value="100" />
<parameter
    name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_EMAC_PTP_REF_CLOCK"
    value="100" />
<parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_EMAC0_RX_CLK_IN" value="10
<parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_EMAC0_TX_CLK_IN" value="10
<parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_EMAC0_MD_CLK" value="100
<parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_EMAC0_GTX_CLK" value="100
<parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_EMAC1_RX_CLK_IN" value="10
<parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_EMAC1_TX_CLK_IN" value="10
<parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_EMAC1_MD_CLK" value="100
<parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_EMAC1_GTX_CLK" value="100
<parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_QSPI_SCLK_OUT" value="100
<parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_SDIO_CLK_IN" value="100"
```

```
 <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_SDIO_CCLK" value="100" />
 <parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_USB0_CLK_IN" value="100" />
 <parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_USB1_CLK_IN" value="100" />
 <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_SPIM0_SCLK_OUT" value="10
 <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_SPIM1_SCLK_OUT" value="10
 <parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_SPIS0_SCLK_IN" value="100"
 <parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_SPIS1_SCLK_IN" value="100"
 <parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_I2C0_SCL_IN" value="100" />
 <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_I2C0_CLK" value="100" />
 <parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_I2C1_SCL_IN" value="100" />
 <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_I2C1_CLK" value="100" />
 <parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_I2C2_SCL_IN" value="100" />
 <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_I2C2_CLK" value="100" />
 <parameter name="FPGA_PERIPHERAL_INPUT_CLOCK_FREQ_I2C3_SCL_IN" value="100" />
 <parameter name="FPGA_PERIPHERAL_OUTPUT_CLOCK_FREQ_I2C3_CLK" value="100" />
 <parameter name="device_name" value="5CSXFC6D6F31C8ES" />
 <parameter
    name="quartus_ini_hps_ip_enable_all_peripheral_fpga_interfaces"
    value="false" />
 <parameter
    name="quartus_ini_hps_ip_enable_emac0_peripheral_fpga_interface"
    value="false" />
 <parameter name="quartus_ini_hps_ip_enable_test_interface" value="false" />
 <parameter name="quartus_ini_hps_ip_fast_f2sdram_sim_model" value="false" />
 <parameter name="quartus_ini_hps_ip_suppress_sdram_synth" value="false" />
 <parameter
    name="quartus_ini_hps_ip_enable_low_speed_serial_fpga_interfaces"
    value="false" />
 <parameter name="quartus_ini_hps_ip_enable_bsel_csel" value="false" />
 <parameter name="quartus_ini_hps_ip_f2sdram_bonding_out" value="false" />
</module>
<module
  kind="altera_jtag_avalon_master"
  version="14.0"
  enabled="1"
  name="master_0">
 <parameter name="USE_PLI" value="0" />
 <parameter name="PLI_PORT" value="50000" />
 <parameter name="COMPONENT_CLOCK" value="0" />
 <parameter name="FAST_VER" value="0" />
 <parameter name="FIFO_DEPTHS" value="2" />
 <parameter name="AUTO_DEVICE_FAMILY" value="Cyclone_V" />
 <parameter name="AUTO_DEVICE" value="5CSXFC6D6F31C8ES" />
</module>
```

```
<module
   kind="processor_module"
   version="1.0"
   enabled="1"
   name="processor_module_0">
  <parameter name="AUTO_CLOCK_CLOCK_RATE" value="500000000" />
</module>
<connection
   kind="clock"
   version="14.0"
   start="clk_0.clk"
   end="hps_0.h2f_axi_clock" />
<connection
   kind="clock"
   version="14.0"
   start="clk_0.clk"
   end="hps_0.f2h_axi_clock" />
<connection
   kind="clock"
   version="14.0"
   start="clk_0.clk"
   end="hps_0.h2f_lw_axi_clock" />
<connection kind="clock" version="14.0" start="clk_0.clk" end="master_0.clk"
<connection
   kind="reset"
   version="14.0"
   start="clk_0.clk_reset"
   end="master_0.clk_reset" />
<connection
   kind="clock"
   version="14.0"
   start="clk_0.clk"
   end="processor_module_0.clock" />
<connection
   kind="avalon"
   version="14.0"
   start="hps_0.h2f_lw_axi_master"
   end="processor_module_0.avalon_slave_0">
  <parameter name="arbitrationPriority" value="1" />
  <parameter name="baseAddress" value="0x0000" />
  <parameter name="defaultConnection" value="false" />
</connection>
<connection
   kind="avalon"
```

```
    version="14.0"
    start="master_0.master"
    end="processor_module_0.avalon_slave_0">
   <parameter name="arbitrationPriority" value="1" />
   <parameter name="baseAddress" value="0x0000" />
   <parameter name="defaultConnection" value="false" />
  </connection>
  <connection
    kind="reset"
    version="14.0"
    start="clk_0.clk_reset"
    end="processor_module_0.reset_sink" />
  <interconnectRequirement for="$system" name="qsys_mm.clockCrossingAdapter" v
  <interconnectRequirement for="$system" name="qsys_mm.maxAdditionalLatency" v
  <interconnectRequirement for="$system" name="qsys_mm.insertDefaultSlave" valu
</system>
```

A.8. socfpga.dts

```
/*
 *  Copyright (C) 2012 Altera Corporation <www.altera.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 * dtc -O dtb -o socfpga.dtb socfpga.dts
 */

/dts-v1/;
/include/ "socfpga.dtsi"

/ {
        model = "Altera SOCFPGA Cyclone V";
        compatible = "altr,socfpga-cyclone5", "altr,socfpga";
```

```
chosen {
        bootargs = "console=ttyS0,57600";
};

memory {
        name = "memory";
        device_type = "memory";
        reg = <0x0 0x40000000>; /* 1 GB */
};

aliases {
        /* this allow the ethaddr uboot environmnet variable contents
         * to be added to the gmac1 device tree blob.
         */
        ethernet0 = &gmac1;
};

soc {
        clkmgr@ffd04000 {
                clocks {
                        osc1 {
                                clock-frequency = <25000000>;
                        };
                };
        };

        dcan0: d_can@ffc00000 {
                status = "disabled";
        };

        dcan1: d_can@ffc10000 {
                status = "disabled";
        };

        dwmmc0@ff704000 {
                num-slots = <1>;
                supports-highspeed;
                broken-cd;
                altr,dw-mshc-ciu-div = <4>;
                altr,dw-mshc-sdr-timing = <0 3>;

                slot@0 {
                        reg = <0>;
```

```
                bus-width = <4>;
        };
};

ethernet@ff700000 {
        status = "disabled";
};

ethernet@ff702000 {
        phy-mode = "rgmii";
        phy-addr = <0xffffffff>; /* probe for phy addr */
};

i2c1: i2c@ffc05000 {
        status = "disabled";
};

i2c2: i2c@ffc06000 {
        status = "disabled";
};

i2c3: i2c@ffc07000 {
        status = "disabled";
};

qspi: spi@ff705000 {
                compatible = "cadence,qspi";
                #address-cells = <1>;
                #size-cells = <0>;
                reg = <0xff705000 0x1000>,
                        <0xffa00000 0x1000>;
                interrupts = <0 151 4>;
                master-ref-clk = <400000000>;
                ext-decoder = <0>;  /* external decoder */
                num-chipselect = <4>;
                fifo-depth = <128>;
                bus-num = <2>;

                flash0: n25q00@0 {
                        #address-cells = <1>;
                        #size-cells = <1>;
                        compatible = "n25q00";
                        reg = <0>;      /* chip select */
                        spi-max-frequency = <100000000>;
```

```
                                page-size = <256>;
                                block-size = <16>; /* 2^16, 64KB */
                                quad = <1>;         /* 1-support quad
                                tshsl-ns = <200>;
                                tsd2d-ns = <255>;
                                tchsh-ns = <20>;
                                tslch-ns = <20>;

                                partition@0 {
                                        /* 8MB for raw data. */
                                        label = "Flash 0 Raw Data";
                                        reg = <0x0 0x800000>;
                                };
                                partition@800000 {
                                        /* 8MB for jffs2 data. */
                                        label = "Flash 0 jffs2 Filesys
                                        reg = <0x800000 0x800000>;
                                };
                        };

                };

sysmgr@ffd08000 {
        cpu1-start-addr = <0xffd080c4>;
};

timer0@ffc08000 {
        clock-frequency = <100000000>;
};

timer1@ffc09000 {
        clock-frequency = <100000000>;
};

timer2@ffd00000 {
        clock-frequency = <25000000>;
};

timer3@ffd01000 {
        clock-frequency = <25000000>;
};

serial0@ffc02000 {
        clock-frequency = <100000000>;
```

```
};

serial1@ffc03000 {
        clock-frequency = <100000000>;
};

usb0: usb@ffb00000 {
        status = "disabled";
};

usb1: usb@ffb40000 {
        ulpi-ddr = <0>;
};

i2c0: i2c@ffc04000 {
        speed-mode = <0>;
};

leds {
        compatible = "gpio-leds";
        hps0 {
                label = "hps_led0";
                gpios = <&gpio1 15 1>;
        };

        hps1 {
                label = "hps_led1";
                gpios = <&gpio1 14 1>;
        };

        hps2 {
                label = "hps_led2";
                gpios = <&gpio1 13 1>;
        };

        hps3 {
                label = "hps_led3";
                gpios = <&gpio1 12 1>;
        };
};

lightweight_bridge: bridge@0xff200000 {
        #address-cells = <1>;
        #size-cells = <1>;
```

```
                        ranges = < 0x0 0xff200000 0x200000 >;

                        compatible = "simple-bus";

                        plazer: plazer@0 {
                                compatible = "altr,plazer";
                                reg = <0x0 0x3c>;
                        };
                };
        };
};

&i2c0 {
        lcd: lcd@28 {
                compatible = "newhaven,nhd-0216k3z-nsw-bbw";
                reg = <0x28>;
                height = <2>;
                width = <16>;
                brightness = <8>;
        };

        eeprom@51 {
                compatible = "atmel,24c32";
                reg = <0x51>;
                pagesize = <32>;
        };

        rtc@68 {
                compatible = "dallas,ds1339";
                reg = <0x68>;
        };
};
```

A.9. fpga.h

```
#ifndef FPGA_H_
#define FPGA_H_

#define FPGA_BASE_ADDRESS   (0x00)

// end is 1 past the end, so we can use < comparison
#define DATA_START              (FPGA_BASE_ADDRESS + 0x00)
#define DATA_CONV_START         (DATA_START + 8)
#define DATA_CONV_END           (DATA_CONV_START + 32)
```

```
#define DATA_END                (DATA_START + 48)

#define FILTER_START            (DATA_END)
#define FILTER_END              (DATA_END + 8)

#define RESULT_LOC              (56) // FILTER_END)
#define READY_MASK              (0x00000001)
#define MAXVAL_MASK             (0xffff0000)
#define MAXVAL_OFFSET           (16)
#define MAXPOS_MASK             (0x0000ff00)
#define MAXPOS_OFFSET           (8)

#define PLAZER_SIZE             (60)
#define PLAZER_SIZE_32          (15)

#endif
```

A.10. plazer.h

```
#ifndef _PLAZER_DRIVER_H
#define _PLAZER_DRIVER_H

#include <linux/ioctl.h>
#include "fpga.h"

#define PLAZER_DIGITS 8

typedef struct {
    unsigned char   left_fill[DATA_CONV_START - DATA_START];
    unsigned char   data[DATA_CONV_END - DATA_CONV_START];
    unsigned char   right_fill[DATA_END - DATA_CONV_END];

    unsigned short  convmax;
    unsigned char   maxpos;
} plazer_arg_t;

typedef struct {
    unsigned char   conv[FILTER_END - FILTER_START];
} plazer_conv_t;

typedef struct {
    plazer_arg_t    data;
    plazer_conv_t   conv;
} plazer_mem_t;
```

```
#define PLAZER_MAGIC 'q'

/* ioctls and their arguments */
#define PLAZER_CONV_MAX      _IOWR(PLAZER_MAGIC, 1, plazer_arg_t *)
#define PLAZER_SET_CONV      _IOW (PLAZER_MAGIC, 2, plazer_conv_t *)
#define PLAZER_READ_MEMORY   _IOWR(PLAZER_MAGIC, 3, plazer_mem_t *)
#define PLAZER_RESET         _IOW (PLAZER_MAGIC, 4, plazer_mem_t *)
#define PLAZER_GET_BUFFER    _IOWR(PLAZER_MAGIC, 5, unsigned char *)

#endif
```

A.11. plazer.c

```
/*
 * Device driver for the plazer max-convolution accelerator
 *
 * A Platform device implemented using the misc subsystem
 *
 * Robert Ying, Xingzhou He, Peiqian Li, Minh Trang Nguyen
 * Columbia University
 *
 * References:
 * Linux source: Documentation/driver-model/platform.txt
 *               drivers/misc/arm-charlcd.c
 * http://www.linuxforu.com/tag/linux-device-drivers/
 * http://free-electrons.com/docs/
 *
 * "make" to build
 * insmod plazer.ko
 *
 * Check code style with
 * checkpatch.pl --file --no-tree plazer.c
 */

#include <linux/module.h>
#include <linux/init.h>
#include <linux/errno.h>
#include <linux/version.h>
#include <linux/kernel.h>
#include <linux/platform_device.h>
#include <linux/miscdevice.h>
#include <linux/slab.h>
#include <linux/io.h>
```

```c
#include <linux/of.h>
#include <linux/of_address.h>
#include <linux/fs.h>
#include <linux/uaccess.h>
#include "plazer.h"

#define DRIVER_NAME "plazer"

/*
 * Information about our device
 */
struct plazer_dev {
    struct resource res; /* Resource: our registers */
    void __iomem *virtbase; /* Where registers can be accessed in memory */
    u8 buffer[PLAZER_SIZE];
} dev;

void ioread32_rep_fixed(unsigned long offset, u8 * pr, size_t num) {
    u32 * ptr = (u32 *) pr;
    int i = 0;
    for (i = 0; i < num; ++i) {
        ptr[i] = ioread32(dev.virtbase + offset + i * 4);
    }
}

void iowrite32_rep_fixed(unsigned long offset, u8 * pr, size_t num) {
    u32 * ptr = (u32 *) pr;
    int i = 0;
    for (i = 0; i < num; ++i) {
        iowrite32(ptr[i], dev.virtbase + offset + i * 4);
    }
}

static long plazer_conv_max(plazer_arg_t * user_arg_ptr) {
    plazer_arg_t arg;
    if (copy_from_user(&arg, user_arg_ptr, sizeof(plazer_arg_t))) {
        return -EACCES;
    }

    memcpy(dev.buffer + DATA_START, &arg, DATA_END - DATA_START);
    iowrite32_rep_fixed(DATA_START, dev.buffer + DATA_START, (DATA_END - DATA

    *((u32*)&(dev.buffer[RESULT_LOC])) = ioread32(dev.virtbase + RESULT_LOC);
```

```
    arg.convmax = ((u16)dev.buffer[RESULT_LOC] << 8) | dev.buffer[RESULT_LOC -
    arg.maxpos = dev.buffer[RESULT_LOC + 2];

    if (copy_to_user(user_arg_ptr, &arg, sizeof(plazer_arg_t))) {
        return -EACCES;
    }
    return 0;
}

static long plazer_set_convolution(plazer_conv_t * user_arg_ptr) {
    plazer_conv_t arg;
    if (copy_from_user(&arg, user_arg_ptr, sizeof(plazer_conv_t))) {
        return -EACCES;
    }

    memcpy(dev.buffer + FILTER_START, arg.conv, FILTER_END - FILTER_START);

    iowrite32_rep_fixed(FILTER_START, dev.buffer + FILTER_START, (FILTER_END -

    return 0;
}

static long plazer_read_memory(plazer_mem_t *user_arg_ptr) {
    plazer_mem_t arg;

    if (copy_from_user(&arg, user_arg_ptr, sizeof(plazer_mem_t))) {
        return -EACCES;
    }

    ioread32_rep_fixed(0, dev.buffer, PLAZER_SIZE_32);

    memcpy(arg.data.left_fill, dev.buffer, DATA_CONV_START - DATA_START);
    memcpy(arg.data.data, dev.buffer + DATA_CONV_START, DATA_CONV_END - DATA_C
    memcpy(arg.data.right_fill, dev.buffer + DATA_CONV_END, DATA_END - DATA_CO
    memcpy(arg.conv.conv, dev.buffer + FILTER_START, FILTER_END - FILTER_START

    arg.data.convmax = ((u16)dev.buffer[RESULT_LOC] << 8) | dev.buffer[RESULT
    arg.data.maxpos = dev.buffer[RESULT_LOC + 2];

    if (copy_to_user(user_arg_ptr, &arg, sizeof(plazer_mem_t))) {
        return -EACCES;
    }
    return 0;
}
```

```
static long plazer_get_buffer(unsigned char *user_arg_ptr) {
   if (copy_to_user(user_arg_ptr, dev.buffer, sizeof(dev.buffer))) {
       return -EACCES;
   }
   return 0;
}

static long plazer_reset() {
    memset(dev.buffer, 0, PLAZER_SIZE);
    iowrite32_rep_fixed(0, dev.buffer, PLAZER_SIZE_32);
    ioread32_rep_fixed(dev.virtbase, dev.buffer, PLAZER_SIZE_32);
    return 0;
}

/*
 * Handle ioctl() calls from userspace:
 * Read or write the segments on single digits.
 * Note extensive error checking of arguments
 */
static long plazer_ioctl(struct file *f, unsigned int cmd, unsigned long arg)
{
    switch (cmd) {
        case PLAZER_CONV_MAX:
            return plazer_conv_max(arg);
        case PLAZER_SET_CONV:
            return plazer_set_convolution(arg);
        case PLAZER_READ_MEMORY:
            return plazer_read_memory(arg);
        case PLAZER_RESET:
            return plazer_reset();
        case PLAZER_GET_BUFFER:
            return plazer_get_buffer(arg);
        default:
            return -EINVAL;
    }
    return 0;
}

static int plazer_open(struct inode *i, struct file *f) {
    return 0;
}

static int plazer_close(struct inode *i, struct file *f) {
```

```
    return 0;
}

/* The operations our device knows how to do */
static const struct file_operations plazer_fops = {
    .owner      = THIS_MODULE,
    .open       = plazer_open,
    .release    = plazer_close,
    .unlocked_ioctl = plazer_ioctl,
};

/* Information about our device for the "misc" framework -- like a char dev */
static struct miscdevice plazer_misc_device = {
    .minor      = MISC_DYNAMIC_MINOR,
    .name       = DRIVER_NAME,
    .fops       = &plazer_fops,
};

/*
 * Initialization code: get resources (registers) and display
 * a welcome message
 */
static int __init plazer_probe(struct platform_device *pdev)
{
    static unsigned char welcome_message[PLAZER_DIGITS] = {
        0x3E, 0x7D, 0x77, 0x08, 0x38, 0x79, 0x5E, 0x00};
    int i, ret;

    /* Register ourselves as a misc device: creates /dev/plazer */
    ret = misc_register(&plazer_misc_device);

    /* Get the address of our registers from the device tree */
    ret = of_address_to_resource(pdev->dev.of_node, 0, &dev.res);
    if (ret) {
        ret = -ENOENT;
        goto out_deregister;
    }

    /* Make sure we can use these registers */
    if (request_mem_region(dev.res.start, resource_size(&dev.res),
                           DRIVER_NAME) == NULL) {
        ret = -EBUSY;
        goto out_deregister;
    }
```

```
    /* Arrange access to our registers */
    dev.virtbase = of_iomap(pdev->dev.of_node, 0);
    if (dev.virtbase == NULL) {
        ret = -ENOMEM;
        goto out_release_mem_region;
    }

    /* Display a welcome message */
    // for (i = 0; i < PLAZER_DIGITS; i++)
    //     write_digit(i, welcome_message[i]);

    return 0;

out_release_mem_region:
    release_mem_region(dev.res.start, resource_size(&dev.res));
out_deregister:
    misc_deregister(&plazer_misc_device);
    return ret;
}

/* Clean-up code: release resources */
static int plazer_remove(struct platform_device *pdev)
{
    iounmap(dev.virtbase);
    release_mem_region(dev.res.start, resource_size(&dev.res));
    misc_deregister(&plazer_misc_device);
    return 0;
}

/* Which "compatible" string(s) to search for in the Device Tree */
#ifdef CONFIG_OF
static const struct of_device_id plazer_of_match[] = {
    { .compatible = "altr,plazer" },
    {},
};
MODULE_DEVICE_TABLE(of, plazer_of_match);
#endif

/* Information for registering ourselves as a "platform" driver */
static struct platform_driver plazer_driver = {
    .driver = {
        .name   = DRIVER_NAME,
        .owner  = THIS_MODULE,
```

```
        .of_match_table = of_match_ptr(plazer_of_match),
    },
    .remove = __exit_p(plazer_remove),
};

/* Called when the module is loaded: set things up */
static int __init plazer_init(void)
{
    pr_info(DRIVER_NAME ": init\n");
    return platform_driver_probe(&plazer_driver, plazer_probe);
}

/* Called when the module is unloaded: release resources */
static void __exit plazer_exit(void)
{
    platform_driver_unregister(&plazer_driver);
    pr_info(DRIVER_NAME ": exit\n");
}

module_init(plazer_init);
module_exit(plazer_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Robert Ying, Xingzhou He, Peiqian Li, Minh Trang Nguyen");
MODULE_DESCRIPTION("Convolution accelerator");
```

A.12. app.c

```
/*
 * Userspace program that communicates with the led_vga device driver
 * primarily through ioctls
 *
 * Stephen A. Edwards
 * Columbia University
 */

#include <stdio.h>
#include "plazer.h"
#include <math.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
```

```
#include <unistd.h>

int plazer_fd;

int emulation_mode = 0;

plazer_mem_t emulation_buffer;

void print_buf(unsigned char * buf, size_t length) {
    int i;
    printf("\n--------------------------------------------------\n");
    for (i = 0; i < length; ++i) {
        if (i % 8 == 0) {
            printf(" %02d: ", i);
        }

        printf("0x%02x ", buf[i]);

        if (i % 8 == 7) {
            printf("\n");
        }
    }
    if (i % 8 != 0) {
        printf("\n");
    }
    printf("--------------------------------------------------\n");
}

void plazer_reset() {
    if (emulation_mode) {
        memset(&emulation_buffer, 0, sizeof(plazer_mem_t));
    } else {
        if (ioctl(plazer_fd, PLAZER_RESET, 0)) {
            perror("ioctl(PLAZER_RESET) failed");
            return;
        }
    }
}

void plazer_read_memory() {
    plazer_mem_t mem;

    if (emulation_mode) {
        memcpy(&mem, &emulation_buffer, sizeof(plazer_mem_t));
```

```
    } else {
        if (ioctl(plazer_fd, PLAZER_READ_MEMORY, &mem)) {
            perror("ioctl(PLAZER_READ_MEMORY) failed");
            return;
        }
    }

    printf("left fill:\n");
    print_buf(mem.data.left_fill, sizeof(mem.data.left_fill));

    printf("data:\n");
    print_buf(mem.data.data, sizeof(mem.data.data));

    printf("right fill:\n");
    print_buf(mem.data.right_fill, sizeof(mem.data.right_fill));

    printf("conv matrix:\n");
    print_buf(mem.conv.conv, sizeof(mem.conv.conv));

    printf("result:\n");
    printf("max: %d at position %d\n", mem.data.convmax, mem.data.maxpos);
}

void plazer_set_convolution(unsigned char * convdat) {
    plazer_conv_t conv;
    memcpy(conv.conv, convdat, sizeof(conv.conv));

    if (emulation_mode) {
        memcpy(&emulation_buffer.conv, &conv, sizeof(plazer_conv_t));
    } else {
        if (ioctl(plazer_fd, PLAZER_SET_CONV, &conv)) {
            perror("ioctl(PLAZER_SET_CONV) failed");
            return;
        }
    }
}

void plazer_conv_max(plazer_arg_t * arg) {
    int i, j;
    if (emulation_mode) {
        memcpy(&emulation_buffer.data, arg, sizeof(plazer_arg_t));

        arg->convmax = 0;
        arg->maxpos = 0;
```

```
        for (i = 0; i < (DATA_CONV_END - DATA_CONV_START); ++i) {
            unsigned int accum = 0;

            unsigned char * dptr = (unsigned char *) arg;

            for (j = 0; j < FILTER_END - FILTER_START; ++j) {
                unsigned int filter_val = emulation_buffer.conv.conv[j];
                accum += dptr[i + j] * filter_val;
                accum += dptr[i + (FILTER_END - FILTER_START) * 2 - j] * filte
            }

            if (accum > arg->convmax) {
                arg->convmax = accum;
                arg->maxpos = i;
            }
        }
    } else {
        if (ioctl(plazer_fd, PLAZER_CONV_MAX, arg)) {
            perror("ioctl(PLAZER_CONV_MAX) failed");
        }
    }
    memcpy(&emulation_buffer.data, arg, sizeof(plazer_arg_t));
}

void generate_gaussian(float sigma, unsigned char * buffer, size_t buflen) {
    int i;
    double s2 = sigma * sigma;
    for (i = 0; i < buflen; ++i) {
        double x = buflen - i;
        double val = exp(-(x * x) / s2) / sqrt(2 * M_PI * s2);

        buffer[i] = val * 0xff;
    }
}

void plazer_read_buffer() {
    unsigned char bytes[60];

    if (ioctl(plazer_fd, PLAZER_GET_BUFFER, bytes)) {
        perror("ioctl(PLAZER_READ_MEMORY) failed");
        return;
    }
```

```
    print_buf(bytes, 60);

}

int main()
{
    static const char filename[] = "/dev/plazer";
    int i;
    unsigned char conv[8];
    plazer_arg_t arg;

    if ((plazer_fd = open(filename, O_RDWR)) == -1) {
        fprintf(stderr, "could not open %s\n", filename);
        emulation_mode = 1;
    }

    printf("plazer userspace program started\n");

    if (emulation_mode) {
        printf("running in emulation mode\n");
    }

    plazer_reset();
    plazer_read_memory();

    generate_gaussian(3, conv, sizeof(conv));

    printf("gaussian matrix:");
    print_buf(conv, sizeof(conv));

    plazer_set_convolution(conv);

    plazer_read_memory();

    memset(&arg, 0, sizeof(plazer_arg_t));
    arg.data[4] = 0xff;

    plazer_conv_max(&arg);

    plazer_read_memory();
    //plazer_read_buffer();

    //printf("result:\n");
    //printf("max: %d at position %d\n", arg.convmax, arg.maxpos);
```

```
    printf("plazer userspace program terminating\n");
    return 0;
}
```

A.13. syscon-test.tcl

```
# A Tcl script for the Qsys system console

# Start Qsys, open your soc_system.qsys file, run File->System Console,
# then execute this script by selecting it with Ctrl-E

# The System Console is described in Chapter 10 of Volume III of
# the Quartus II Handbook

# Alternately,
# system-console --project_dir=. --script=syscon-test.tcl
#
# system-console --project_dir=. -cli
#   and then "source syscon-test.tcl"

# Base addresses of the peripherals: take from Qsys
set vga_led 0x0

puts "Started system-console-test-script"

# Using the JTAG chain, check the clock and reset"

set j [lindex [get_service_paths jtag_debug] 0]
open_service jtag_debug $j
puts "Opened jtag_debug"

puts "Checking the JTAG chain loopback: [jtag_debug_loop $j {1 2 3 4 5 6}]"
jtag_debug_reset_system $j

puts -nonewline "Sampling the clock: "
foreach i {1 1 1 1 1 1 1 1 1 1 1 1} {
    puts -nonewline [jtag_debug_sample_clock $j]
}
puts ""

puts "Checking reset state: [jtag_debug_sample_reset $j]"

close_service jtag_debug $j
```

```
puts "Closed jtag debug"

# Perform bus reads and writes

set m [lindex [get_service_paths master] 0]
open_service master $m
puts "Opened master"

# zero out all memory
foreach i {0 4 8 12 16 20 24 28 32 36 40 44 48 52} {
    master_write_32 $m [expr $vga_led + $i] 0x00000000
}

# write convolution buffer
foreach {r v} {48 0x000000ff 52 0xff000000} {
    master_write_32 $m [expr $vga_led + $r] $v
}

# write point location
master_write_8 $m [expr $vga_led + 0x05] 0xff

foreach i {0 4 8 12 16 20 24 28 32 36 40 44 48 52 56} {
    set x [master_read_32 $m [expr $vga_led + $i] 1]
    puts "$i: $x"
}

set ready [master_read_8 $m [expr $vga_led + 59] 1]
set pos [master_read_8 $m [expr $vga_led + 58] 1]
set val [master_read_16 $m [expr $vga_led + 56] 1]

puts "pos: $pos"
puts "val: $val"
puts "ready: $ready"

# Write a test pattern to the various registers
#foreach {r v} {0 0xff 1 0x1 2 0x2 3 0x4 4 0x8 5 0x10 6 0x20 7 0x40} {
#    master_write_8 $m [expr $vga_led + $r] $v
#}

close_service master $m
puts "Closed master"
```

A.14. userland.cpp

```
/*
 * Userspace program
 * PLazeR
 * Columbia University
 */

#include <stdio.h>
#include "plazer.h"
#include <math.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>

#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

const double w0 = 15138.3, w1 = -1.32275;

int plazer_fd;

int emulation_mode = 0;

plazer_mem_t emulation_buffer;

void print_buf(unsigned char * buf, size_t length) {
    int i;
    printf("\n--------------------------------------------------------\n");
    for (i = 0; i < length; ++i) {
        if (i % 8 == 0) {
            printf(" %02d: ", i);
        }

        printf("0x%02x ", buf[i]);

        if (i % 8 == 7) {
            printf("\n");
        }
    }
    if (i % 8 != 0) {
        printf("\n");
    }
```

```
    printf("--------------------------------------------------\n");
}

void plazer_reset() {
    if (emulation_mode) {
        memset(&emulation_buffer, 0, sizeof(plazer_mem_t));
    } else {
        if (ioctl(plazer_fd, PLAZER_RESET, 0)) {
            perror("ioctl(PLAZER_RESET) failed");
            return;
        }
    }
}

void plazer_read_memory() {
    plazer_mem_t mem;

    if (emulation_mode) {
        memcpy(&mem, &emulation_buffer, sizeof(plazer_mem_t));
    } else {
        if (ioctl(plazer_fd, PLAZER_READ_MEMORY, &mem)) {
            perror("ioctl(PLAZER_READ_MEMORY) failed");
            return;
        }
    }

    printf("left fill:\n");
    print_buf(mem.data.left_fill, sizeof(mem.data.left_fill));

    printf("data:\n");
    print_buf(mem.data.data, sizeof(mem.data.data));

    printf("right fill:\n");
    print_buf(mem.data.right_fill, sizeof(mem.data.right_fill));

    printf("conv matrix:\n");
    print_buf(mem.conv.conv, sizeof(mem.conv.conv));

    printf("result:\n");
    printf("max: %d at position %d\n", mem.data.convmax, mem.data.maxpos);
}

void plazer_set_convolution(unsigned char * convdat) {
    plazer_conv_t conv;
```

```
    memcpy(conv.conv, convdat, sizeof(conv.conv));

    if (emulation_mode) {
        memcpy(&emulation_buffer.conv, &conv, sizeof(plazer_conv_t));
    } else {
        if (ioctl(plazer_fd, PLAZER_SET_CONV, &conv)) {
            perror("ioctl(PLAZER_SET_CONV) failed");
            return;
        }
    }
}

void plazer_conv_max(plazer_arg_t * arg) {
    int i, j;
    if (emulation_mode) {
        memcpy(&emulation_buffer.data, arg, sizeof(plazer_arg_t));

        arg->convmax = 0;
        arg->maxpos = 0;

        for (i = 0; i < (DATA_CONV_END - DATA_CONV_START); ++i) {
            unsigned int accum = 0;

            unsigned char * dptr = (unsigned char *) arg;

            for (j = 0; j < FILTER_END - FILTER_START; ++j) {
                unsigned int filter_val = emulation_buffer.conv.conv[j];
                accum += dptr[i + j] * filter_val;
                accum += dptr[i + (FILTER_END - FILTER_START) * 2 - j] * filte
            }

            if (accum > arg->convmax) {
                arg->convmax = accum;
                arg->maxpos = i;
            }
        }
    } else {
        if (ioctl(plazer_fd, PLAZER_CONV_MAX, arg)) {
            perror("ioctl(PLAZER_CONV_MAX) failed");
        }
    }
    memcpy(&emulation_buffer.data, arg, sizeof(plazer_arg_t));
}
```

```
void generate_gaussian(float sigma, unsigned char * buffer, size_t buflen) {
    int i;
    double s2 = sigma * sigma;
    for (i = 0; i < buflen; ++i) {
        double x = buflen - i;
        double val = exp(-(x * x) / s2) / sqrt(2 * M_PI * s2);

        buffer[i] = val * 0xff;
    }
}

int main(int argc, char **argv) {
    if(argc != 2) {
        fprintf(stderr, "Usage: %s image-file\n", argv[0]);
        return 1;
    }

    static const char dev_filename[] = "/dev/plazer";
    unsigned char conv[8];
    plazer_arg_t arg;

    if ((plazer_fd = open(dev_filename, O_RDWR)) == -1) {
        fprintf(stderr, "could not open %s\n", dev_filename);
        emulation_mode = 1;
    }

    printf("plazer userspace program started\n");

    cv::Mat image = cv::imread(argv[1], CV_LOAD_IMAGE_GRAYSCALE);
    unsigned char raw[image.rows][image.cols];
    for(int i=0; i<image.rows; ++i)
        for(int j=0; j<image.cols; ++j)
            raw[i][j] = image.at<uchar>(i, j);

    plazer_reset();
    plazer_read_memory();

    generate_gaussian(3, conv, sizeof(conv));

    printf("gaussian matrix:");
    print_buf(conv, sizeof(conv));

    plazer_set_convolution(conv);
```

```
    plazer_read_memory();

    memset(&arg, 0, sizeof(plazer_arg_t));

    size_t size_leftfill = sizeof(arg.left_fill);
    size_t size_data = sizeof(arg.data);
    size_t size_rightfill = sizeof(arg.right_fill);
    for(int i = 0; i < image.rows; ++i) {
        int row_convmax = -1;
        int row_maxpos = -1;
        for(int j = 0; j < image.cols; j += size_data) {
            memcpy(arg.data, raw[i] + j, size_data);

            //left filler
            if(j) {
                memcpy(arg.left_fill, raw[i] + j - size_leftfill, size_leftfi
            } else {
                memset(arg.left_fill, 0, size_leftfill);
            }

            //right filler
            if(j + size_data < image.cols) {
                memcpy(arg.right_fill, raw[i] + j + size_data, size_rightfill
            } else {
                memset(arg.right_fill, 0, size_rightfill);
            }

            plazer_conv_max(&arg);

            if(arg.convmax > row_convmax) {
                row_convmax = arg.convmax;
                row_maxpos = j + arg.maxpos;
            }
        }
        printf("row = %d; distance = %.4f\n", i, w0 * pow(row_maxpos, w1));
    }
    printf("plazer userspace program terminating\n");

    return 0;
}
```