

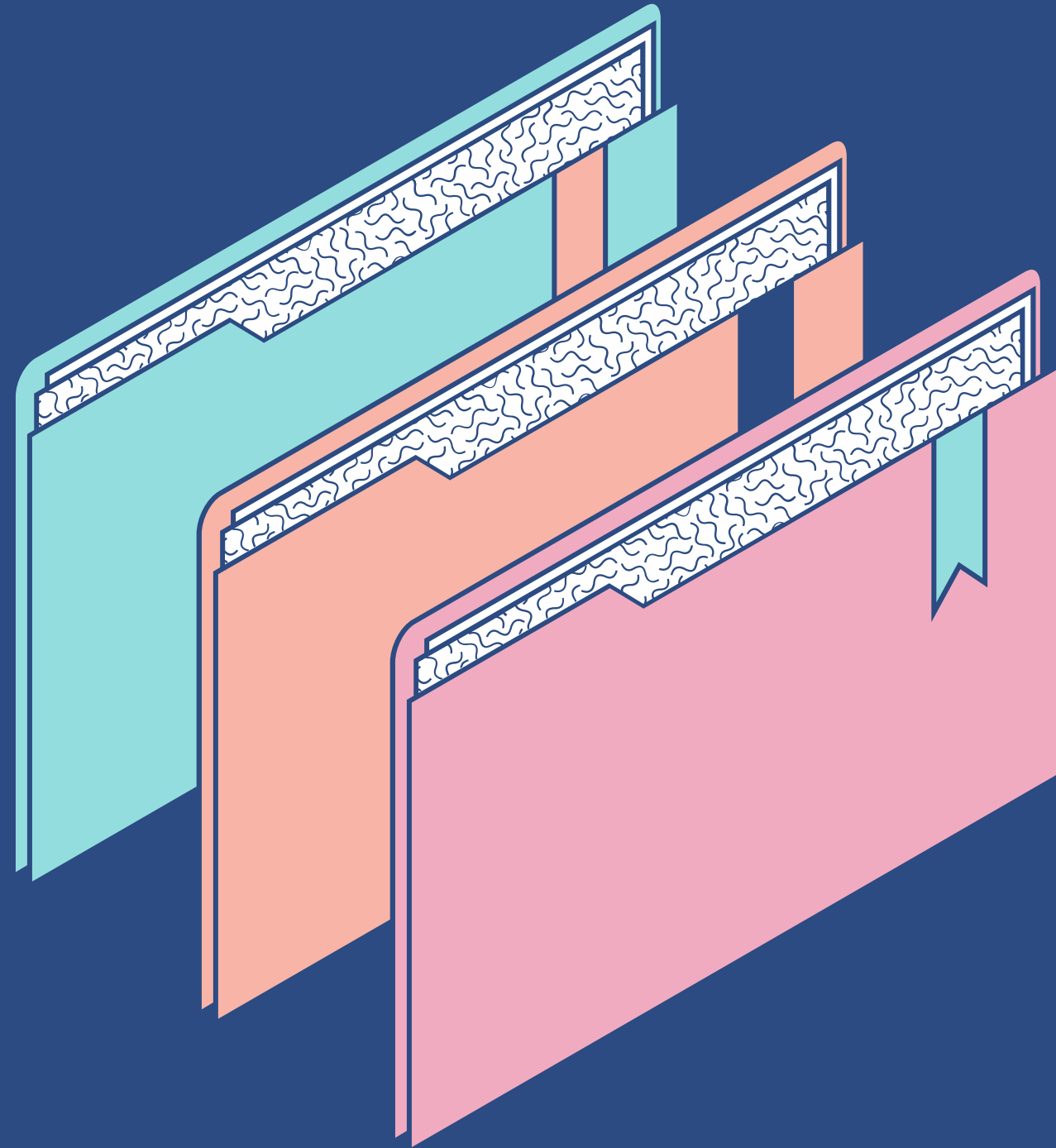


晶片程式設計實作營

多元學習成果

新竹女中 周語泠

檔案目錄



| | |
|------------------|-----|
| 參與動機與目標..... | p3 |
| 活動內容與課程學習摘要..... | p4 |
| • 實作經驗..... | p5 |
| • 挑戰突破..... | p9 |
| 自我成長與反思..... | p12 |
| 延伸統整..... | p13 |
| 未來展望與跨領域應用..... | p14 |



參與動機與目標

高一暑假時我參與了由 EDC 專長探索中心舉辦的「晶片設計實作營」。過去的我對晶片與硬體程式設計**懷有強烈好奇心**，卻從未接觸過任何程式語言。正因如此，我更希望能踏出第一步，了解程式設計與數位邏輯的世界。這次營隊便成為我**自我挑戰的起點**，期望自己能：

- 習得一種能實際應用的**程式語言**
- 初步**理解晶片**背後的運作邏輯
- 認識**數學邏輯**和**程式**的關聯

活動內容與課程學習摘要

將抽象邏輯轉化為具體電路

營隊中我學習了**數位邏輯設計流程**，從生活情境需求出發，**列出真值表、化簡布林式、繪製卡諾圖**，最後以**邏輯閘實作**實際電路。以警報系統為例，我透過判斷門戶與窗戶開關時間的條件，推導出 $Y = AB'CD + ABC'D' + ABC'D + ABCD$ 的邏輯式，並以圖示方式理解其運作。這樣的學習讓我首次體會到課堂中所學邏輯數學如何應用於真實世界，也培養了我**系統化拆解問題**的能力。

同時，我也首次接觸了**硬體描述語言 Verilog**，學習了**assign陳述句**與**資料流模型**的使用方式，並透過 HDLBits 線上平台練習。從基礎語法到模擬邏輯閘，我一步步建構起對程式語言的理解。



實作邏輯設計流程

1

STEP

根據實際應用情形
訂定所需規格

2

STEP

將所需規格
列成真值表

3

STEP

利用最小項展開，
將真值表轉換成
布林表達式

4

STEP

將布林表達式以
卡諾圖化為最簡型式

5

STEP

根據化簡後的
布林表達式用邏輯閘
組合成實際邏輯電路

完成！

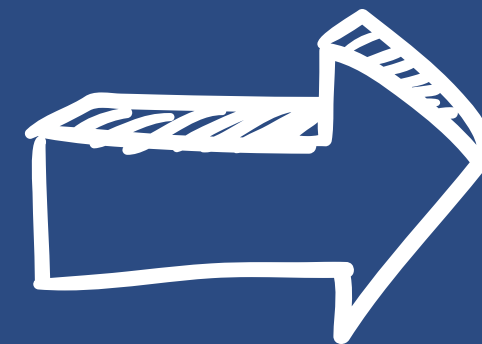
數位邏輯設計實作

以警報器為例

要讓警報器響，警報器開關必須在開啟的狀態。
警報器開啟後，如果門沒關，或是晚上六點之後
窗戶沒關，警報器就會響。

理解為 \Rightarrow 只有在 警報器開關開啟 且 門沒關 或 (晚上六點 後 且 窗戶沒關)，警報器才會響。

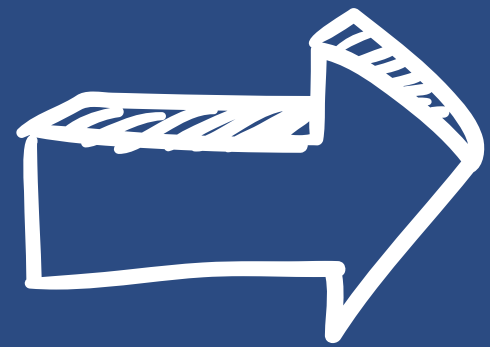
A B
C D Y



列出真值表

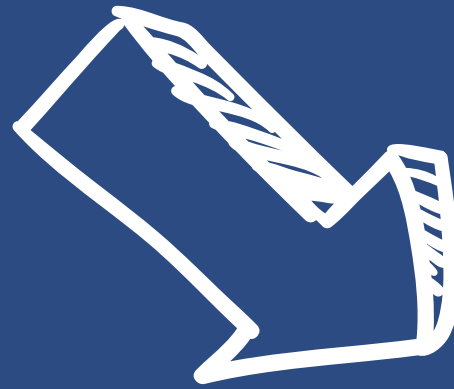
| A | B | C | D | Y |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

數位邏輯設計實作



最小項展開

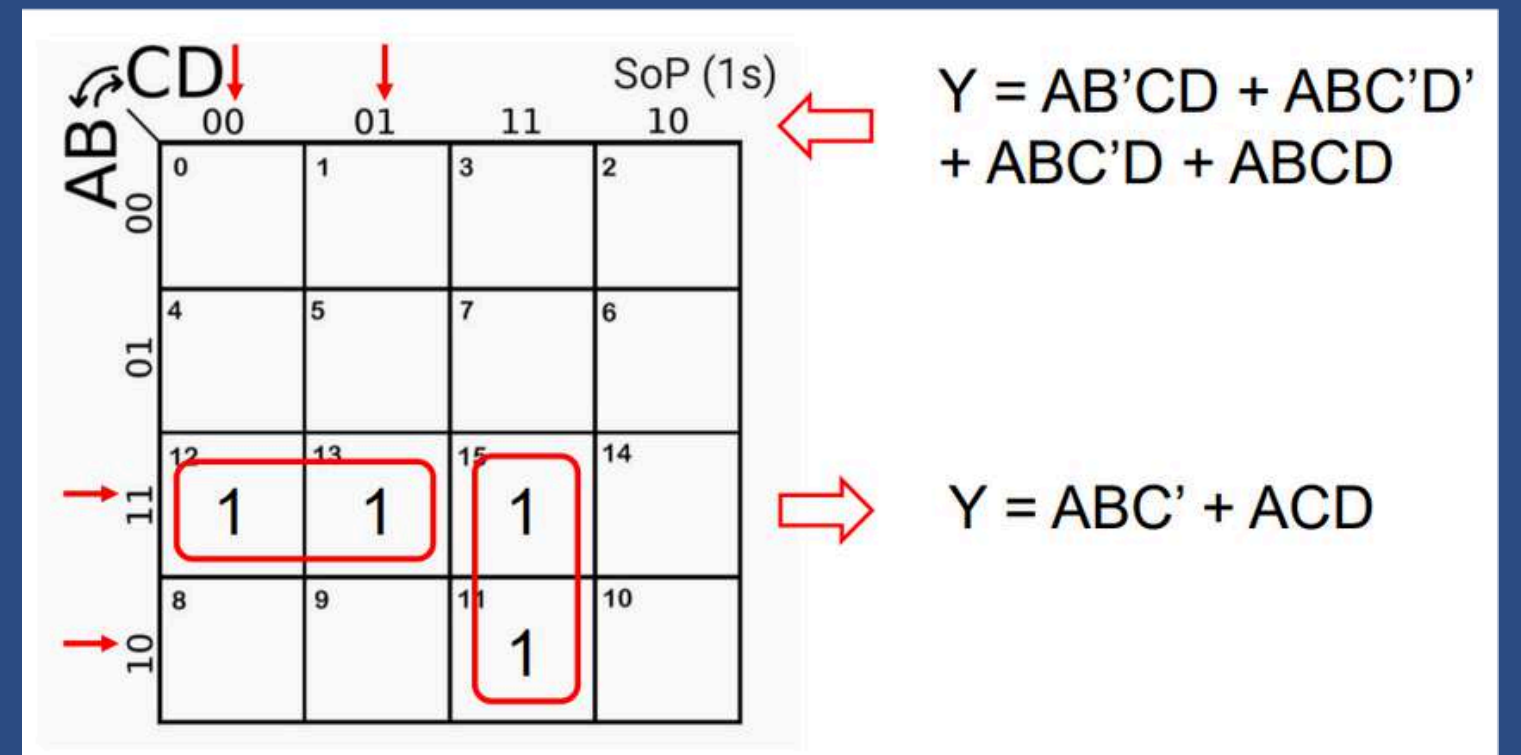
| A | B | C | D | Y |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |



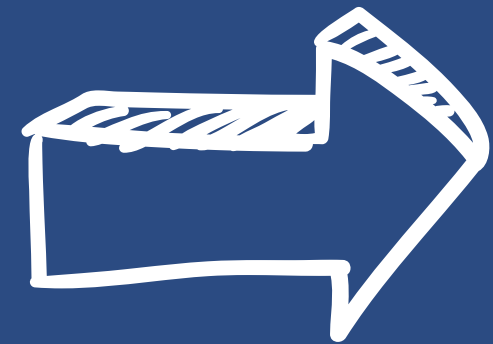
卡諾圖畫簡

轉換成布林表達式

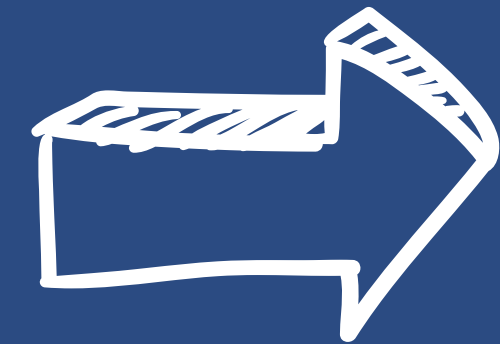
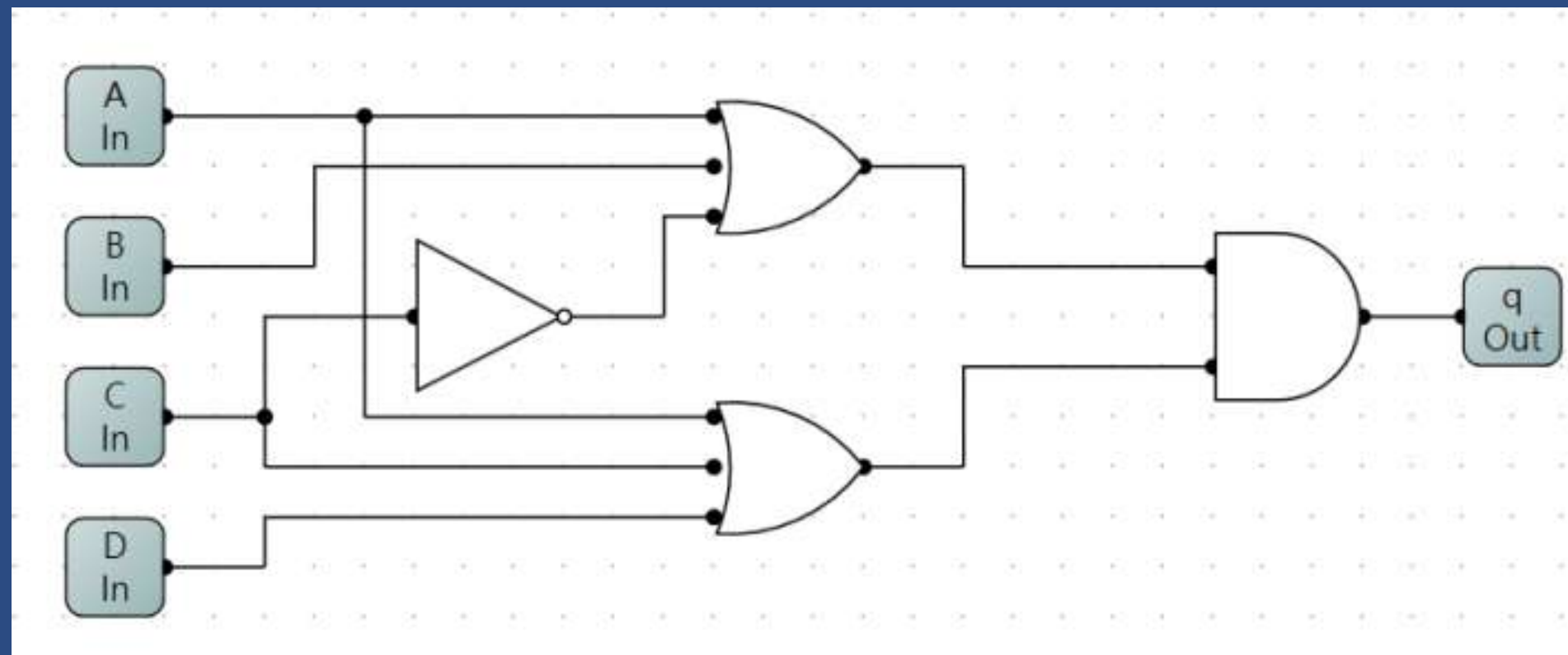
$$Y = AB \cdot CD + ABC \cdot D' + ABC' \cdot D + ABCD$$



數位邏輯設計實作



實際邏輯電路



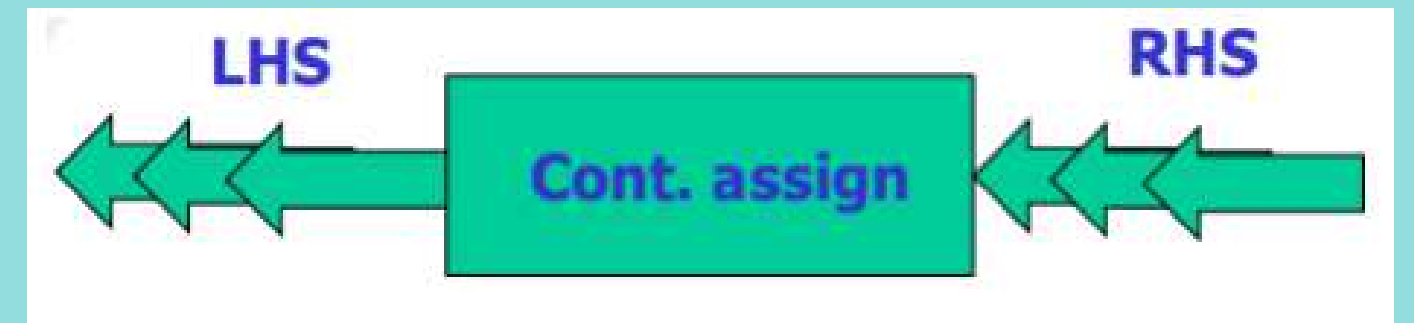
完成！

利用日常生活中出現的例子來說明程式設計的最雛形。
來自台大機械系的教授將所有流程做成一套SOP讓初學者們更好上手，
也用**清晰的數學邏輯思維**一步步推導結果。將最後結果用邏輯閘展現，
讓我對製作程式的過程**豁然開朗**，愈發**對程式設計的興趣**。

硬體描述語言Verilog

資料流模型的敘述

- 設計重點：說明資料如何在電路中的傳送過程
- assign
 - 等號左式只能是wire，右式可以是 wire 或 reg
 - 用於描述組合邏輯電路
 - 必須避免使用迴路式的寫法：assign a = b + a;
- 電路範例
 - $X = A \cdot C \cdot D' + B \cdot C' + B \cdot D + C \cdot D$
 - $Y = A' + B + C$



```
1 module boolean(a, b, c, d, x, y);  
2     input a, b, c, d;    //wire  
3     output x, y;        //wire  
4  
5     assign x = (a & c & (!d)) | (b & (!c)) | (b & d) | (c & d);  
6     assign y = (!a) | b | c;  
7  
8 endmodule
```

硬體描述語言Verilog

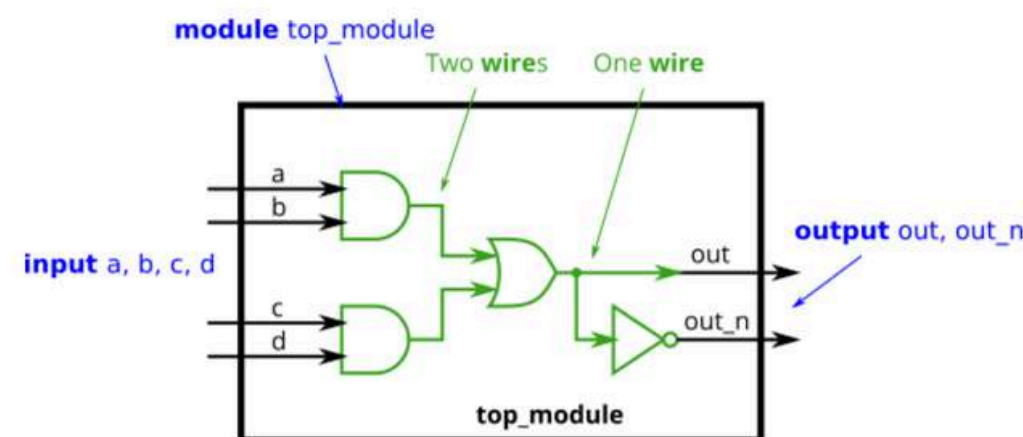
Verilog 練習

Practice

Implement the following circuit. Create two intermediate wires (named anything you want) to connect the AND and OR gates together. Note that the wire that feeds the NOT gate is really wire out, so you do not necessarily need to declare a third wire here. Notice how wires are driven by exactly one source (output of a gate), but can feed multiple inputs.

If you're following the circuit structure in the diagram, you should end up with four assign statements, as there are four signals that need a value assigned.

(Yes, it is possible to create a circuit with the same functionality without the intermediate wires.)



https://hdlbits.01xz.net/wiki/Main_Page

我們利用此網址進行多個Verilog 的基礎練習。
主要的練習模式是網頁顯示邏輯閘，
再讓我們自己利用那些邏輯閘寫出與它相符合的
Verilog 運算式。

附圖為其中一練習題的螢幕截圖

硬體描述語言Verilog

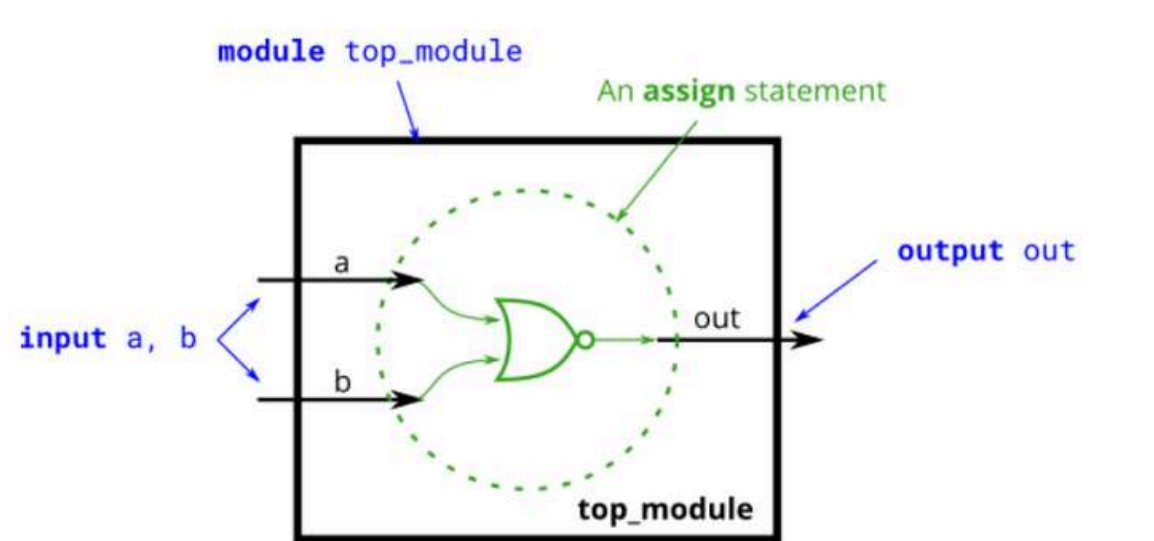
我卡關最久的練習題

Norgate ✓

← andgate ✓ Previous Next xnorgate ✓ →

Create a module that implements a NOR gate. A NOR gate is an OR gate with its output inverted. A NOR function needs two operators when written in Verilog.

An assign statement drives a wire (or "net", as it's more formally called) with a value. This value can be as complex a function as you want, as long as it's a *combinational* (i.e., memory-less, with no hidden state) function. An assign statement is a *continuous assignment* because the output is "recomputed" whenever any of its inputs change, forever, much like a simple logic gate.



```
module top_module
    input a, b
    output out
    assign out = ~(a | b);
endmodule
```

在 Verilog 練習中，我遭遇了初學者常見的錯誤。我一開始寫 `assign out = a ! | b;`，導致語法錯誤。經過**反覆測試、比對邏輯運算順序**，我終於理解必須將整個 `or` 運算括號包起，改為 `assign out = !(a | b);`，才能實現「整個 "a或b" 的否定」。雖然只是小細節，卻讓我深刻體會到**細心與邏輯**在寫程式時的重要性，也在錯誤中學會**冷靜面對挫折、逐步拆解問題的能力**與**細節敏感度**，更激發了我對程式設計的**成就感與興趣**。

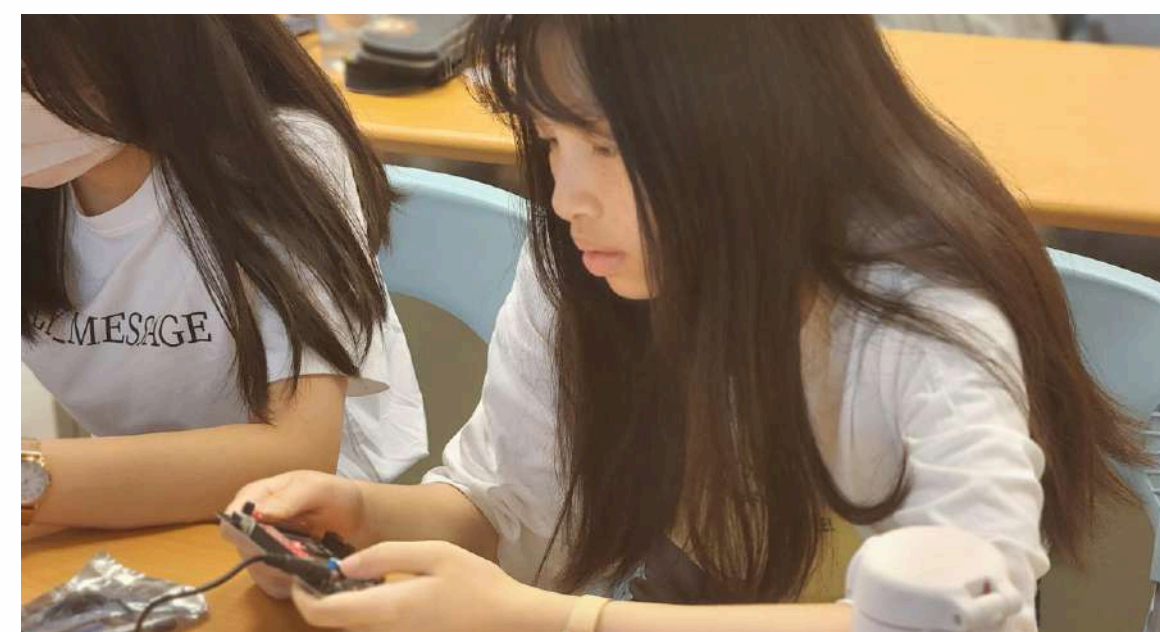
自我成長與反思

這次營隊不僅是我第一次寫程式，更是我第一次將**數學邏輯思維**轉化為**具體實作**。從一開始的焦慮與迷惘，到後來成功寫出正確的邏輯式與 Verilog 程式，我對自己的**理解力**與**實作能力**有了全新的肯定。我發現自己具備良好的**邏輯推理能力**與**細節敏感度**，能**在錯誤中調整方向**，並**有毅力**從零開始學會一項技術。這樣的特質不僅適用於工程技術領域，更是面對**跨領域**問題時的重要資產。

此外，我也展現了**主動學習**與**自我驅動**的能力。即使面對技術限制，我仍能**換位思考、積極觀摩、勇於提問**。這種彈性與適應力，是未來無論在管理、科技還是產品開發等領域都珍貴的素養。這場營隊經驗，讓我更**認識自己的學習風格與潛力**，並建立了對**科技應用與系統設計**的高度**興趣與責任感**。

延伸統整

這份基礎**啟發了我**後續的多元學習：我進一步完成了C++ 的進階學習與製作學習成果，並在微電腦機電整合課程中組裝並程控一台具備拾球、投球功能的機器人；我也參與了AI 流行音樂製作營，開始探索人工智慧的創意應用。為了深化對此科技應用的理解，我也自主學習半導體光感測器課程，並在陽明交大的邀請下，將我的自學經驗在高中生自學講座中分享。此後更參與陽明交大的人工智慧與深度學習人才培育課程，在同年暑假也進一步參加台大資工營，深入接觸**演算法與資訊結構**的運作邏輯。



未來展望與跨領域應用

這場營隊是我對邏輯、電路與程式世界的**第一步**，也開啟我整合「**數理基礎、邏輯思考、程式設計、跨域應用**」的學習之路。我期許自己未來不論在**工程、資訊、管理或產業應用**等不同領域中，都能持續以**實作能力**與**冷靜解決問題**的態度貢獻自己的力量。面對未知領域，我始終**保持好奇、勇於嘗試**，也願意投入時間**反覆修正、不怕失敗**。這正是我在這場營隊中得到最重要的收穫。



附圖為參與證明