Advanced Computer-Aided VLSI System Design
Midterm Project: Single-Layer Convolution Engine with Quantization

TA: 邱仁皓 r12943008@ntu.edu.tw

TA: 蔡岳峰 f12943014@ntu.edu.tw

Due Tuesday, Apr. 29, 13:59

Data Preparation

1. Decompress 1132_midterm.tar with following command

```
tar -xvf 1132_midterm.tar
```

Folder	File	Description
00_TESTBED	testfixture.v	Testbench for top
00_TESTBED/test_patterns	p*i*.dat	Input data (*i* = 0, 1, 2)
00_TESTBED/test_patterns	p*i*_golden.dat	Output golden (*i* = 0, 1, 2)
00_TESTBED/axi	axi_ram.v	External memory
01_RTL	rtl_01.f	File list for RTL simulation
01_RTL	01_run	RTL simulation bash file
01_RTL	top.v	Your design
02_SYN	filelist.v	File list for synthesis
02_SYN	top_syn.tcl	TCL script for synthesis
02_SYN	top_syn.sdc	Constraints for synthesis
03_GATE	rtl_03.f	File list for gate-level simulation
03_GATE	03_run	Gate-level simulation bash file
05_POWER	pt_script.tcl	TCL script for power estimation

All libraries needed for synthesis, simulation can be found in previous homework.

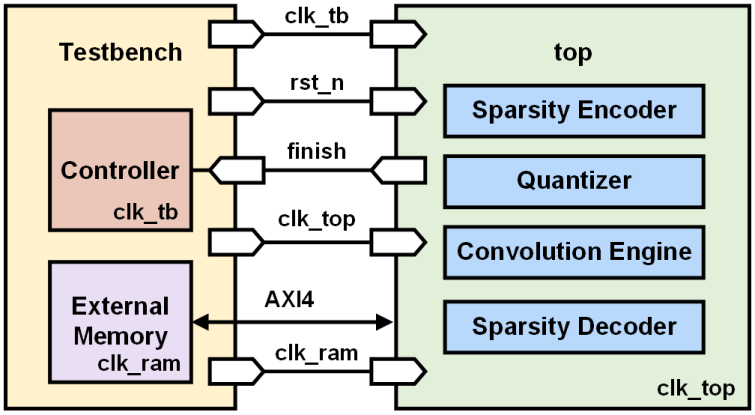
Only worst-case library is used in this homework.

Introduction

In this homework, you are asked to design a **convolution engine** for an MNIST classifier. The input encoded image and convolution parameters are stored in the external memory. The engine needs to fetch the image and parameters from the memory via AXI protocol and then perform convolution on the image. The computation results are then written back to the memory via AXI protocol. There are three time domains, which are for control signal, for memory, and for your design, respectively. After storing the results back to the memory, raise the finish signal for three cycles, and the system

will check the correctness of the results and terminate the simulation.

Block Diagram



Specifications

1. Top module name: **top**
2. Input/output description:

Signal Name	I/O	Width	Simple Description
clk_tb	I	1	Clock for control signal (positive edge trigger). Inputs have a **half-cycle delay**. Outputs should be synchronized at clock **rising** edge.
clk_top	I	1	Clock for your design (positive edge trigger).
clk_ram	I	1	Clock for AXI channel (positive edge trigger). Inputs are synchronized with the positive edge clock. Outputs should be synchronized at clock **rising** edge.
rst_n	I	1	Active **low** synchronous reset.
finish	O	1	Output finish signal. The signal should **be asserted high for three cycles** to indicate the end of computation.
awaddr araddr	O	15	AXI Write / Read request channel. Address of first transfer in a transaction.
awlen arlen	O	8	AXI Write / Read request channel. Total number of transfers in a transaction.
awsize arsize	O	3	AXI Write / Read request channel. Maximum number of bytes in a transfer within a transaction.
awburst arburst	O	2	AXI Write / Read request channel. Mode of the address increment.

awvalid arvalid	O	1	AXI Write / Read request channel. Write / Read request valid indicator.
awready arready	I	1	AXI Write / Read request channel. Write / Read request ready indicator.
wdata rdata	O I	8	AXI Write / Read data channel. Write / Read data.
wlast rlast	O I	1	AXI Write / Read data channel. Indicator of the last Write / Read transfer in a transaction.
wvalid rvalid	O I	1	AXI Write / Read data channel. Write / Read data valid indicator.
wready rready	I O	1	AXI Write / Read data channel. Write / Read data ready indicator.
wstrb	O	1	AXI Write data channel. Indicator of which byte lanes of write data contain valid data.
bresp rresp	I	2	AXI Write response / Read data channel. Response for transactions in Write response / Read data channels.
bvalid bready	I O	1	AXI Write response channel. Write response valid / ready indicator.

3. Only worst-case libraries are used for synthesis.
4. The slack for setup time and hold time should be non-negative.
5. No timing violations or glitches in the gate-level simulation after reset.

Design Description

1. Convolution

   In CVSD HW3, you had been asked to implement convolution operations. In ACVSD Midterm Project, the input data is a fixed 32×32 grayscale image (each pixel is **unsigned 8-bit** integer). The depth is fixed to eight, and the kernel weights are **signed 8-bit integers**. Bias, ReLU and max pooling are also applied on the convolution result. After computing the convolution, you need to add a bias to the result, filter out the negative elements with ReLU, and output a 16×16 by choosing maximum in each 2×2 groups with max pooling. Biases are **signed 24-bit integers**.

2. Bitmap Encoding

   The input image is bitmap-encoded and stored in the external memory. Bitmap encoder first chooses an element with highest frequency of occurrence (skipped

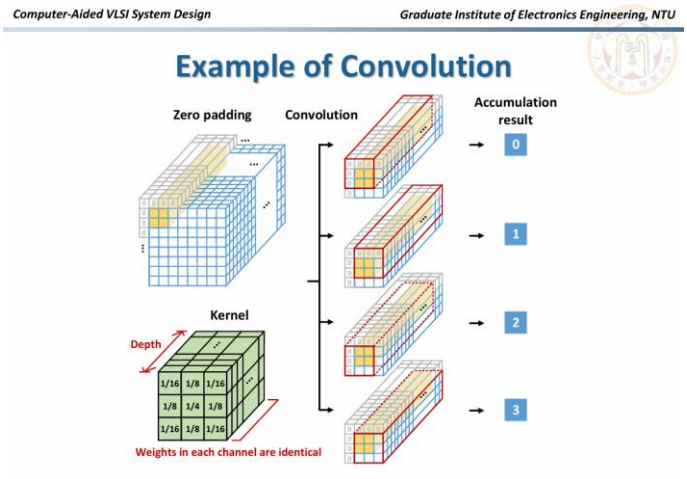


Fig. 1 Convolution in CVSD HW3

element) and generates a 32×32 binary bitmap indicating the locations of skipped and un-skipped elements ("0" = skipped, "1" = un-skipped). Next, the encoder records all the elements not equal to the skipped element in raster-scan ordering. Finally, the image is stored in the sequence of bitmap → skipped elements → all un-skipped elements.

3. Quantization

You have to quantize the result after computing convolution → ReLU → max pooling. Given a scale factor, you have to multiply the result with the scale factor and round it to the nearest integer. The scale factor is in **unsigned 32-bit** fixed-point format, and all the bits are fraction bits. For example, $(1100\ldots0)$ represents $(0.11)_2 = (0.75)_{10}$. The scale factor must scale and round the result into the range from 0 to 255, and it is stored in the external memory.

4. Memory Mapping

The external memory has word size of 8 bits and can store up to 32768 words. To simplify notation, the *i*-th word in the memory is denoted as MEM[*i*].

4.1. Input Image

Input image is bitmap-encoded, and its bitmap is stored from MEM[0] to MEM[127] in raster-scan ordering. Input skipped element is stored in MEM[128], and un-skipped elements are stored from MEM[129] to MEM[1151]. Un-skipped elements do not fill up all the memories from MEM[129] to MEM[1151], and the vacancies are filled with zeros.

4.2. Convolution Parameters

There are eight kernels, and the kernel size is 3×3. Each kernel has nine **signed 8-bit** weights, one **signed 24-bit** bias and one **unsigned 32-bit** scale factor. MEM[1152:1167] stores the parameters of kernel #0, in the sequence of weights → bias → scale factor. Weights are stored in raster-scan ordering. Bias and scale factor are stored from the most significant bit (MSB). Each kernel requires 16-word memory, and eight kernels are stored sequentially.

4.3. Results

After computing convolution → ReLU → max pooling → quantization, you have to bitmap-encode the result into a 16×16 binary bitmap, one **unsigned 8-bit** skipped elements, and multiple **unsigned 8-bit** un-skipped elements. The result of kernel #0 is stored in MEM[1280:1567], in the sequence of bitmap → skipped element → un-skipped elements. Bitmap is stored in raster-scan ordering, and the sequence of un-skipped elements is also determined in raster-scan ordering. Un-skipped elements do not fill up MEM[1313:1567], and the testbench will not check the vacancies. Eight results are stored sequentially.

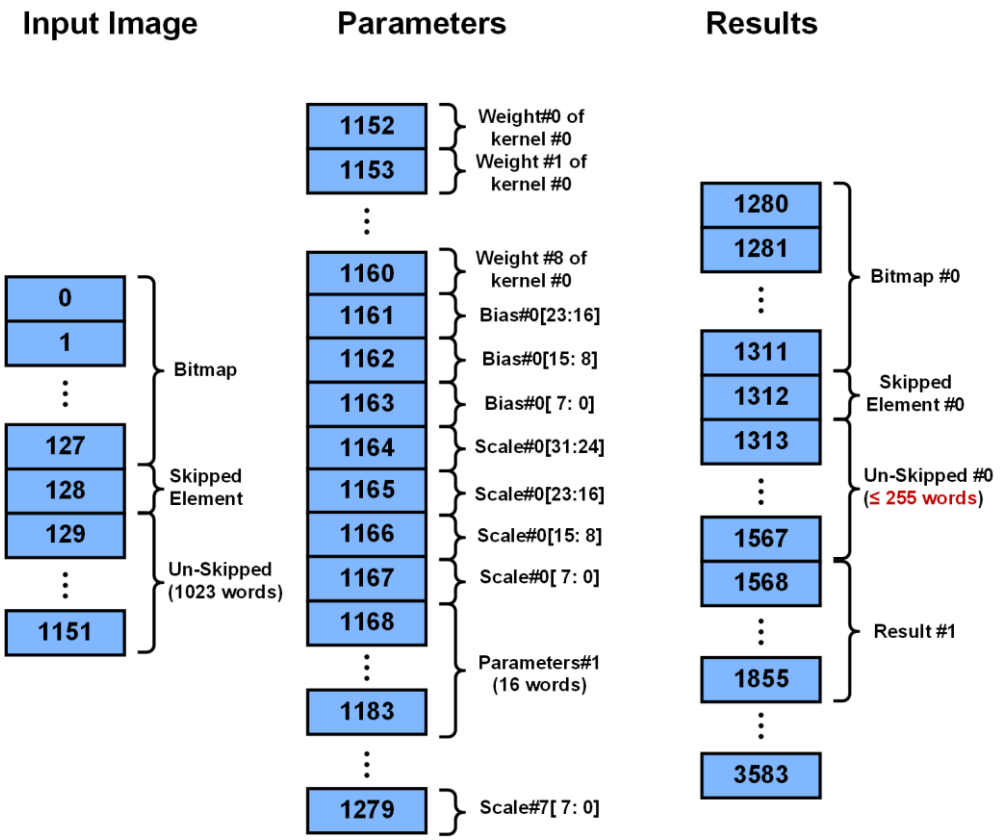
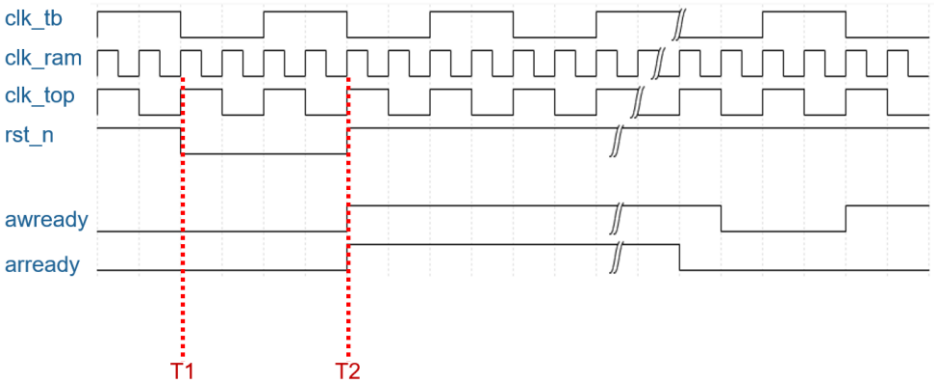
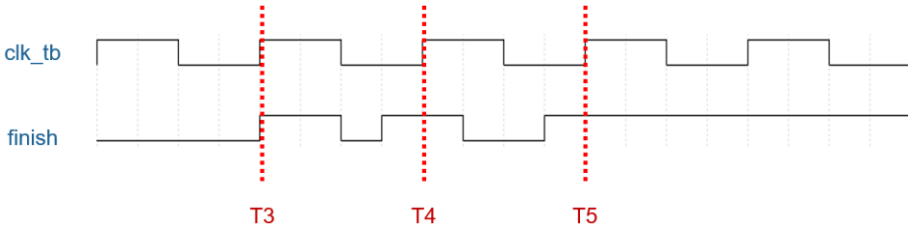


Fig. 2 Storing of the external memory

Timing Diagram



1. The system is initialized between T1 and T2. No need to consider reset synchronizer.
2. There is no start signal in this design, and you can start whenever the memory is ready (after T2).
3. AXI signals are synchronized with clk_ram, you should perform multi-bit CDC techniques on them.



4. You are required to finish all your computation and store the results back to AXI memory before T3. The finish signal should be set to high for three times. The finish signal is synchronized with clk_tb, and you should perform single-bit CDC techniques on the finish signal.
5. Once the finish signal is set to high for three times at T5, testbench will check the result in the memory with golden answer and terminate the simulation.

Submission

1. Submission in Workstation

1.1. Create a folder named **studentID_midterm (lowercase)**, and follow the hierarchy below.

```
r12943008_midterm
        ├─ 01_RTL
        │    ├─ rtl_01.f
        │    ├─ xxx.v  (other Verilog files)
        │    ├─ xxx.sv (other SystemVerilog files)
        │    └─ top.v
        ├─02_SYN
        │    ├─ top_syn.tcl
        │    ├─ top_syn.sdc
        │    ├─ top_syn.timing_min
        │    ├─ top_syn.timing_max
        │    ├─ top_syn.area
        │    └─ top_syn.ddc
        ├─03_GATE
        │    ├─ rtl_03.f
        │    ├─ top_syn.v
        │    └─ top_syn.sdf
        ├─04_UPF (optional)
        │    ├─ top.rtl.upf
        │    └─ top.syn.upf
        ├─05_POWER
        │    └─ p0_2.power
        └─ report.txt
```

1.2. 04_UPF files are optional, and no submission of UPF files will not cause any penalty.

1.3. Content of the report.txt

Record (1) processing time and power consumption for three public patterns with gate-level simulation (2) area estimated in synthesis.

```
StudentID: r12943008
Clock period: 5.0 (ns)
Area: 50000.00 (um^2)
Is UPF used? No

----------------------
p0 time: 50000 (ns)
p0 Power: 5 (mW)
----------------------
p1 time: 50000 (ns)
p1 Power: 5 (mW)
----------------------
p2 time: 50000 (ns)
p2 Power: 5 (mW)
```

1.4. Compress the folder **studentID_midterm** into a **tar.gz file** named **acvsdxxx_midterm.tar.gz**

```
tar -zcvf acvsdxxx_midterm.tar.gz studentID_midterm
```

TA will only check the latest version of your homework.

2. Submission in NTU COOL
   2.1. Create a folder named **studentID_midterm (lowercase)**, and follow the hierarchy below.

```
r12943008_midterm
        ├─ 01_RTL
        │     ├─ rtl_01.f
        │     ├─ xxx.v  (other Verilog files)
        │     ├─ xxx.sv (other SystemVerilog files)
        │     └─ top.v
        ├─02_SYN
        │     ├─ top_syn.tcl
        │     └─ top_syn.sdc
        ├─04_UPF (optional)
        │     ├─ top.rtl.upf
        │     └─ top.syn.upf
        └─ report.txt
```

   2.2. UPF files and report are the same as the submission in NTU COOL.
   2.3. Compress the folder **studentID_midterm** into a **tar.gz file** named **acvsdxxx_midterm_v*k*.tar.gz** (*k* is the version, *k* = 1, 2, …)

```
tar -zcvf acvsdxxx_midterm_vk.tar.gz studentID_midterm
```

Grading Policy

1. TA will use **01_run** and **03_run** to run your code at RTL and gate-level simulation with three public patterns and three hidden patterns.

2. Simulation (70%)

	Score
RTL simulation	30%
Gate-level simulation	30%
Hidden pattern (gate-level)	10%

3. Performance (30%)
   Score for performance to be considered:

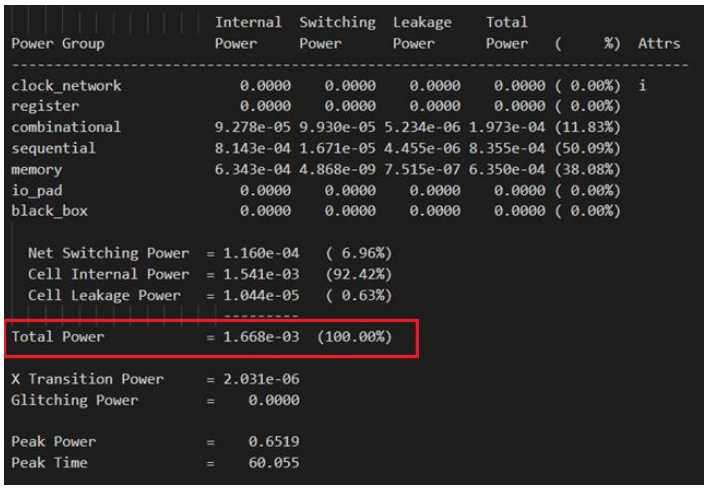
$$\text{Score} = \left(\sum_{i=0}^{2} \text{power}_i \times \text{time}_i\right) \times \text{area}$$

Unit: power (mW), time (ns), area(um^2)

Baseline = 10^{10}

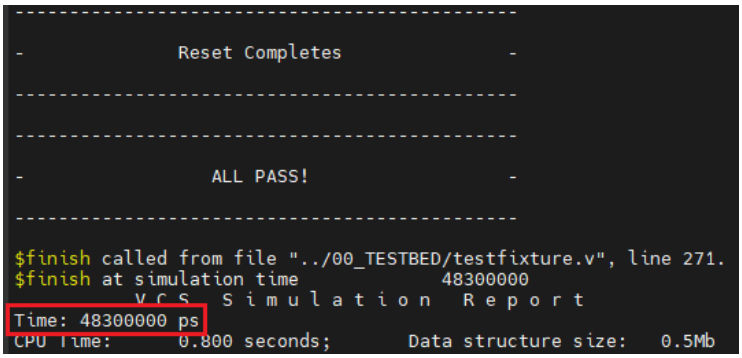
	Score
Baseline (need passing hidden patterns)	10%
Ranking (need passing baseline)	20%

4. Precise explanation of the performance terms:
   4.1. Power: power report by Primetime analyzing gate-level waveform files (ex. 1.668 mW below). There are three patterns in total.



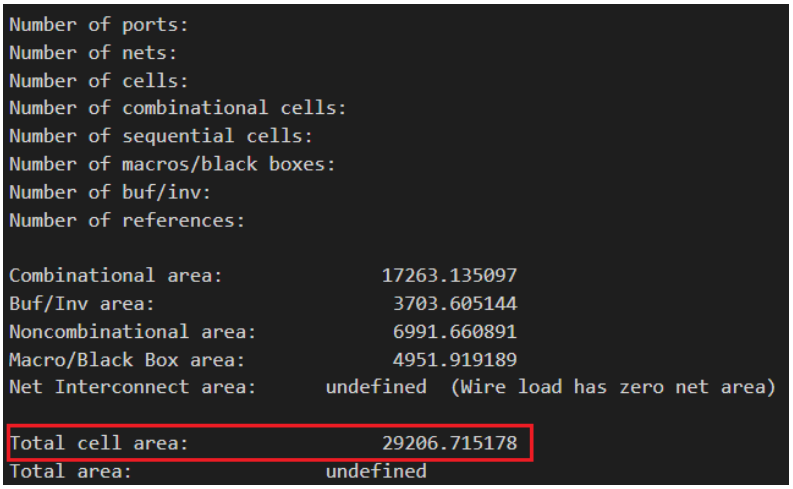
```
                          Internal  Switching  Leakage    Total
Power Group               Power     Power      Power      Power   (     %)  Attrs
--------------------------------------------------------------------------------
clock_network               0.0000    0.0000    0.0000    0.0000 ( 0.00%)  i
register                    0.0000    0.0000    0.0000    0.0000 ( 0.00%)
combinational           9.278e-05 9.930e-05 5.234e-06 1.973e-04 (11.83%)
sequential              8.143e-04 1.671e-05 4.455e-06 8.355e-04 (50.09%)
memory                  6.343e-04 4.868e-09 7.515e-07 6.350e-04 (38.08%)
io_pad                      0.0000    0.0000    0.0000    0.0000 ( 0.00%)
black_box                   0.0000    0.0000    0.0000    0.0000 ( 0.00%)

  Net Switching Power  = 1.160e-04   ( 6.96%)
  Cell Internal Power  = 1.541e-03   (92.42%)
  Cell Leakage Power   = 1.044e-05   ( 0.63%)
                         ---------
Total Power            = 1.668e-03  (100.00%)

X Transition Power     = 2.031e-06
Glitching Power        =    0.0000

Peak Power             =    0.6519
Peak Time              =    60.055
```

   4.2. Time: processing time from gate-level simulation (ex. 48300 ns below)



```
---------------------------------------------

-              Reset Completes               -

---------------------------------------------

---------------------------------------------

-                 ALL PASS!                  -

---------------------------------------------

$finish called from file "../00_TESTBED/testfixture.v", line 271.
$finish at simulation time              48300000
           V C S   S i m u l a t i o n   R e p o r t
Time: 48300000 ps
CPU Time:      0.800 seconds;       Data structure size:   0.5Mb
```

   4.3. Area: total cell area in synthesis report (ex. 29206.72 um^2 below)



```
Number of ports:
Number of nets:
Number of cells:
Number of combinational cells:
Number of sequential cells:
Number of macros/black boxes:
Number of buf/inv:
Number of references:

Combinational area:              17263.135097
Buf/Inv area:                     3703.605144
Noncombinational area:            6991.660891
Macro/Black Box area:             4951.919189
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 29206.715178
Total area:                 undefined
```

Reference

[1] MNIST: GTDLBench

[2] Convolution: [ML 2021 (English version)] Lecture 9: Convolutional Neural Networks

[3] Convolution: CVSD HW3.

[4] Bitmap encoding: C. -W. Chang, I. -T. Lin and C. -H. Yang, "A 101mW, 280fps Scene Graph Generation Processor for Visual Context Understanding on Mobile Devices," *2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, 2024.

[5] Quantization: tinyML_Talks-_Marios_Fournarakis_210929.pdf, p.26.

[6] AXI protocol: AMBA AHB Protocol Specification