

## Computer-Aided VLSI System Design, Fall 2018

### Verilog Homework 1 – ALU Design

TA: 林邑政 (dawn1dusk1fate@gmail.com)

#### 1. 作業目標

從 ALU 基本運算單元的例子讓大家熟悉如何完成 RTL level 的電路敘述，並在工作站上利用 simulator (NC-verilog) 配合 testbench 跑 RTL-level simulation。配合 Notes 以及 Appendix 講解設計硬體時與 Verilog Coding (Coding Style) 要注意的地方。

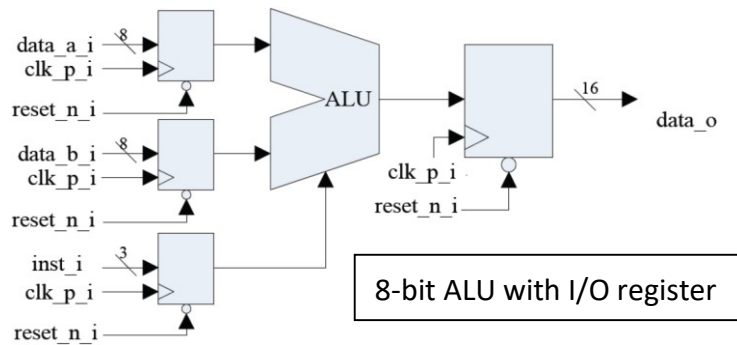
#### 2. ALU 介紹

Arithmetic logic unit (ALU)是在電腦處理器之中其中一個組成單元，ALU 有數學、邏輯、還有一些設計過的運算在電腦之中。基於 LAB1, 一些特別的 instructions 為了影像處理需要被設計並且完成-ALU with I/O register。請設計下表中 specification **8-bit ALU with I/O register , 3-bit instruction set** 的電路。

Inst_i [2:0]	Operation	Notes
000	Signed Addition	$data\_o = data\_a\_i + data\_b\_i$
001	Signed Subtraction	$data\_o = data\_b\_i - data\_a\_i$
010	Unsigned Multiplication	$data\_o = data\_a\_i * data\_b\_i$
011	AND	$data\_o = data\_a\_i \& data\_b\_i$
100	XOR	$data\_o = data\_a\_i \oplus data\_b\_i$
101	Absolute Value	$data\_o =  data\_a\_i $
110	Addition & Divide by 2	$data\_o = (data\_b\_i + data\_a\_i) / 2$
111	Mod	$data\_o = (data\_b\_i \% data\_a\_i)$

[Notes] 數字系統與訊號之間的注意事項

- Output signal “data\_o”有 16bits 並且用 **2’s complement** 表示。
- Input data(a & b)對於 instruction 010/011/100/110/111 為 **unsigned**，但對於 instruction 000/001/101 是 **signed**。
- 此次作業不需要考慮 **overflow** 的問題，也就是 output 訊號 16bit 對於運算結果夠用。
- 對於訊號在特定數字系統(2’s complement)下運算前後是需要被注意的
  - 做 zero extension 當 output 永遠為正 (instruction 010/101/110/111)
  - 做 **sign-extension** 當 output 有可能為負的 (instruction 000/001)
  - 做 zero extension 當”AND”和”XOR”運算
- 如果是 fix point 的運算，則需要注意精準度與面積間的拿捏，也就是總 bit 數與小數點的位置(shift bit)選取，可以在整數部分不會 overflow 且小數部分精確度夠情況下使 bit 數最少(面積)。(此次作業無須注意)



1. 作業提供了一個未完成 RTL template program，請依照上述 spec.表完成程式，為了加速 debug 的過程，建議每完成一個 instruction 就模擬一次。在 testbench 之中:

- If assign “test\_all\_ins” = 1, 則所有的 instruction 會被測試(default setting)
- If assign “test\_all\_ins” = 0, 則指定特定的"test\_instruction"可以測試相對應的 instruction

(ex: test\_all\_ins = 0, test\_instruction = 000 : 只測試 signed addition 功能)

Filename	Description
<i>HW1_alu.v</i>	Unfinished Verilog RTL template
<i>HW1_test_alu.v</i>	Testbench for ALU design

2. 請利用提供的 testbench 來驗證你的 ALU with I/O registers 設計，我們會依照 testbench 對你的 design 評分。
3. 對於 000 到 001 的 instruction 為基本運算，而 101 到 111 為較進階的運算。

### 3. 繳交要求

直接上傳RTL檔案，命名形式如下：

***StudentID\_HW1\_alu.v*** (e.g., R05943001\_HW1\_alu.v).

[Notes]

1. 在你的 design 之中，請依照原本提供 Verilog template 的 port name 以及 port order，否則你的 design 可能會在測試中 fail 進而影響成績。
2. 如果你想要修改你的 code，則在繳交時新的檔案命名為 ***StudentID\_HW1\_alu\_v2.v*** 之後版本請依序類推

### 4. 繳交期限：10/02(二) 中午11:59AM

### 5. 評分標準

該次作業ALU一個功能測試無誤得12.5分，八個功能都對則滿分100分。需要注意的是，批改作業時，測試每個功能的時候會使用額外的測資。

## 5. Appendix

### [Naming Rules]

一個好的有共識 Naming Rules 會對 Code 維持性以及可讀性大大提升，最重要的是可以跟硬體架構精準的相對應，可以參考以下所提供的命名規則來對 design 內變數進行命名。

1. “\_i”與“\_o”應該被加在一個 module 中 input 以及 output 變數的最後面。
2. 如果是 clk 或 reset signal，則“\_p” or “\_n”可以用來表示他是正緣觸發還是負緣處發。

3. 如果變數在電路中為 flipflop，則在 reg 變數名稱後

- Flipflop input 加“\_w”，表示是 write，flipflop output 加“\_r”，表示是 read。

- Flipflop input: a\_nxt, Flipflop output: a

(注意這裏強調為 flipflop 不是 reg，因為只要是宣告在 block 內的變數都要是 reg 型態，所以有可能實際上是拉線還是宣告為 reg，但 block 外的變數必須宣告為 wire 且一定也是拉線)

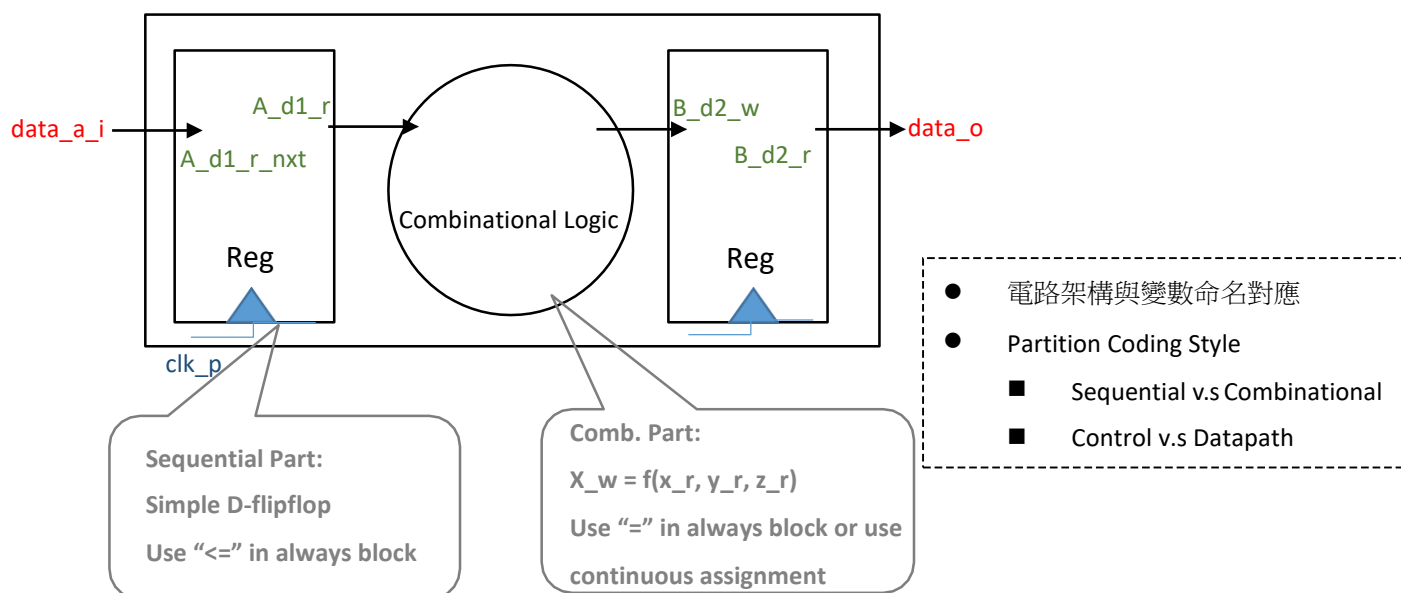
4. 隨著系統內 pipeline 的層數，訊號會 delay x 個 cycle，可以在之後加“\_dx”已表示為在第 x 層的訊號。

ex: a\_d1\_w (第一層的 flipflop input)

5. 如果是不得已對於 a\_w 中間操作要用到 truncate(例如 fixpoint operation)，因語法限制無法一行寫完則 combinational circuit 內中間 wire 可命名為 tmp 再接給 a\_w，如右圖。

Ex:

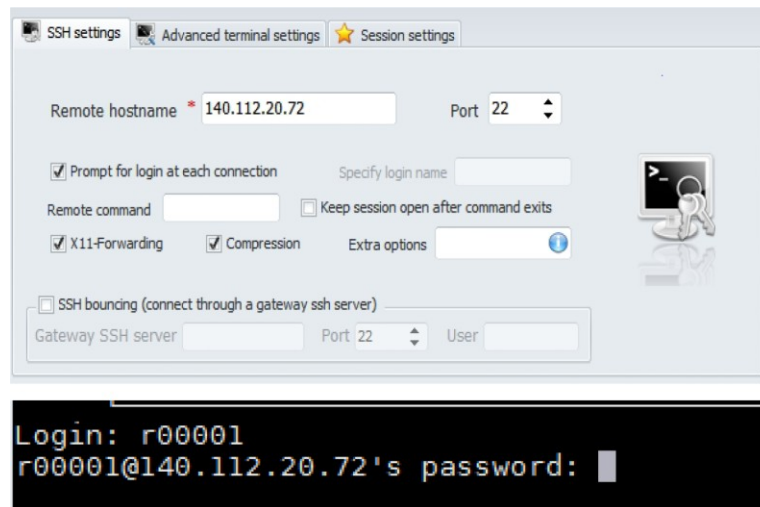
```
a_tmp = b_r + c_r;  
a_w = a_tmp[5:3]
```



## [Run NC-Verilog Simulation on GIEE's Workstation]

### (1) Connect to GIEE's Workstation: [Take MobaXterm for illustration]

Available workstation list: [http://cad.ee.ntu.edu.tw/ws\\_list.htm](http://cad.ee.ntu.edu.tw/ws_list.htm)



### (2) Environment Setup: [type following commands in the console]

1. Making Directory:

**mkdir HW1**

2. Enter the Directory:

**cd HW1**

3. Source the default file:

**source /usr/cadence/cshrc**

### (3) Check Verilog Code by NC-Verilog

1. Type this command:

**ncverilog [^design\_filename]**

**ex: ncverilog HW1\_alu.v**

### (4) Run Simulation with a test bench

1. Type this command:

**ncverilog [^testbench\_filename] [^design\_filename]**

**ex: ncverilog HW1\_test\_alu.v HW1\_alu.v**

PS: 從大module依序到小module，如果未來有tsmc13.v則要加在最後面讓simulator可以將foundry standard電路module轉成看得懂的basic behavior block (ex:and)，又或著是include memory、IO pad額外非standard cell module的external-IP behavior model。

### (5) If you want to check the waveform, (原則上該次作業不需看波型即可完成)

1. Type the command:

**ncverilog [^testbench\_filename] [^design\_filename] +access+r**

For more GIEE's workstation information, please refer to: <http://cad.ee.ntu.edu.tw/>

For more workstation commands, please refer to:

[http://linux.vbird.org/linux\\_basic/redhat6.1/linux\\_06command.php](http://linux.vbird.org/linux_basic/redhat6.1/linux_06command.php)