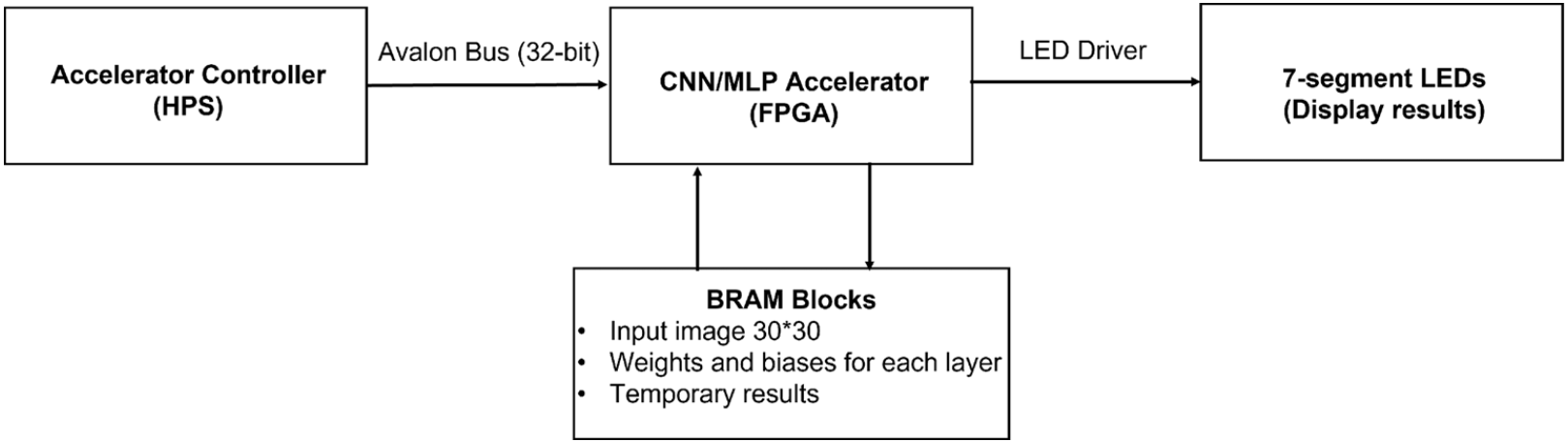
CNN Accelerator based on FPGA

CSEE 4840 Embedded Systems Course Project

Tianchen Yu, Haichun Zhao, Haomiao Li, Qixiao Zhang, Yue Niu

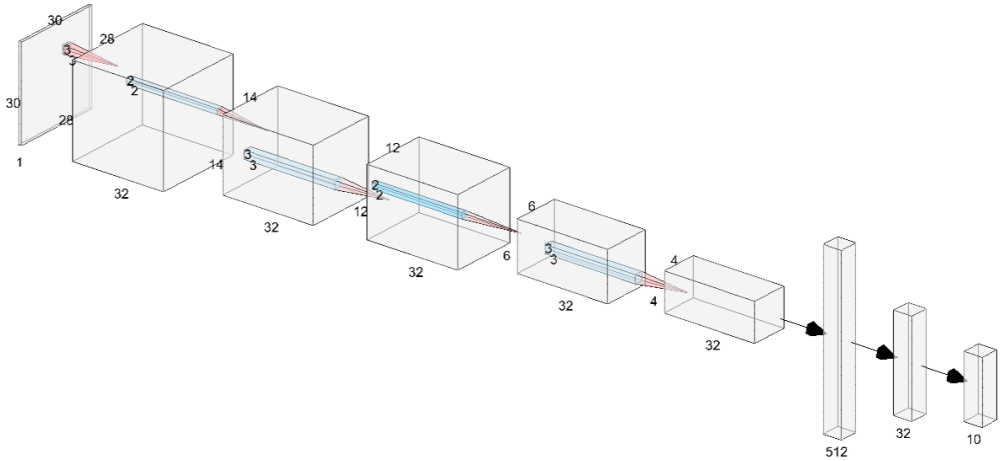
Project Intro

Target: Use FPGA to accelerate the inference process of a CNN architecture



Import parameters and image → Run inference purely on hardware → Display result

CNN Architecture

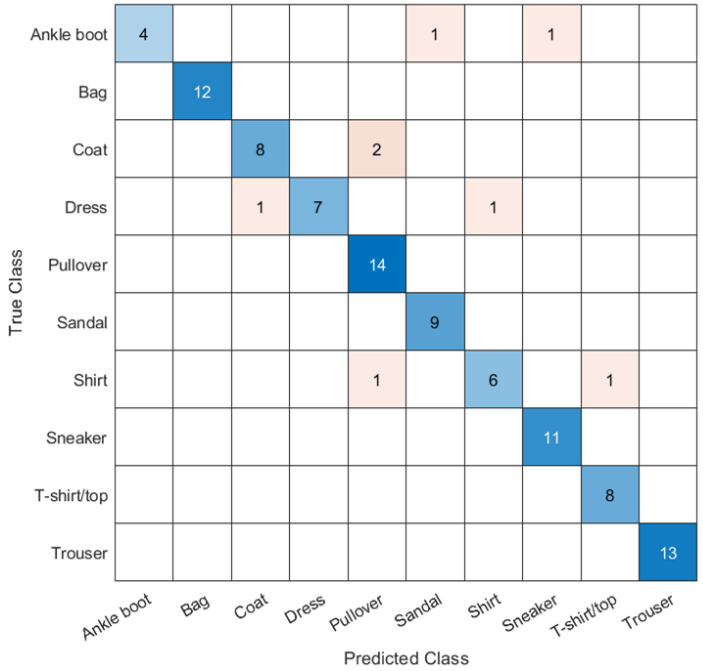



```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(30, 30, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(10))
```

MATLAB Golden Model

Purpose: Model every single output from accelerator for verification

```
%perform the convolution and max pooling process
conv2d1_result = conv2d(img_f8,  conv2d1_weight_2D_f8, conv2d1_bias_f8, F8, wordlength8, fractionlength8,  1);      %f
pooling1_result = maxpooling2by2(conv2d1_result, F8, wordlength8, fractionlength8);                              %fir

conv2d2_result = conv2d(pooling1_result, conv2d2_weight_2D_f8, conv2d2_bias_f8, F8, wordlength8, fractionlength8, 1); '
pooling2_result = maxpooling2by2(conv2d2_result, F8, wordlength8, fractionlength8);                               %s

conv2d3_result = conv2d(pooling2_result, conv2d3_weight_2D_f8, conv2d3_bias_f8, F8, wordlength8, fractionlength8, 1); '

flatten_result = flatten(conv2d3_result, F8, wordlength8, fractionlength8);

dense1_result = dense(flatten_result, dense1_weight_f8, dense1_bias_f8, F8, wordlength8, fractionlength8, 1);

dense2_result = dense(dense1_result, dense2_weight_f8, dense2_bias_f8, F8, wordlength8, fractionlength8, 0);
```



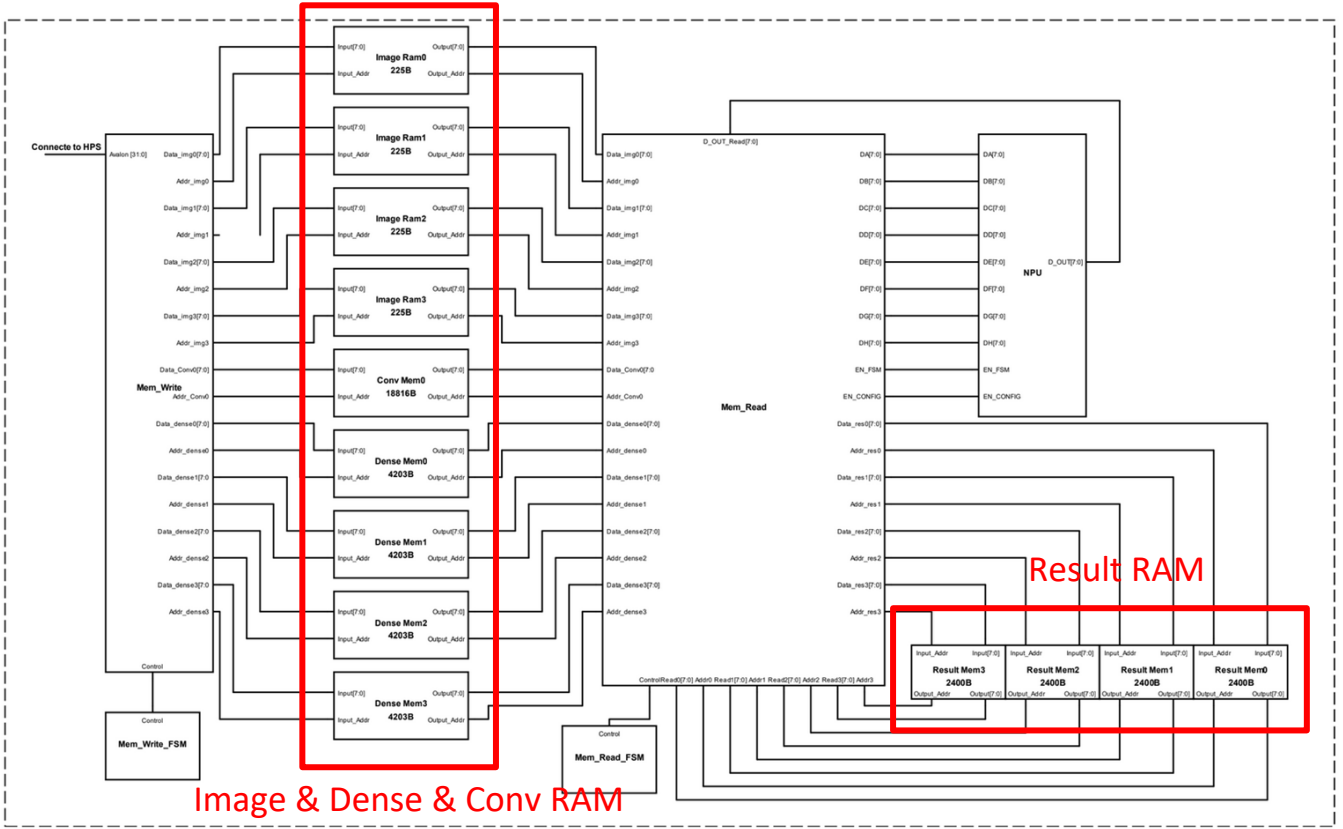
Input/Output: 8-bit fixed point
Computation (Addition & Multiplication): 16-bit fixed point

Accuracy in TF: ~95%
Accuracy in MATLAB: ~92%

Hardware Top Level

Hardware Blocks:

- Mem_Read
- Mem_Write
- 4 image RAM
- 4 Dense RAM
- 4 Result RAM
- 1 Conv RAM
- NPU (Processing Core)



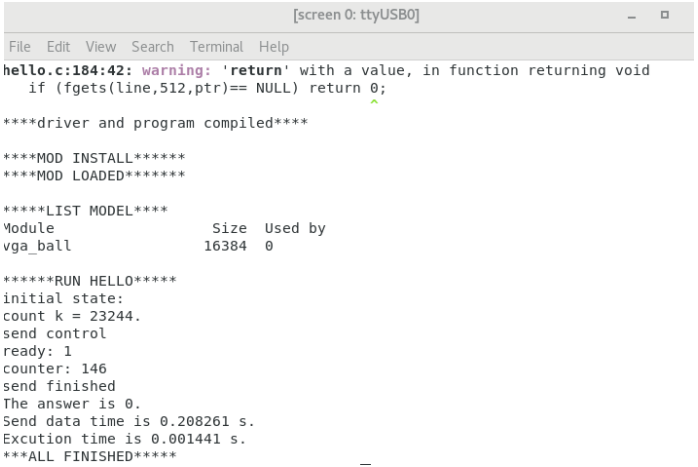
HW/SW Interface

- 32-bit Avalon bus: pass the image, weights and biases data before all the computation.
- Bus connected to “data_reg” and “control_reg” 32-bit registers in hardware
- “set_data” function passes Avalon bus data to “data_reg”
- “set_control” functions passes Avalon bus data to “control_reg”
- “read_ready” functions continuously check if the computation is completed
- “read_answer” functions read the computation result from the FPGA

HW/SW Interface

Current Problem:

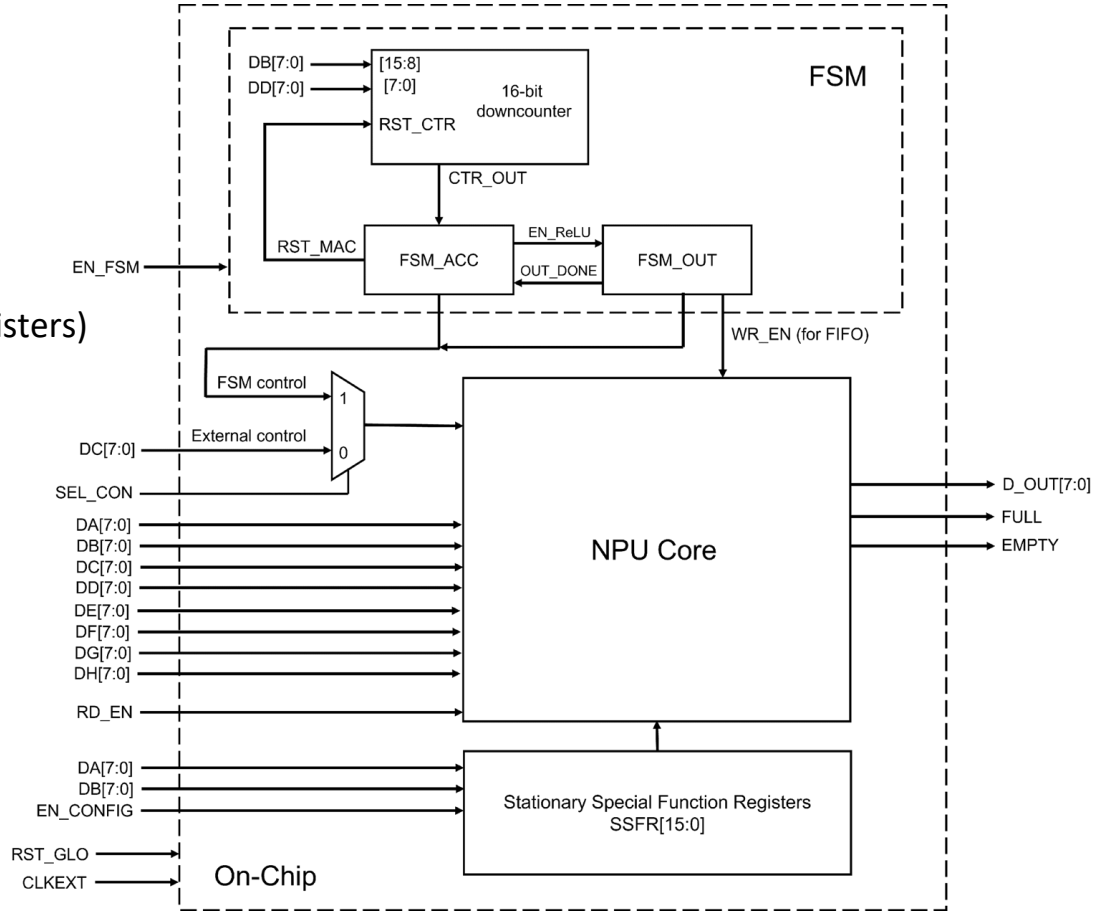
- The frequency of software (HPS) is higher than the frequency of hardware (FPGA)
- Timing requirement of Mem_Write is very strict, which cannot be fulfilled by HPS
- It is difficult to perfectly align the timing of software and hardware



NPU Architecture

Three main blocks:

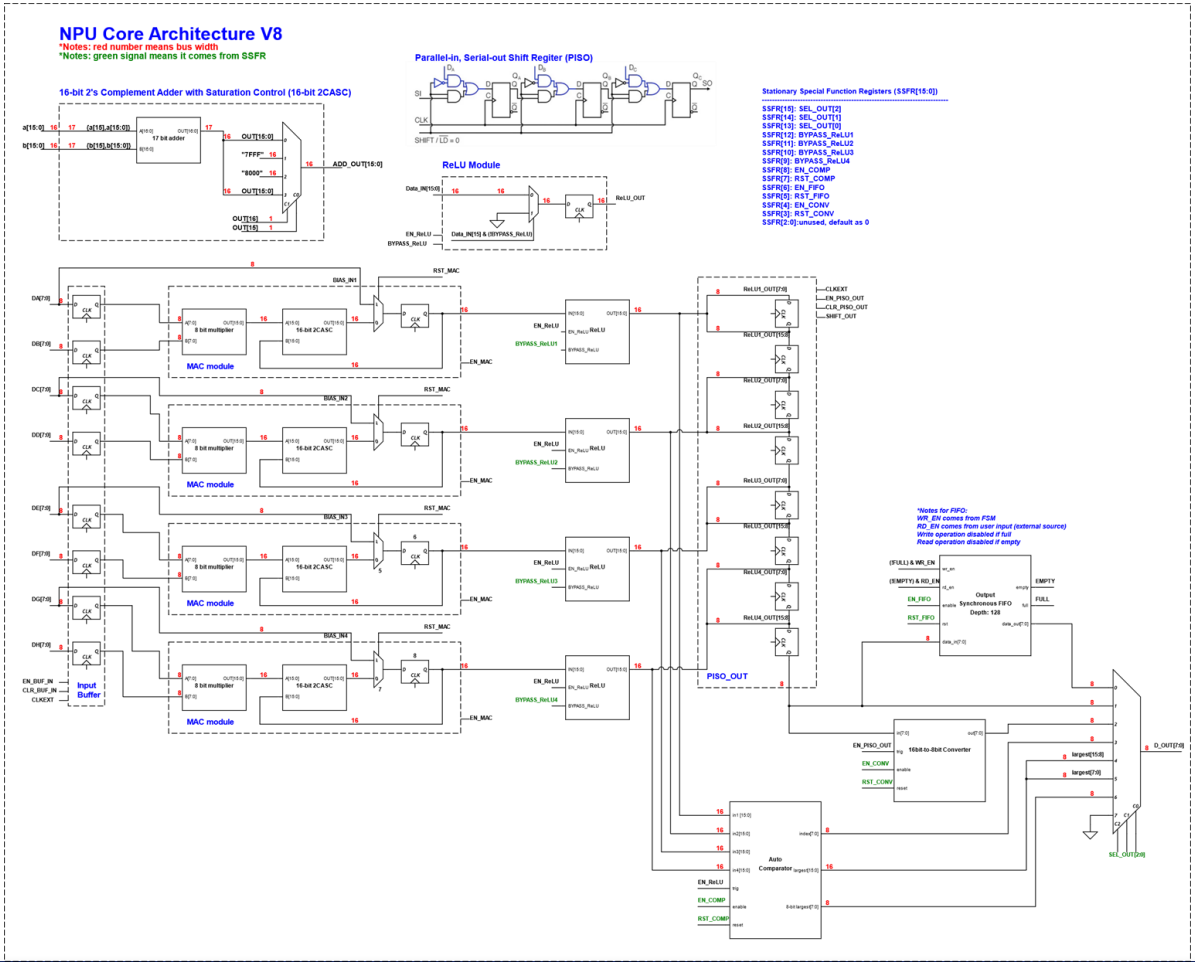
- FSM
- NPU Core
- SSFR (Stationary Special Function Registers)



NPU Architecture

Features:

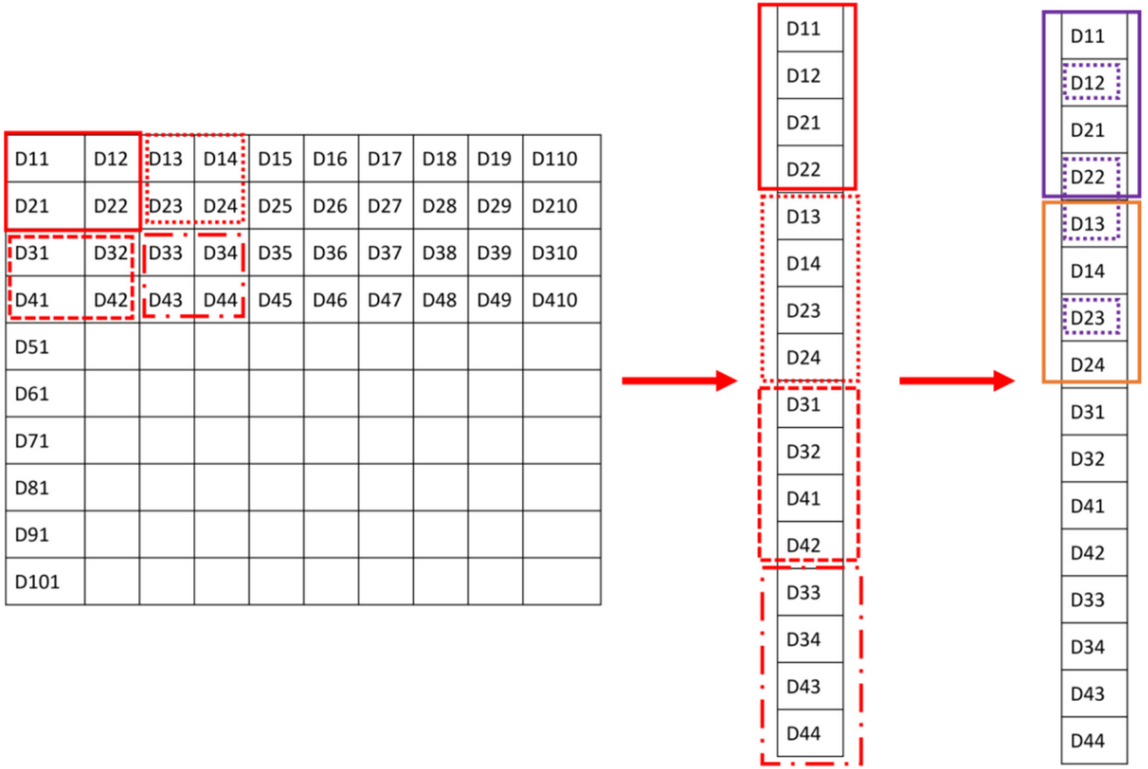
- 4 parallel MAC channels
- 8-bit input
- 8-bit/16-bit output
- 7 flexible output options
- Auto saturation control
- Auto result recording
- Auto maxpooling support
- Auto output neuron comparison
- Activation function choice



Memory Operation

D11	D12	D13	D14	D15	D16	D17	D18	D19	D110
D21	D22	D23	D24	D25	D26	D27	D28	D29	D210
D31	D32	D33	D34	D35	D36	D37	D38	D39	D310
D41	D42	D43	D44	D45	D46	D47	D48	D49	D410
D51									
D61									
D71									
D81									
D91									
D101									

Example of a 2D image (10 by 10)



Temporary buffer approach for generated the required sequence

Memory Operation

D11	D12	D13	D14	D15	D16	D17	D18	D19	D110
D21	D22	D23	D24	D25	D26	D27	D28	D29	D210
D31	D32	D33	D34	D35	D36	D37	D38	D39	D310
D41	D42	D43	D44	D45	D46	D47	D48	D49	D410
D51	D52	D53	D54	D55	D56	D57	D58	D59	D510
D61	D62	D63	D64	D65	D66	D67	D68	D69	D610
D71									
D81									
D91									
D101									

D11	D12	D13	D14	D15	D16	D17	D18	D19	D110
D21	D22	D23	D24	D25	D26	D27	D28	D29	D210
D31	D32	D33	D34	D35	D36	D37	D38	D39	D310
D41	D42	D43	D44	D45	D46	D47	D48	D49	D410
D51	D52	D53	D54	D55	D56	D57	D58	D59	D510
D61	D62	D63	D64	D65	D66	D67	D68	D69	D610
D71									
D81									
D91									
D101									

The transformation of four 4 by 4 regions after convolution and maxpooling

Inference Procedure