Computer Architecture and System Laboratory

# Laboratory 4
# Xilinx Vivado 介紹 & Verilog 簡介

Department of Electrical Engineering
National Cheng Kung University

國立成功大學電機系

# 實驗目的

- 瞭解硬體描述語言 Verilog
- 熟悉Vivado

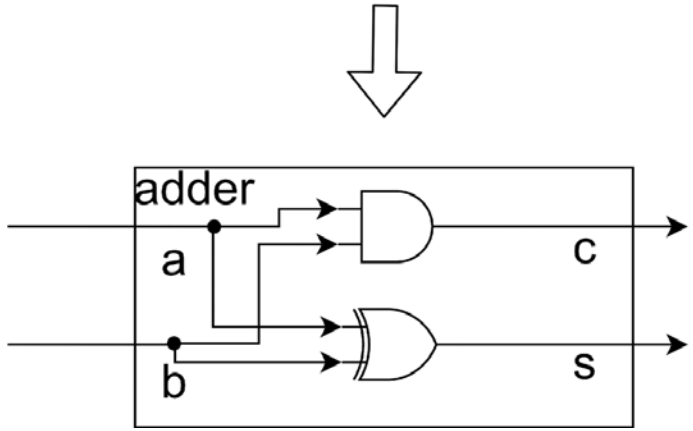
# Outline

- Verilog 簡介
- 使用Verilog 邏輯閘描述電路
- 如何使用Vivado模擬
- 實作題 1
  - 使用Verilog實作**保全系統**與**接線生**
- 實作題 2
  - 使用Verilog實作**全加器**與**半加器**

# Verilog Introduction

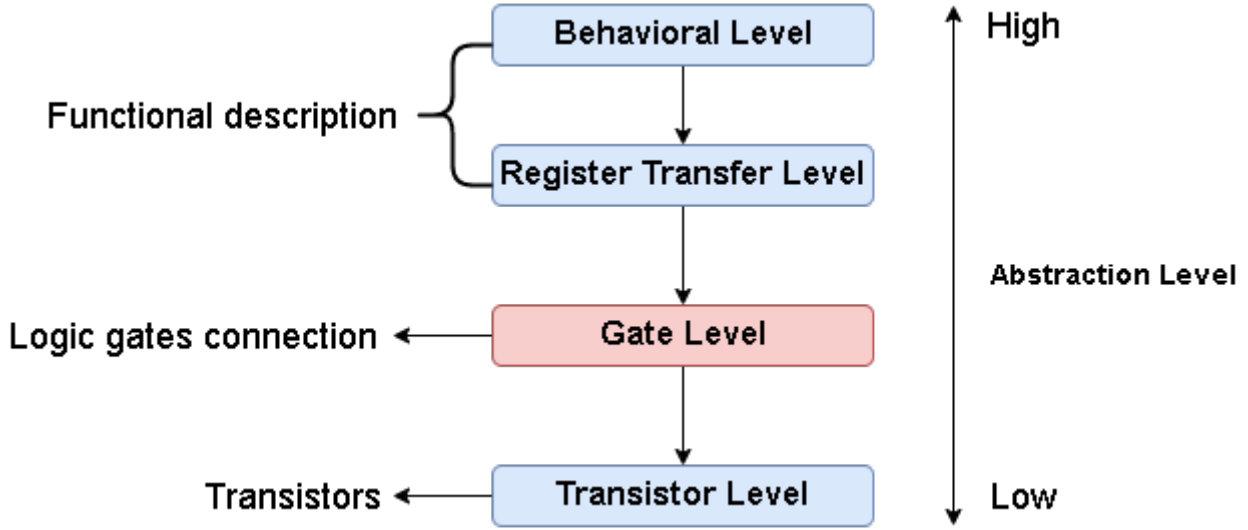
- Verilog 是個用來描述硬體的語言
- 以語言描述硬體避免複雜的模擬

```
module adder (input a, input b, output c, output s);
        assign c = a & b;
        assign s = a ^ b;
endmodule
```



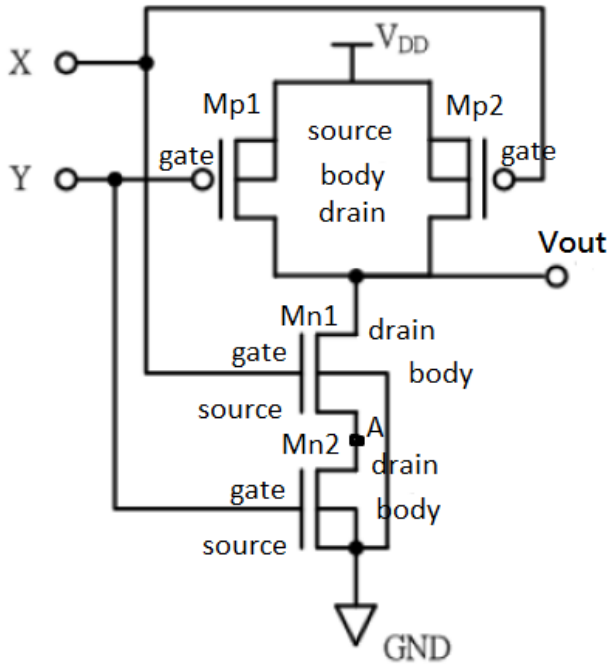
# Verilog Introduction

- Verilog 主要可以分成四個 Level 來描述電路,由低到高分別是 Transistor Level,Gate Level,Register Transfer Level,Behavioral Level
- Verilog 允許不同Level描述混合使用
- 這次的實作題會利用 Gate Level 來描述目標電路



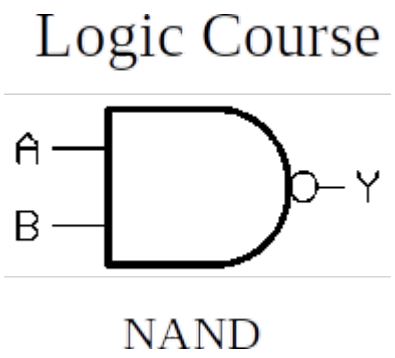
# Verilog Introduction

- Transistor Level to Gate Level




NAND Gate in **Transistor Level**



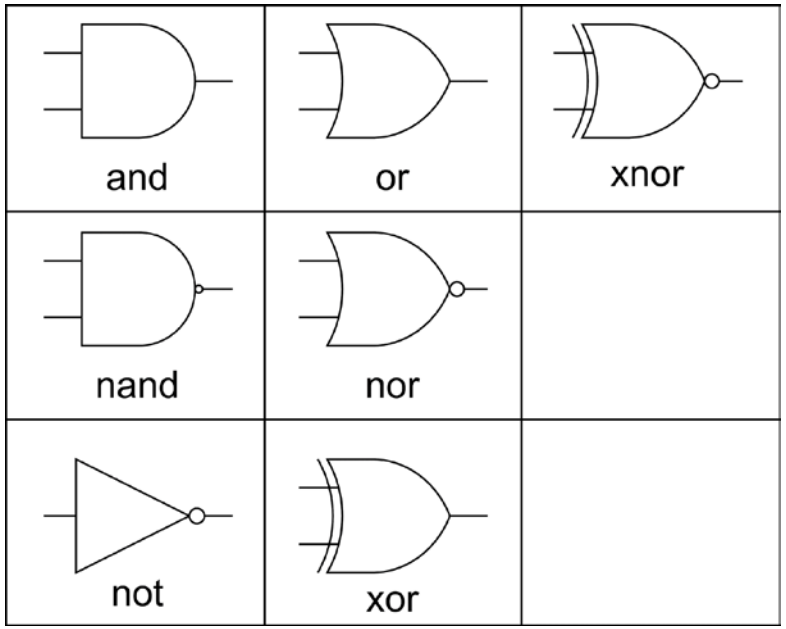



NAND Gate in **Gate Level**

Source : Prof. Chung-Ho Chen

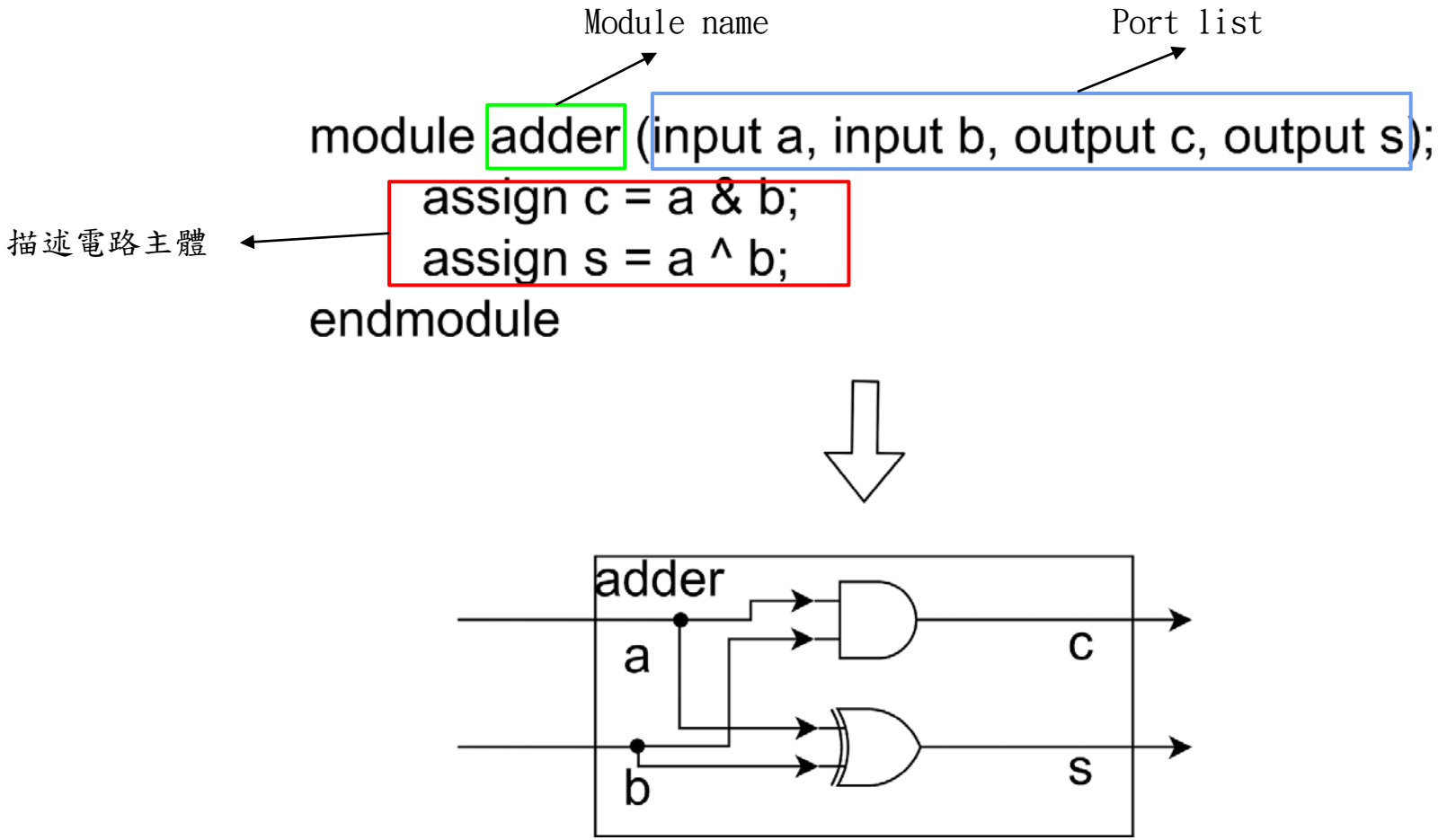
# Describe circuitry with verilog

- 這次會實作在Gate Level層級，描述電路的基本元件為**邏輯閘**
- Verilog 有提供一些內建的邏輯閘可以直接使用，如下圖所示。接下來會利用這些內建的邏輯閘來完成實作題1和實作題2



# Constructing Circuitry

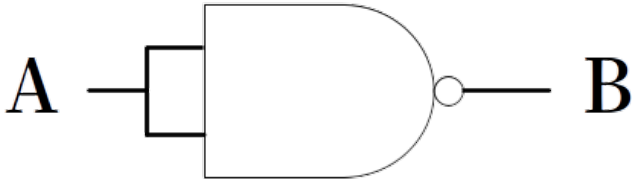
- Verilog 基礎架構



# Constructing Circuitry

<Gate Type> <Identifier>(Output,Input1,Input2)

- Verilog 基本邏輯閘宣告
  - **Gate type**: verilog 提供內建的邏輯閘名稱 (and, nand, or, xor, nor, xnor, not ……)
  - **Identifier (Optional)**：邏輯閘名稱或編號，可以簡略不寫




Lab 1 problem ： 以NAND實作NOT

```
module NOT (input A, output B);
  nand (B,A,A);
endmodule
```

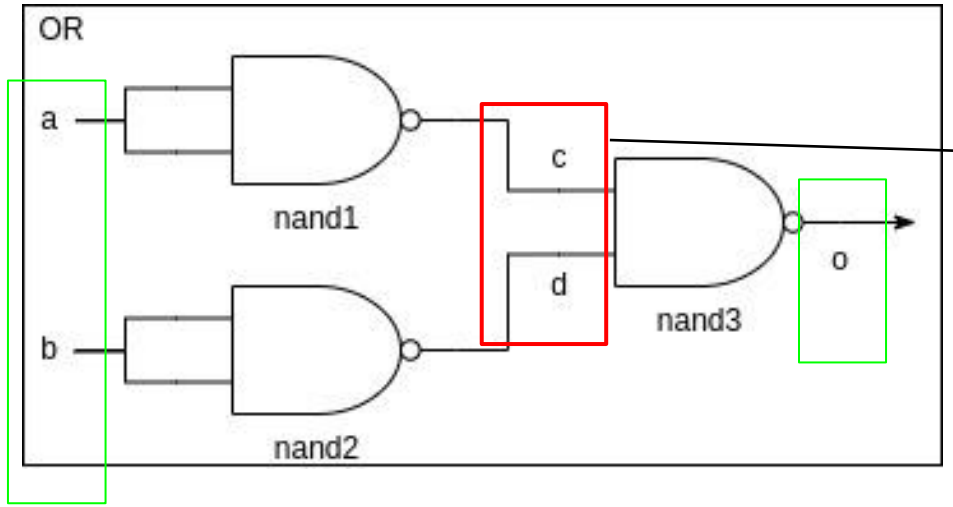
(output,input,input)

Gate Type

省略 Identifier

# Constructing Circuitry

- 使用 wire 來連接不同的邏輯閘



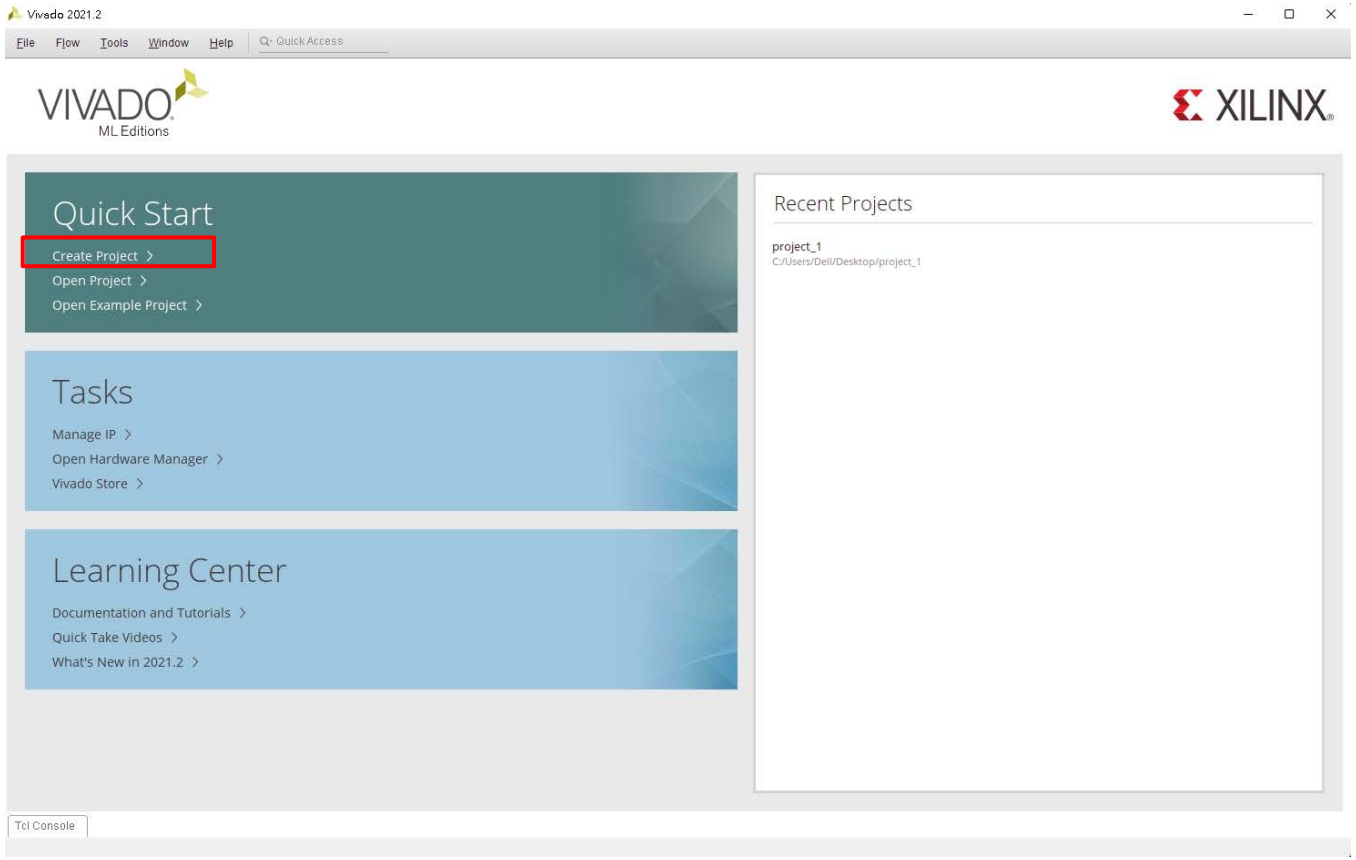

```
module OR (input a,input b,output o);
  wire c,d;
  nand nand1(c,a,a);
  nand nand2(d,b,b);
  nand nand3(o,c,d);
endmodule
```

Gate Type　　Identifier　　(output,input,input)

Lab 1 problem : 以NAND實作OR

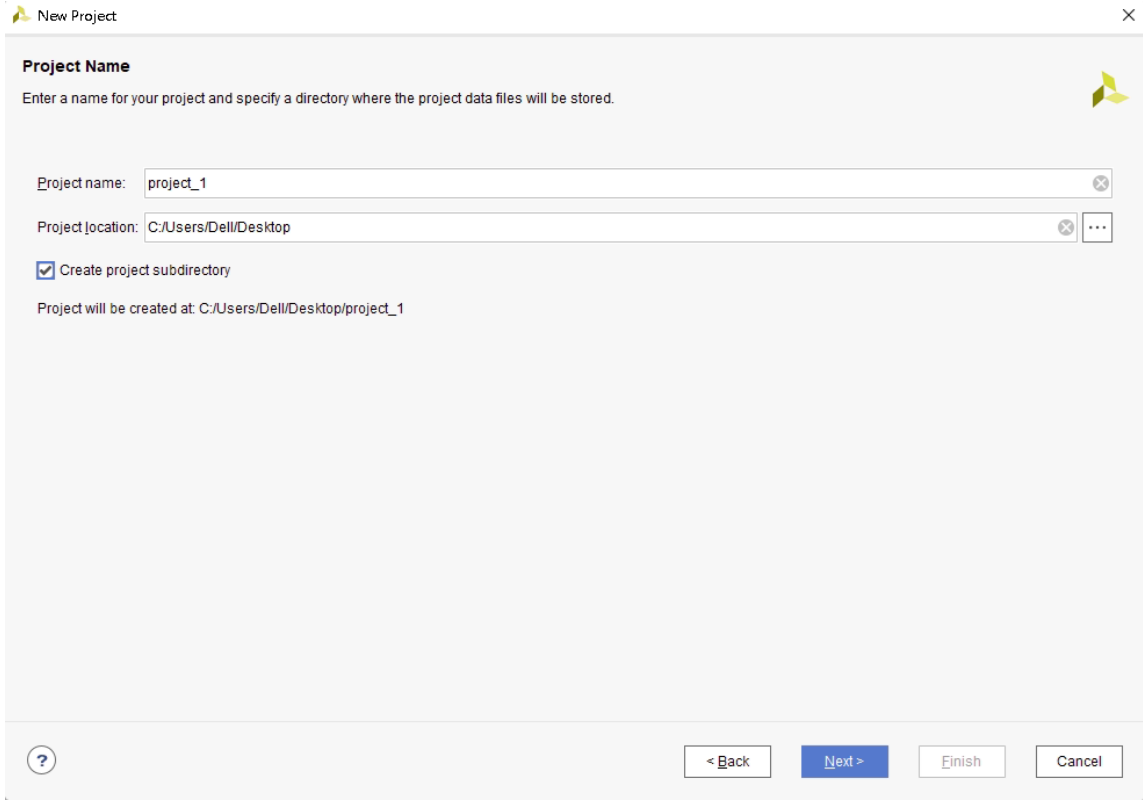
# Xilinx Vivado

- 執行Vivado並創建一個新的專案



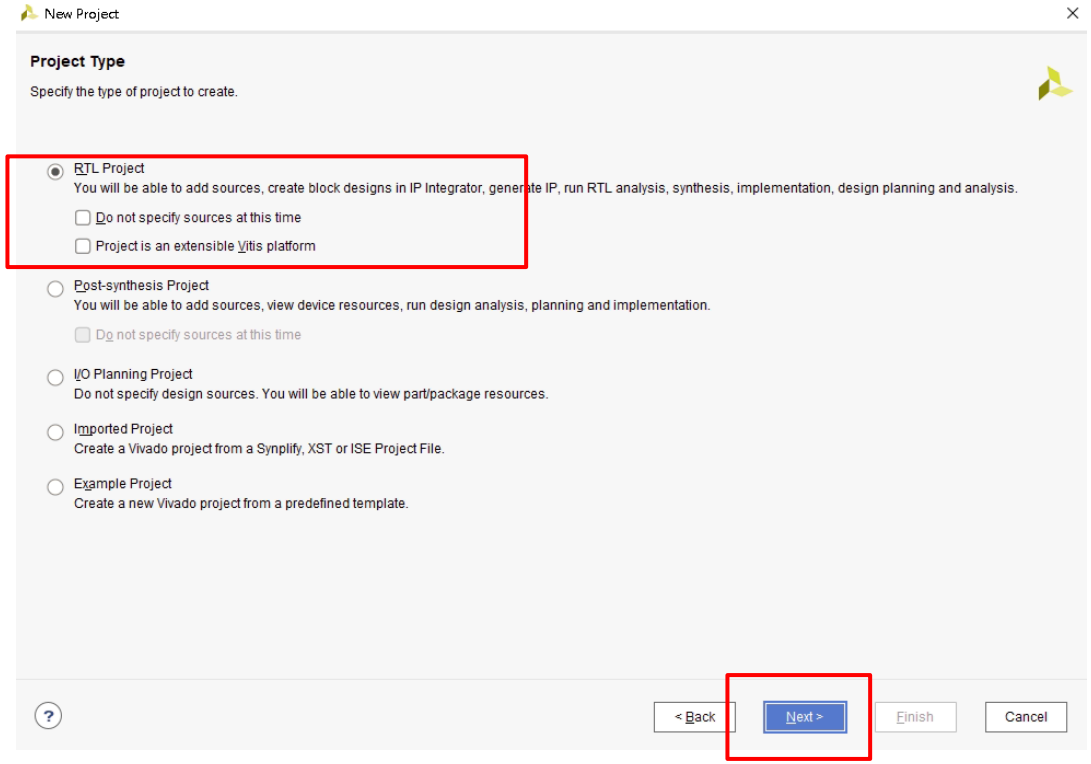
# Xilinx Vivado

- 選擇路徑



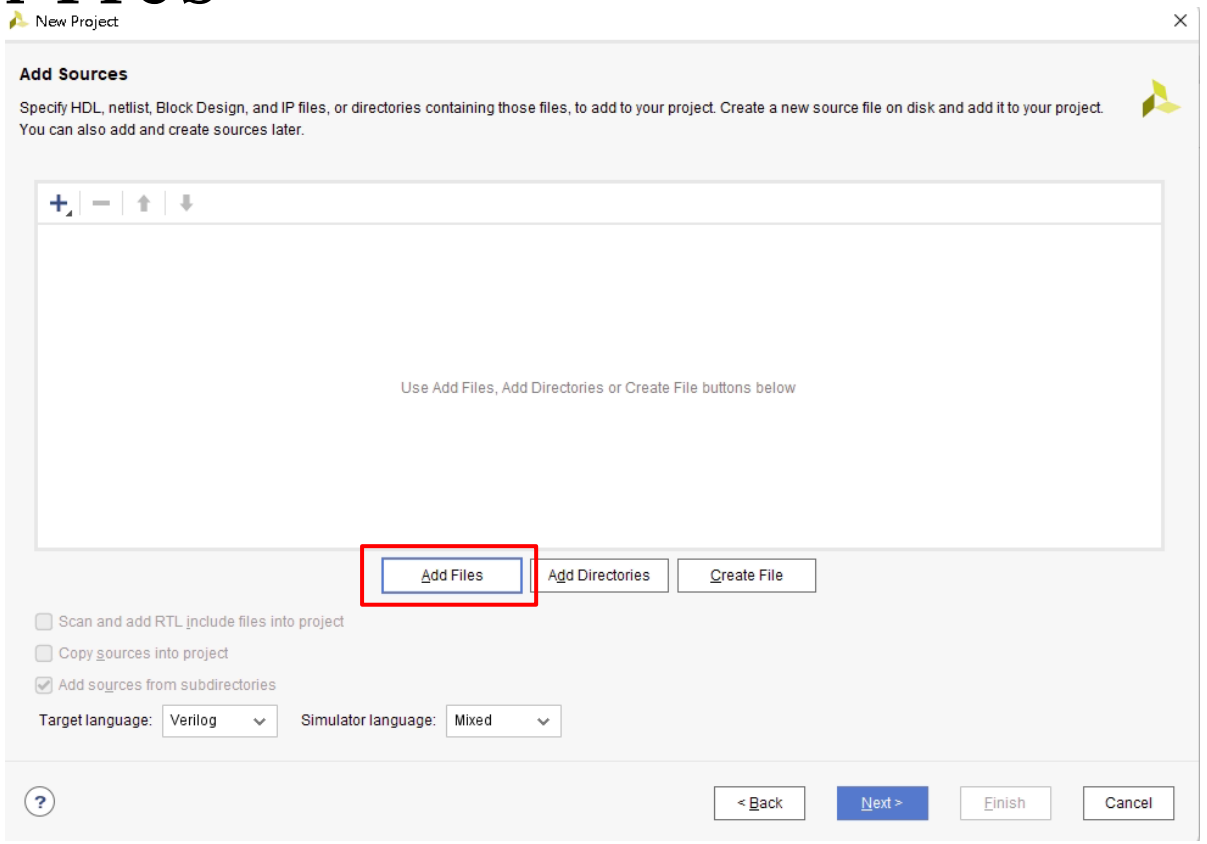
# Xilinx Vivado

- 勾選RTL後直接點Next



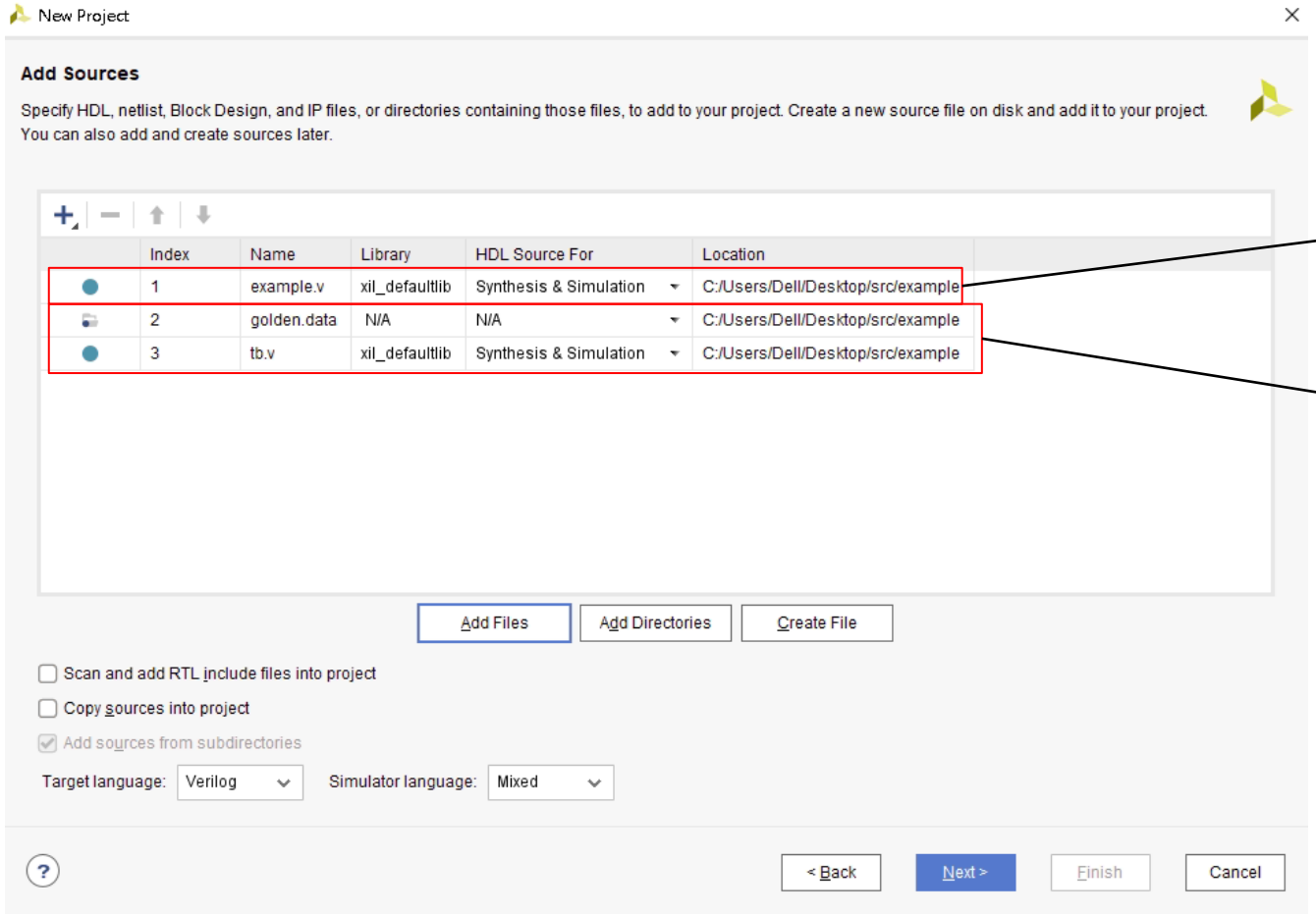
# Xilinx Vivado

- 選擇Add Files



# Xilinx Vivado

- TA提供的檔案加入專案

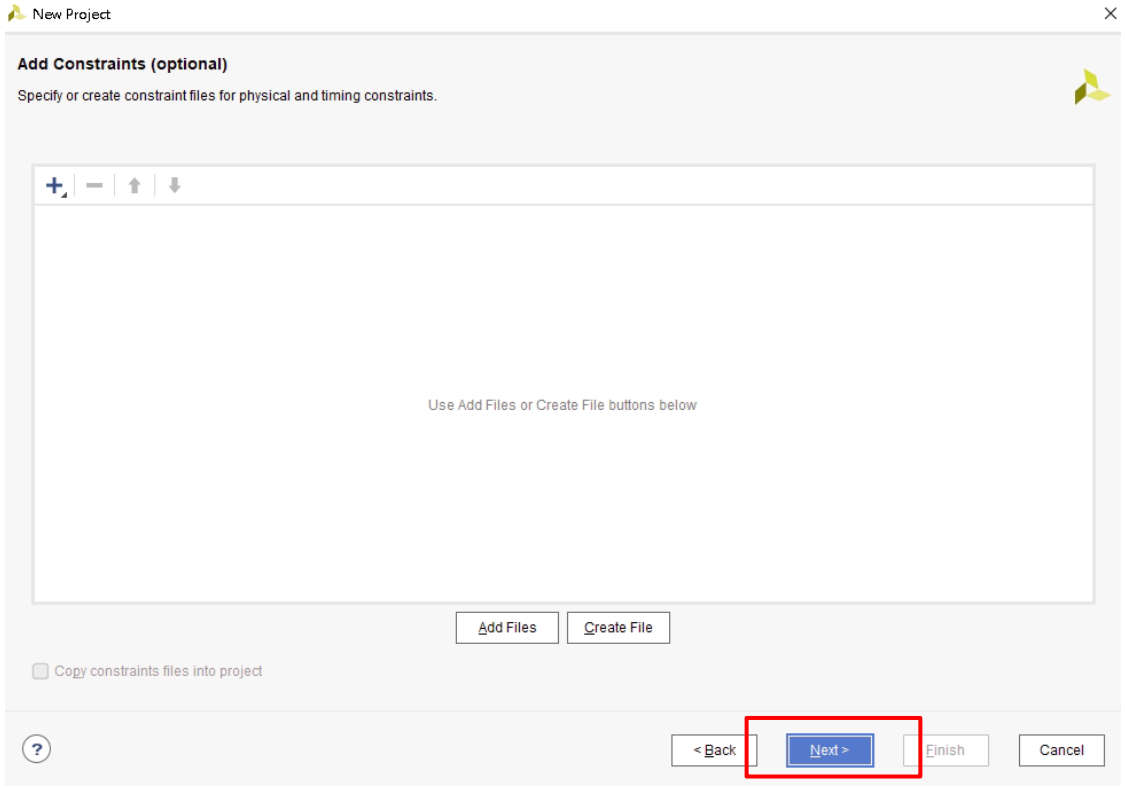



需要實作的Verilog檔案

用來驗證的檔案

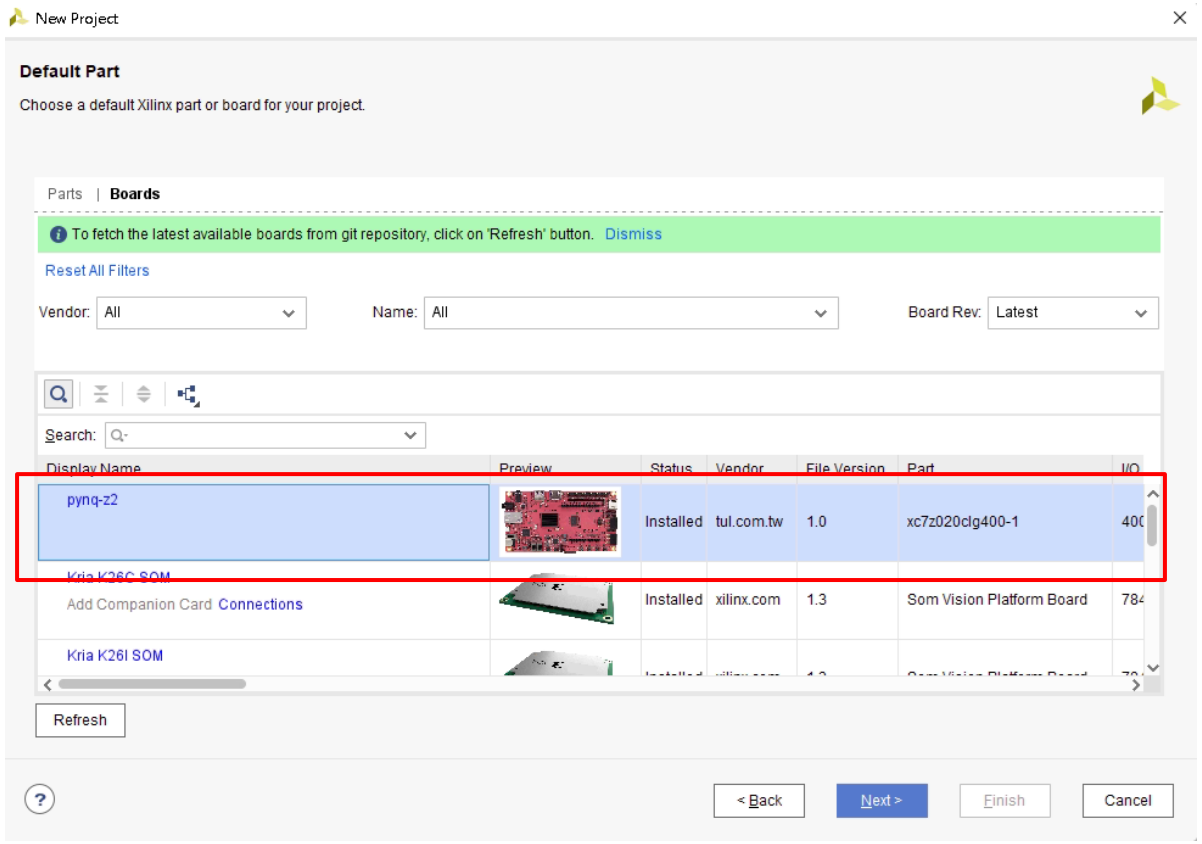
# Xilinx Vivado

- 點選Next



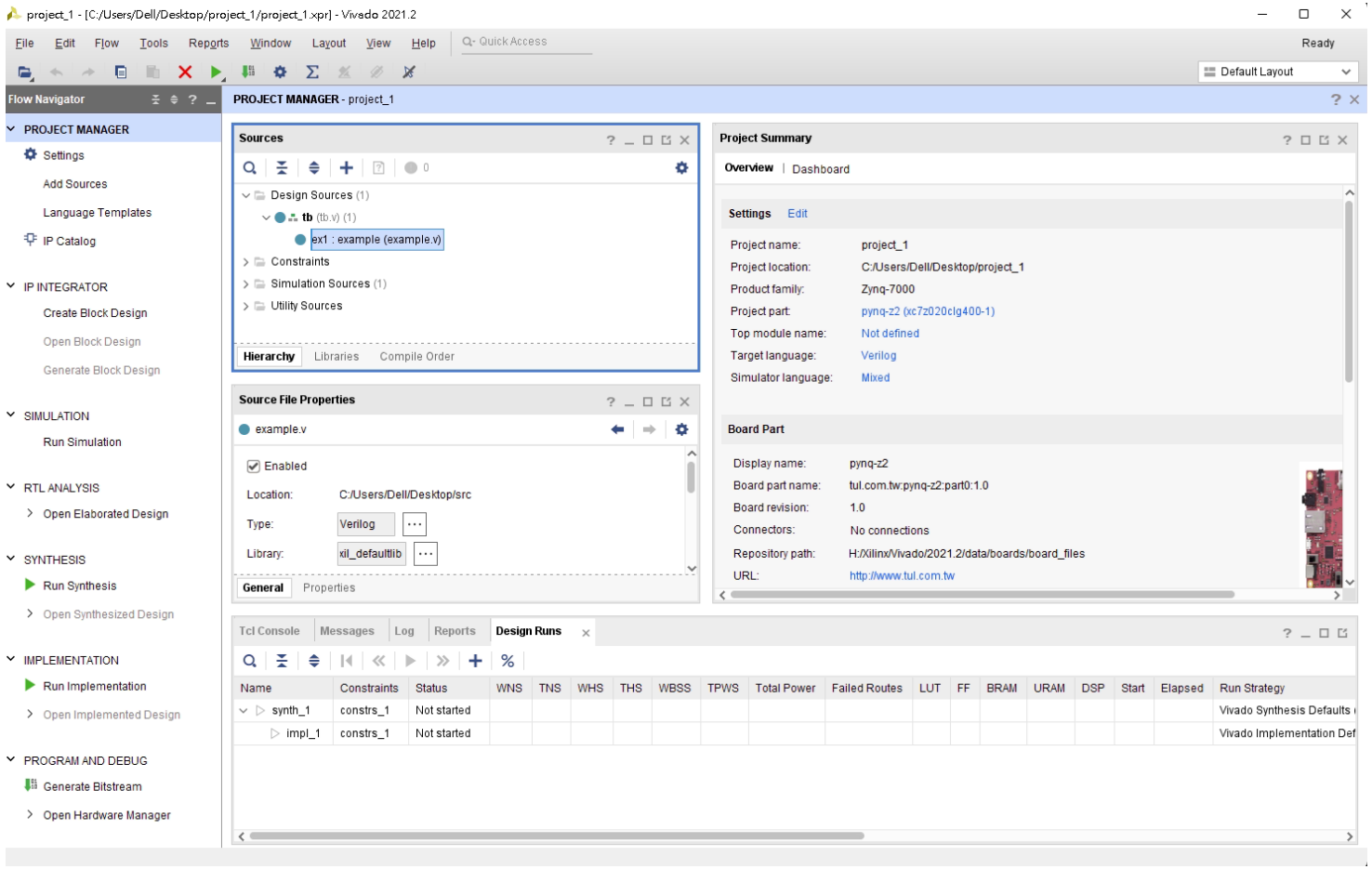
# Xilinx Vivado

- 選擇pynq-z2的板子



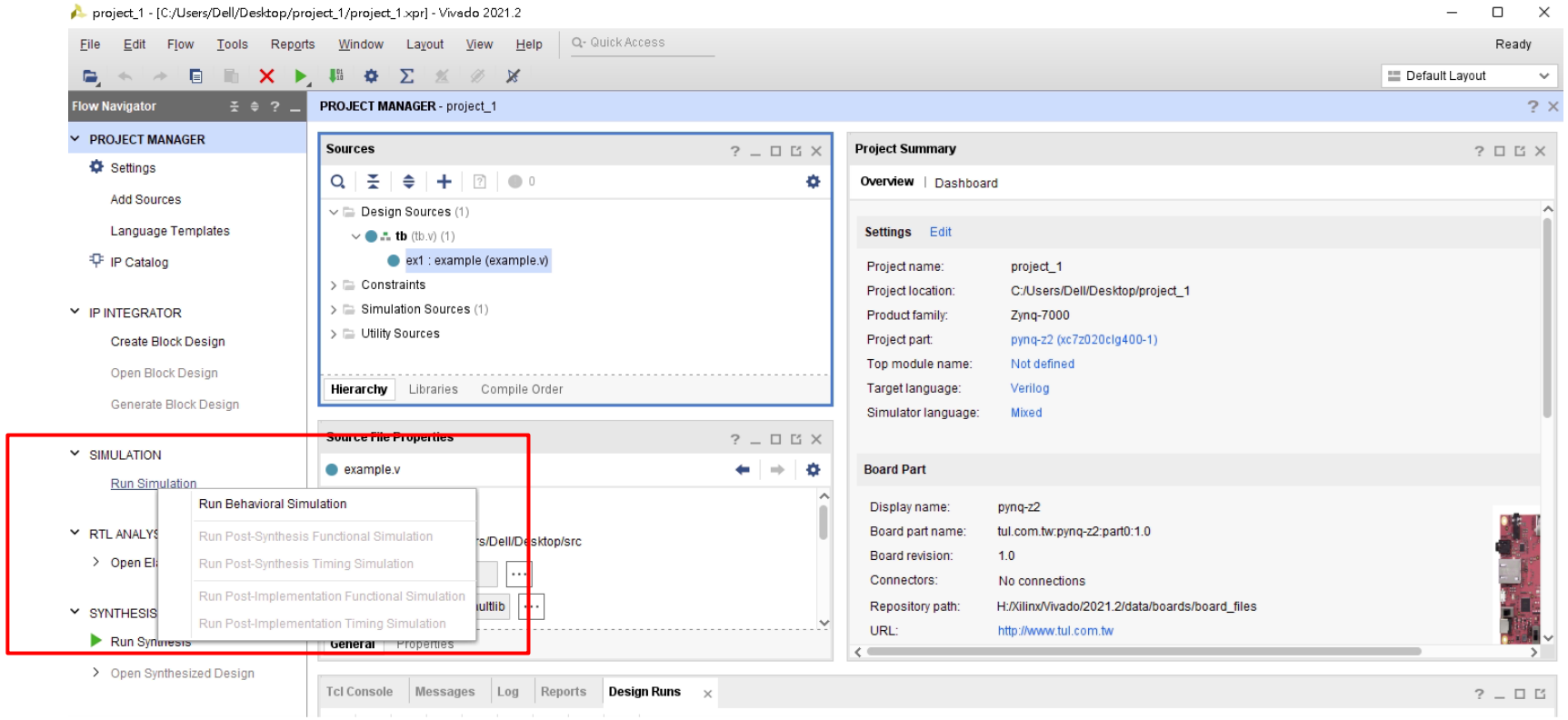
# Xilinx Vivado

- 完成後就會進入主要頁面



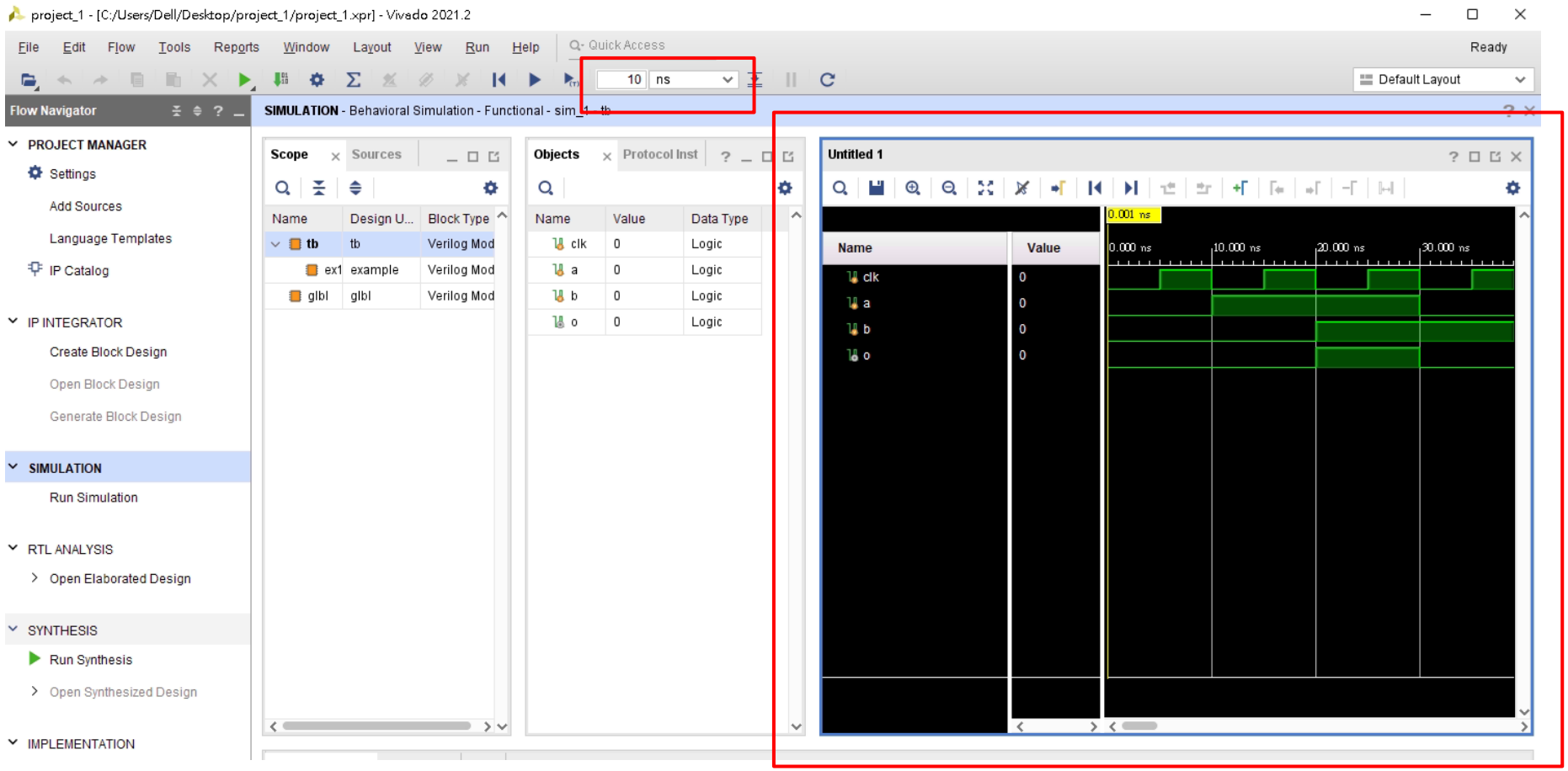
# Xilinx Vivado

- 選擇左邊的SIMULATION->Run Simulation->Run Behavioral Simulation



# Xilinx Vivado

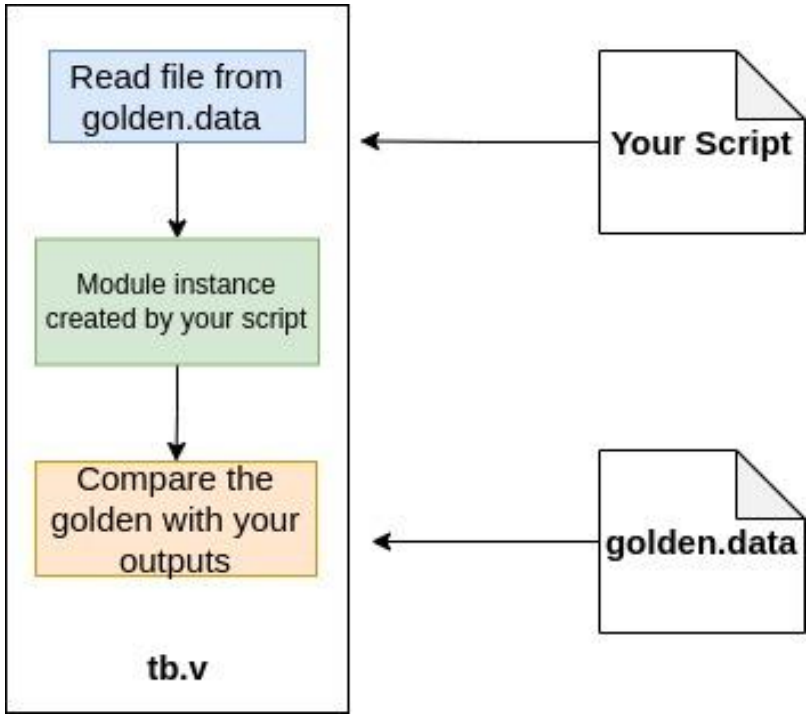
- 將間隔改為10 NS 並利用右邊的waveform來檢查自己所寫的電路



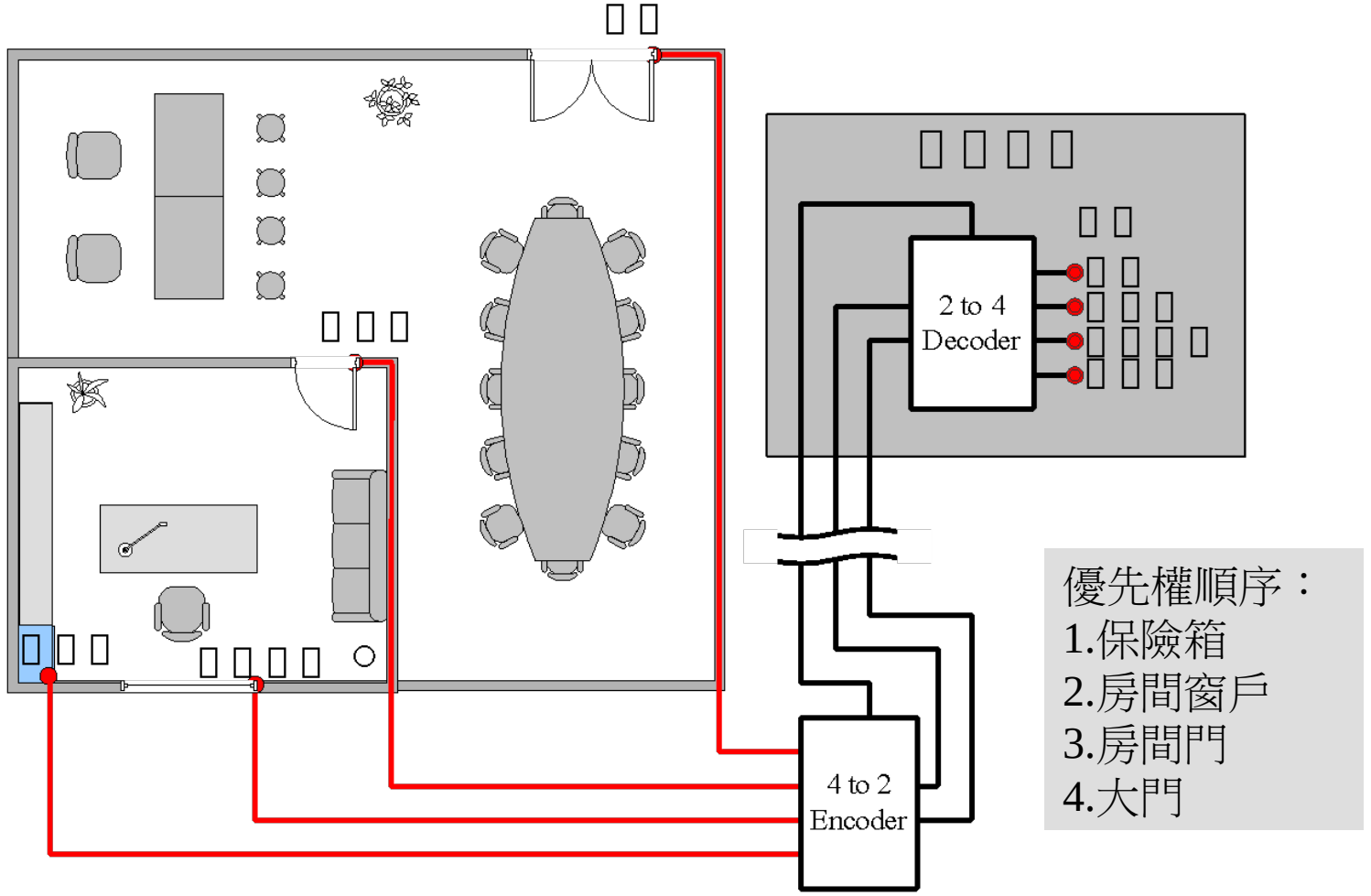
# TestBench

- Testbench 驗證流程
  - 從golden.data讀資料
  - 利用寫完的verilog file在tb.v裡面生成一個module instance
  - 比較輸出跟golden.data的差異

Note:module name & port list 必須要跟 testbench file 裡面的一樣

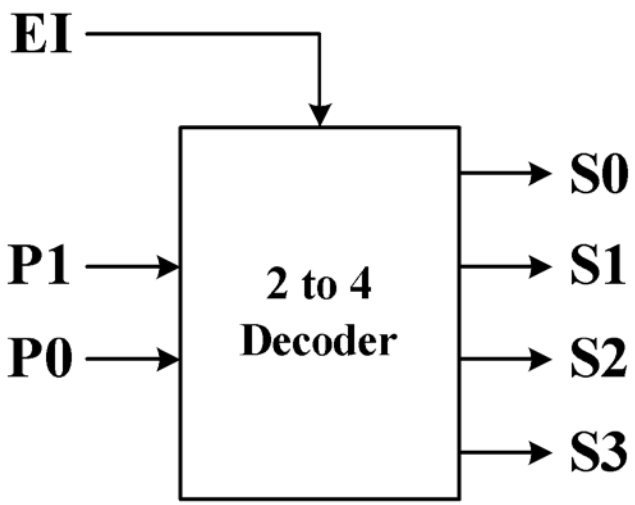


# 實作題 1-1：保全系統



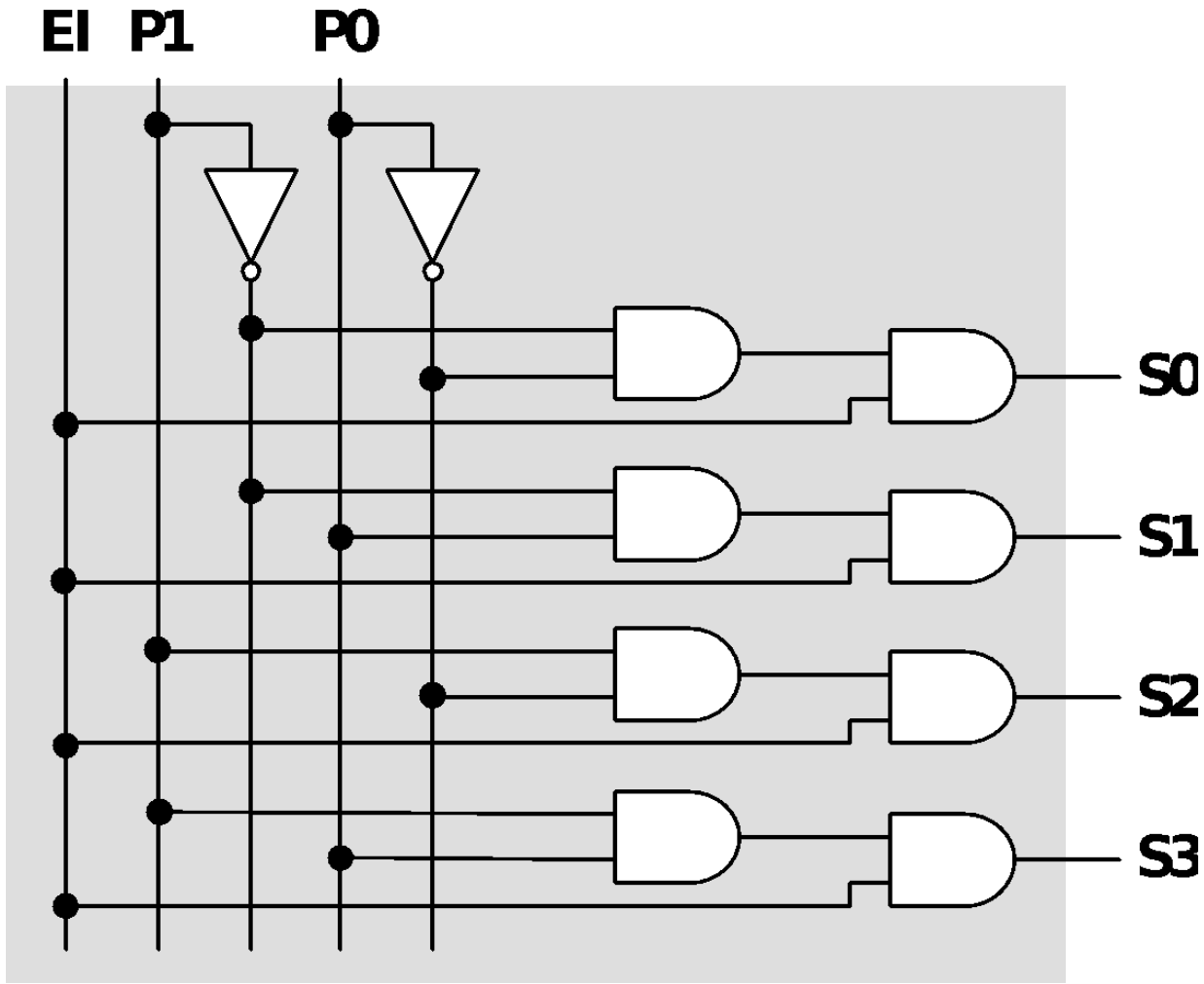

優先權順序：
1.保險箱
2.房間窗戶
3.房間門
4.大門

# 編/解碼器：解碼器(Decoder)

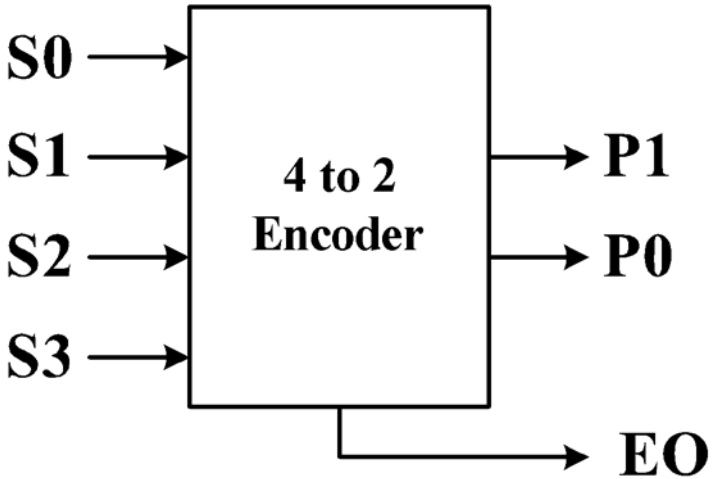



| Input | | | Output | | | |
|---|---|---|---|---|---|---|
| EI | P1 | P0 | S0 | S1 | S2 | S3 |
| 0 | x | x | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |

# 編/解碼器：解碼器(Decoder)

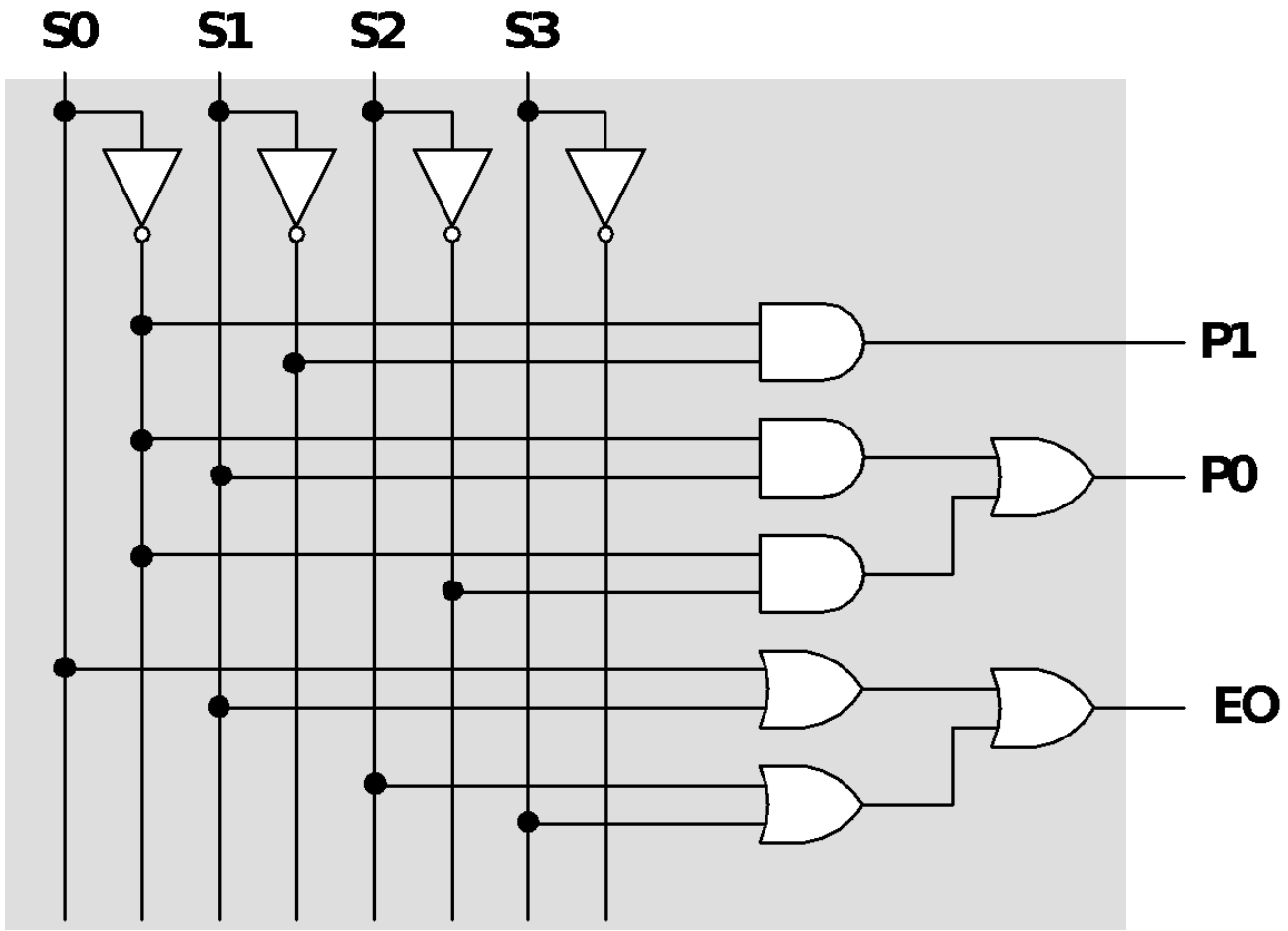


# 編/解碼器：編碼器(Encoder)




| Input | | | | Output | | |
|---|---|---|---|---|---|---|
| S0 | S1 | S2 | S3 | P1 | P0 | EO |
| 1 | x | x | x | 0 | 0 | 1 |
| 0 | 1 | x | x | 0 | 1 | 1 |
| 0 | 0 | 1 | x | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | x | x | 0 |

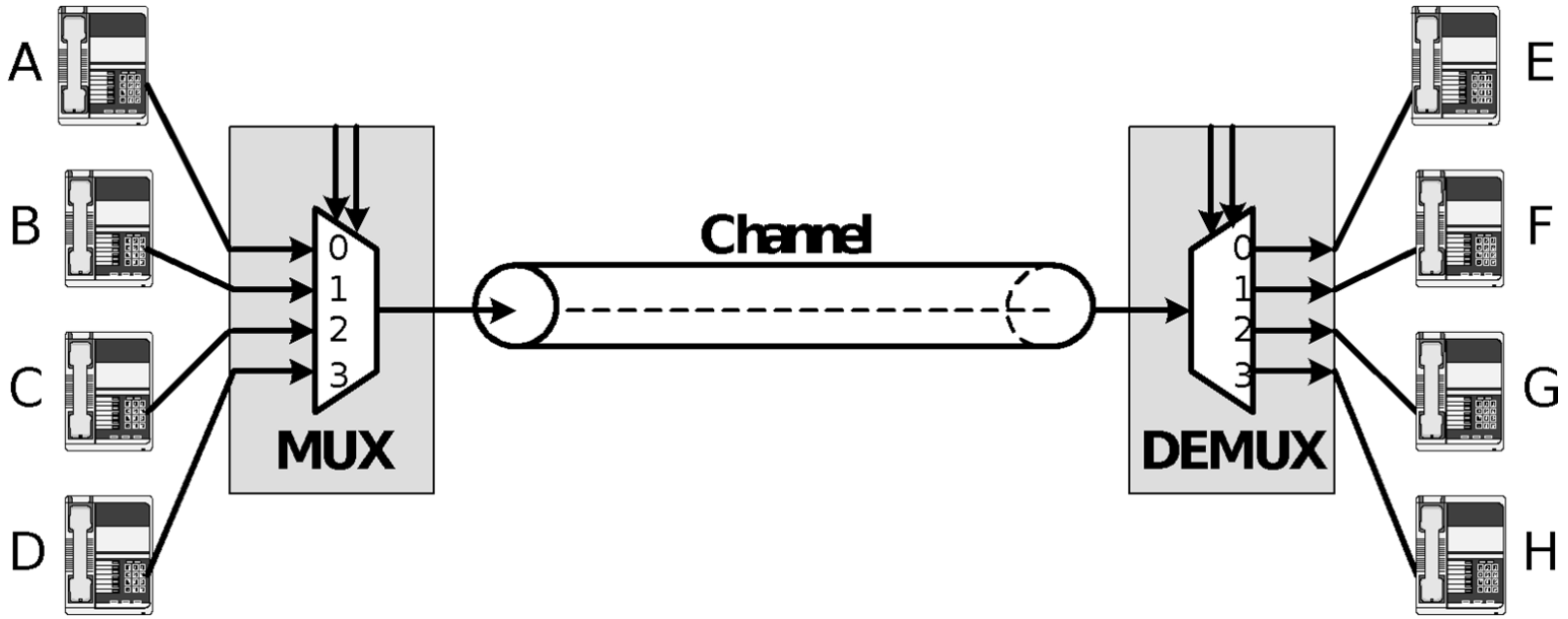
# 編/解碼器：編碼器(Encoder)



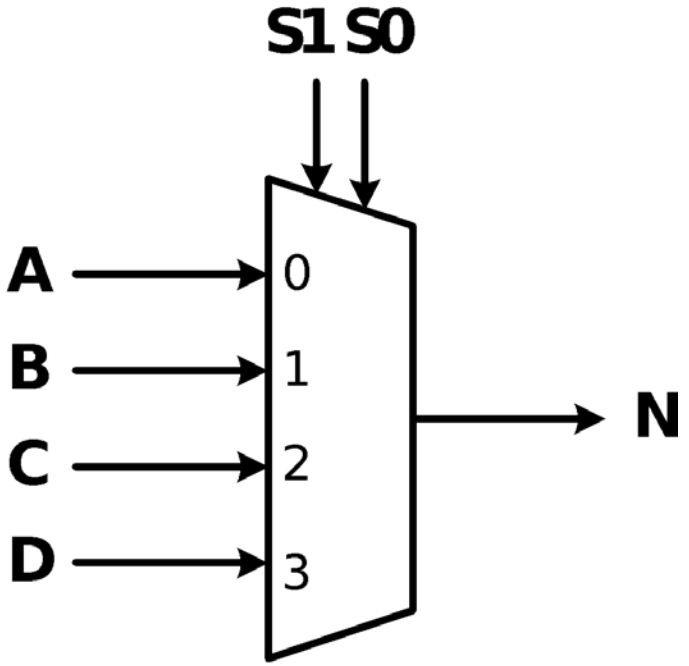
# 保全系統 Port list

- Input
  - s0 : 保險箱
  - s1 : 房間窗戶
  - s2 : 房間門
  - s3 : 大門
- Output
  - e : 保險箱
  - f : 房間窗戶
  - g : 房間門
  - h : 大門

# 實作題 1-2：接線生

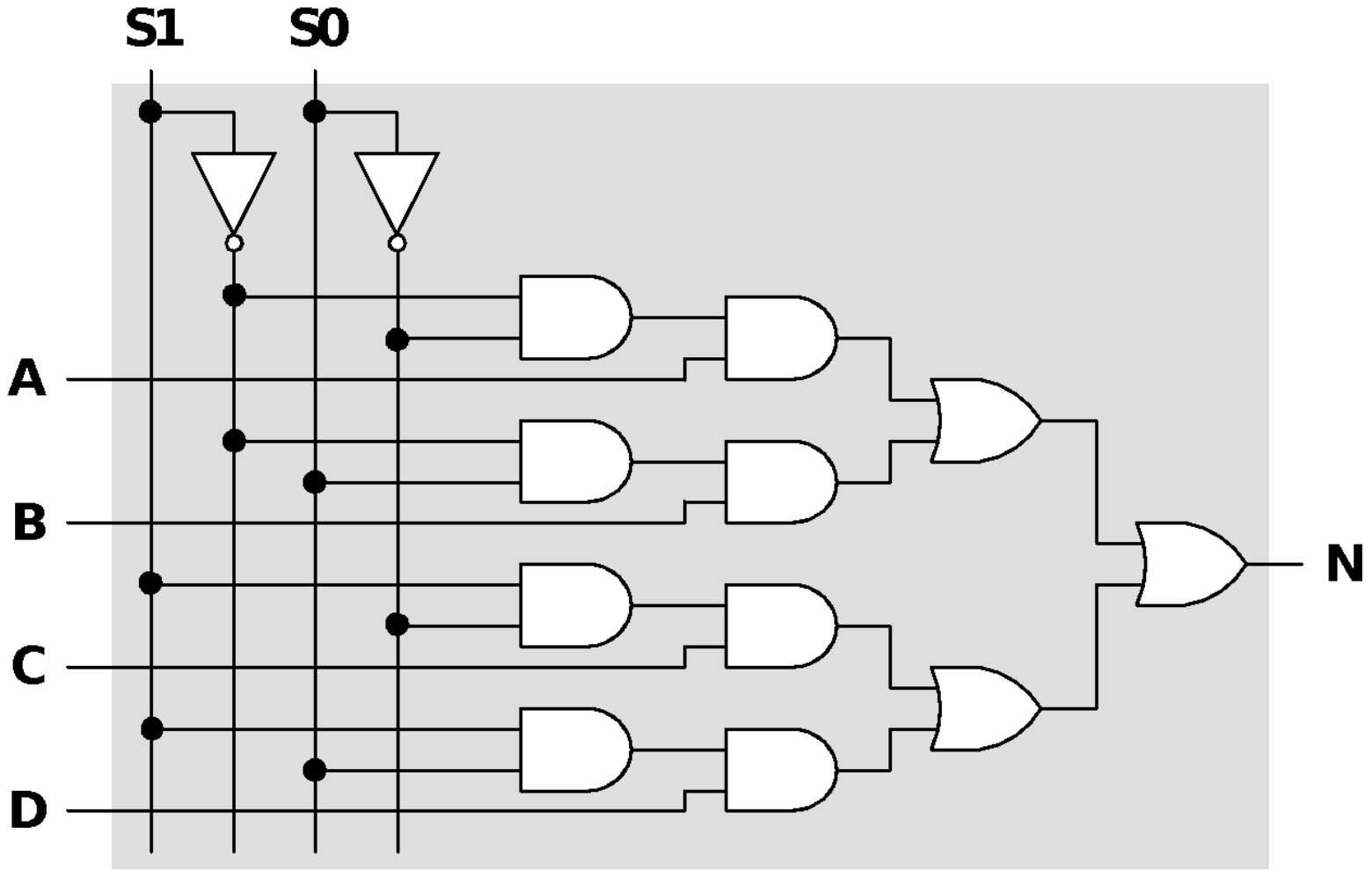


# 多工器(MUX)

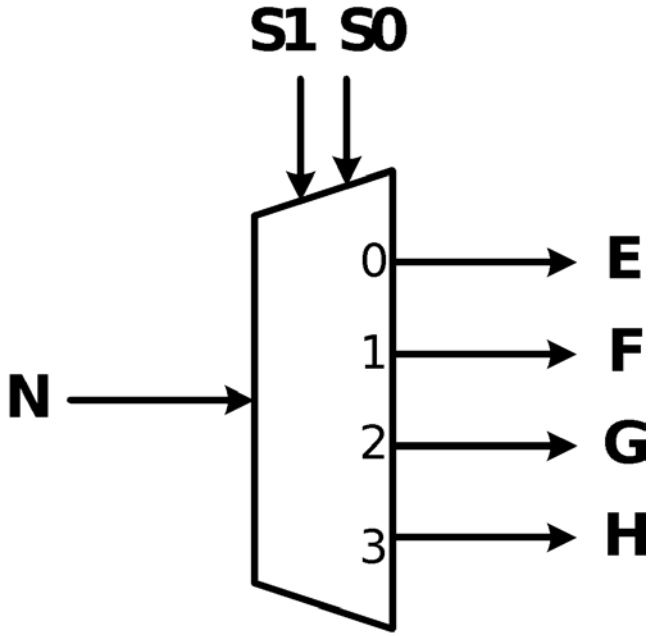



| S1 | S0 | Output |
|---|---|---|
| 0 | 0 | N=A |
| 0 | 1 | N=B |
| 1 | 0 | N=C |
| 1 | 1 | N=D |

# 多工器(MUX)

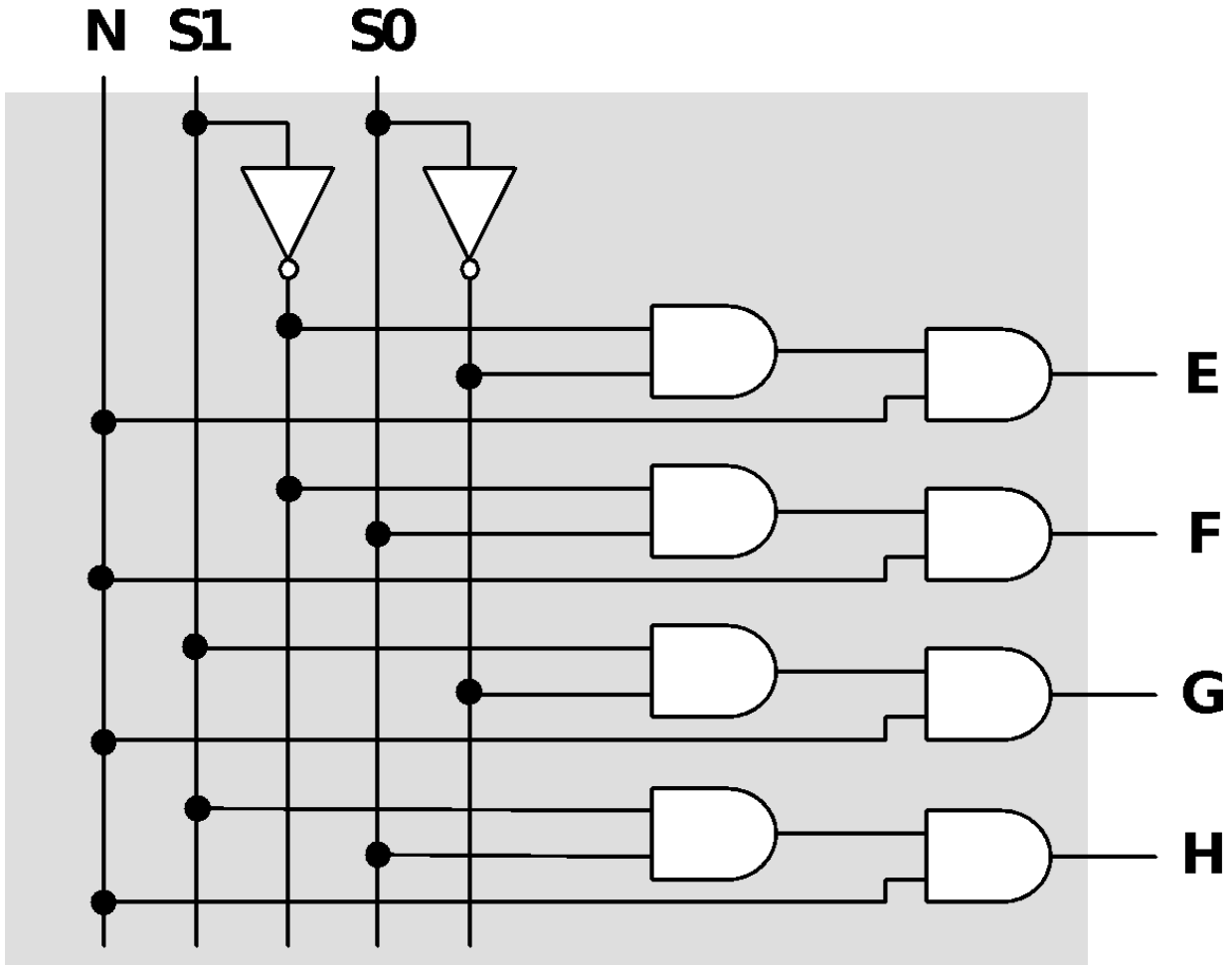


# 解多工器(DEMUX)




| S1 | S0 | E | F | G | H |
|---|---|---|---|---|---|
| 0 | 0 | N | x | x | x |
| 0 | 1 | x | N | x | x |
| 1 | 0 | x | x | N | x |
| 1 | 1 | x | x | x | N |

# 解多工器(DEMUX)

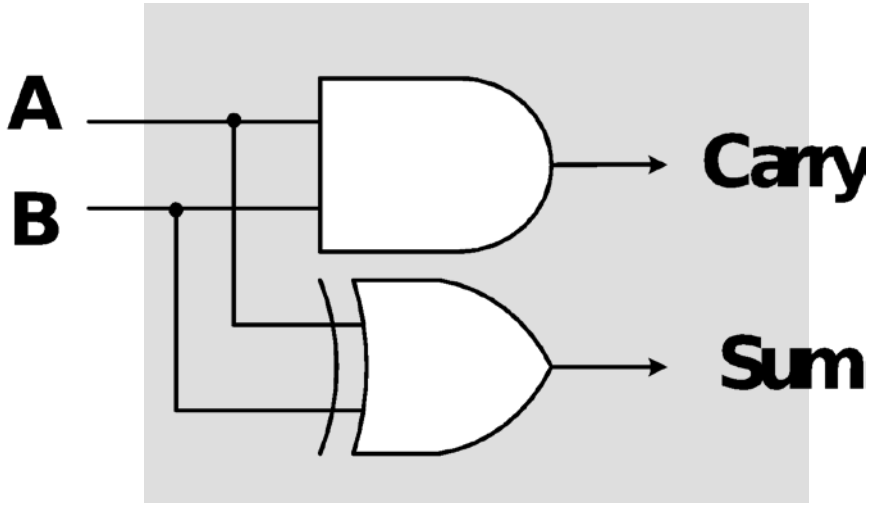


# 接線生 Port list

- Input
  - a, b, c, d: Mux input signals
  - mux_s0, mux_s1: mux s0, s1 signal
  - demux_s0, demux_s1: demux s0, s1 signal
- Output
  - e, f, g, h : demux output signals

# 實作題 2-1：半加器

| A | B | Carry | Sum |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

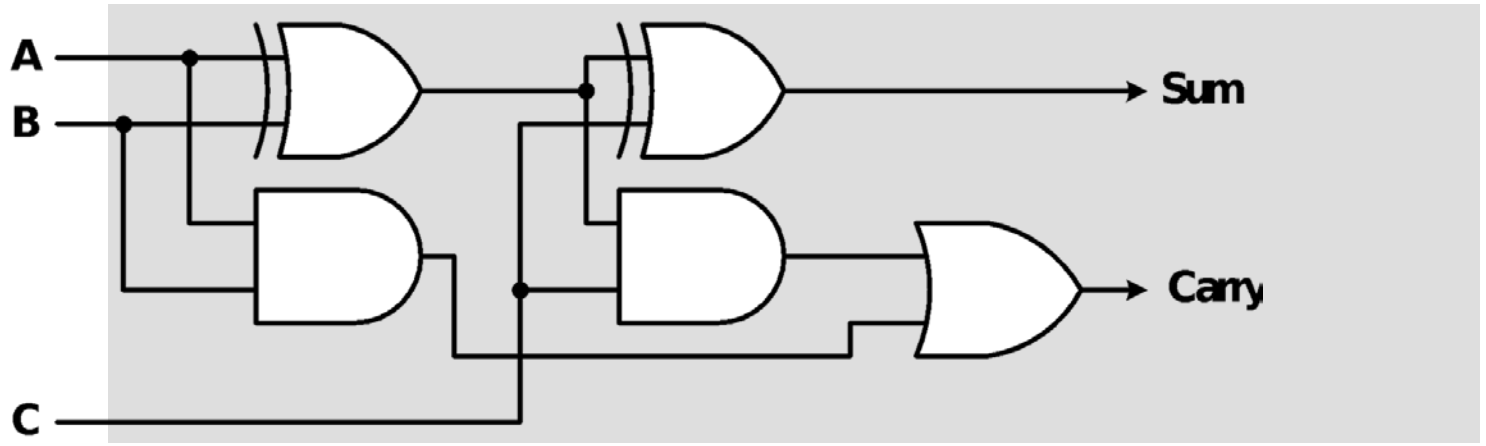


# 半加器 Port list

- Input
  - a,b:input signals
- Output
  - c: carry output signal
  - s: sum output signal

# 實作題 2-2：全加器

| A | B | C | Carry | Sum |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |



# 全加器 Port list

- Input
  - a, b, c:input signals
- Output
  - carry: carry output signal
  - s: sum output signal

# 本次需要檢查的內容

- 實作題 1 & 2 Pass

```
*****************************          _.-.     _,.-'""`-._
**  Congratulations !!     **        (,-.`._,'(       |\`-/|
**  Simulation1 PASS!!     **            `-.-' \ )-`( , o o)
*****************************                  `-    \`_`"'-
```

# 本次結報內容

- 實作(一)
  - **保全系統** 波形截圖並解釋
  - **接線生** 波形截圖並解釋
- 實作(二)
  - **半加器** 波形截圖並解釋
  - **全加器** 波形截圖並解釋
- 實驗心得

# 本次結報需要附上的內容

- 實作題 1-1 ： 保全系統
- 五種可能各截一張並解釋

| 保險箱 | 房間窗戶 | 房間門 | 大門 | Output |
|---|---|---|---|---|
| 1 | X | X | X | 保險箱 |
| 0 | 1 | X | X | 房間窗戶 |
| 0 | 0 | 1 | X | 房間門 |
| 0 | 0 | 0 | 1 | 大門 |
| 0 | 0 | 0 | 0 | 沒有輸出 |

# 本次結報需要附上的內容

- 實作題 1-2 ： 接線生
- 根據真值表截出所有波形並解釋

| A | B | C | D | Mux Sel | Demux Sel | Output |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | A | H | H=1 |
| 1 | 0 | 0 | 0 | D | E | All=0 |
| 0 | 1 | 0 | 0 | B | G | G=1 |
| 0 | 1 | 0 | 0 | C | F | All=0 |
| 0 | 0 | 1 | 0 | C | E | E=1 |
| 0 | 0 | 1 | 0 | A | G | All=0 |
| 0 | 0 | 0 | 1 | D | F | F=1 |
| 0 | 0 | 0 | 1 | B | H | All=0 |

# 本次結報需要附上的內容

- 實作題 2-1 ： 半加器
- 根據真值表截出所有波形並解釋

| A | B | Carry | Sum |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

# 本次結報需要附上的內容

- 實作題 2-2 ： 全加器
- 根據真值表截出所有波形並解釋

| A | B | C | Carry | Sum |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |