2024 Autumn ICLAB Final Project: Image Signal Processor (ISP) for Camera Auto Focus & Auto Exposure Algorithm

Jui-Huang Tsai
erictsai.ee12@nycu.edu.tw

Abstract

The objective of this assignment is to design camera 3A algorithms using hardware description language, focusing primarily on autofocus—to calculate the optimal focal length within a given range—and auto-exposure algorithms, which adjust the image to a specified brightness and compute the average brightness of the adjusted image. Concurrently, the design utilizes the AXI4 transmission protocol to read and write image data from and to external DRAM.

To accommodate large volumes of image data within the design, the use of SRAM is encouraged in this assignment to reduce area overhead. This project is expected to enhance students' understanding and proficiency in hardware description language writing through the implementation of fundamental camera 3A algorithms. Furthermore, it aims to provide practical insight into the read and write operations of the AXI4 communication protocol and to strengthen skills and techniques related to SRAM operations.

Through this hands-on experience, students will gain a deeper comprehension of hardware design principles and their application in real-world imaging systems, while also developing expertise in efficient memory management and protocol implementation.

Index Terms

Auto focus, Auto exposure, AXI4

I. Introduction

In this assignment, we will primarily focus on designing Verilog implementations of camera 3A algorithms, specifically the auto focus and auto exposure algorithms. The following sections will briefly introduce the processes of autofocus and auto-exposure algorithms. Please use `tar xvf ~iclabTA01/Final_Project.tar` to extract the exercise file.

A. Auto Focus

The auto focus algorithm primarily works by simulating different focal lengths to identify the area with the maximum focus. The image size is fixed to $3 * 32 * 32$. The specific procedure is as follows:

I. Given 3 contrasts, from 0 to 2, cut the center of image to process. The coordinate of center of image for 4 contrasts are shown bellow (count from 0):

- 0: $15 \leq x \leq 16, \quad 15 \leq y \leq 16, \quad 2 * 2$, total 4 points
- 1: $14 \leq x \leq 17, \quad 14 \leq y \leq 17, \quad 4 * 4$, total 16 points
- 2: $13 \leq x \leq 18, \quad 13 \leq y \leq 18, \quad 6 * 6$, total 36 points

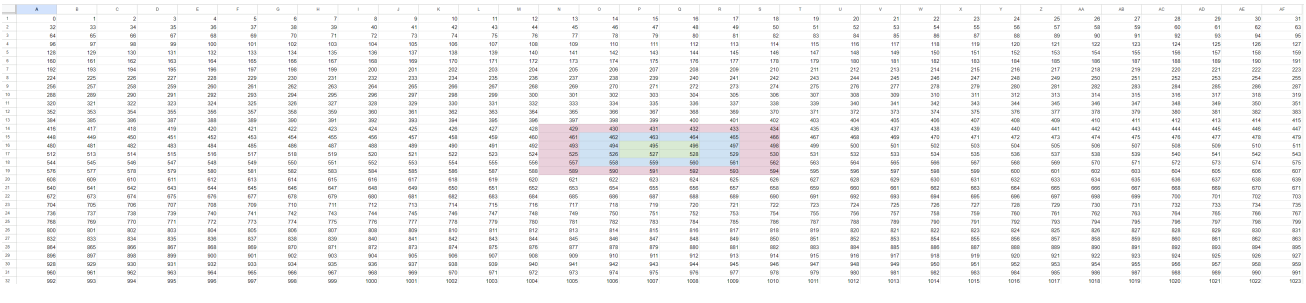


Fig. 1: The selection area for each center

II. Deal with the center area of given image for each contrasts, need to do 3 time.

$$I_{center} \in \mathbb{N}^{3 \times H \times W}, \quad H = W, \quad W = \{2, 4, 6\}$$

III. Convert the selection area to grayscale. This can be represented by the following formula:

$$G_{center}(i,j) = I_{center}(i,j,0) \cdot 0.25 + I_{center}(i,j,1) \cdot 0.5 + I_{center}(i,j,2) \cdot 0.25$$

The definition of each symbol:

$$[I_{center} \in \mathbb{N}^{3\times H\times W}, 0 \le I \le 255], \quad [G_{center} \in \mathbb{N}^{1\times H\times W}, 0 \le G \le 255],$$

IV. Calculate the difference between each element on the x-axis and y-axis separately, and take the absolute value. Sum up all the calculated differences.

$$D_{contrast\_h*w} = \sum_{j=0}^{H-1}\sum_{i=0}^{W-2} |G_{center}(i+1,j) - G_{center}(i,j)| + \sum_{i=0}^{W-1}\sum_{j=0}^{H-2} |G_{center}(i,j+1) - G_{center}(i,j)|$$

$$D_{contrast\_h*w} \in \mathbb{N}^3, h = w, w = 2, 4, 6$$

V. Divide the summed value by the processed image's area, and record this value.

$$D_{contrast\_h*w} = D_{contrast\_h*w}/H^2$$

VI. Repeat steps III. to V. for four contrast. Find the maximum contrast value, which is the optimal solution. If there are two or above value are same, output the smaller one's contrast value. E.g. if contrast value 0 and 2 both's solution are same but larger than 1's, please output 0.

$$max\_contrast = \arg\max(D_{contrast\_h*w})$$

The python and numpy-like pseudo code for color auto focus function is shown below.

Algorithm 1 Python and Numpy-like pseudo code for function Color Auto Focus.

```
def color_auto_focus(image):
    """
    Process auto-focus algorithm for color images
    :param image: 32x32x3 color image, numpy array
    :return: Index of the best "focus"
    """
    def calculate_contrast(img):
        # Convert to grayscale using dot product
        gray = np.dot(img[...,:3], [0.25, 0.5, 0.25])
        gray = np.array(gray)

        # Calculate differences in 'gray'
        dx = np.diff(gray, axis=1)  # horizontal diff
        dy = np.diff(gray, axis=0)  # vertical diff

        return int(np.sum(np.abs(dx)) + np.sum(np.abs(dy))) / (img.size[0]*img.size[1])

    contrasts = []
    center_point = image.shape[0] // 2

    for i in range(0, 3):  # Simulate different "focus", iterate over center area
        # Extract different sized centers
        center = image[center_point-1-i:center_point+1+i,
                       center_point-1-i:center_point+1+i, :]
        contrasts.append(calculate_contrast(center))

    # Find the maximum value's index in the contrasts list
    return np.argmax(contrasts)
```

B. Auto Exposure

The purpose of this algorithm is to adjust the exposure ratio of a given image, which size is fixed to 3*32*32.

$$0 \le i, j \le 31$$

I. First, input the entire image and the desired exposure adjustment ratio. There are 4 options, from 0 to 3, corresponding to 0.25, 0.5, 1, and 2 times the original exposure.

$$(input, ratio) = \{(0, 0.25), (1, 0.5), (2, 1), (3, 2)\}$$

II. Adjust the original image according to the chosen exposure ratio and save the adjusted picture. If any adjusted value exceeds the maximum (255), set that value to 255.

$$I_{i,j} = \begin{cases} I_{i,j} * ratio & \text{for } 0 \leq (I_{i,j} * ratio) < 256 \\ 255 & \text{other} \end{cases}$$

III. Finally, calculate the average value of the adjusted image and return this average. The detailed method is as follows:

- Multiply each element in the R and B channels by 0.25
- Multiply each element in the G channel by 0.5
- Sum all these values and divide by the original image size (1024)
- This result is the average value.

$$Avg = \frac{\sum_{i=0}^{32}\sum_{j=0}^{32}[I(i,j,0)\cdot 0.25 + I(i,j,1)\cdot 0.5 + I(i,j,2)\cdot 0.25]}{32 * 32}$$

IV. Due to possible precision errors in the calculations, the answer allows a margin of error of ±1. ($|golden - ans| < 2$)

The python and numpy-like pseudo code for color auto exposure function is shown below.

Algorithm 2 Python and Numpy-like pseudo code for function Color Auto Exposure.

```python
def color_auto_exposure(image, ratio):
    """
    Process auto-exposure algorithm for color images
    :param image: 32x32x3 color image, numpy array
    :param ratio: Adjustment ratio
    :return: Mean value of adjusted image
    """

    # Adjust each color channel
    # Use np.clip to limit values between 0-255
    adjusted_image = np.clip(image * ratio, 0, 255).astype(np.uint8)

    # return mean value of adjusted image
    return np.mean(np.dot(adjusted_image[...,:3], [0.25, 0.5, 0.25]))
```

C. Average of Min and Max in the Picture

In this function, you need to find out the maximum and minimum value in each RGB channels in the selected picture from the DRAM. You need to sum up three maximum values in each channel then divide this value by 3, and so does the minimum value. After you getting the average value of maximum value and minimum value, get the average value of them. The math formula for this function is shown bellow.

$$max_{R,G,B} \in 3 * N, min_{R,G,B} \in 3 * N,$$

$$max_{R,G,B} = Max(Pic_{pic\_idx(R,G,B)}), min_{R,G,B} = Min(Pic_{pic\_idx(R,G,B)})$$

$$max = \frac{max_R + max_G + max_B}{3}, min = \frac{min_R + min_G + min_B}{3}$$

$$result = \frac{max + min}{2}$$

Here is an example for this function, and the numbers in this example are shown in decimal. We got 2, 4, 6 as the minimum value for each channel, and 80, 160, 140 for the maximum value for each channel, respectively. Now you should get the average minimum and maximum value, which is 4 and 126. Now, you can output the computation result to the PATTERN, which is 65.

II. Design

The system is composed of three parts: PATTERN.v, ISP.v, and pseudo_DRAM.v. PATTERN.v is responsible for sending instructions and verifying the correctness of the answers. ISP.v implements the algorithm for auto focus and auto exposure, and AXI4 for reading and writing data with pseudo_DRAM.v. If large amounts of data need to be stored, SRAM can be added within ISP.v to facilitate temporary storage of large data volumes. pseudo_DRAM.v stores all image data, which ISP.v needs to read and write using AXI4, the AXI4 function inside pseudo_DRAM has been implement by TA. The connectivity of the ISP system can refer to the fig. 2. The channel length and description between ISP.v and PATTERN.v is shown in table I.

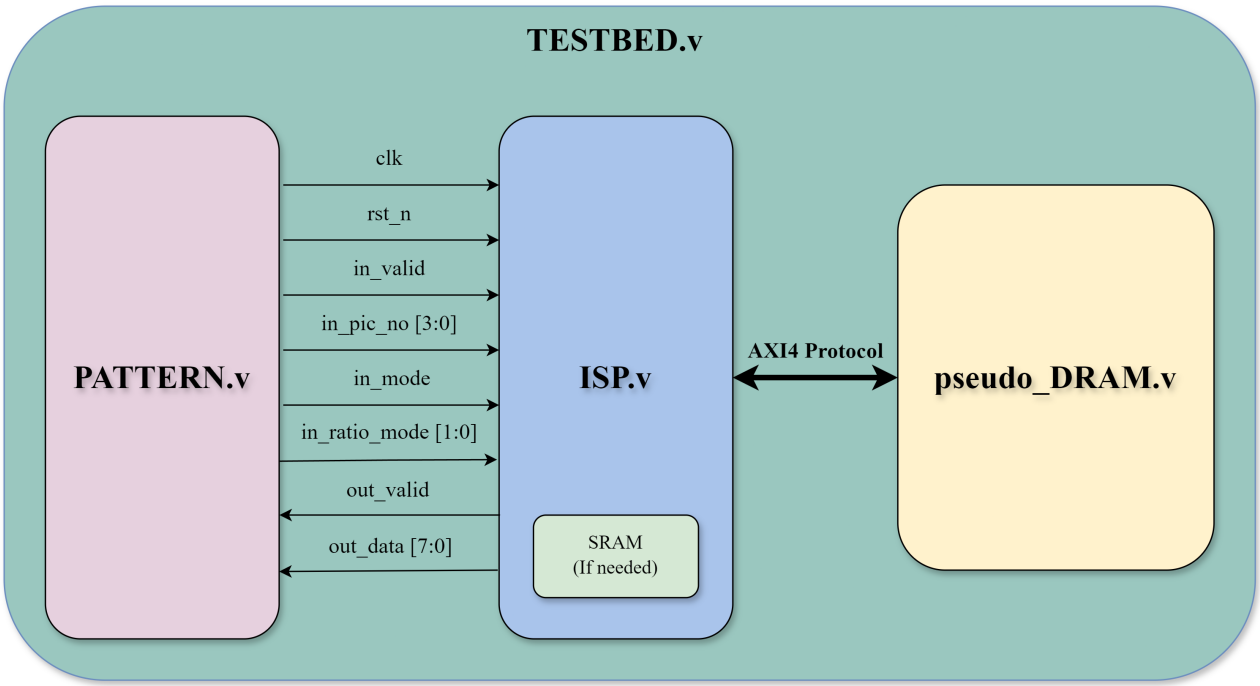


Fig. 2: ISP System for midterm project.

For write data into pseudo_DRAM.v, you need to comply with the AXI4 protocol, which can be refereed from the AXI slide. Note that the clk and rst_n signal should be sent into pseudo_DRAM.v from ISP.v. In table II, it is related to write address control. Note that the head of image's memory site is started from 32'h10000 + (3*32*32)*pic_no. In the table III is about the write data channel controlling, and the table IV is for write response channel controlling. For reading data from pseudo_DRAM.v, please refer to tabel V, VI.

The DRAM content data is shown in fig. 3, note that each location in DRAM can store 8 bits data. The first picture was started at 32'h10000, if the design need to get other picture, the formula for location calculation is $32'h10000 + pic\_no * 3 * 32 * 32$. If you want to get whole picture in a round, you can set burst_len to 192, this is calculated from

$$\frac{32 * 32 * 3 * 8bits}{128bits}$$

For in_mode is set to 0 when in_invalid is high, your design should execute auto focus algorithm. First, you should get data from the DRAM with corresponding picture through the AXI4 protocol and save into your design. After that, the design can start conducting auto focus algorithm as mentioned in the chapter I-A. After calculating, you should return the best contrast to the PATTERN.v through out_data when out_valid is high.

If the mode is auto exposure algorithm now, same as the previous one, you should get the data from the DRAM and refer to the chapter I-A to transform the image to the desired brightness. You need to write back the image after adjusting to DRAM, or you may fail if encounter the same picture in next coming testing. You also need to return the mean brightness of the adjusted picture through out_data when out_valid is high to PATTERN.v.

For the PATTERN.v, you can use the code bellowing to mimic the pseudo_DRAM, which will lead easier for your pattern to check the result from your design.

```
parameter DRAM_p_r = "../00_TESTBED/DRAM/dram.dat";
reg [7:0]   DRAM_r  [0:196607];

initial begin
$readmemh(DRAM_p_r, DRAM_r);
end
```

Listing 1: example code for PATTERN.v to mimic pseudo_DRAM.

As known, there are some delay caused by other issue in the real DRAM. Our DRAM has simulated this situation too. The LAT_MAX is the maximum latency that may happened in read data mode; hence, the LAT_min is the minimum latency in read data mode. During the demo, TA will use the 'PERF' mode list in the bellowing verilog code. The parameter 'd_DRAM_p_r' is the location for the pseudo_DRAM to read your data. 'd_DRAM_R_LAT' is the fixed latency for read data, available when

TABLE I: Signal Description between PATTERN.v and ISP.v

Signal	Width	From	Description
clk	1	PATTERN.v	System clock.
rst_n	1	PATTERN.v	Reset all system.
in_valid	1	PATTERN.v	When in_valid=1, means the signal in_mode, in_pic_no and in_ratio_mode signals are valid. Will be high for only 1 cycle.
in_pic_no	4	PATTERN.v	Select which picture from the DRAM. Only valid when in_valid=1
in_mode	2	PATTERN.v	Select algorithm type. 0 for Auto Focus, 1 for Auto Exposure, and 2 for Average of min max value. Only valid when in_valid=1.
in_ratio_mode	1	PATTERN.v	Select the ratio for auto exposure. 0 means 0.25, 1 means 0.5, 2 means 1, 3 means 2. Only valid when in_valid=1 and in_mode=1.
out_valid	1	ISP.v	When out_valid=1, means the our_data is valid. Note that cannot be overlapped with in_valid.
out_data	8	ISP.v	Return the best contrasts when algorithm mode is auto focus, means value for the picture adjusted after auto exposure.

TABLE II: AXI Signal for Write Address Channel

Signal	Width	From	Description
awid_s_inf	4	ISP.v	Write address ID. In this project, we only use this to recognize master, reordering method is not supported.
awaddr_s_inf	32	ISP.v	Write address. Address start from 32'h10000, and each pic location is 32'h10000 + 3072*pic_no.
awsize_s_inf	3	ISP.v	Burst size. Only support 3'b100 in this exercise.
awburst_s_inf	2	ISP.v	Burst type. Only INCR(2'b01) support in this Project.
awlen_s_inf	8	ISP.v	Burst length. The burst length gives the exact number of transfers in a burst.
awvalid_s_inf	1	ISP.v	Write address valid. 1 = address and control information available; 0 = address and control information not available.
awready_s_inf	1	pseudo_DRAM.v	Write address ready. 1 = slave ready; 0 = slave not ready.

TABLE III: AXI Signal for Write Data Channel

Signal	Width	From	Description
wdata_s_inf	128	ISP.v	Write data.
wlast_s_inf	1	ISP.v	Write last. This signal indicates the last transfer in a write burst.
wvalid_s_inf	1	ISP.v	Write valid. 1 = write data and strobes available; 0 = write data and strobes not available.
wready_s_inf	1	pseudo_DRAM.v	Write ready. 1 = slave ready; 0 = slave not ready.

TABLE IV: AXI Signal for Write Response Channel

Signal	Width	From	Description
bid_s_inf	4	pseudo_DRAM.v	Response ID. The BID value must match the AWID value.
bresp_s_inf	2	pseudo_DRAM.v	Write response. In this project we only issue OKAY(2'b00)
bvalid_s_inf	1	pseudo_DRAM.v	Write response valid. 1 = write response available; 0 = write response not available.
bready_s_inf	1	ISP.v	Response ready. 1 = master ready; 0 = master not ready.

TABLE V: AXI Signal for Read Address Channel

Signal	Width	From	Description
arid_s_inf	4	ISP.v	Read address ID. In this project, we only use this to recognize master, reordering method is not supported
araddr_s_inf	32	ISP.v	Read address. Each image location is started from 32'h10000 + 3072*pic_no.
arlen_s_inf	8	ISP.v	Burst length.
arsize_s_inf	3	ISP.v	Burst size. Only support 3b'100.
arburst_s_inf	2	ISP.v	Burst type. Only INCR: 2b'01 support in this Project.
arvalid_s_inf	1	ISP.v	Read address valid. 1 = address and control information valid; 0 = address and control information not valid.
arready_s_inf	1	pseudo_DRAM.v	Read address ready. 1 = slave ready; 0 = slave not ready.

TABLE VI: AXI Signal for Read Data Channel

Signal	Width	From	Description
rid_s_inf	4	pseudo_DRAM.v	Read ID tag. The RID value is generated by the slave and must match the ARID value.
rdata_s_inf	128	pseudo_DRAM.v	Read data.
rresp_s_inf	2	pseudo_DRAM.v	Read response. In this project we only issue OKAY (2'b00).
rlast_s_inf	1	pseudo_DRAM.v	Read last.
rvalid_s_inf	1	pseudo_DRAM.v	Read valid. 1 = read data available; 0 = read data not available.
rready_s_inf	1	ISP.v	Read ready. 1= master ready; 0 = master not ready.

'd_RANDOM_R_LAT' is 0. 'd_DRAM_W_LAT' is the latency for writing data mode. 'd_RANDOM_R_LAT' can decide the latency for read mode is fixed or random.

```
`ifdef SAMPLE
`define LAT_MAX 10
`define LAT_MIN 1
`endif
`ifdef FUNC
`define LAT_MAX 10
`define LAT_MIN 1
`endif
`ifdef PERF
`define LAT_MAX 500
`define LAT_MIN 300
`endif

// Modify your "d_DRAM_p_r" in this directory path to initialized DRAM Value
// Modify d_DRAM_R_LAT           for Initial Read Data Latency,
//        d_DRAM_W_LAT           for Initial Write Data Latency
//        d_RANDOM_R_LAT         for 1: Random Read Data Latency 0: DRAM_R_LAT Read Data
    Latency

`define d_DRAM_p_r "../00_TESTBED/DRAM/dram.dat"
`define d_DRAM_R_LAT 1
`define d_DRAM_W_LAT 1
`define d_RANDOM_R_LAT 1
```

Listing 2: The verilog code in pseudo_DRAM about setting up latency parameter.

III. Design Rules

I. After rst_n, all output (including AXI) signals should be set to 0.
II. You need to raise out_valid only when your design has done current input operation. TA will check the out_data when the out_valid is high.
III. out_valid signal cannot overlap with in_valid.
IV. The next in_valid signal will be high in next 2 cycle after out_valid is high.
V. The system will NOT check the data stored in DRAM.
VI. The AXI write mode should be completed when the writing action is done, which means the w_last should be high when the last data is writing back to the DRAM. Otherwise, it will fail when next time attempt to write data back to the DRAM.
VII. We encourage you use the SRAM, or the area may exceed the limitation. If you decide to use, remember to add the relative path of your DRAM to the filelist.f, and add the correspond .db to the 02_SYN/.synopsys_dc.setup.
VIII. TA has provided some python file for this lab under 00_TESTBED folder:
- pic_generation.py: It can generate dram.dat and dram.json file. dram.dat can be used by PATTERN.v and pseudo_DRAM.v. dram.json has the same data with dram.dat, but it is more readability for helping user to debugging.
- cam3A.py: It provide the algorithm for auto focus and auto exposure in python manner, you can refer the algorithm from this file.
- check_pattern.py: It can read the dram.dat and calculate the data by python, you can use this program to help u to debug.

IX. TA will use dram1.dat, dram2.dat and dram3.dat provide to you inside the 00_TESTBED/DRAM folder to demo your design. However, TA will also use some hidden cases to verify your design too. Fortunately, TA will only use the .dat file randomly produce from the pic_generation.py to test.

IV. Specifications

A. *Top module*

I. Top module name: ISP (filename: ISP.v)
II. Your design should trigger at positive edge clock, and the pattern will trigger and receive input/output signal at negative edge clock.
III. Remember to add the SRAM .v file to filelist.f if you have utilize SRAM in your design.
IV. Remember ensure the same dram.dat that your pattern.v and pseudo_DRAM.v using!

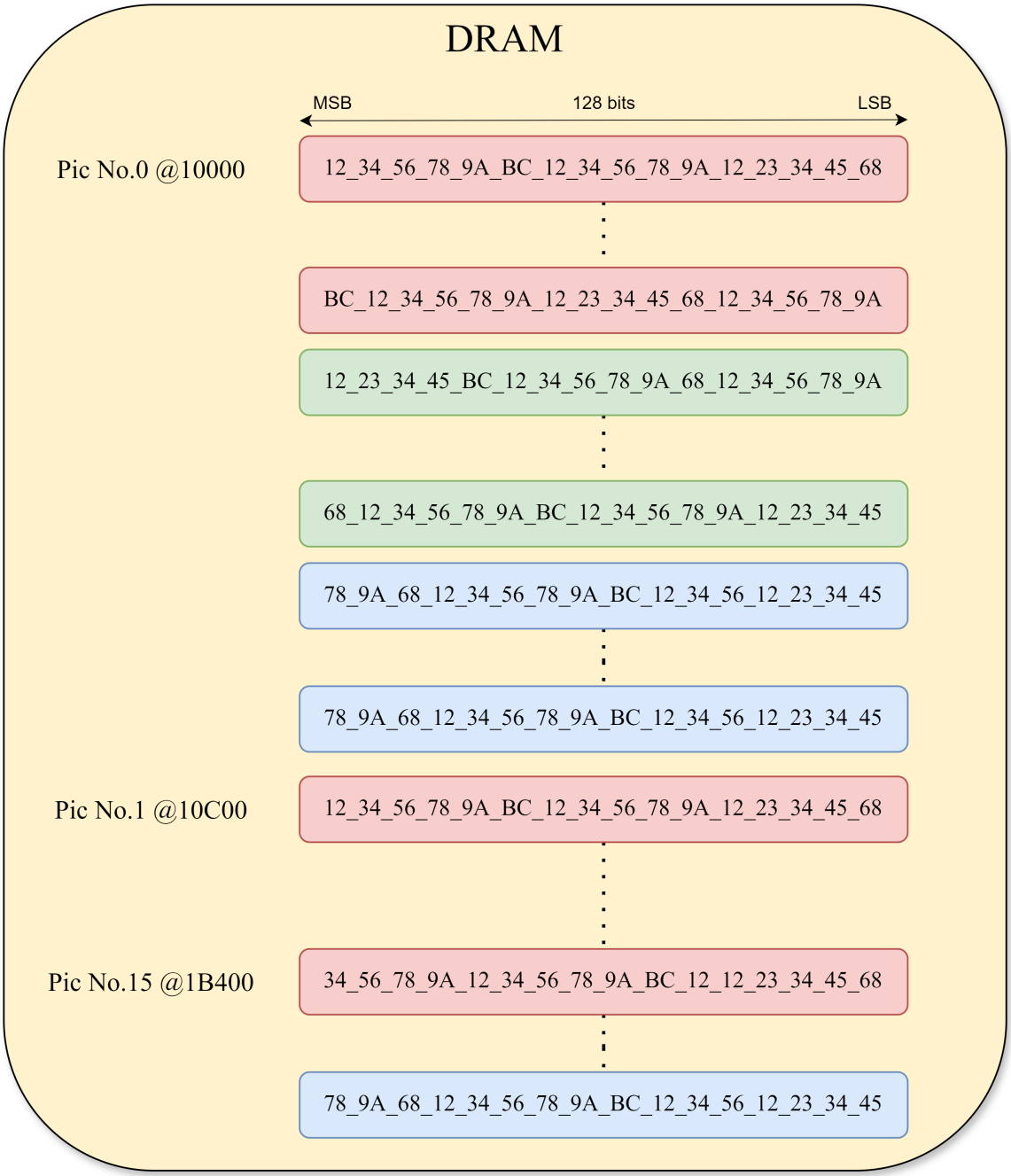


Fig. 3: DRAM content illustration.

B. Reset

I. It is asynchronous reset and active-low architecture. If you use synchronous reset (reset after clock starting) in your design, you may fail to reset signals.
II. The reset signal (rst_n) would be given only once at the beginning of simulation. All output signals (including AXI signals) should be reset after the reset signal is asserted.
III. All signal should be reset to 0.

C. Design Constraints

I. The maximum clock period is 20 ns.
II. Your latency should be less than 10000 cycles for each operation.
III. All outputs are synchronized at clock rising edge.
IV. You CANNOT use any designware IP in this lab.

D. Synthesis

I. The input delay and the output delay are $\frac{clock\_period}{2}$.
II. The output load should be set to 0.05.
III. The synthesis result of data type cannot include any LATCH.
IV. Synthesis time should less than 1 hour.
V. The total area (ISP.v and your SRAM) should be less than 750,000.
VI. Remeber to add .db file to the .synopsys_dc.setup file if you using SRAM.

E. Gate Level Simulation

I. The gate-level simulation cannot include any timing violations without the notimingcheck command.

F. APR

I. Core to IO boundary must be more than 100.
II. Hard macro must be placed inside core.
III. Power ring must be wire group, interleaving, at least 4 pairs, width at least 9.
IV. Stripes sets distance should be less than 200, width at least 4.
V. Timing slack shouldn't have any negative slacks after setup/hold time analysis (including SI).
VI. The DRV of (fanout, cap, tran) should be all 0 after post_Route setup/hold time analysis (including SI).
VII. No LVS violation after "verify connectivity".
VIII. No DRC violation after "verify geometry".
IX. Core filler must be added.

G. Supplement

I. Don't use any wire/reg/submodule/parameter name called *error*, *congratulation*, *latch* or *fail* otherwise you will fail the lab. Note: * means any char in front of or behind the word. e.g: error_note is forbidden.
II. Don't write Chinese or other language comments in the file you sent.
III. Verilog command for design compiler optimizations are forbidden.
IV. Any error messages during synthesize and simulation, regardless of the result will lead to failure in this lab.
V. Any form of display or printing information in Verilog design is forbidden. You may use this methodology during debugging, but the file you turn in should not contain any coding that is not synthesized.
VI. Plagiarism is strictly prohibited!

V. Grading

- 70%: Functionality
- 30%: (ISP Area, including SRAM) * (latency * cycle_time)^2
- 30% off for 2nd demo.

VI. Notes

I. Submit your files throught 09_SUBMIT in the final project folder
   - 1st_demo deadline: 2024/12/31 (Tue) 12:00:00 (noon)
   - 2nd_demo deadline: 2025/01/03 (Fri) 12:00:00 (noon)

II. The file needs to be submit in final project:
   - ISP.v
   - filelist.f
   - .synopsys_dc.setup
   - 04_MEM folder (Should contain all the necessary MEMORY files (*.v, *.db, *.lef, *.lib) for the synthesis and APR.)
   - APR files: CHIP.inn, CHIP.io, CHIP.sdc, CHIP.sdf, CHIP.v, CHIP.inn.dat

   The cycle time of your design will be submitted by '00_tar xx.x'.

VII. Conclusion

This project presents a comprehensive exercise in designing and implementing crucial components of an Image Signal Processor (ISP) for camera systems. By focusing on the auto-focus and auto-exposure algorithms, students are challenged to apply their knowledge of hardware description languages in a practical context that mirrors real-world imaging applications. The integration of the AXI4 transmission protocol for DRAM interaction adds an extra layer of complexity, simulating the constraints and considerations of actual system designs.

This project not only reinforces fundamental concepts in digital design but also introduces students to advanced topics in image processing and system-on-chip architecture. The emphasis on efficient memory management through SRAM utilization encourages students to consider hardware resource optimization, a critical skill in VLSI design. Furthermore, the strict design rules and synthesis constraints provide valuable experience in meeting industry-standard requirements for timing, area, and power efficiency.

By engaging with this multifaceted assignment, students will gain a deeper understanding of the intricate relationship between algorithms, hardware design, and system-level considerations. This holistic approach to learning will undoubtedly prepare them for the challenges they may face in their future careers in the field of electronic engineering and computer architecture.

VIII. Reference Waveform

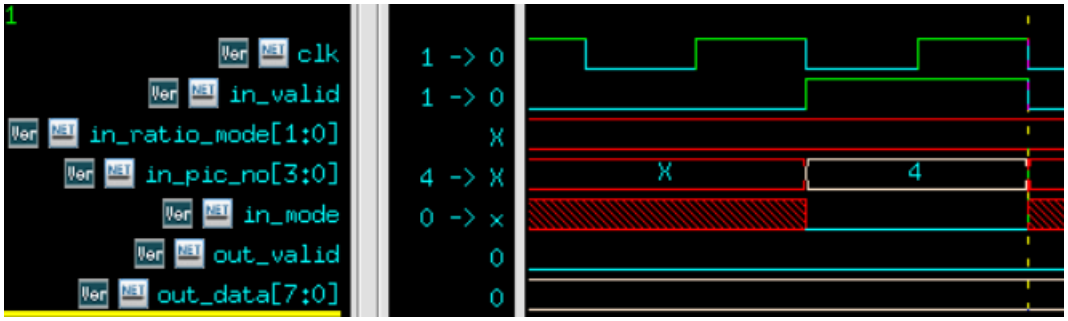


Fig. 4: Waveform for input mode 0: Auto Focus.

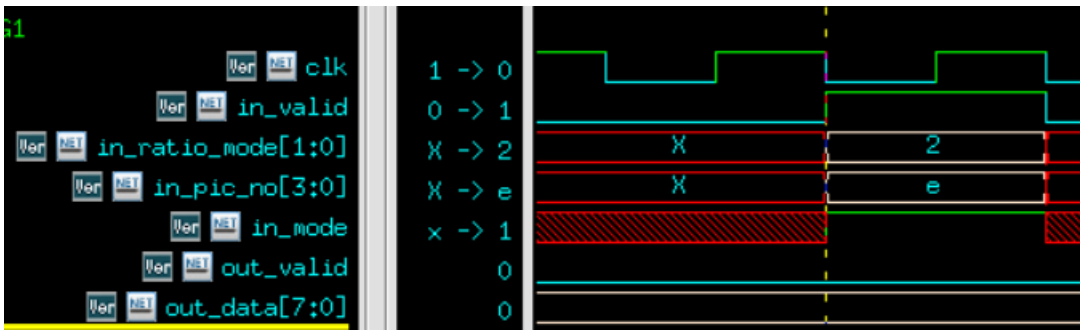


Fig. 5: Waveform for input mode 1: Auto Exposure.

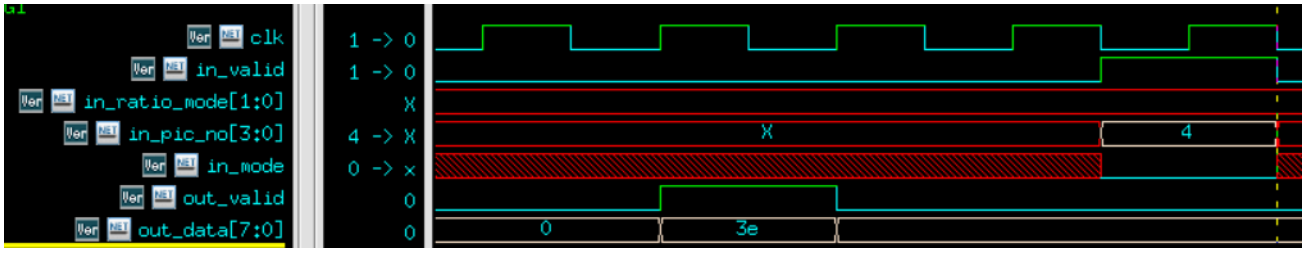


Fig. 6: Output waveform, the next input valid signal will come after 1 cycle.

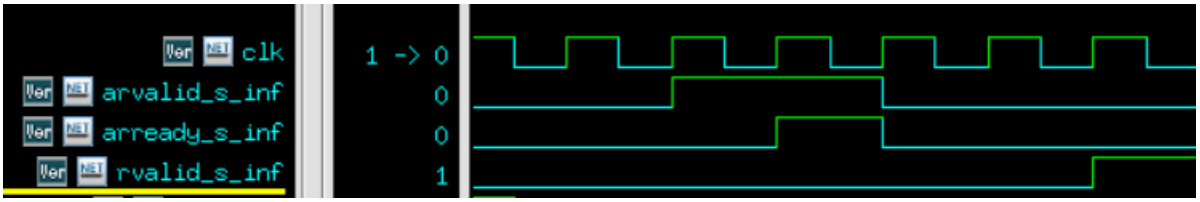


Fig. 7: Waveform for read latency set to 1.

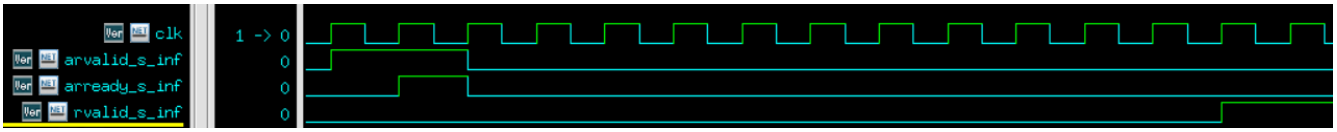
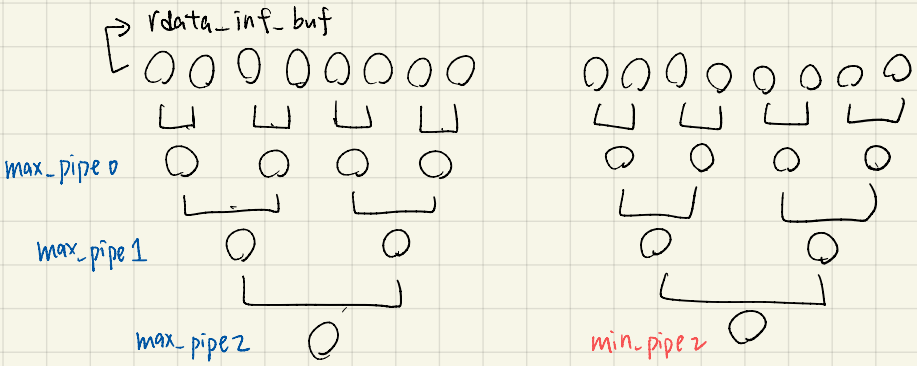


Fig. 8: Waveform for read latency set to 10.



don't use sorting since more register is needed.....

0 1 2 3 4

buf

buf 0

0 1

1 2] max和0比, min和2比

2

最後算cnt存到max_R

always @(*)

case(cnt)...

max_image_in1 = max_image_out;

max_image_in2 =

max_image