TAIWAN TECH National Taiwan University of Science and Technology

An Energy-Efficient CNN Accelerator Specialized for PYNQ-Z2 FPGA Device

Student: Chao-Chia Lin

Dept. Electronic and Computer Engineering

National Taiwan University of Science and Technology

Outline

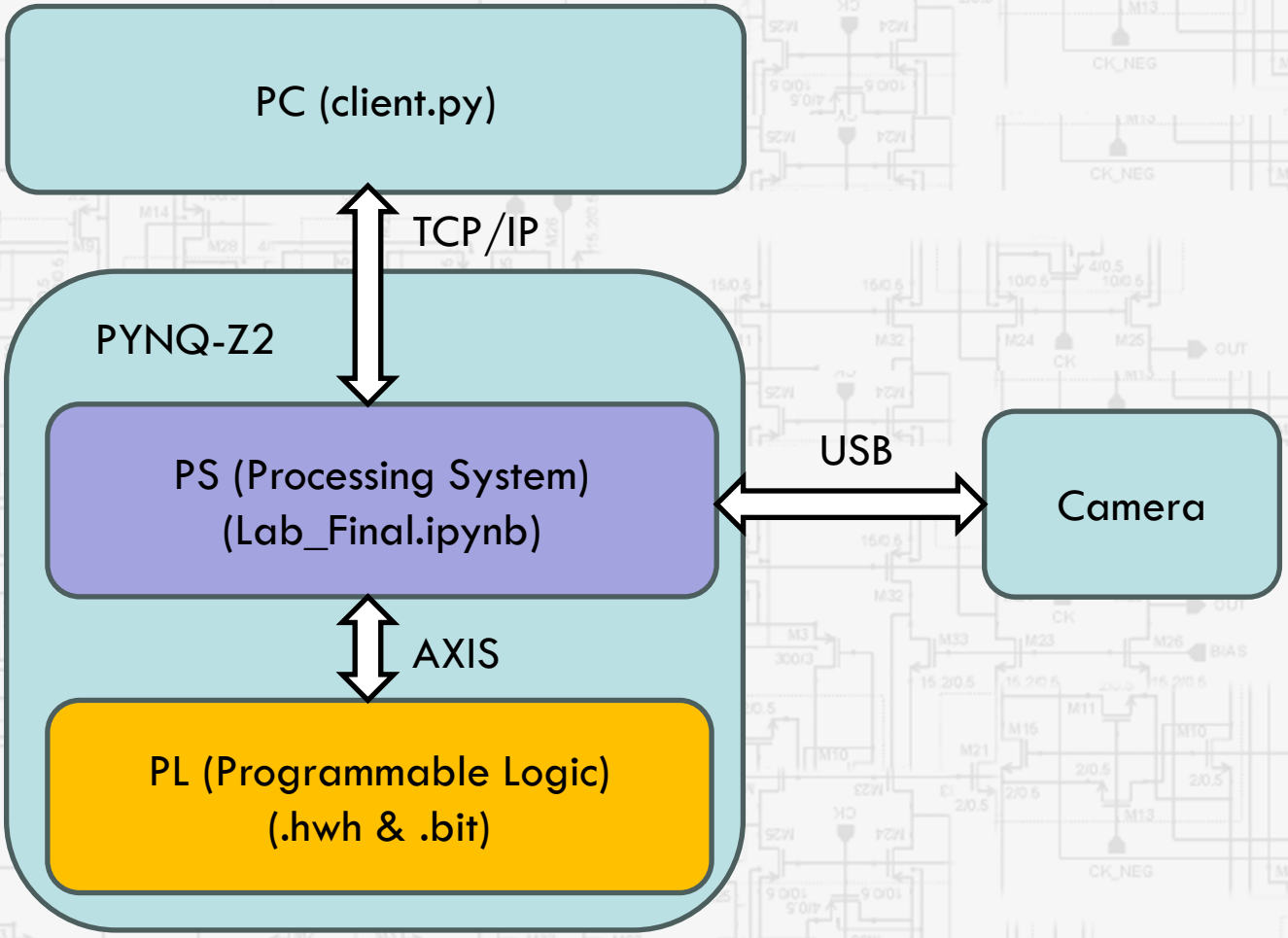
- Introduction
- CNN Model
- Hardware architecture
- Design flow
- Result
- Conclusion
- References

Introduction

- The aim of this study is to build a low power CNN accelerator specialized for PYNQ-Z2.
- The employed CNN model is to classify the gender based on input face images.
- Based on the model architecture, we construct all the layers by pure logical circuits.
- The system is complete from software to hardware to real-time application.

Introduction

- System architecture:

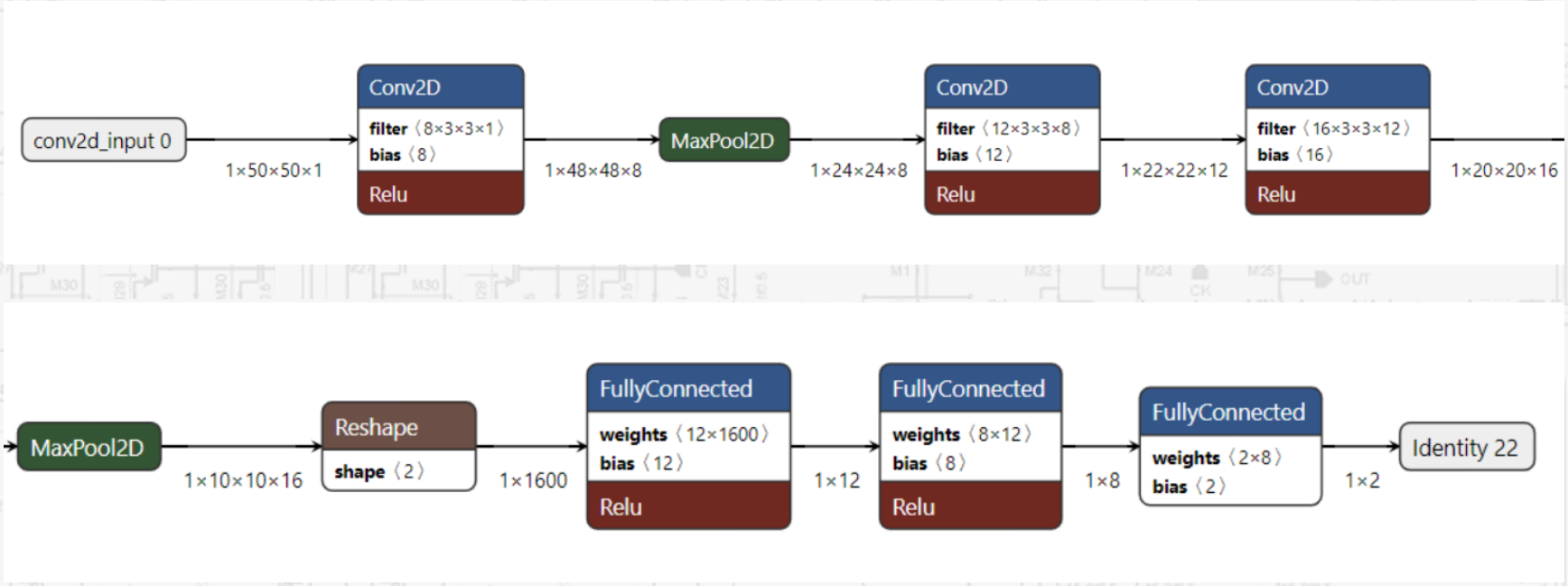


Objective

- To deploy a software-based model into a hardware circuits
- To leverage the virtue of quantization
- To make data available for hardware implementation
- To design the hardware architecture for CNN layers
- To realize the communication between heterogeneous systems.
- To validate the correctness of the design
- To turn a complete hard-software system into a real-time application

CNN Model

- The trained model is built on the following architecture:



Quantization

- Integer quantization

$$r = S(q - Z)$$

r: real number
q: integer
S: scale
Z: zero point

Quantization

- Calculate convolution in integer

$$r_3 = \sum r_1 r_2$$

$$S_3(q_3 - Z_3) = \sum S_1(q_1 - Z_1) S_2(q_2 - Z_2)$$

$$q_3 = Z_3 + \mathrm{M} \sum (q_1 - Z_1)(q_2 - Z_2)$$

$$\mathrm{M} = \frac{S_1 S_2}{S_3} = 2^{-n} \mathrm{M}_0$$

Quantization

- Reduce the computation of quantized convolution

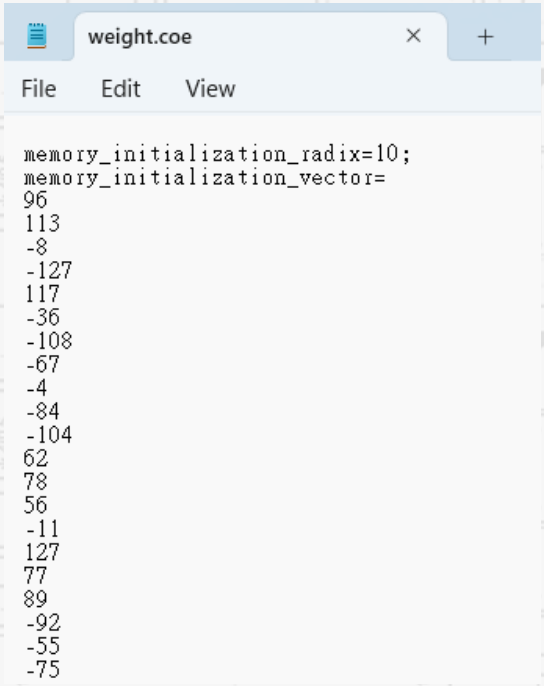
$$q_3 = Z_3 + \mathrm{M} \sum^{N} (q_1 - Z_1)(q_2 - Z_2)$$

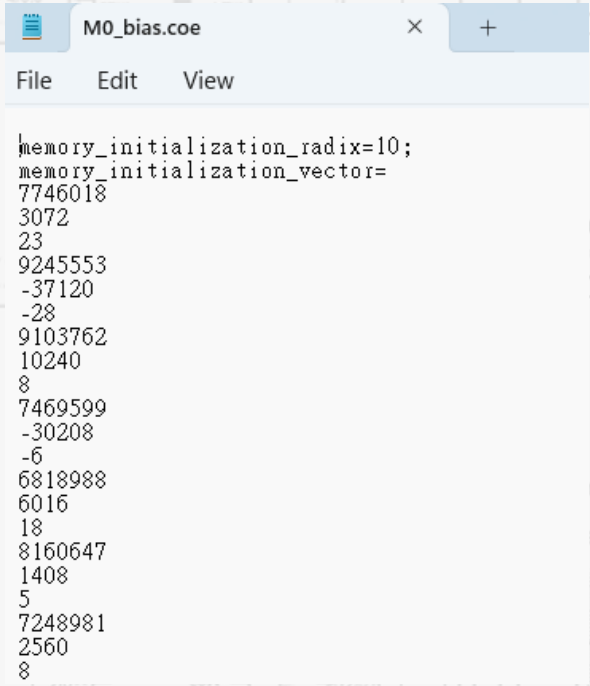
$$q_3 = Z_3 + M\left(NZ_1Z_2 - Z_2a_1 - Z_1a_2 + \sum q_1q_2\right)$$

$$a_1 = \sum^{N} q_1 \qquad a_2 = \sum^{N} q_2$$

Quantization

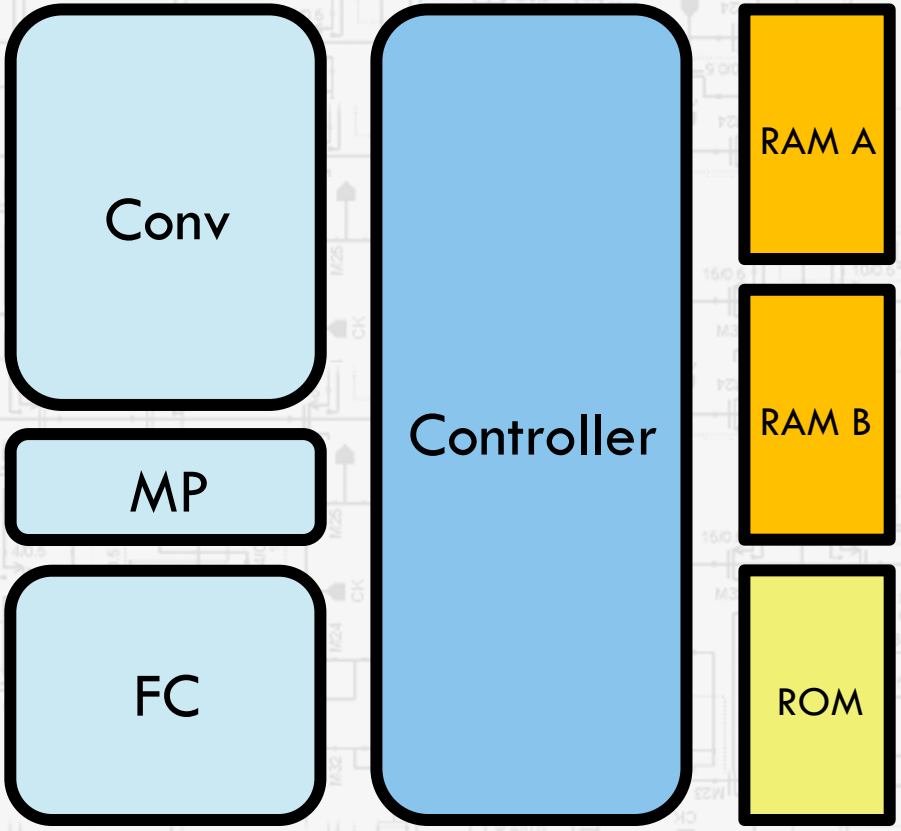
- Weight and Quantization parameters are saved in .coe file.
- They will be the reference source for ROMs.





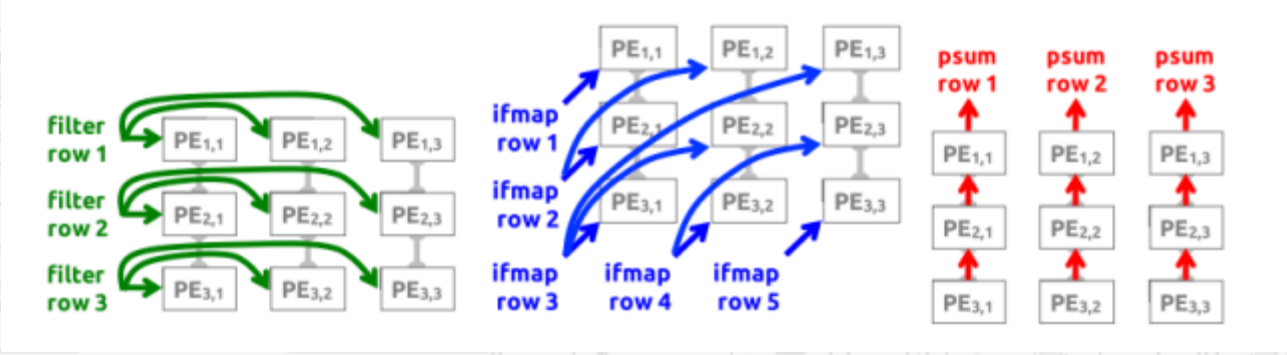
Hardware architecture

- PL architecture:



Hardware architecture

- Convolution is built based on the Eyeriss architecture to achieve best efficiency.
- 5 rows of input data and 3 rows of filter are first buffered simultanously and then go through a convolution pipeline.
- The partial sums of each row are produced every cycle.

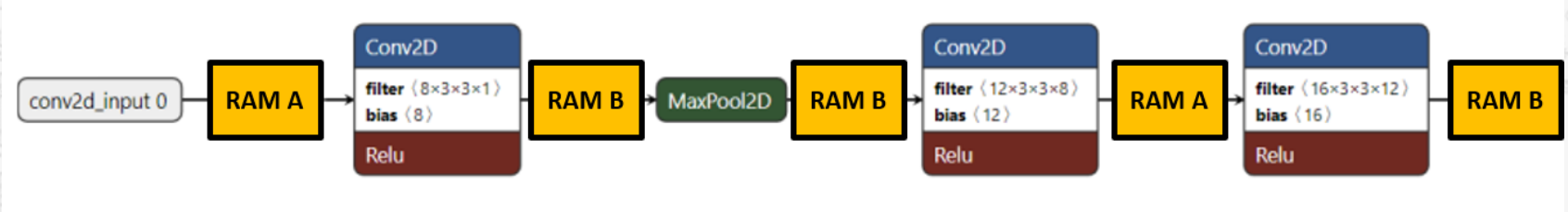


Hardware architecture

- Efficient partial sum (psum) accumulation and minimizing data movement become essential to conserve storage space and memory read/write energy.
- However, achieving maximum input data reuse while immediately reducing psums is challenging, as psums generated by Multiply and Accumulate (MAC) operations using the same filter or ifmap value are not readily reducible.
- Our strategy is to complete the psum of the output feature map at the earliest possible, also leverage synchronous computing to balance the reuse ratio and storage resources.

Hardware architecture

- Our Convolution circuit is constructed to be size-flexible for every layer, by setting the parameters as input from Controller
- Beside Convolution reuse, RAM reuse is another power saving strategy by reducing the number of active RAM
- Maxpooling is implemented in SISO style

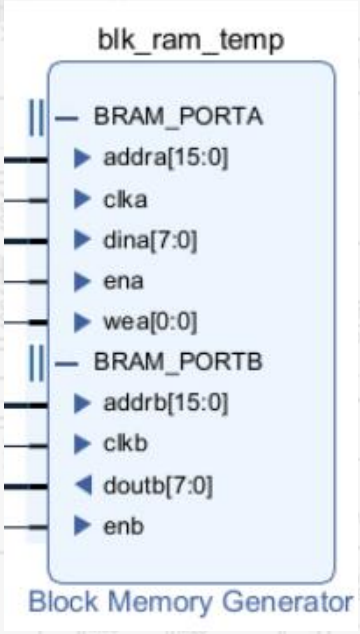


Hardware architecture

- The first layer of Fully Connected Layers require the most computation resources and memory cost
- It is not possible to load every feature map and weight into the chip and perform cross-multiplication at the same time
- To balance the hardware requirement between layers, we apply partial sum strategy, which load a small amount of data every stage and accumulate the multiplication result
- For the last two layers, we reuse the same hardware and can infer each output node immediately without any accumulation

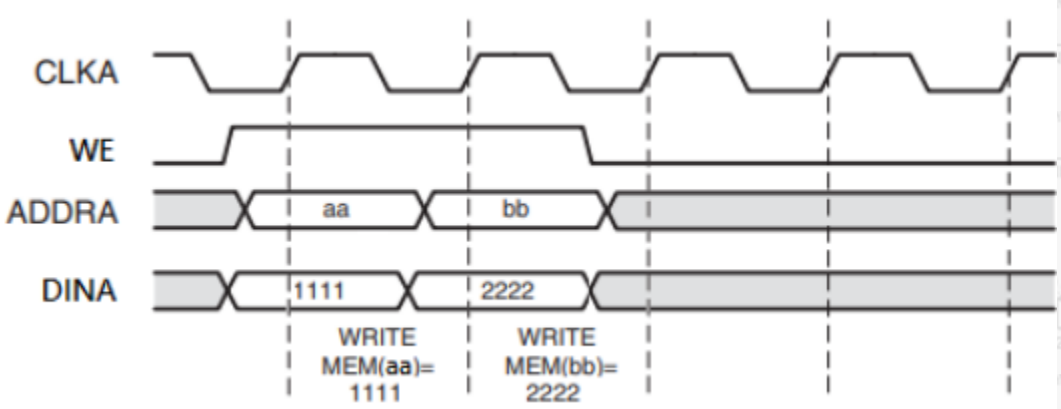
Hardware architecture

- Data Block RAM
  - Can be used to store temporary data. Storing data in BRAM instead of the register can reduce circuit area
  - The Simple Dual-port RAM provides two ports, A and B. Write access to the memory is allowed through port A, and Read access is allowed through port B



Hardware architecture

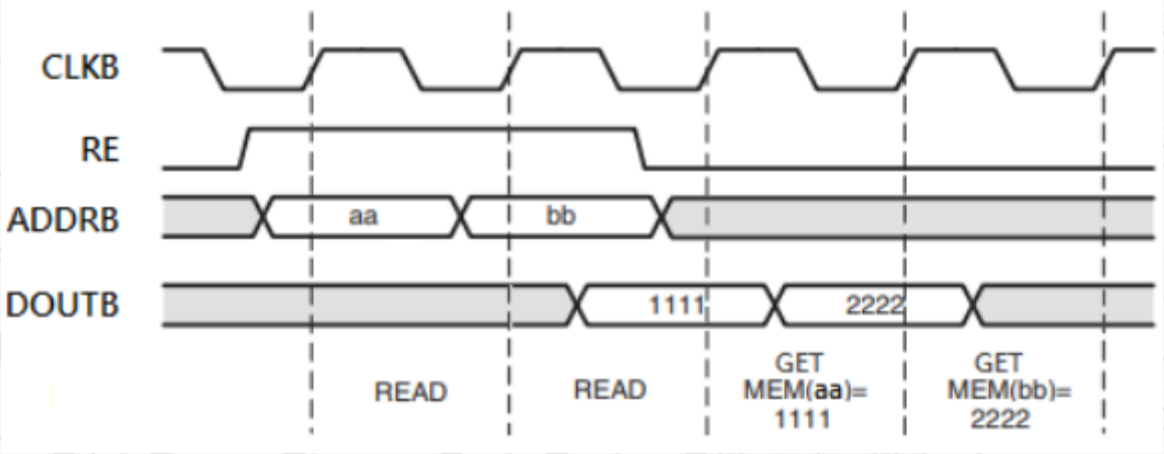
- Data Block RAM (Port A/Write)



BRAM_PORTA	
clka	Write Clock
wea	Write Enable
addra[8:0]	Write Address
dina[31:0]	Data Input
ena	Enable

Hardware architecture

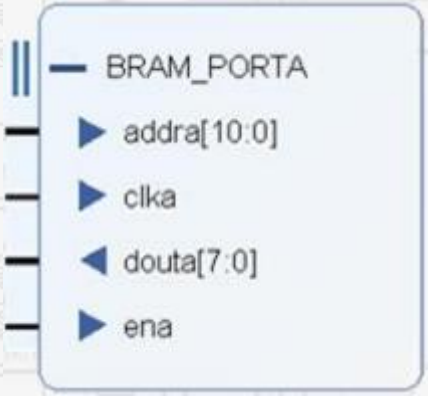
- Data Block RAM (Port B/Read)



BRAM_PORTB	
clkb	Read Clock
rea	Read Enable
addrb[8:0]	Read Address
doutb[31:0]	Data Output
enb	Enable

Hardware architecture

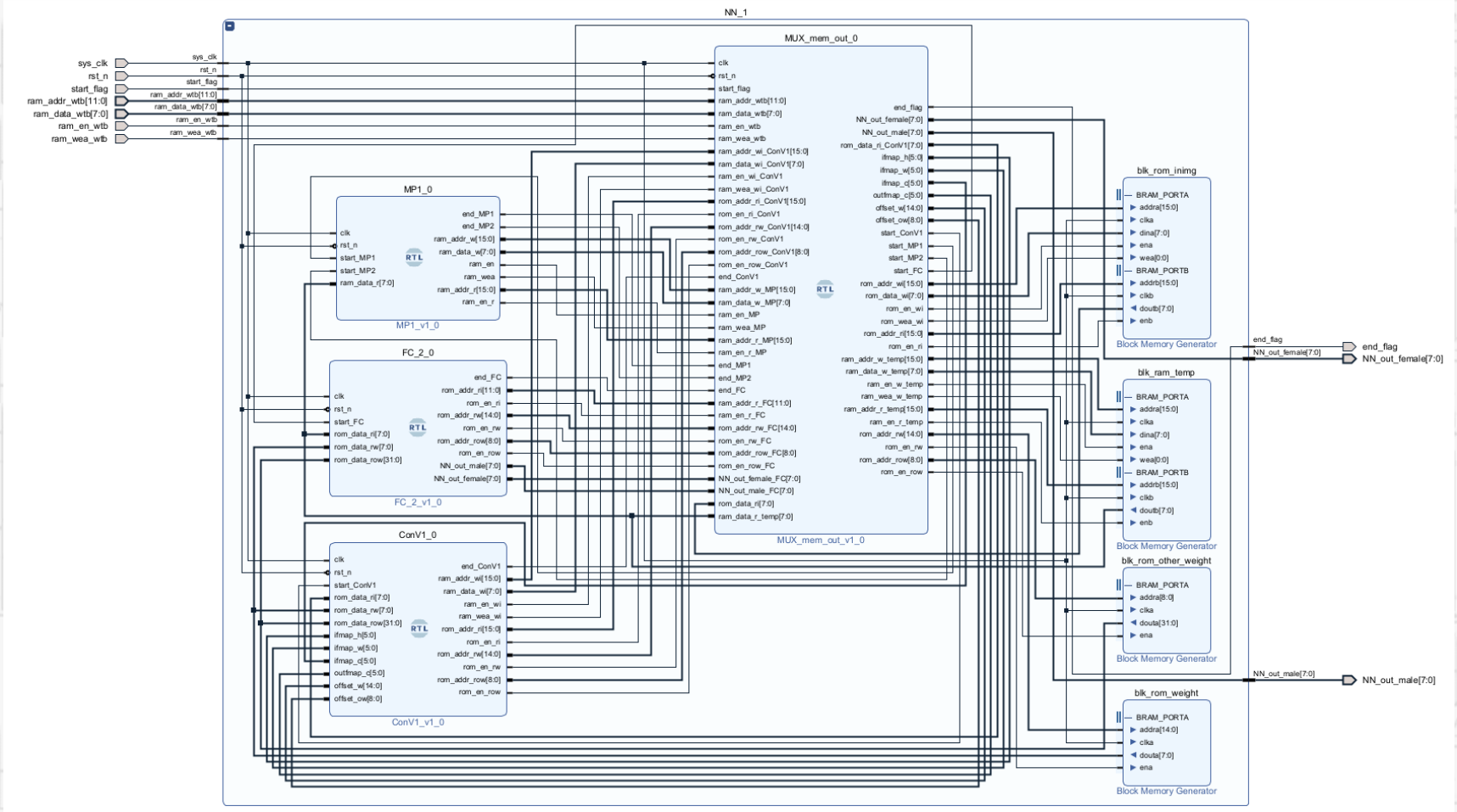
- Weight Block ROM
  - All weights have been stored to weight ROM
  - The Single-port ROM allows Read access to the memory space through a single port.



BRAM_PORTA	
clka	Read Clock
ena	Read Enable
addra[10:0]	Read Address
doutb[7:0]	Data Output

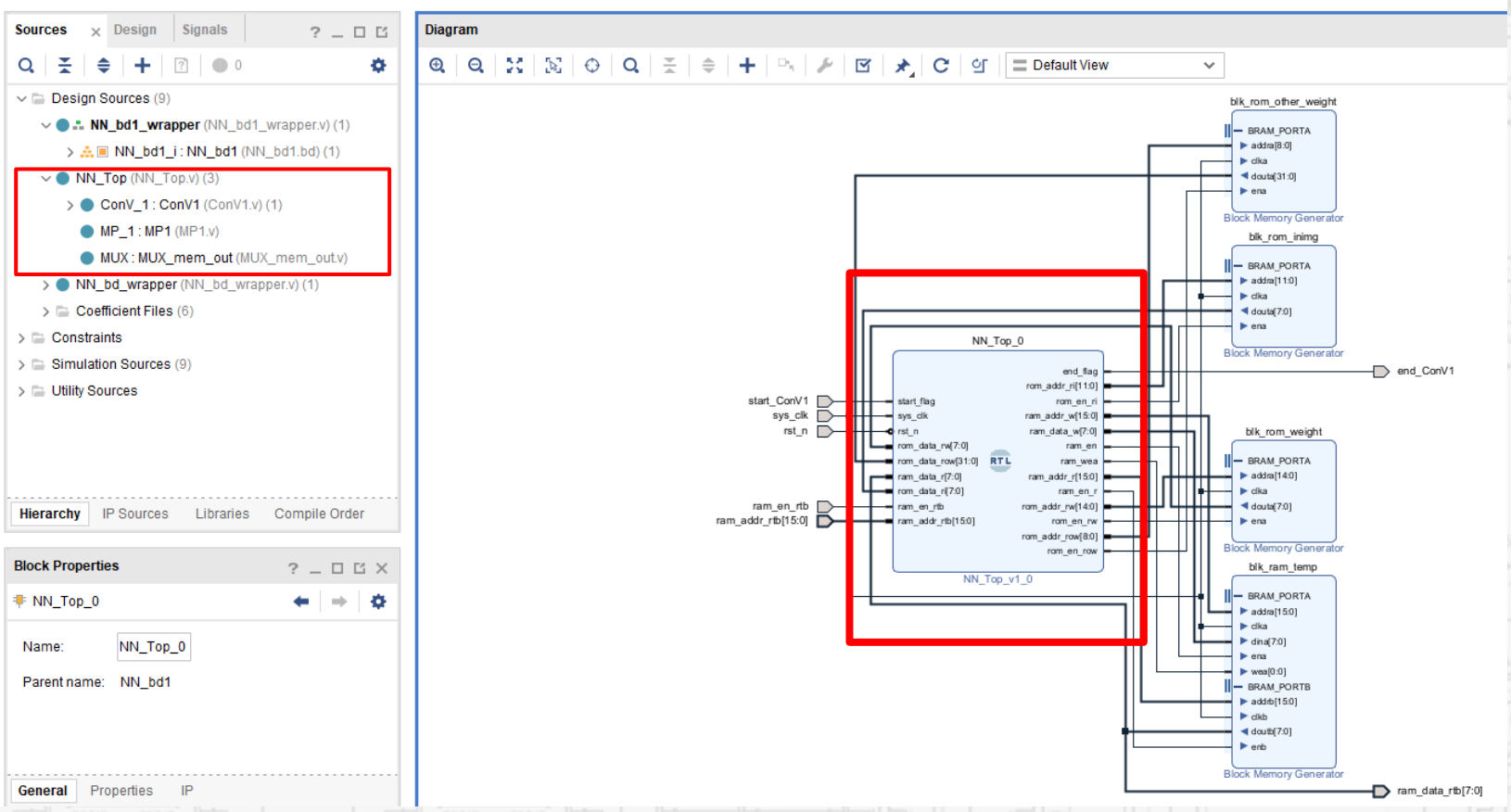
Hardware architecture

- Final block design of CNN model:



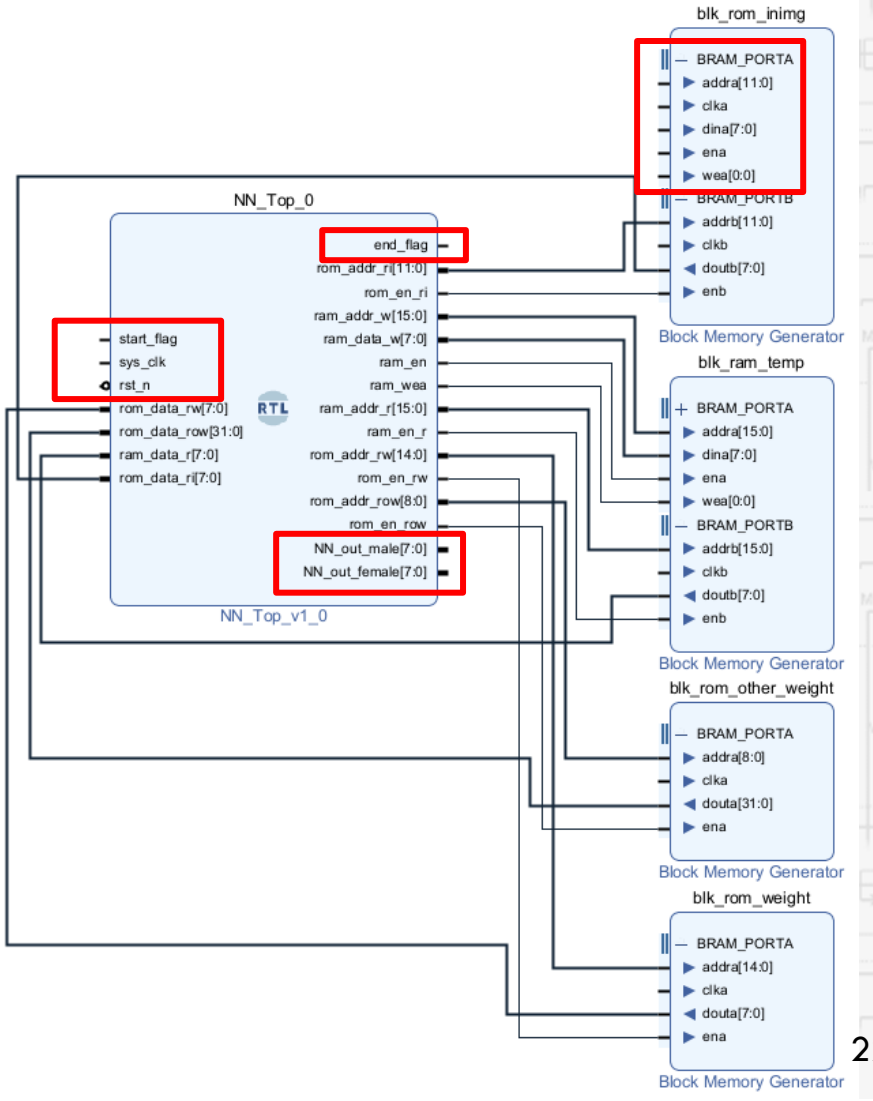
Design Flow

- Step 1: Package the CNN block design



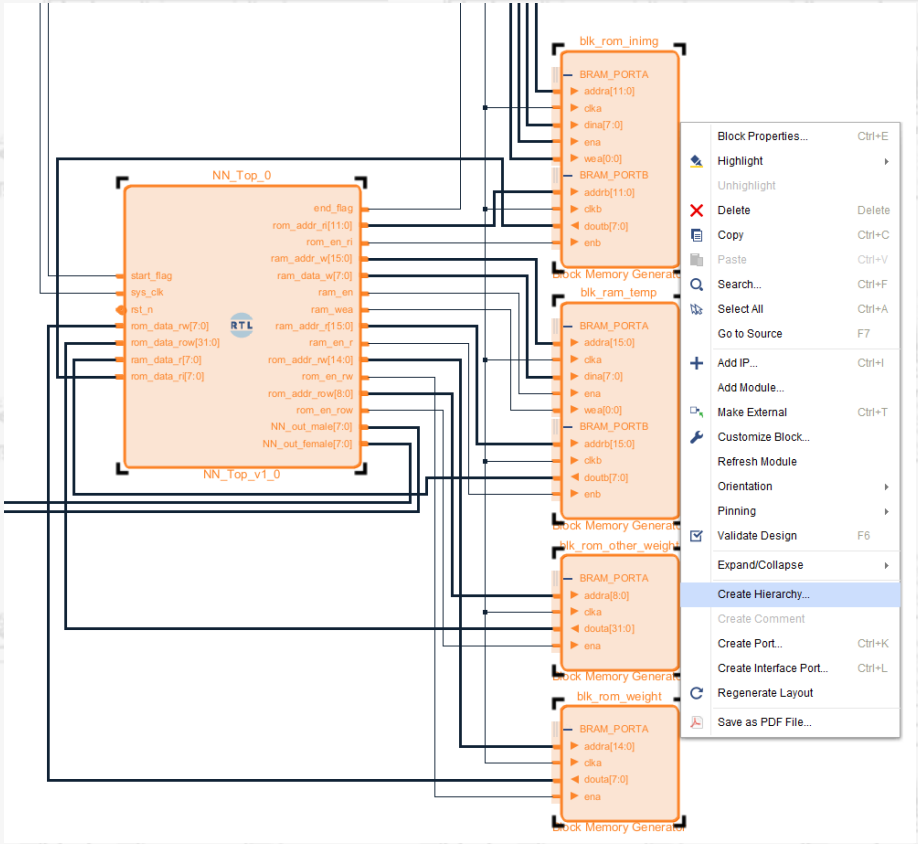
Design Flow

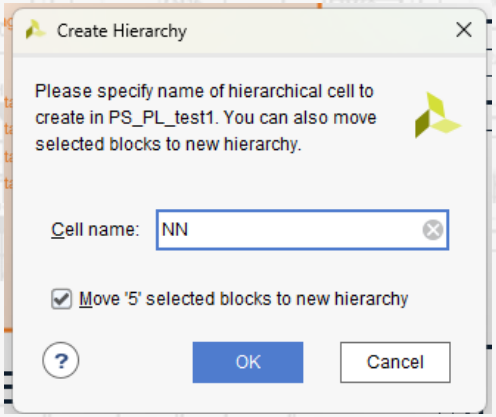
- Step 2: Wiring the components
- Preserve ports for interface
  - Global
    - sys_clk
    - reset_n
  - RAM_inimg
    - PORTA all
  - NN_top
    - NN_start
    - NN_end
    - [7:0]NN_out_male
    - [7:0]NN_out_female

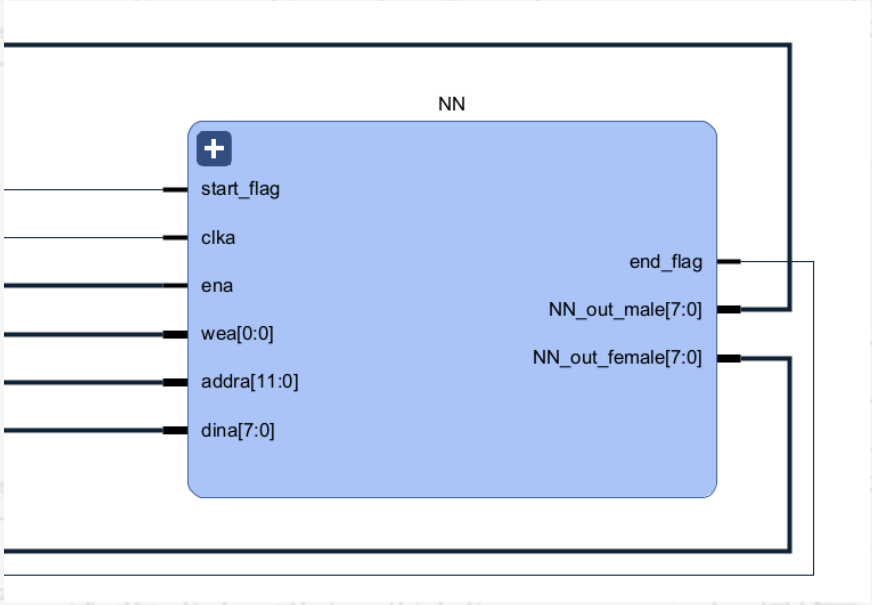


Design Flow

- Step 3: Create hierarchy

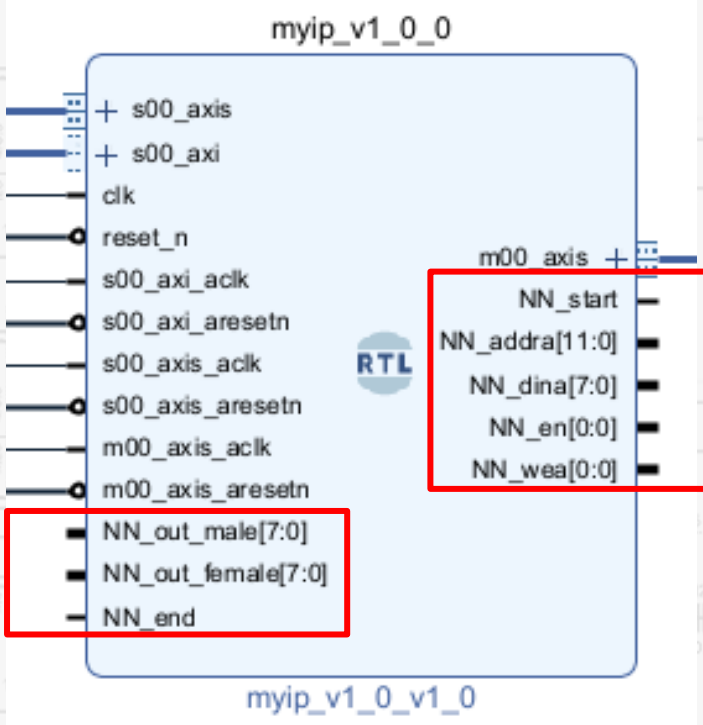






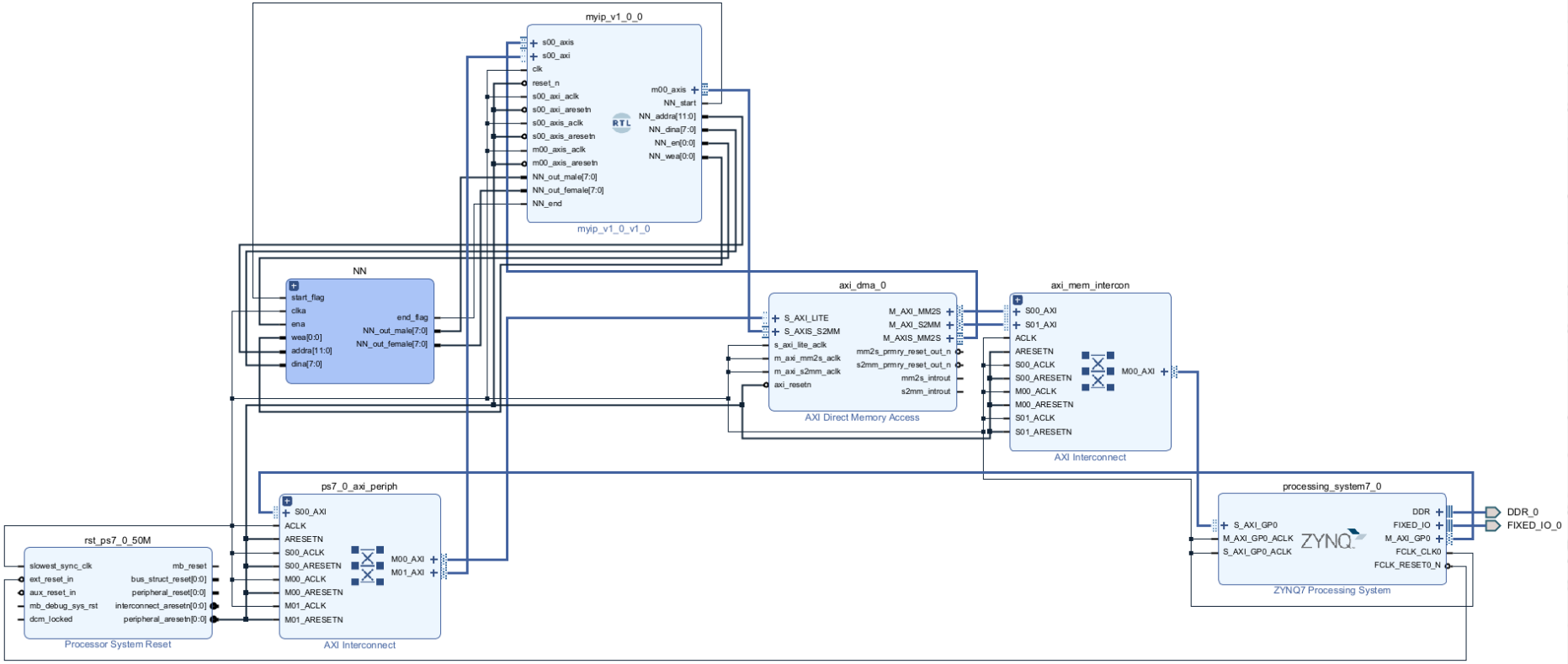
Design Flow

- Step 4: Connect the CNN and myip
  - myip is the communication bridge between PS and NN



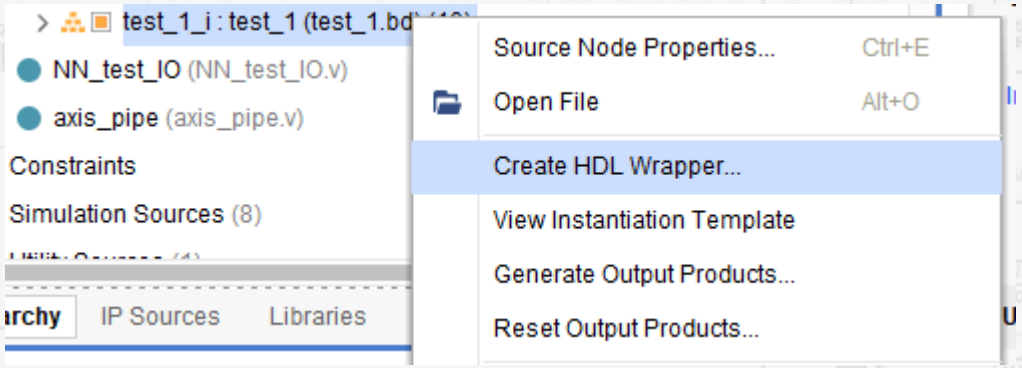
Design Flow

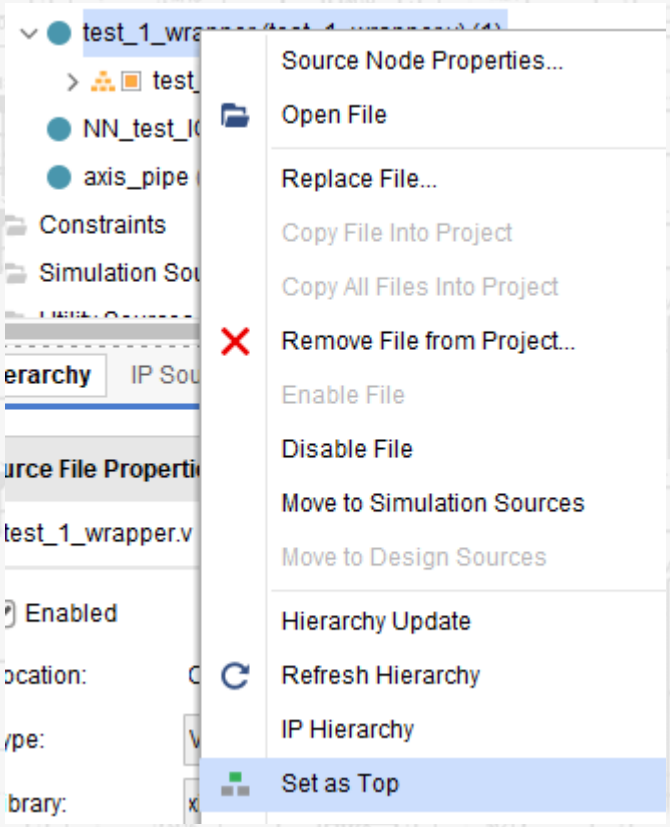
- Block design



Design Flow

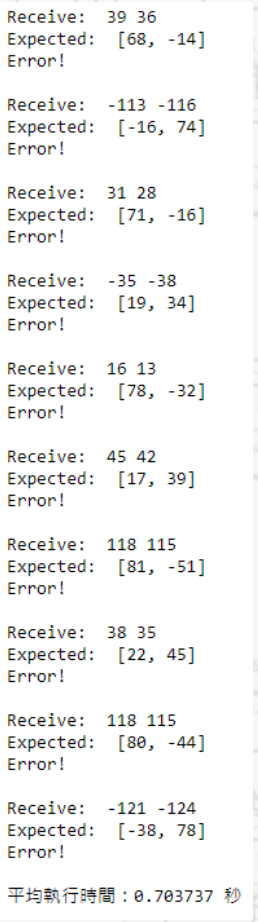
- Create HDL Wrapper
- Set as Top
- Run generate bitstream





Integration Workflow

1. Connect PYNQ to PC and Webcam
2. Open Vivado project
3. Generate Bitstream directly
4. Create a new folder on PYNQ server
5. Upload Bitstream (.hwh & .bit) to the folder
6. Upload and run Lab_Final.ipynb
7. Check the Output
8. Run client.py on PC
9. Type 192.168.2.99 (default PYNQ's IP)
10. It will show the output in real-time



Result

- Utilization report:

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	Block RAM Tile (140)	DSPs (220)	Bonded IOPADs (130)	BUFGCTRL (32)
N Lab_final_bd_wrapper	40.65%	16.88%	4.61%	3.37%	49.41%	40.34%	0.94%	30.36%	25.45%	100.00%	3.13%
Lab_final_bd_i (Lab_final_bd)	40.65%	16.88%	4.61%	3.37%	49.41%	40.34%	0.94%	30.36%	25.45%	0.00%	3.13%

- Utilization of each component in CNN:

NN_1 (NN_1_imp_1SXG1K6)	36.33%	13.77%	4.61%	3.37%	42.56%	36.33%	0.00%	28.93%	25.45%	0.00%	0.00%
blk_ram_temp (Lab_final_	0.02%	<0.01%	0.00%	0.00%	0.10%	0.02%	0.00%	11.43%	0.00%	0.00%	0.00%
blk_rom_inimg (Lab_final_	0.03%	<0.01%	0.00%	0.00%	0.10%	0.03%	0.00%	11.43%	0.00%	0.00%	0.00%
blk_rom_other_weight (La	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.36%	0.00%	0.00%	0.00%
blk_rom_weight (Lab_final	0.04%	<0.01%	0.03%	0.00%	0.08%	0.04%	0.00%	5.71%	0.00%	0.00%	0.00%
ConV1_0 (Lab_final_bd_C	25.38%	10.96%	4.58%	3.37%	30.21%	25.38%	0.00%	0.00%	3.64%	0.00%	0.00%
FC_2_0 (Lab_final_bd_FC	8.93%	1.98%	0.00%	0.00%	10.58%	8.93%	0.00%	0.00%	21.82%	0.00%	0.00%
MP1_0 (Lab_final_bd_MP1	1.69%	0.80%	0.00%	0.00%	2.08%	1.69%	0.00%	0.00%	0.00%	0.00%	0.00%
MUX_mem_out_0 (Lab_fir	0.24%	0.02%	0.00%	0.00%	0.53%	0.24%	0.00%	0.00%	0.00%	0.00%	0.00%

Result

- Timing report after Run Implementation (cycle time constraint is 20ns):

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	1.136 ns	Worst Hold Slack (WHS):	0.033 ns	Worst Pulse Width Slack (WPWS):	9.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	29575	Total Number of Endpoints:	29575	Total Number of Endpoints:	14787

All user specified timing constraints are met.

Conclusion

- In this project, we complete a simple CNN model implemented by hardware and deployed to fit a FPGA devices.
- It is important to note that we did not consider the data transferring network, which is crucial in most of the computational application. They can be achieved with Network on Chip (NoC) and efficient communication system between PEs.
- Additionally, future iterations could explore the integration of more advanced models, such as ResNet or YOLO, to broaden the versatility of the CNN accelerator.

References

- Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks." IEEE journal of solid-state circuits 52.1 (2016): 127-138.