RISC-V 伴伴學

加法器及ALU的實現

第一組
組長：KIM(阿寬)
報告人：名名

先備知識

- 二進位制及其運算
- 邏輯閘
- Verilog基礎語法

學習目標

- **了解加法器的實現原理**
- 淺談CPU – 什麼是ALU?
- 使用Verilog實現ALU

先備知識

二進位制

十進制

- 由0、1、2、3、4、5、6、7、8、9十個數字所組成。
- 1、2、3… 9→10

…	億	千萬	百萬	十萬	萬	千	百	十	個
	1	2	3	4	5	6	7	8	9

16進制

- 由0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F十六個數字所組成。
- 1、2、3... 9、A、B、C、D、E、F→10

二進制

- 僅由 0、1 兩個數字組成
- 0 → 1 → 1 0

權重	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
十進制	256	128	64	32	16	8	4	2	1

先備知識

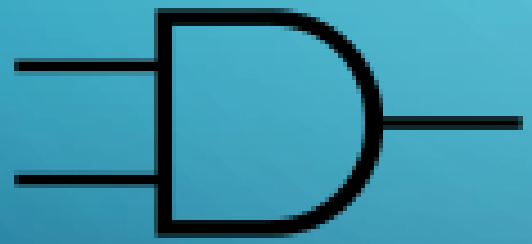
邏輯閘

基本邏輯閘的種類

- AND
- OR
- XOR
- NOT

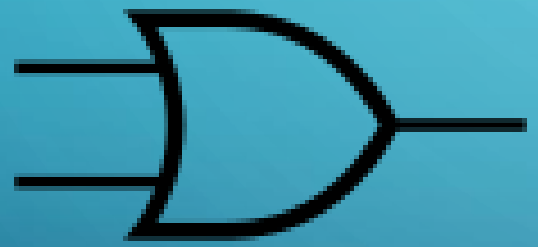
及閘 (AND GATE)

- 輸入皆為 1 ，則輸出 1 ，否則輸出為 0。
- 符號：



或閘 (OR GATE)

- 任一輸入為 1 ，則輸出 1 ，否則為 0。
- 符號：

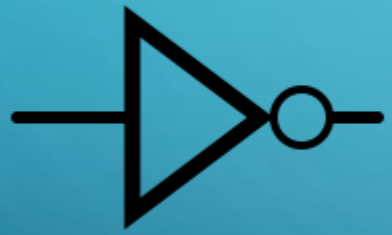


互斥或閘 (XOR GATE)

- Exclusive OR
- 輸入不同，則輸出 1 ，相同則為 0。
- 符號：

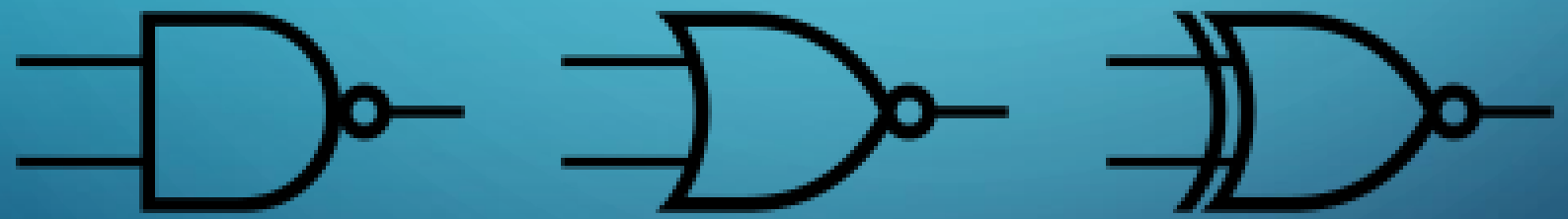
反閘 (NOT GATE)

- 輸出為輸入的相反。
- 符號：



與NOT組合的邏輯閘

- NAND：AND + NOT → 把AND的輸出做NOT。
- NOR　：OR + NOT → 把OR的輸出做NOT。
- XNOR：相同為 1，不同為 0。



先備知識

VERILOG - 硬體描述語言

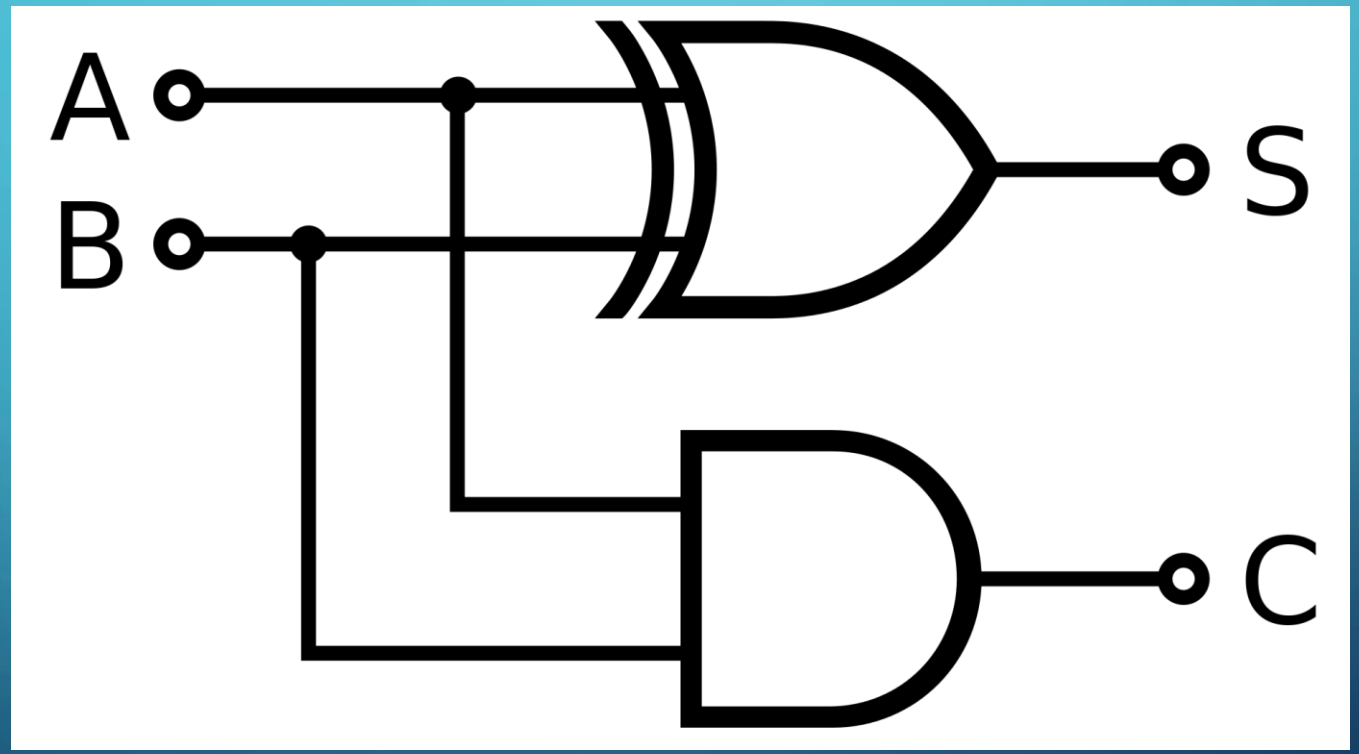
VERILOG

- 硬體描述語言
- 用程式碼描述一張電路圖

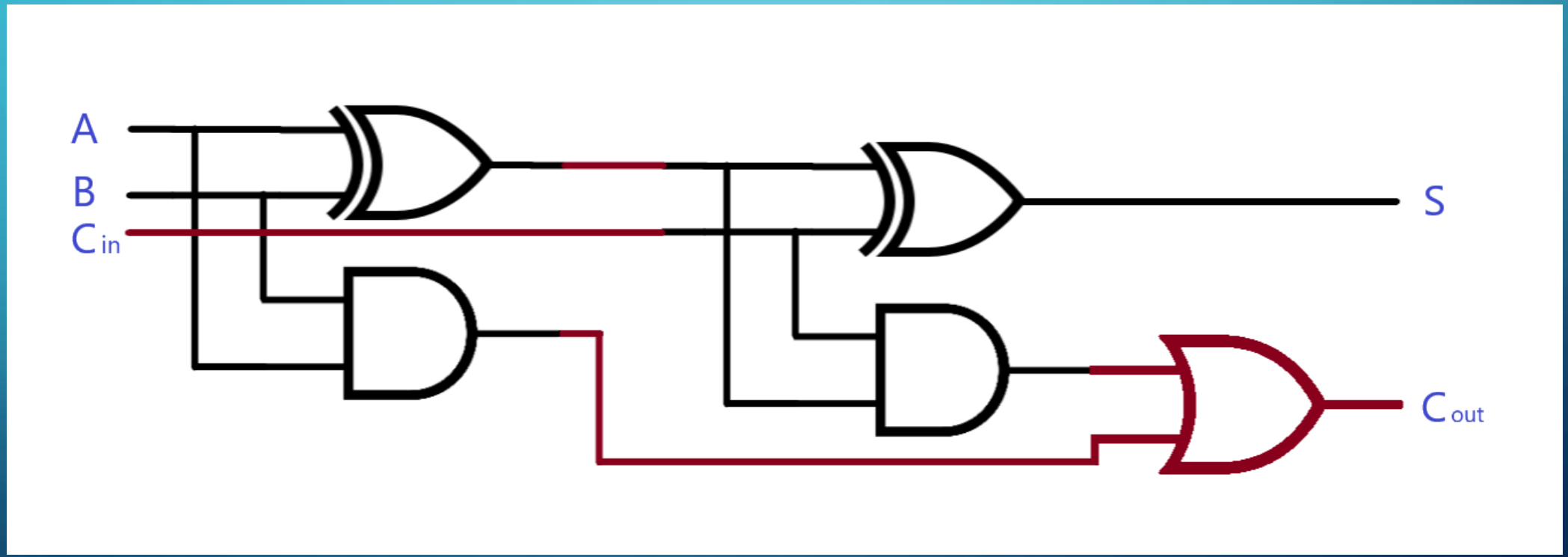
學習重點

加法器電路

半加器 (HALF ADDER)



全加器 (FULL ADDER)



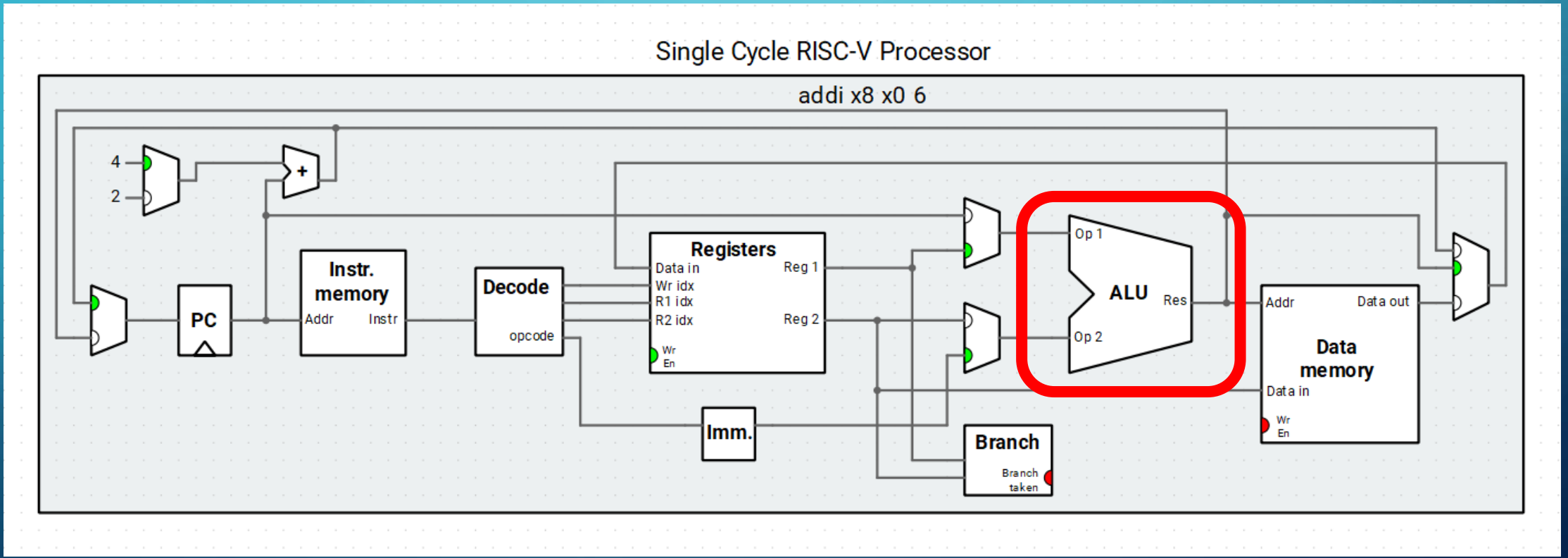
學習重點

淺談CPU

CPU的架構

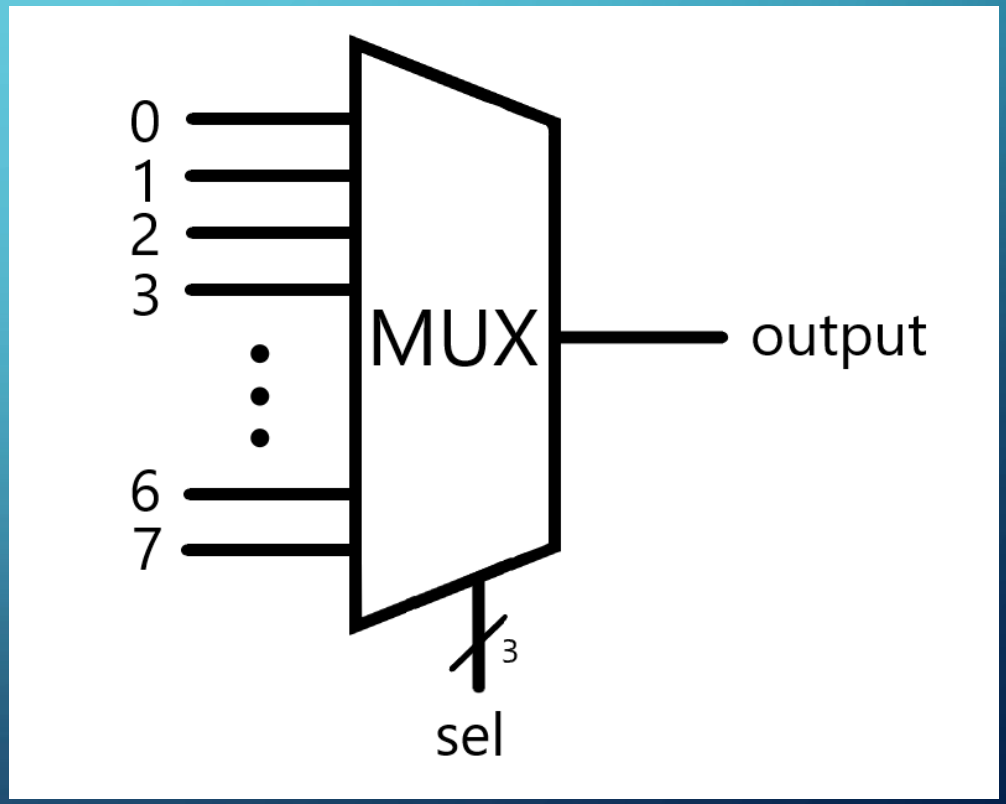
- 馮紐曼模型(普林斯頓架構)
  →馮紐曼瓶頸
- 哈佛架構
  - 5 stage CPU
    instr mem提取指令→Decoder解碼，提取合併reg資料
    →ALU執行→儲存資料到data mem→資料寫進reg

ALU是什麼?

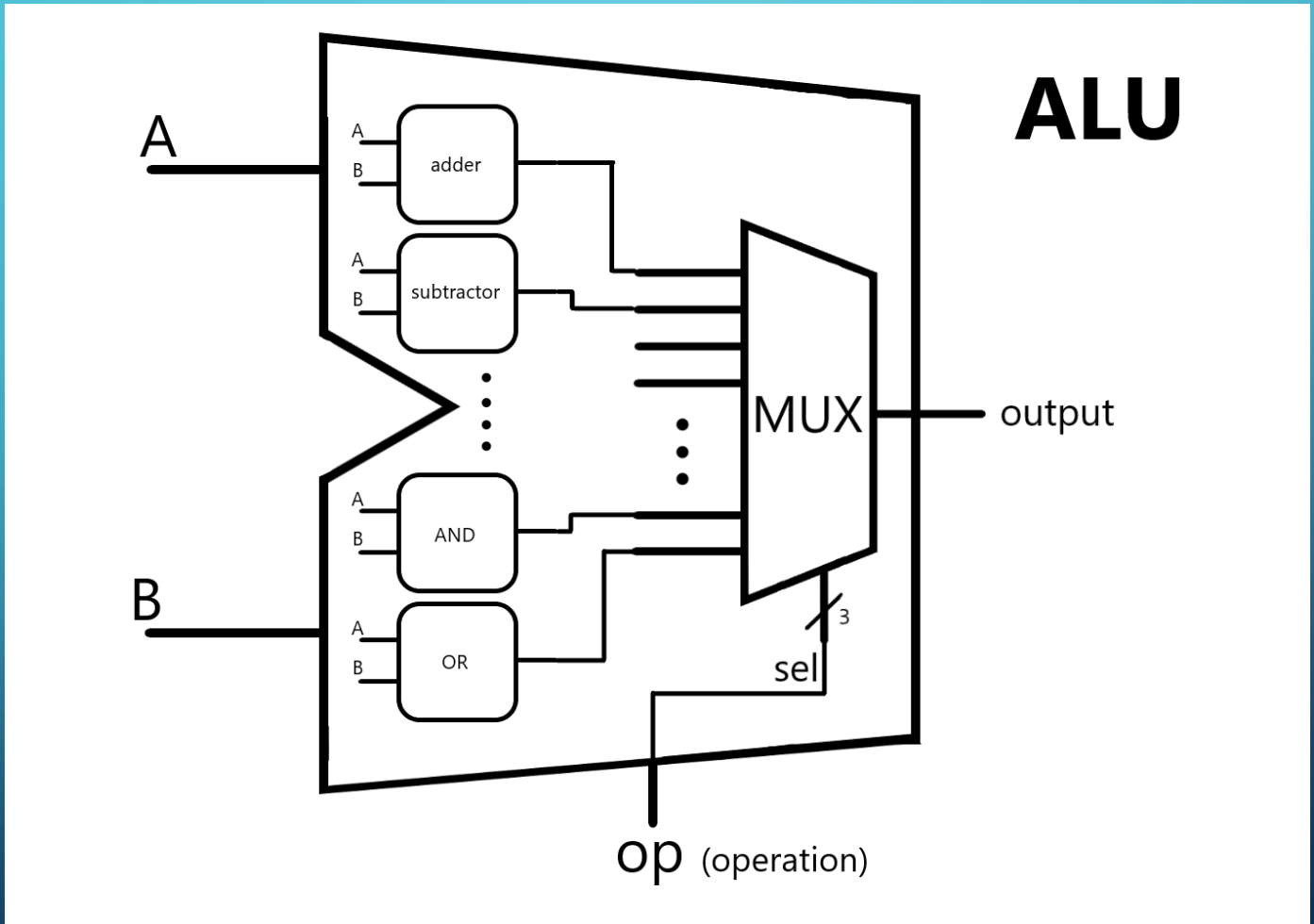

Single Cycle RISC-V Processor
addi x8 x0 6
4
2
+
PC
Instr.
memory
Addr
Instr
Decode
opcode
Registers
Data in
Wr idx
R1 idx
R2 idx
Reg 1
Reg 2
Wr
En
Imm.
Branch
Branch
taken
Op 1
ALU
Res
Op 2
Addr
Data out
Data
memory
Data in
Wr
En

多工器

- multiplexer，MUX
- 又稱選擇器，Data Selector



ALU電路


ALU
A
B
adder
subtractor
AND
OR
A
B
MUX
output
3
sel
op (operation)

學習重點

用VERILOG實現ALU

用VERILOG實現ALU

- 設定要達成的目標：
  - 加減法運算
  - 邏輯閘運算
  - 選擇要使用的運算方式

用VERILOG實現運算電路

- 加法器

```
1   module adder ( input [31:0] a, input [31:0] b, output [31:0] out );
2
3   always@(*) begin
4
5     assign out = a+b;
6
7   end
8
9   endmodule
```

實現選擇電路

```
1   module alu(input [31:0] a, input [31:0] b, input [31:0] op, output reg [31:0] y);
2     always@(*) begin
3       case(op)
4         3'b000: y = a + b;
5         3'b001: y = a - b;
6         3'b010: y = a * b;     // RV32I規範中未包含乘法
7         3'b011: y = a / b;     // RV32I規範中未包含除法
8         3'b100: y = a & b;
9         3'b101: y = a | b;
10        3'b110: y = ~a;        // RV32I規範中未包含NOT
11        3'b111: y = a ^ b;
12        default: y = 0;
13      endcase
14    end
15  endmodule
```