NCKUES

DIGITAL INTEGRATED CIRCUIT DESIGN

2021 fall

Midterm project

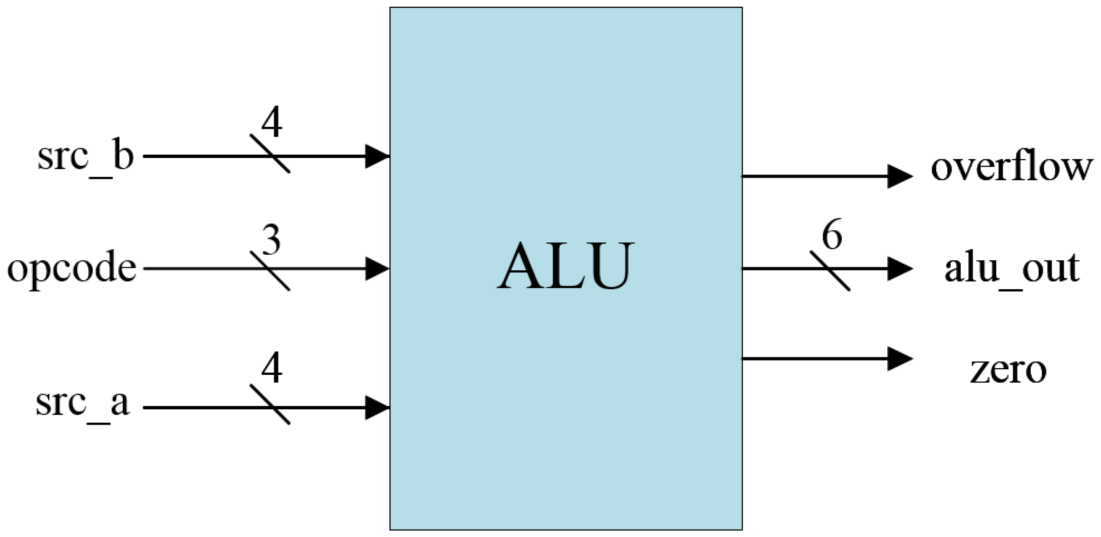
Arithmetic Logic Unit

Professor : Wen-Long Chin

TA : Pin-Wei Chen

VSP LAB

- Design Description
  - An arithmetic logic unit (ALU) is a digital circuit that performs arithmetic and logical operations.
  - Please design an ALU which performs 8 operations.
- Block Diagram



- Specifications
  - Top module name : alu (File name: alu.v).
  - Input ports: src_a[3:0], src_b[3:0], opcode[2:0],
  - Output ports: overflow, alu_out[5:0], zero.

- Functionality
  - Calculation result (alu_out) please reset to 6’b000000 at the beginning.
  - The zero detection bit becomes 1 when the calculation result (alu_out) is equal to 6’b000000, otherwise becomes 0.
  - The src_a, src_b, alu_out are signed number.
  - If the calculation result (alu_out) exceed the range that can represent, the overflow detection bit will become 1.
  - If an overflow occurs, the maximum or minimum value will be output (saturated output).

    Ex. 10002 (-8) * 01102 (6) = 10100002 (-48)

    => overflow = 1, alu_out = 100000 (-32), zero = 0

  - Note that when the operation is left shifting (<<), overflow detection and saturated output is also needed.
  - Note that when the operation is AND, the most significant two bits of alu_out (alu_out[5] , alu_out[4]) must be 0.

opcode	ALU operation
000	src_a AND src_b
001	Max(src_a , src_b)
010	LUT(src_a)
011	src_a * src_b
100	src_a + src_b
101	src_a - src_b
110	PERM(src_a , src_b) ,bit permutation
111	src_a << src_b (barrel shift left , 0 <= src_b < 8)

➢ Example

opcode	**ALU operation example**
000(AND)	src_a=4’b0001, src_b=4’b1110 → alu_out=6’b000000, zero=1’b1, overflow=1’b0
001(Max)	src_a=4’b0001, src_b=4’b1110 → alu_out=6’b000001, zero=1’b0, overflow=1’b0
010 (LUT)	Look up table.

index	content
0	31
1	-7
2	0
3	1
4	3
5	10
6	14
7	16
8	-2
9	8
10	21
11	7
12	-17
13	11
14	2
15	8

	alu_out = table[src_a]
	(此時 src_a 視為無號數)
	src_a=4’b0010, src_b=4’b1110,
	→ alu_out=6’b000000, zero=1’b1, overflow=1’b0

011 (*)	src_a=4’b0001, src_b=4’b1110,
	→ alu_out=6’b111111, zero=1’b0, overflow=1’b0
	src_a=4’b1000, src_b=4’b0111,
	→ alu_out=6’b100000, zero=1’b0, overflow=1’b1

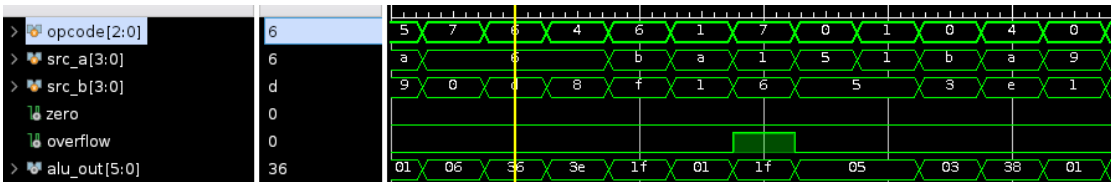
100(＋)	src_a=4’b0001, src_b=4’b1001,
	→ alu_out=6’b111010, zero=1’b0, overflow=1’b0
	src_a=4’b0111, src_b=4’b0111,
	→ alu_out=6’b001110, zero=1’b0, overflow=1’b0

101(－)	src_a=4’b0001, src_b=4’b1001,
	→ alu_out=6’b001000, zero=1’b0, overflow=1’b0
	src_a=4’b0111, src_b=4’b0011,
	→ alu_out=4’b000100, zero=1’b0, overflow=1’b0

110(PERM)	if src_a is odd, alu_out=
	{src_a[2],src_b[2],src_a[1],src_b[1],src_a[0],src_b[0]}
	else
	{src_b[2],src_a[2],src_b[1],src_a[1],src_b[0],src_a[0]}
	src_a=4’b0111, src_b=4’b0011,
	→ alu_out=4’b101111, zero=1’b0, overflow=1’b0
	src_a =4’b1110, src_b=4’b0001,
	→ alu_out=6’b010101, zero=1’b0, overflow=1’b0

111(<<)	src_a=4’b0111, src_b=4’b0100, → alu_out=6’b011111, zero=1’b0, overflow=1’b1

- waveform



testbench 會控制輸入訊號，同學設計的 alu 就是要設法算出對應的輸出然後由 testbench 判斷你輸出的答案是否正確。

➢ 評分標準依以下規定。
   1. A 等級為完成測試樣本之所有 RTL simulation 和
      Post-synthesis Functional simulation 和
      Post-synthesis Timing simulation
   2. B 等級為完成測試樣本之所有 RTL simulation
   3. C 等級為完成測試樣本 50%以上(含)之 RTL simulation
   4. D 等級為完成測試樣本 50%以下之 RTL simulation

➢ A 等級依照 LUT 個數細分成績，同 LUT 時則依照上傳至 moodle 的時間先後來細分成績

➢ B 等級依照上傳至 moodle 的時間先後來細分成績

➢ C、D 等級依照通過測試樣本的比例來細分成績

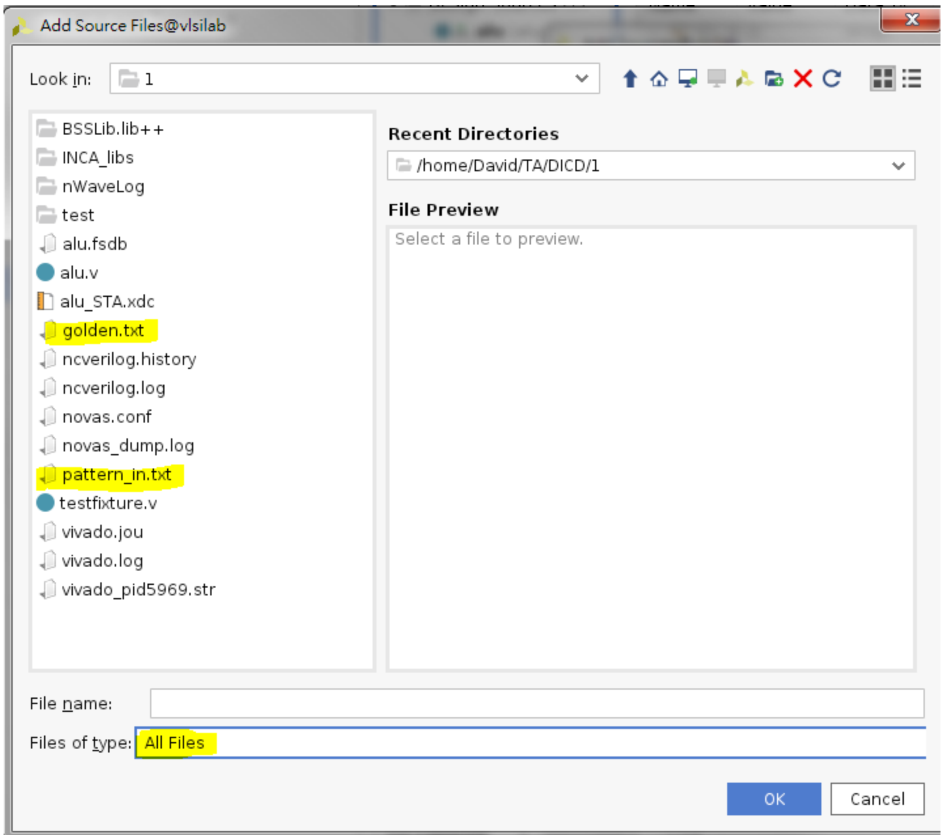
繳交方式

- 上傳 alu.v 至 moodle，未繳交者以 0 分計。
- 上傳 LUT 個數截圖 (optional)
- 上傳通過 simulation 截圖 (optional)

可將檔案壓縮成.zip 再上傳至 moodle

注意事項

1. 請記得將 golden.txt 和 pattern_in.txt 放入 simulation sources


Add Source Files@vlsilab

Look in: 1

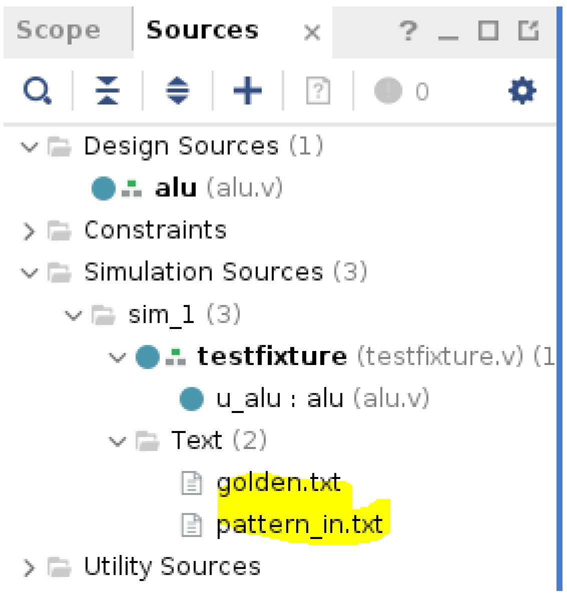
BSSLib.lib++
INCA_libs
nWaveLog
test
alu.fsdb
alu.v
alu_STA.xdc
golden.txt
ncverilog.history
ncverilog.log
novas.conf
novas_dump.log
pattern_in.txt
testfixture.v
vivado.jou
vivado.log
vivado_pid5969.str

Recent Directories: /home/David/TA/DICD/1

File Preview: Select a file to preview.

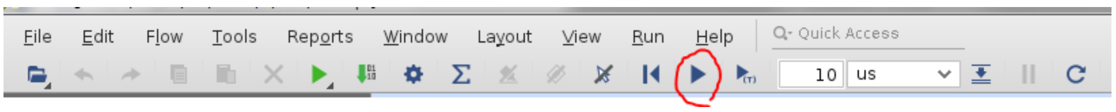
File name:

Files of type: All Files

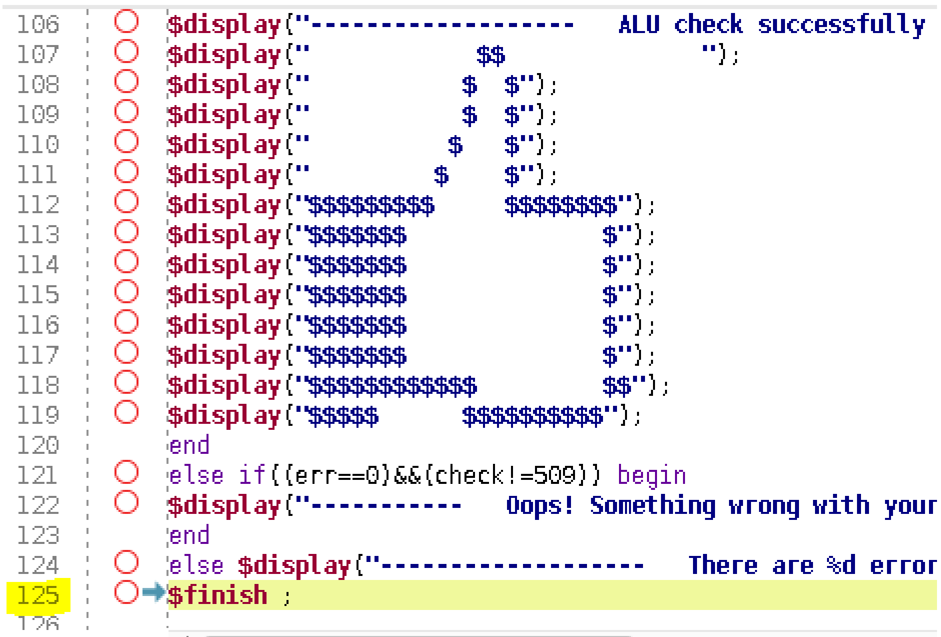


```
Design Sources (1)
    alu (alu.v)
Constraints
Simulation Sources (3)
    sim_1 (3)
        testfixture (testfixture.v) (1
            u_alu : alu (alu.v)
        Text (2)
            golden.txt
            pattern_in.txt
Utility Sources
```

2. 請勿更改 alu.v 裡面的 input/output 腳位
3. 請注意 simulation 是否確時跑完，若無跑完請按 run all



無論程式(你的 alu.v)功能對錯，要跑到$finish 才真的有跑完



```
$display("-------------------   ALU check successfully
$display("            $$              ");
$display("           $  $");
$display("           $  $");
$display("          $   $");
$display("         $    $");
$display("$$$$$$$$$     $$$$$$$$");
$display("$$$$$$$              $");
$display("$$$$$$$              $");
$display("$$$$$$$              $");
$display("$$$$$$$              $");
$display("$$$$$$$              $");
$display("$$$$$$$$$$$$         $$");
$display("$$$$$      $$$$$$$$$$");
end
else if((err==0)&&(check!=509)) begin
$display("-----------   Oops! Something wrong with your
end
else $display("-------------------   There are %d error
$finish ;
```

4. 請記得將 xdc 檔放入 constraint sources，要達到 A 等級請務必通過 Post-synthesis Functional simulation 和 Post-synthesis Timing simulation，執行 Post-synthesis simulation 前請先 report timing summary 去確認 setup/hold time 是否 met
   (之後會有課堂練習教同學如何跑合成後的 sim)
5. 請勿直接針對特定測資做輸出，會有隱藏測資做驗證
6. 請勿抄襲，若抄襲以 0 分計