Final Project
Stack計算機+

Department of Electrical Engineering
National Cheng Kung University

Outline

- Project規劃
- 計算機規格要求
- Project實作建議
- 期末報告注意事項
- 補充說明

Project規劃

- Final Project總成績佔比：30%
- Final Project評分方式
  - Demo：30%
  - Presentation：70%
- 前3週的時間進行Project實作與Demo
  - 5/22
  - 5/29
  - 6/5
- 第4週會讓所有組別上台報告自己的成果
  - 6/12

Stack計算機

- Stack計算機可以有效率的處理Postfix運算式的計算（參考Lab9的說明）
- 疊機器（英語：Stack machine）其實是電腦科學中一種計算模型，這種類型的電腦，記憶體以堆疊（Stack）儲存。
- Java Virtual Machine就是一種Stack machine，經過編譯的Java程式（*.class）就是一連串對於Stack操作的指令
- 本次Final Project將基於Lab9的Stack計算機再新增更多Operation（指令）來實作功能更完整的Stack計算機

計算機規格要求

- 需要實作的運算指令有：

名稱	代號	說明
literal	x	push(x)
add	+	b = pop(); a = pop(); push(a + b)
subtract	-	b = pop(); a = pop(); push(a - b)
right-shift	>>	b = pop(); a = pop(); push(a >> b)
left-shift	<<	b = pop(); a = pop(); push(a << b)
bitwise-and	&	b = pop(); a = pop(); push(a & b)
bitwise-or	|	b = pop(); a = pop(); push(a | b)
exclusive-or	^	b = pop(); a = pop(); push(a ^ b)

計算機規格要求

- 需要實作的特殊指令有：

名稱	代號	說明
duplicate	dup	a = pop(); push(a); push(a);
swap	swap	b = pop(); a = pop(); push(b); push(a)
equal-zero	eqz	a = pop(); **if** a == 0 **then** push(1) **else** push(0)

計算機規格要求

- 所有運算元都是4位元的無號數
- 不必考慮overflow或是underflow，例如：
  - $1100_2 + 0101_2 = 0001_2$
  - $1010_2 >> 1000_2 = 0000_2$
- 輸出必須有一個七段顯示器顯示TOS（Top of stack）
- 輸入的方法大家可以自己設計（報告及Demo時說明），例如：
  - 先使用指撥開關設定要輸入的值，按下`btn[0]`表示push運算元、按下`btn[1]`則代表執行指令
  - 使用Lab9的不同mode以及累加的方法選擇對應的運算元或指令
- 可再加入其他功能（報告及Demo時說明，加分）
  - Stack空時亮紅燈
  - 使用LED來表示目前所在的模式
  - ?

計算機規格要求：Demo

- Demo需成功運算出以下3則運算式的結果（各10%）

1. 基本運算

```
((1 + 2 - 3) | (4 & 5)) << (6 ^ 7) = 8
```

Postfix: `1 2 + 3 - 4 5 & | 6 7 ^ <<`

2. 比較兩個數是否只有一個位元不一樣（QM的compare）

```
int c = a ^ b; return (c & (c - 1)) == 0;
```

Postfix: `a b ^ dup 1 - & eqz  // a b 可以為任意數`

3. 計算一個數的popcount（二進位中1的個數）

Postfix: `a (dup 1 & swap 1 >>)`$_{3x}$ `+ + +`

Project實作建議

- 可以先設計每個module之間的階層關係，以及每個module的功能，最後決定每個module要開哪些I/O port
- 每個module的實作可以分工進行
- 設計好Top module的I/O後再修改xdc檔
- 每個modul設計完後可以先使用testbench進行Behavioral Simulation以驗證功能的正確性（如Stack、ALU等）
  - 請參照testbench的模板
  - 自己設計輸入的訊號並透過波形檢查結果
- 狀態機或是其他序向電路請參考附錄的說明撰寫
  - 比較不容易寫錯（不能合成之類的問題）且助教比較好幫你debug
- 當每個module都沒問題後，再將整個系統整合，並燒錄到FPGA上

期末報告注意事項

- 每組報告 10 分鐘 + QA 2 分鐘
- 請於 6/12 23:59:59 前繳交投影片至 moodle
  - 檔名請以 Final_GroupX 命名
- 報告順序

1	第12組	8	第4組
2	第5組	9	第2組
3	第7組	10	第11組
4	第13組	11	第10組
5	第6組	12	第14組
6	第1組	13	第3組
7	第8組	14	第9組

期末報告注意事項

- 報告內容請包含以下三項，其餘的可自行發揮
- RTL Hardware Design
  - 說明使用了哪些 module、各 module 的功能、有哪些 I/O ports 等，可以以 Block Diagram 呈現
- Finite State Machine
  - 介紹如何實作狀態機，可以以 state diagram 呈現
- Testbench verification
  - 介紹如何驗證 module 的正確性，包含驗證過哪些 module、以哪些 pattern 驗證等

Appendix 1. Constraint Files

- Constraint file是用來描述 Verilog code 中的 port 與 FPGA 上 pin腳的關係
- 如果需用到FPGA作為input / output，則必須定義xdc file
- 這次Project可能會使用到的input / output與xdc檔對應區段名稱
  - RGB LED (RGB LEDs)
  - 單色LED (LEDs)
  - 按鈕 (Buttons)
  - 開關 (Switchs)
  - 數位I/O (Arduino Digital I/O)

Appendix 1. Constraint Files (cont'd)

範例說明：

```
##Buttons

#set_property -dict { PACKAGE_PIN D19   IOSTANDARD LVCMOS33 } [get_ports { btn[0] }]; #IO_L4P_T0_35 Sch=btn[0]
#set_property -dict { PACKAGE_PIN D20   IOSTANDARD LVCMOS33 } [get_ports { btn[1] }]; #IO_L4N_T0_35 Sch=btn[1]
#set_property -dict { PACKAGE_PIN L20   IOSTANDARD LVCMOS33 } [get_ports { btn[2] }]; #IO_L9N_T1_DQS_AD3N_35 Sch=btn[2]
#set_property -dict { PACKAGE_PIN L19   IOSTANDARD LVCMOS33 } [get_ports { btn[3] }]; #IO_L9P_T1_DQS_AD3P_35 Sch=btn[3]
```

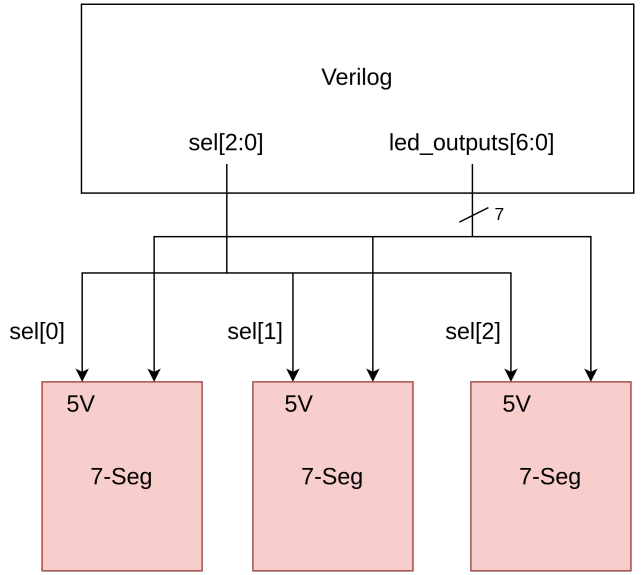
若要使用該pin腳，此處需解開註解（刪除#號），**若未使用該pin腳，務必將此處註解起來**

於此處寫上該硬體對應Verilog port的名稱，需注意此名稱需存在於top module的input或ouput中

在上面的例子中，top module中有一個bit-width為4的訊號`btn`，其中的`btn[0]`會對應到FPGA第0個按鈕，`btn[1]`會對應到第一個，以此類推

Appendix 2. Sweeping Seven-Seg

- 在Pynq-Z2中只有15個數位I/O(Arduino digital I/O)的腳位，若要同時使用超過兩個7段顯示器的話，可以採用sweeping的方法解決
- 範例：使用3個7段顯示器
  - 3個訊號線表示目前所選的顯示器，接到三個7段顯示器5V位置
    - `sel[2:0]`可能為001、010、100
  - 每個顯示器各一個7-bit register儲存要顯示的數值，每個cycle根據sel選擇一個register輸出給7段顯示器
  - 每個cycle輪流輸出一個顯示器要顯示的數值，當clock夠快就會看起來三個七段顯示器同時亮起來

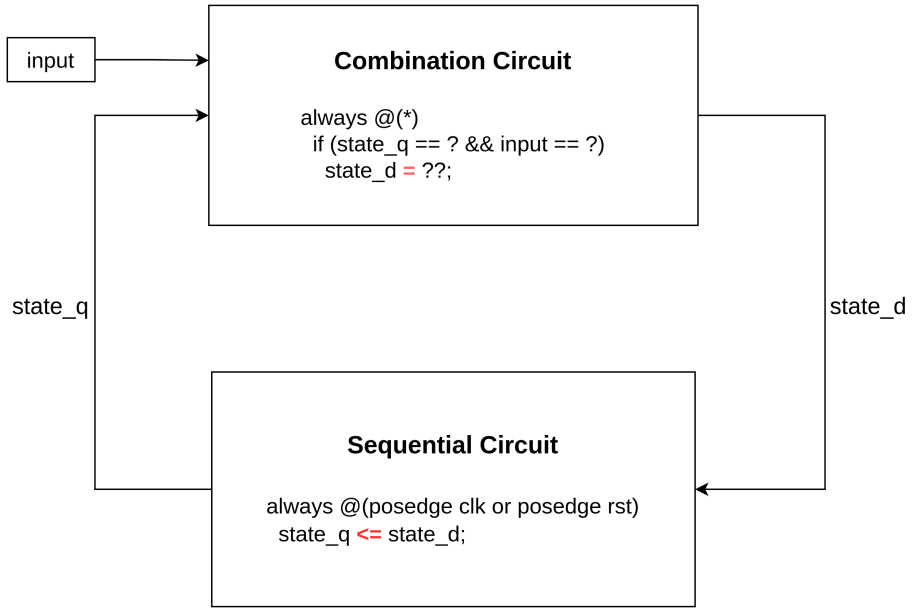


Appendix 3. State Machine

- 在Verilog撰寫state machine時，我們可以將組合電路以及序向電路分開撰寫以方便debug
- 組合電路：
  - 根據目前所在的state (`state_q`) 以及其他條件決定下個cycle應該是哪個state (`state_d`)
  - e.g.,

```
if (state_q == 2'd0)
    state_d = 2'd1;
```

- 序向電路：
  - 根據組合電路的輸出更新新的state的值
  - e.g., `state_q <= state_d;`



Appendix 4. 下拉電阻

- 在使用按鈕或是開關作為輸入的電路，可以使用下拉電阻（Pull-down resistor）來確保當按鈕未被按下時，輸入pin腳的訊號是低電位
- 與之前實驗課的 TTL IC不同的是，若是沒有下拉電阻，pin腳會處於「浮接（Floating）」狀態，會造成pin腳輸入電壓的不確定性