

Laboratory 4

Xilinx Vivado 介紹 & Verilog 簡介



Department of Electrical Engineering
National Cheng Kung University

實驗目的

- 瞭解硬體描述語言 Verilog
- 熟悉Vivado

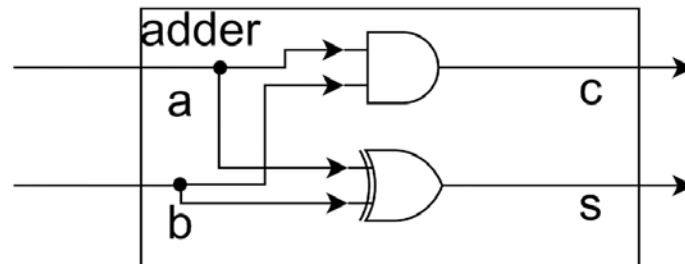
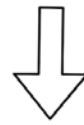
Outline

- 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

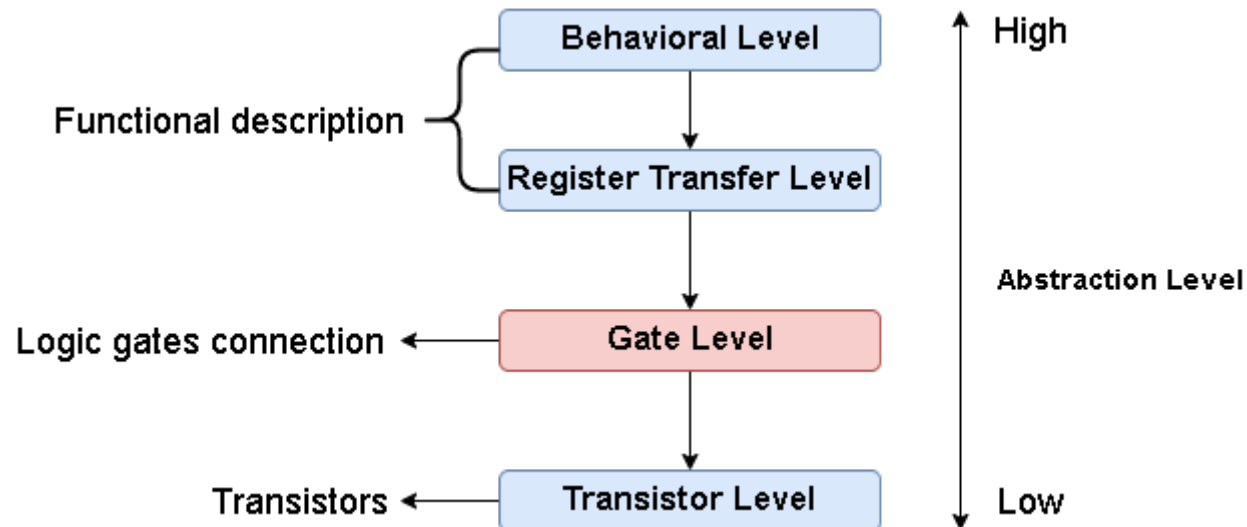
- Verilog 是個用來描述硬體的語言
- 以語言描述硬體避免複雜的模擬

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



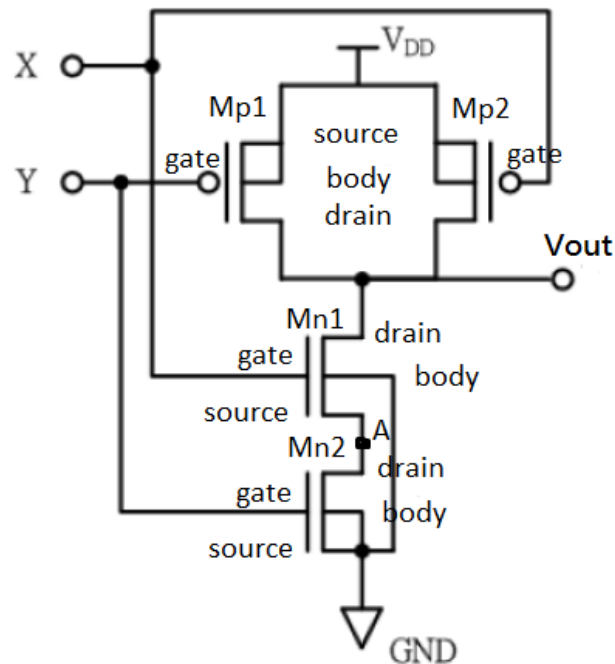
Verilog Introduction

- 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



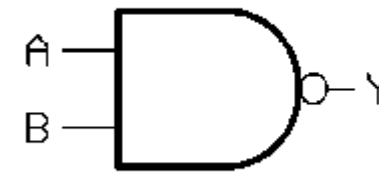
Verilog Introduction

- Transistor Level to Gate Level



NAND Gate in Transistor Level

Logic Course



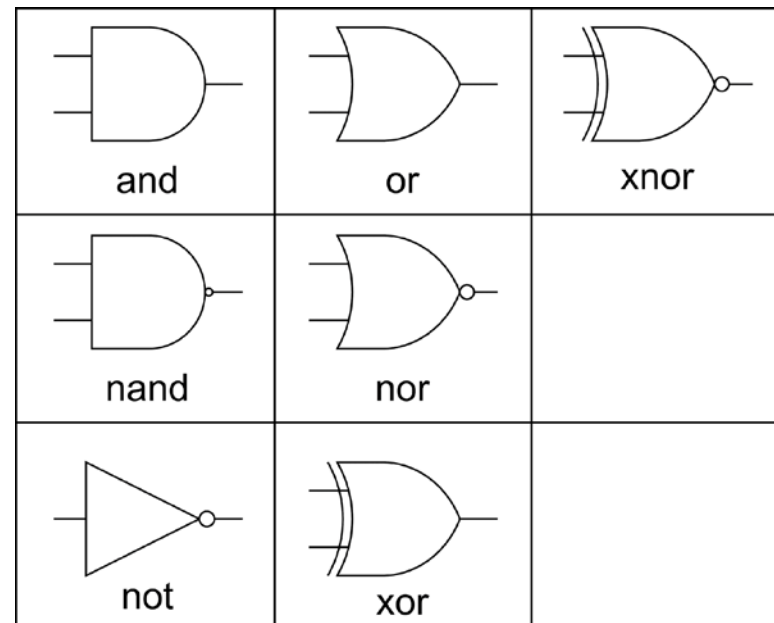
NAND

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

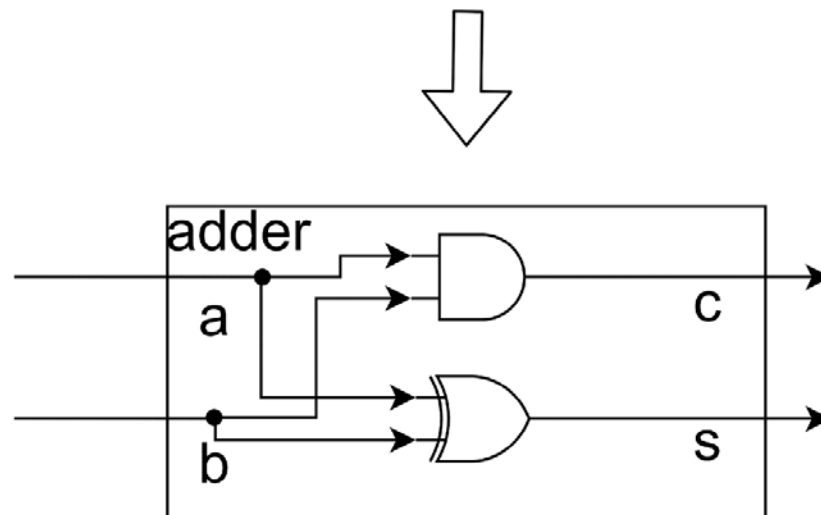
- Verilog 基礎架構

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

Module name

Port list

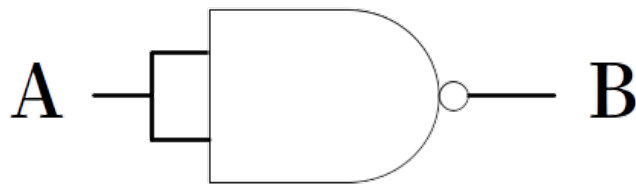
描述電路主體



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
```

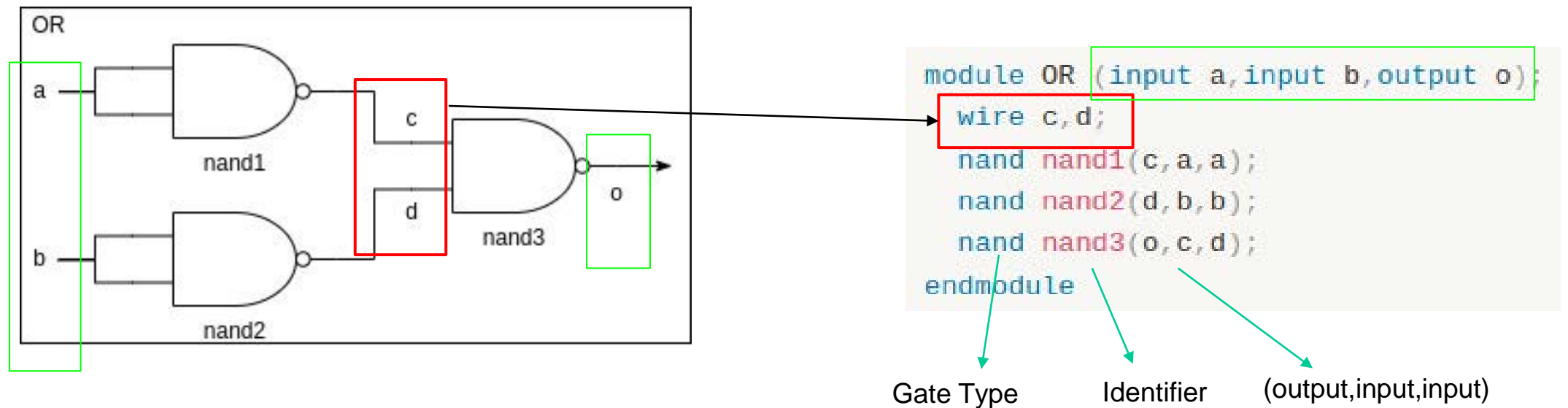
Gate Type

省略 Identifier

(output,input,input)

Constructing Circuitry

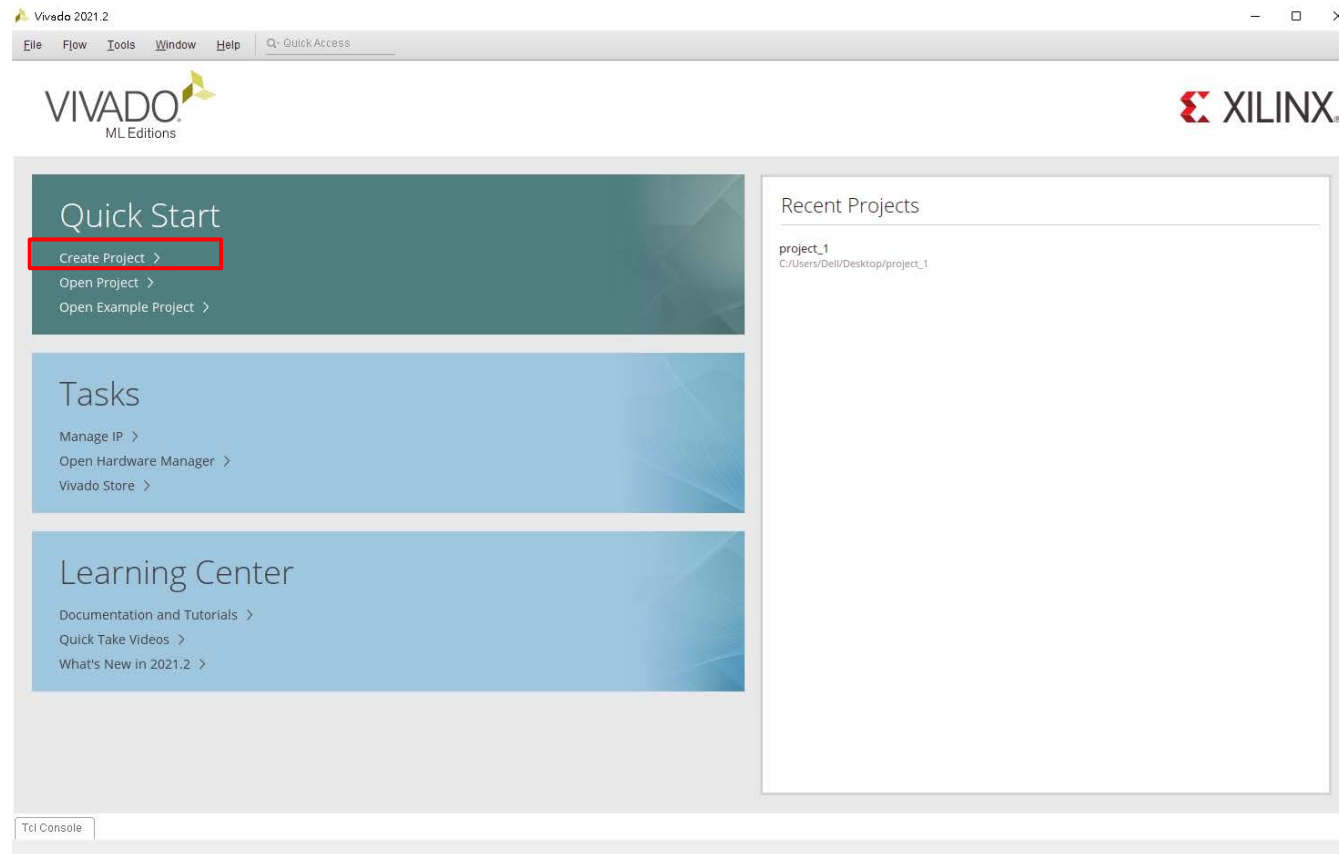
- 使用 **wire** 來連接不同的邏輯閘



Lab 1 problem : 以NAND實作OR

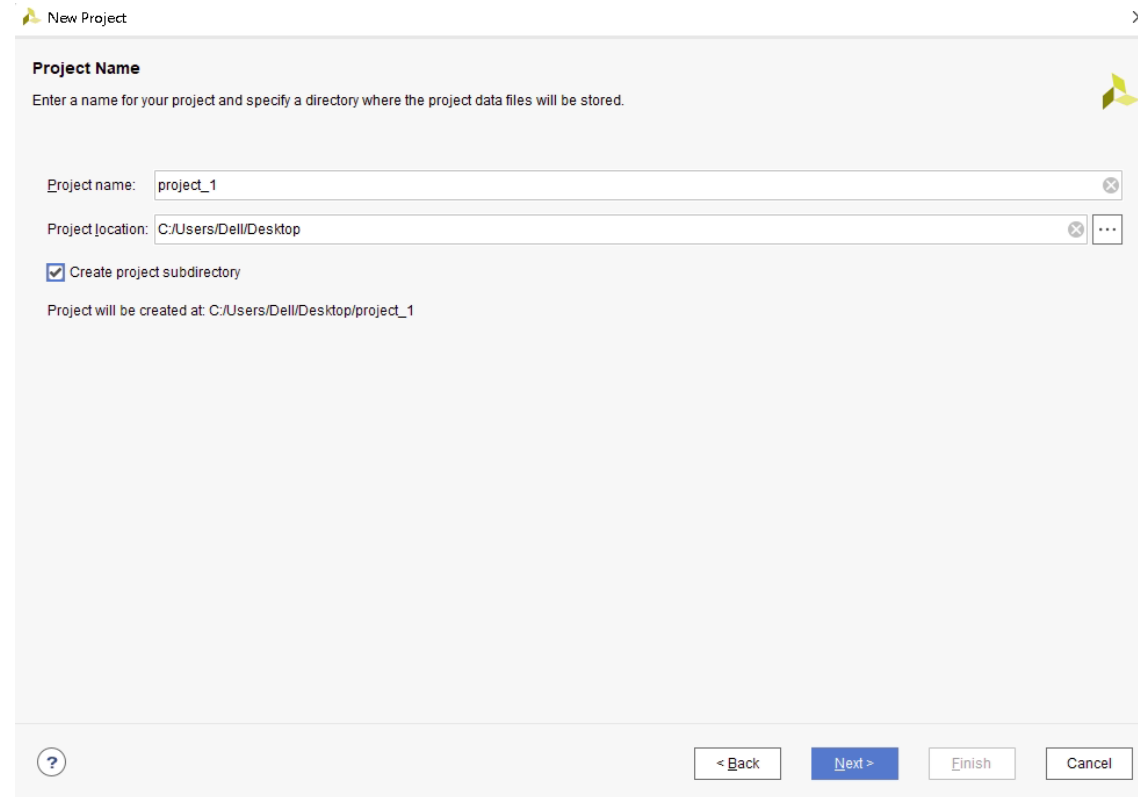
Xilinx Vivado

- 執行Vivado並創建一個新的專案



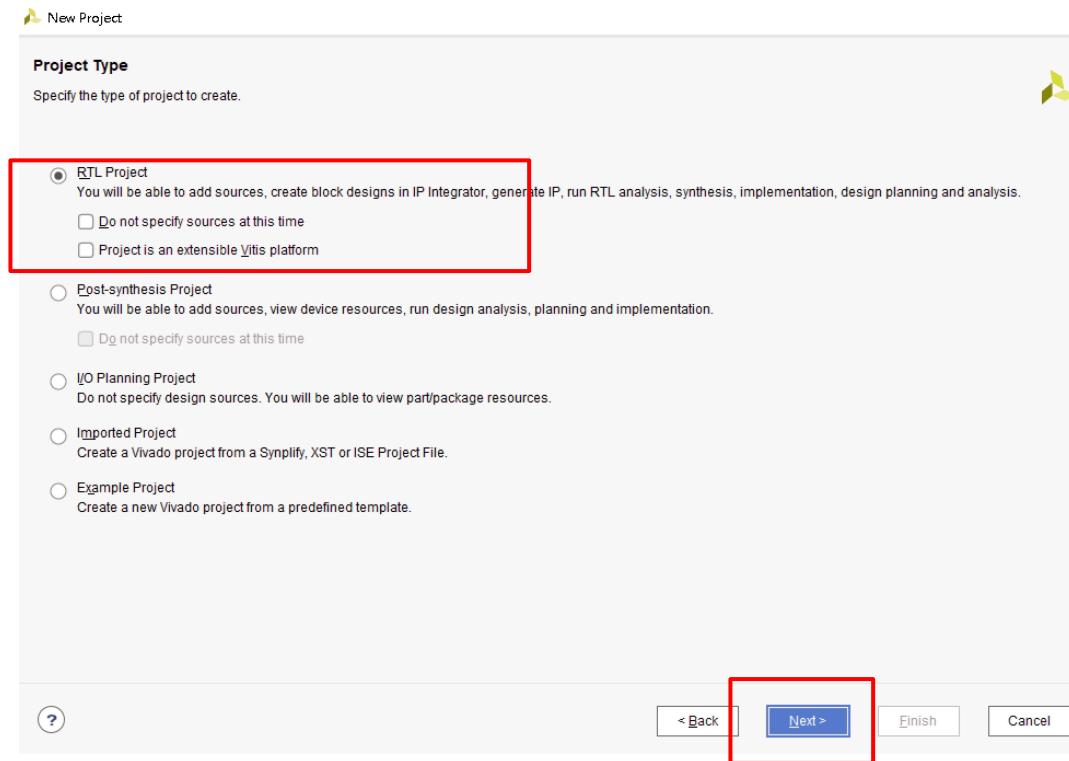
Xilinx Vivado

- 選擇路徑



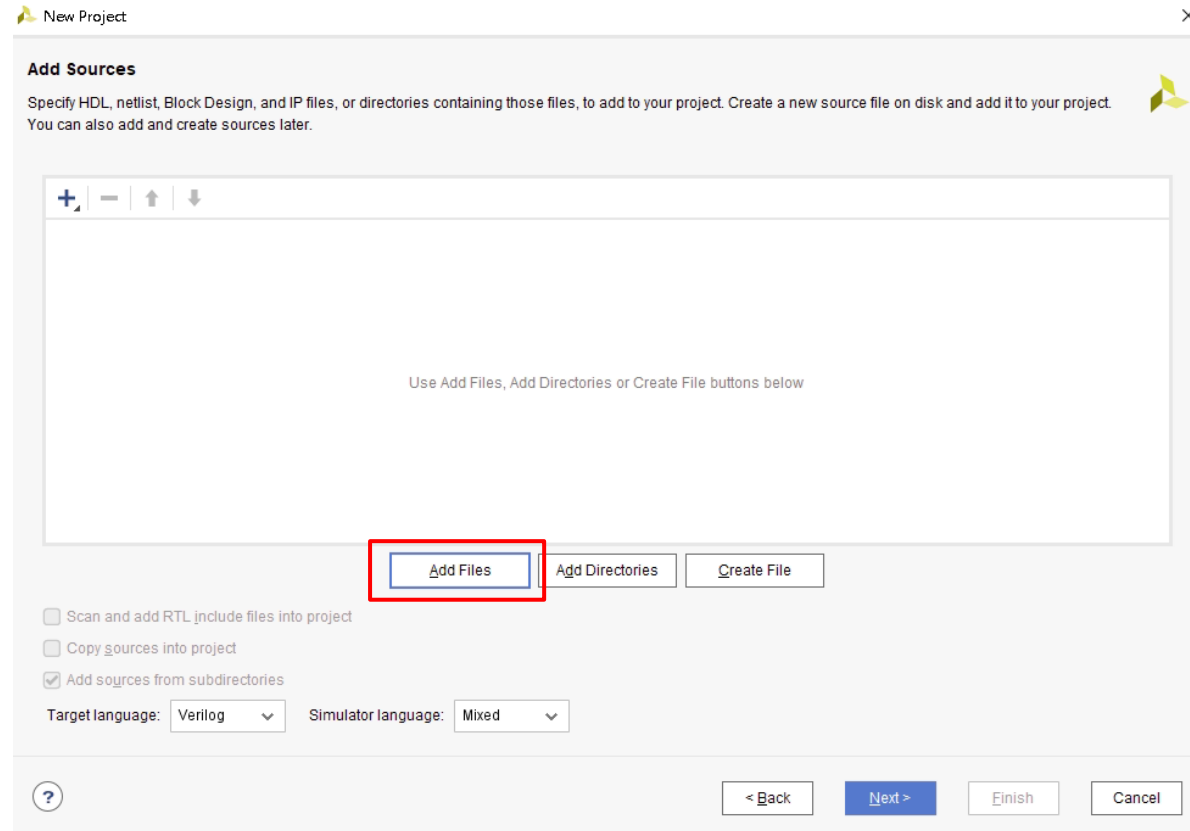
Xilinx Vivado

- 勾選RTL後直接點Next



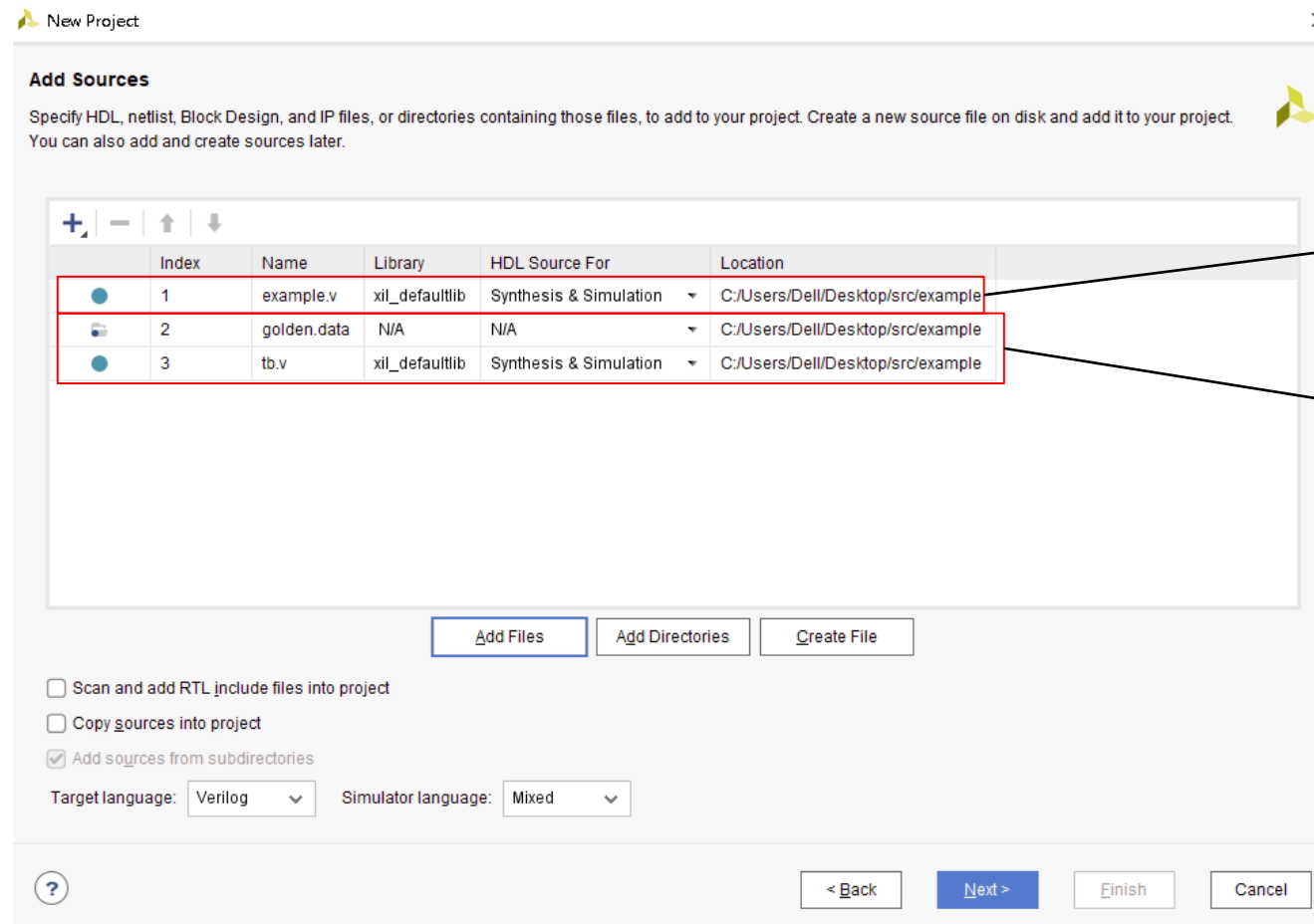
Xilinx Vivado

- 選擇Add Files



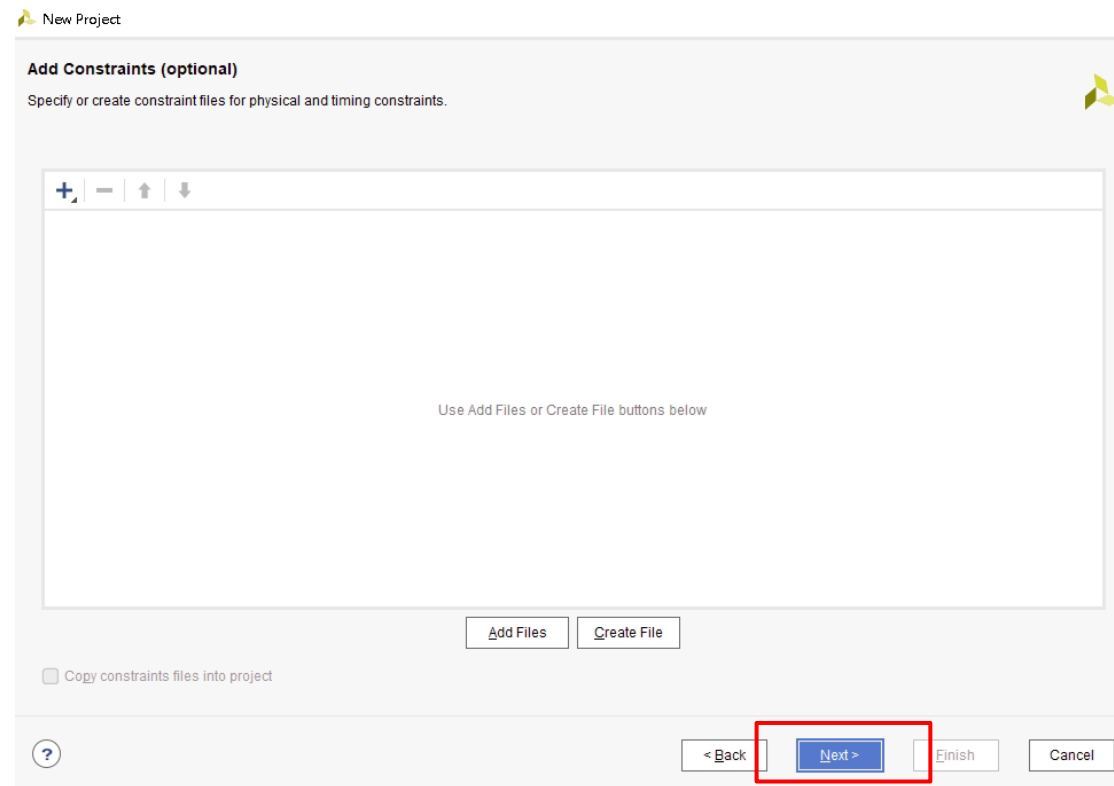
Xilinx Vivado

- TA提供的檔案加入專案



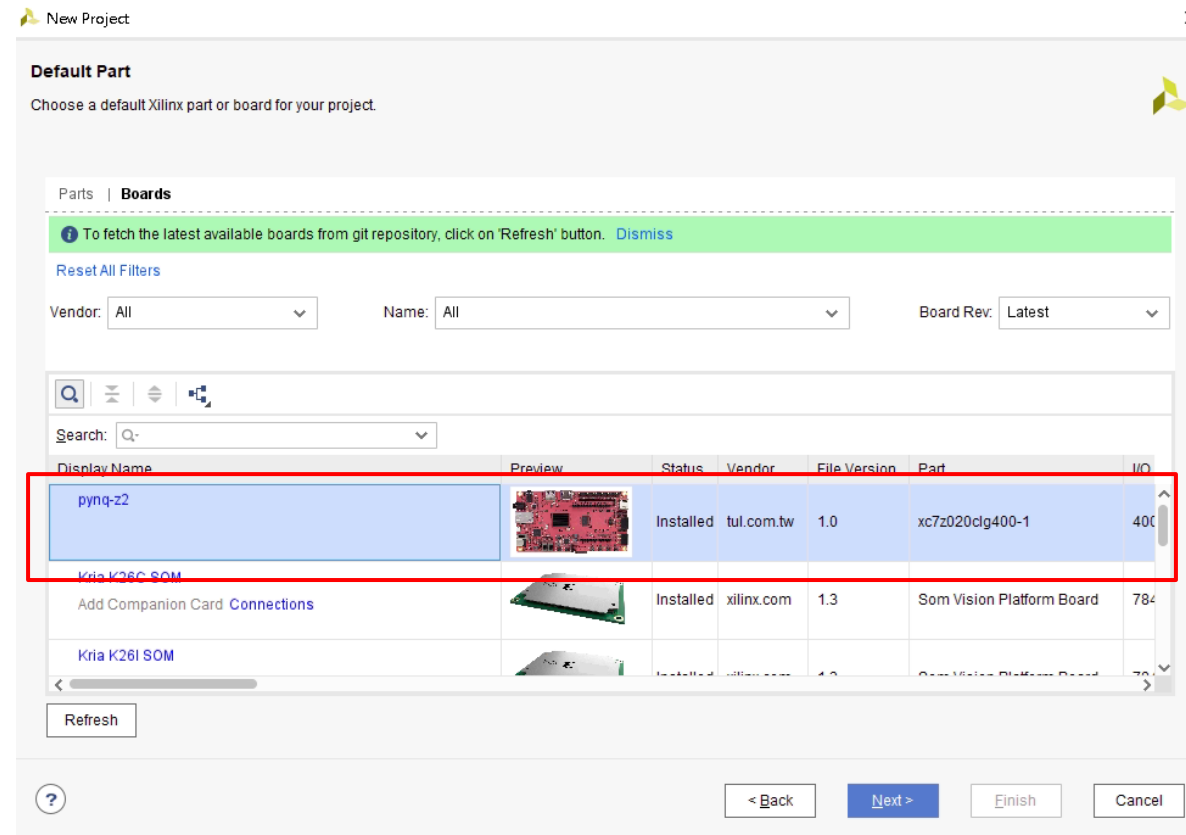
Xilinx Vivado

- 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

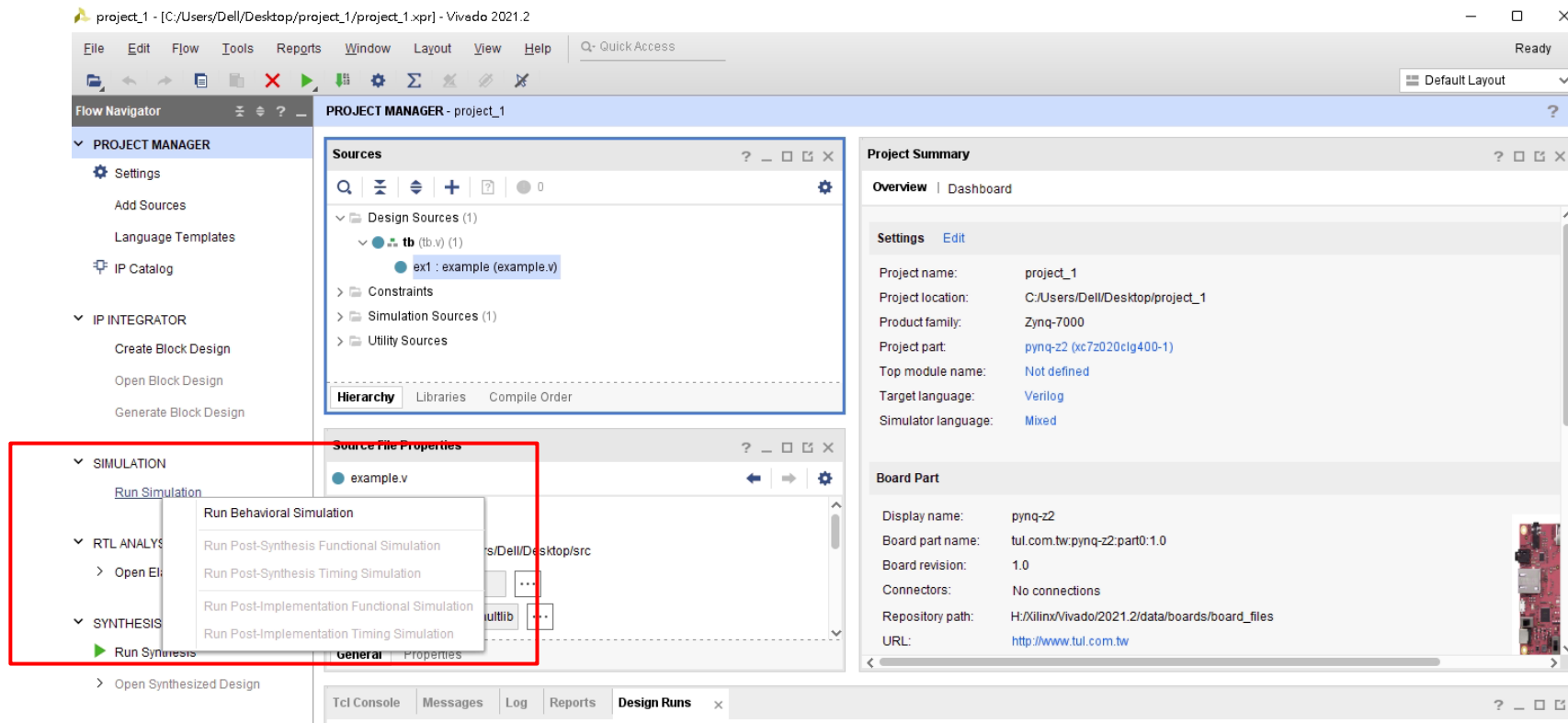
The screenshot displays the Xilinx Vivado 2021.2 software interface. The main window is titled "project_1 - [C:/Users/Dell/Desktop/project_1/project_1.xpr] - Vivado 2021.2". The interface is divided into several panels:

- Flow Navigator:** A sidebar on the left containing project management tasks such as "Settings", "Add Sources", "Language Templates", "IP Catalog", "IP INTEGRATOR", "SIMULATION", "RTL ANALYSIS", "SYNTHESIS", "IMPLEMENTATION", and "PROGRAM AND DEBUG".
- PROJECT MANAGER - project_1:** The central panel showing the project's source files. Under "Design Sources (1)", there is a sub-entry "tb (tb.v) (1)" which contains "ex1 : example (example.v)".
- Source File Properties:** A panel below the sources showing details for the selected "example.v" file, including its location, type (Verilog), and library (xil_defaultlib).
- Project Summary:** A panel on the right providing an overview of the project, including the project name, location, product family (Zynq-7000), project part (pynq-z2), and board part details.
- Design Runs:** A table at the bottom showing the status of various design runs.

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	
synth_1	constrs_1	Not started																	Vivado Synthesis Defaults
impl_1	constrs_1	Not started																	Vivado Implementation Def

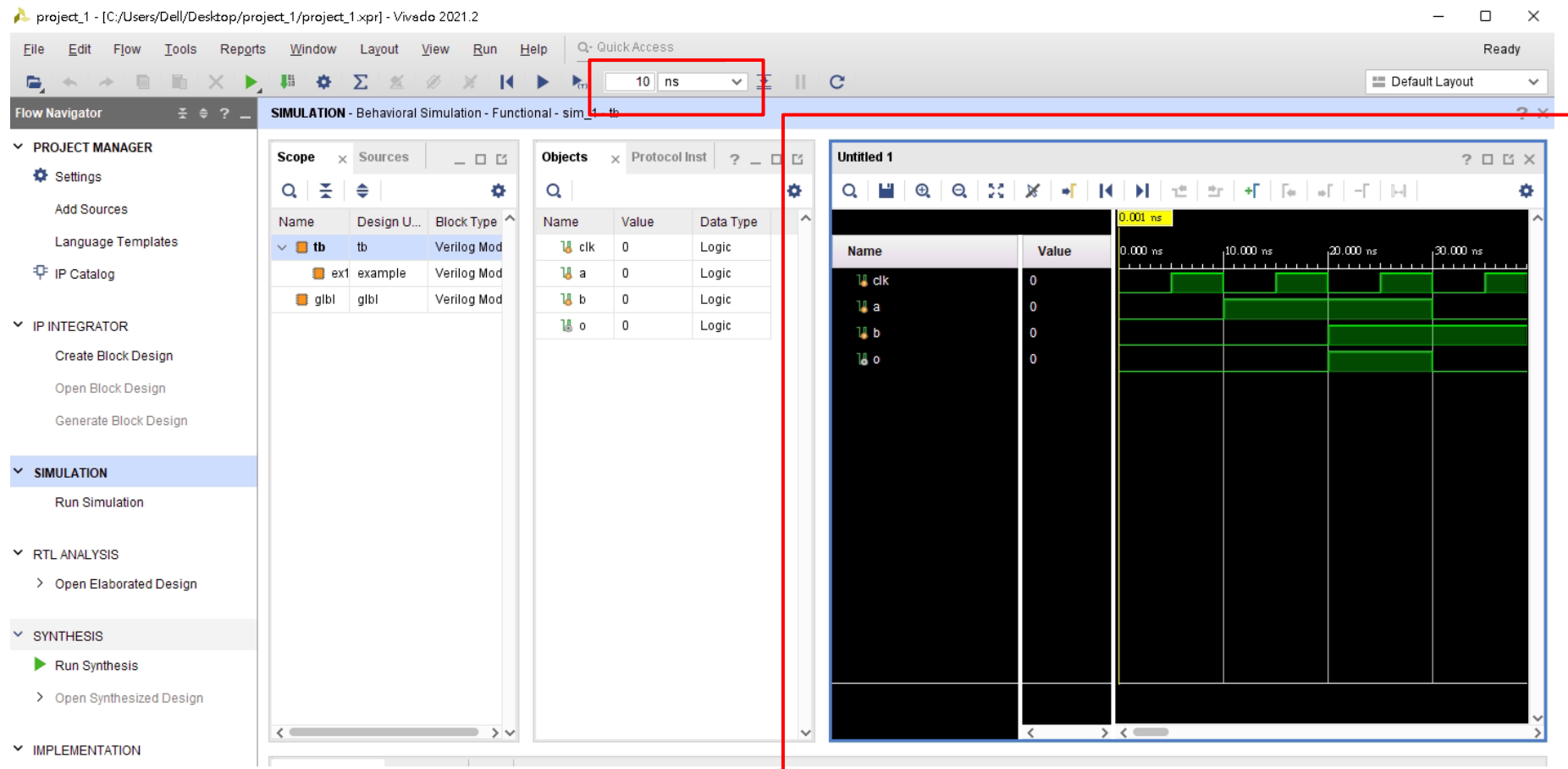
Xilinx Vivado

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



Xilinx Vivado

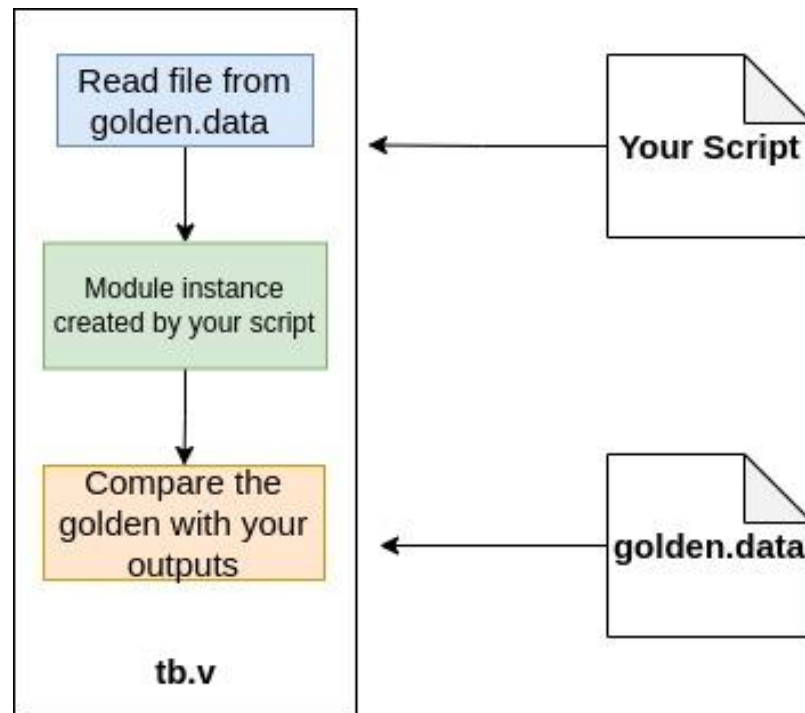
- 將間隔改為10 NS 並利用右邊的waveform來檢查自己所寫的電路



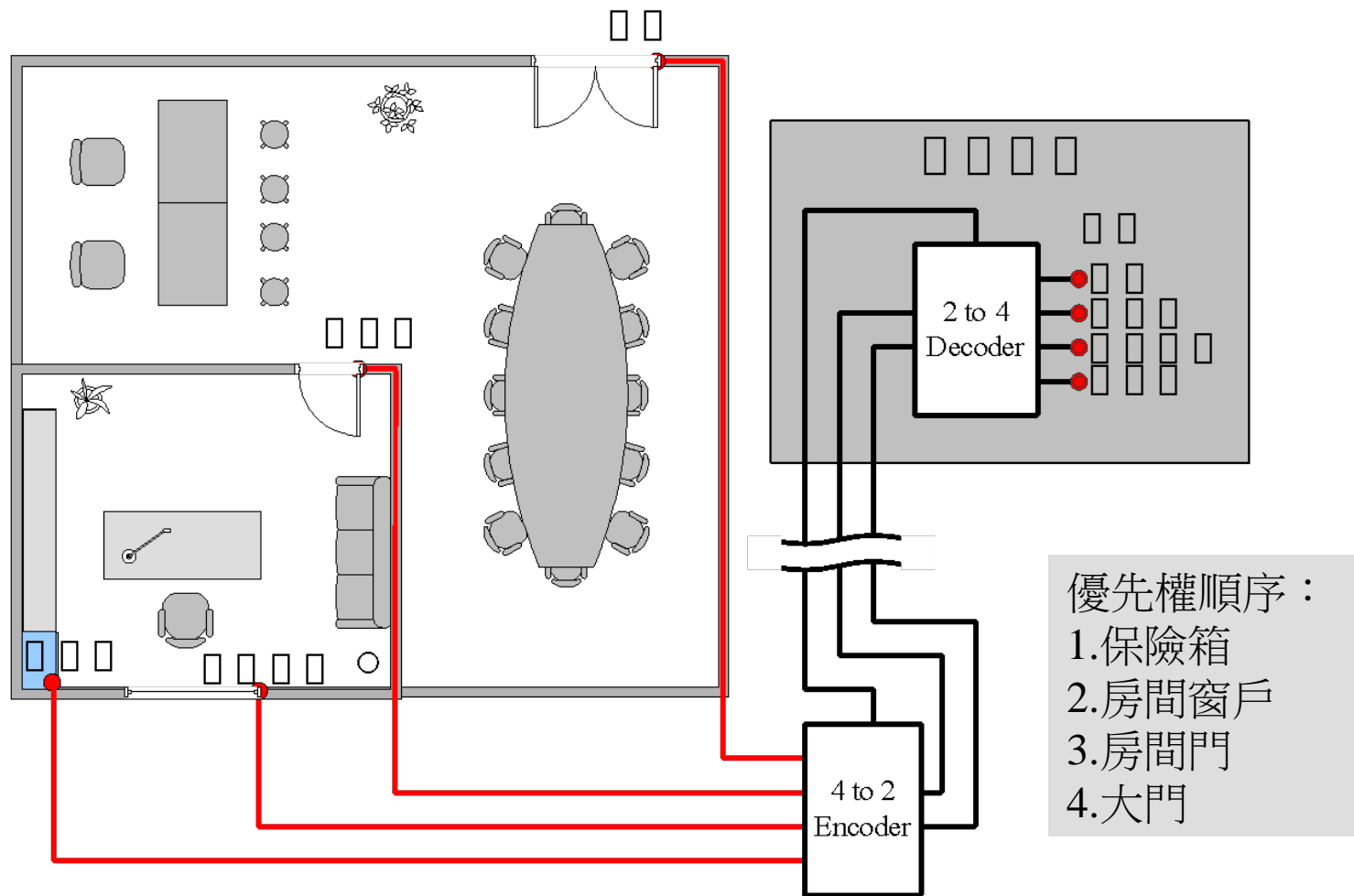
TestBench

- 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

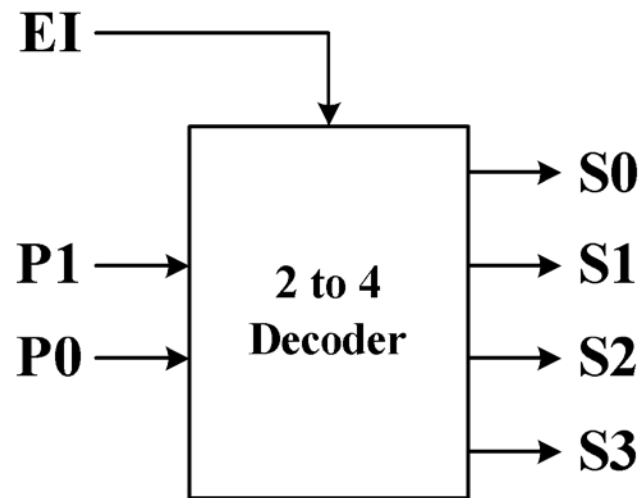
Note: module name & port list 必須要跟 testbench file 裡面的一樣



實作題 1-1：保全系統

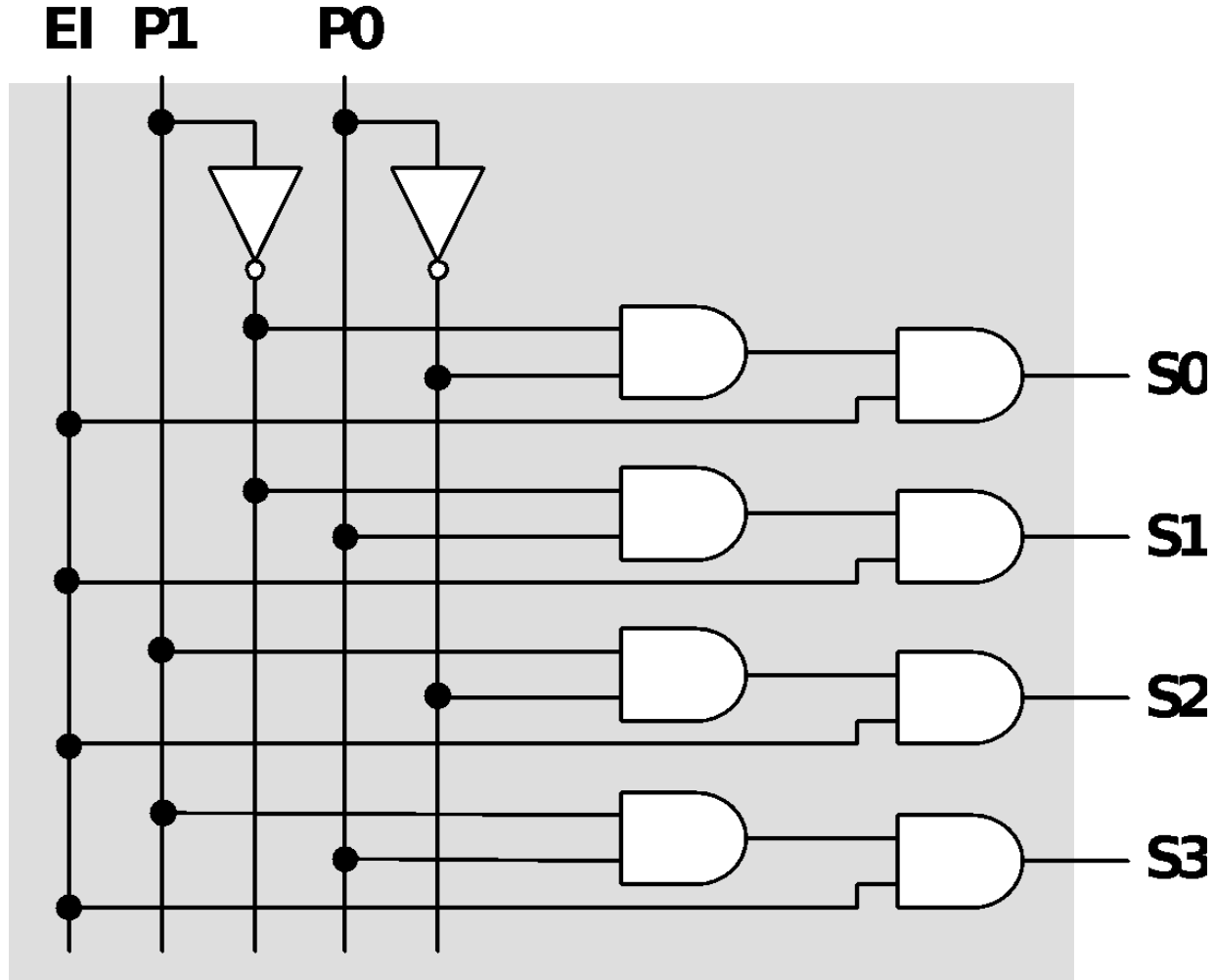


編/解碼器：解碼器(Decoder)

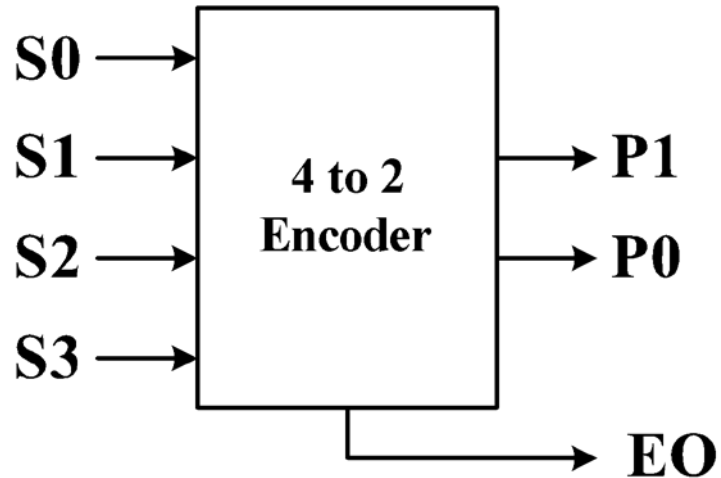


Input			Output			
EI	P1	P0	S0	S1	S2	S3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

編/解碼器：解碼器(Decoder)

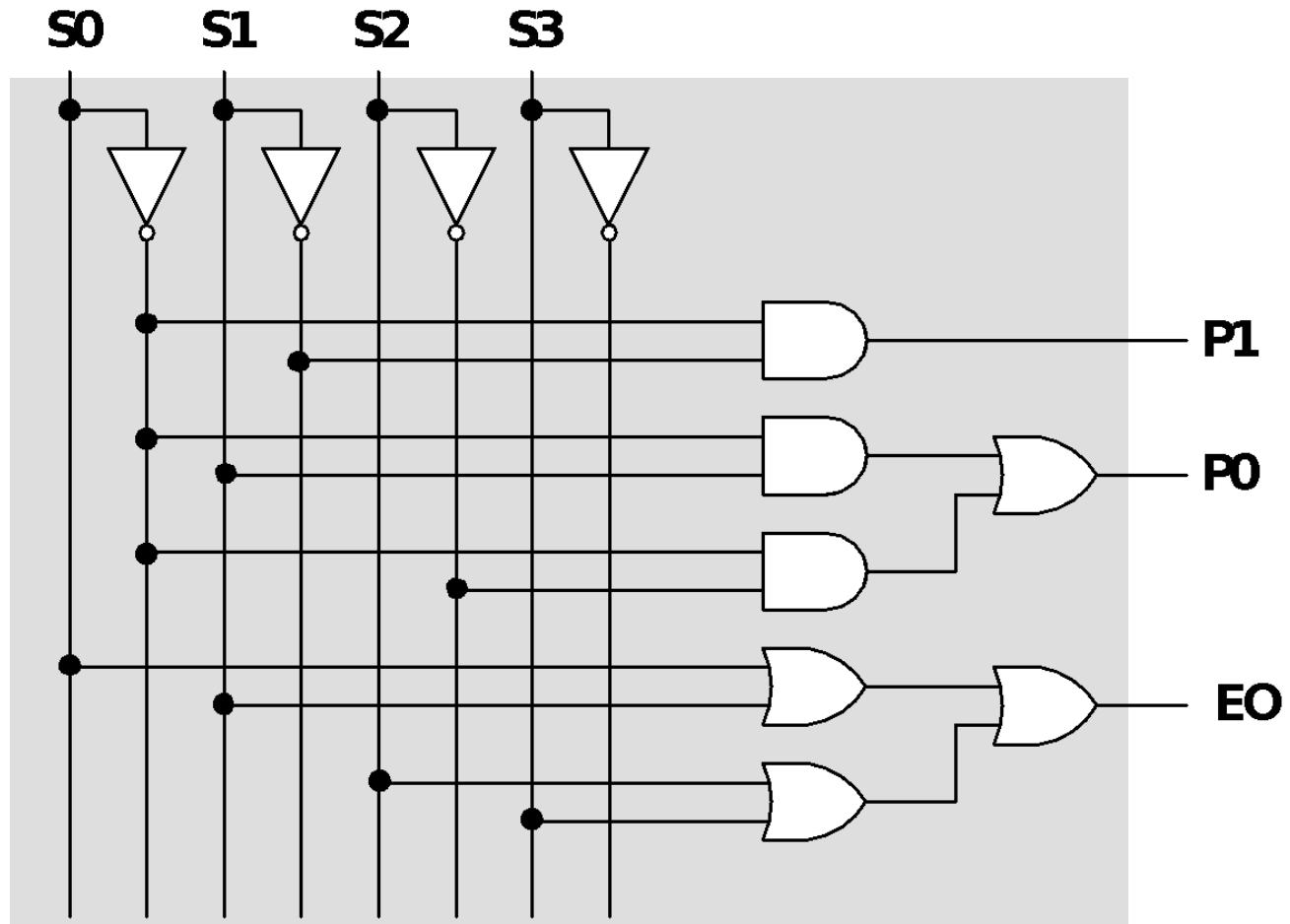


編/解碼器：編碼器(Encoder)



Input				Output		
S_0	S_1	S_2	S_3	P_1	P_0	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

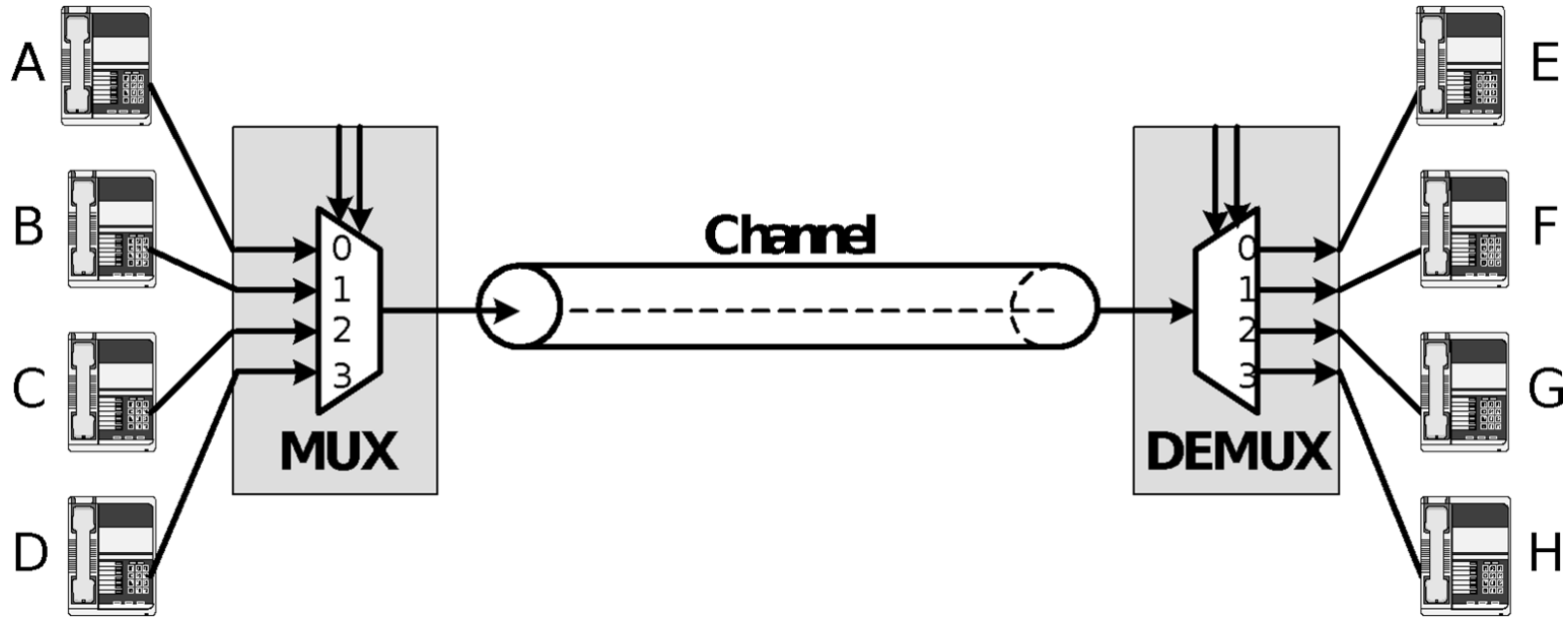
編/解碼器：編碼器(Encoder)



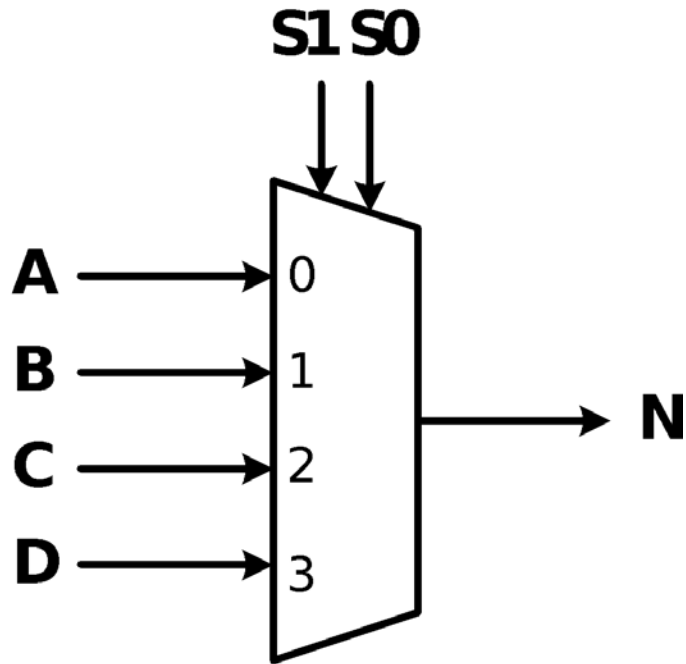
保全系統 Port list

- 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

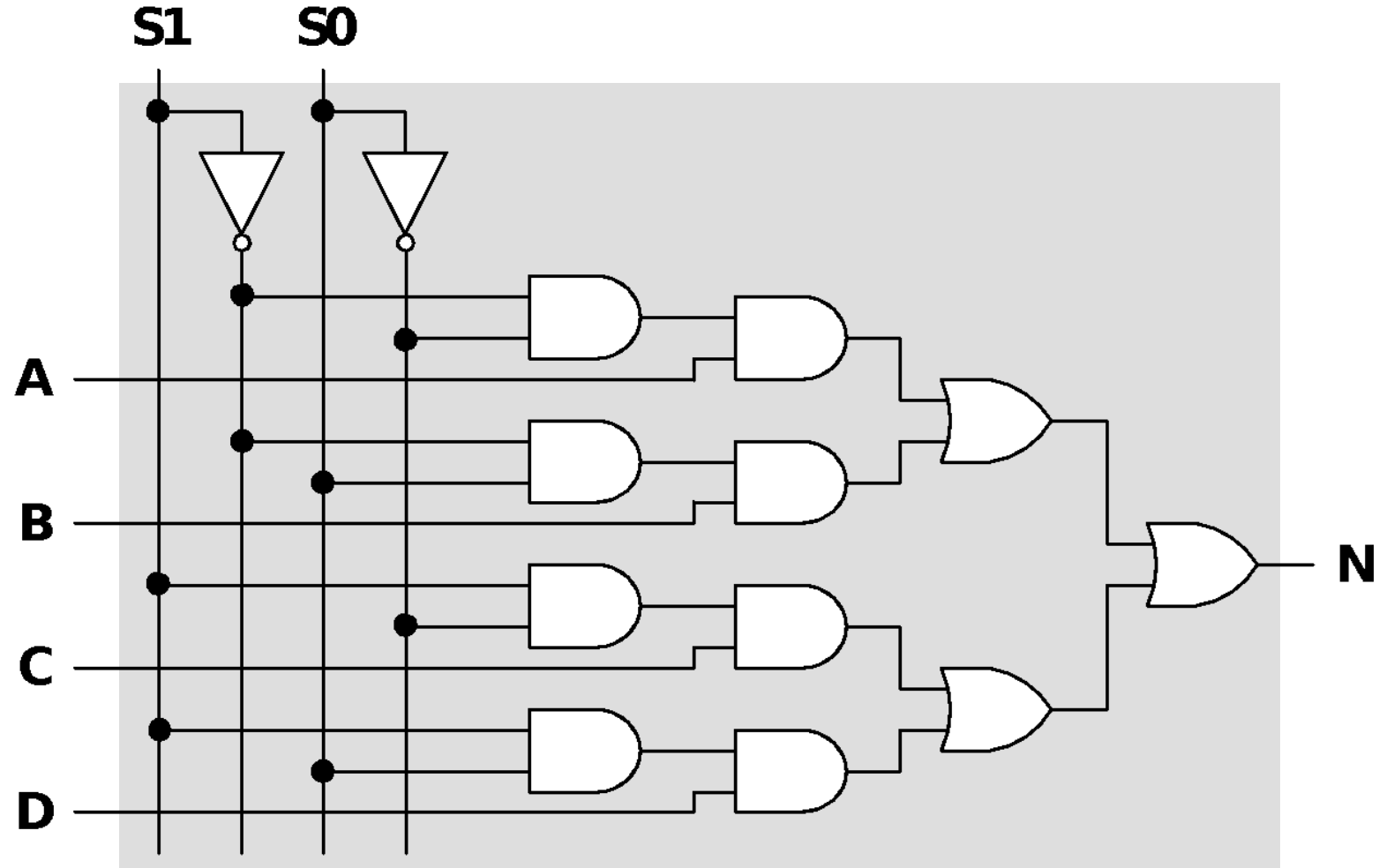


多工器(MUX)

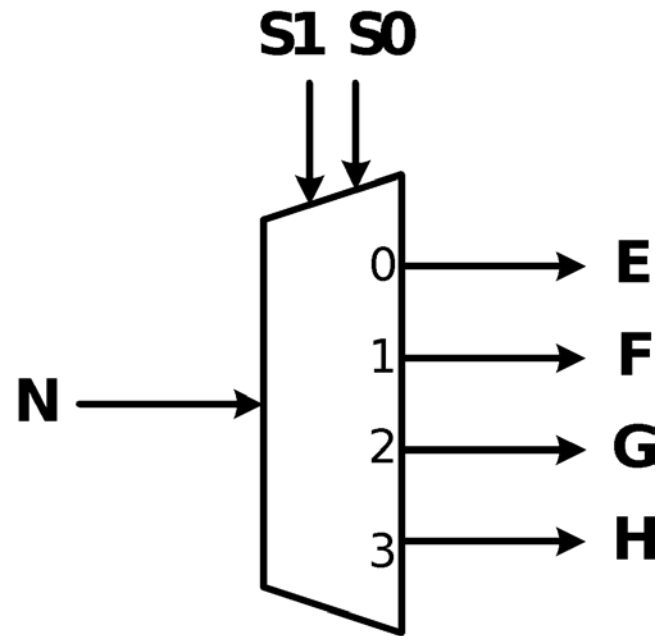


S1	S0	Output
0	0	N=A
0	1	N=B
1	0	N=C
1	1	N=D

多工器(MUX)

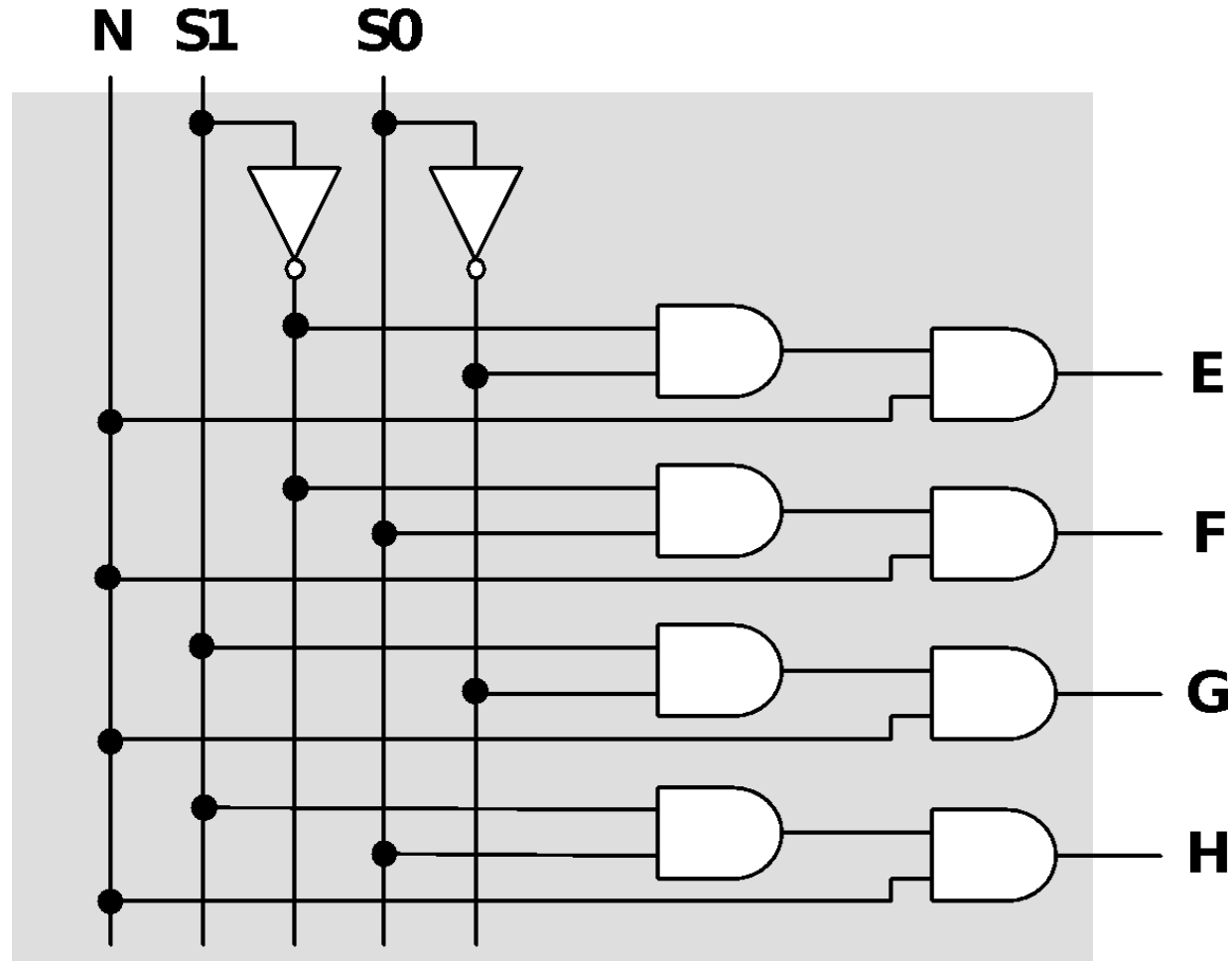


解多工器 (DEMUX)



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)

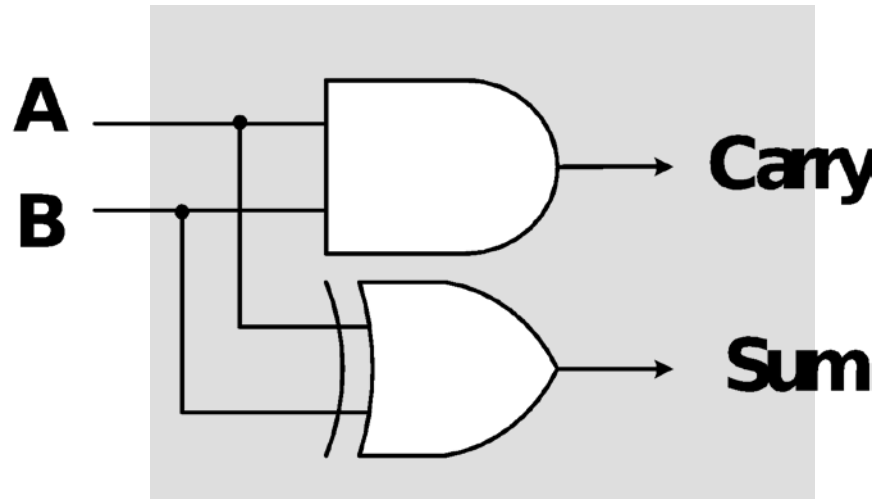


接線生 Port list

- 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

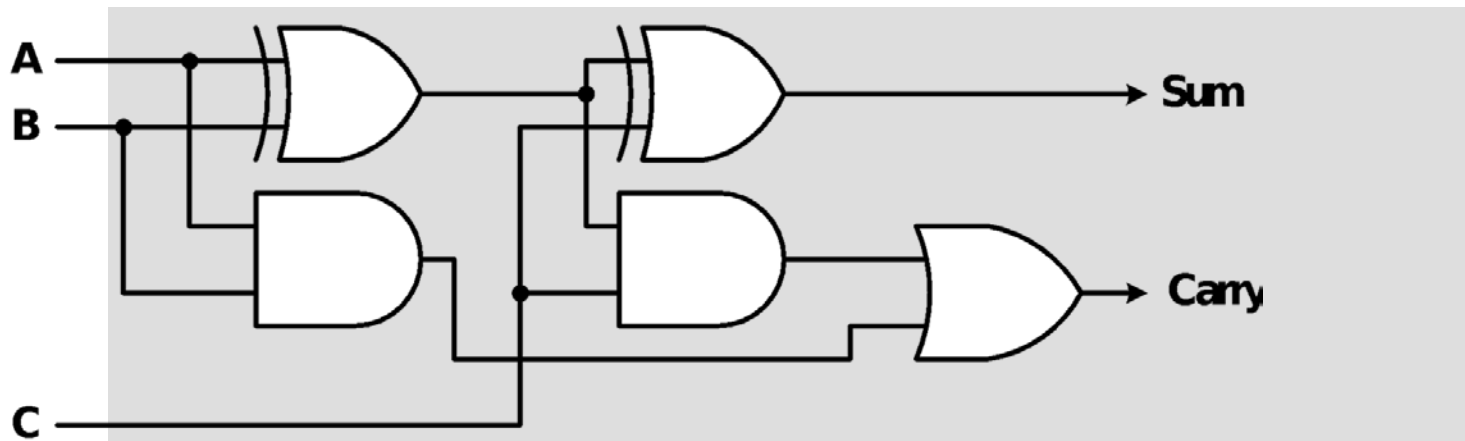


半加器 Port list

- 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!! **  
*****
```



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