NCTU-EE DCS-2019

HW01

Design: MIPS CPU(No clock) + Seven-Segment Display

資料準備

1. 從 TA 目錄資料夾解壓縮

   % tar -xvf ~dcsta01/hw01.tar

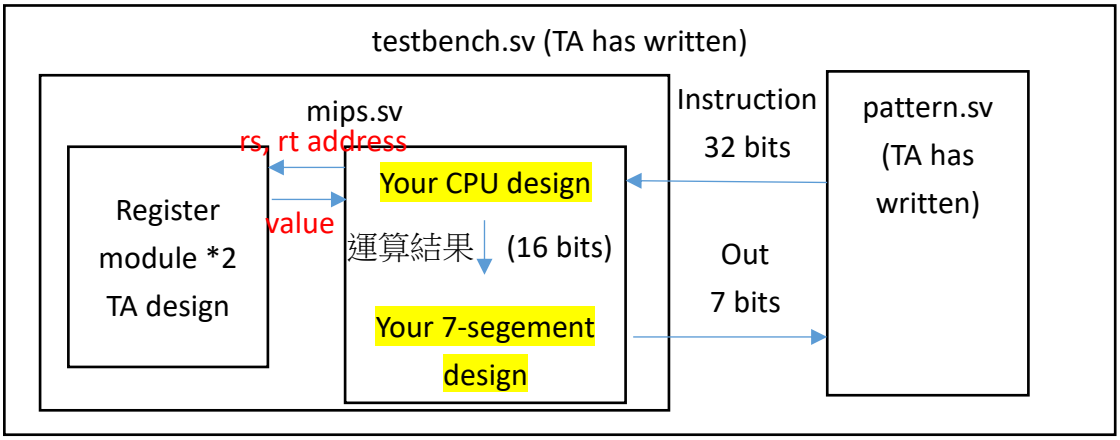
2. 解壓縮資料夾 hw01 包含以下：

   A. 00_TESTBED/

   B. 01_RTL/

   C. 02_SYN/

Block Diagram



設計描述

此次作業目的是設計一個簡單的 CPU 執行簡單運算，輸出值轉換成七字顯示器形式。

每個 instruction 為 32 bits 組成，每個 bit 不是 1 就是 0，並且由 pattern 的 output 給予至你的 design 的 input。

以下表格是我們作業會使用到的指令：

Type	Instruction 32 bits					
R	opcode (6 bits)	Rs (5 bits)	rt (5 bits)	Rd (5 bits)	Shamt (5 bits)	funct (6 bits)
	opcode	funct		Operation		
	000000	100000		rs + rt		
		100100		rs and rt		
		100101		rs or rt		
		100111		rs nor rt		
		000000		rt shift 向左 shamt bits		
		000010		rt shift 向右 shamt bits		
Type	Instruction 32 bits					
I	Opcode (6 bits)	rs (5 bits)	rt (5 bits)	immediate (16 bits)		
	opcode	Operation				
	001000	rs + imm (16bits)				

rs: source register address, rt: target register address,

rd: destination register address, imm :value

Opcode 可以選擇 R-type 或 I-type 指令。例如當 opcode 選擇 001000 就是 I-type 指令，並在本次作業只會有 addi 需要實現，且 addi 中的 **imm 資料皆是正值且不會造成 overflow**；當 opcode 選擇 000000 就是 R-type 指令，我們可以透過 funct 選擇不同指令，本次做作業必須實現 add, and, or, nor, sll(邏輯向左), srl(邏輯向右)。

Rs, Rt, Rd 是 **register(暫存器)的 address，不是值**。本次作業助教會提供兩個 register file 的 module 一個是 RS 的暫存器，一個是 RT 的暫存器，**你們必須透過 module connect(name mapping)，將兩個暫存器與你的 design 連接。**

兩個暫存器位置及內容是一樣，如下：

Address(5 bits)	**Value(16 bits) 注意為 16bits**
01101	12
01110	38
10001	27
10010	150
11011	379
11101	142
11111	1508

Rd address 是用來將運算結果寫回該位置，**但本次作業不需要將運算完的值寫回去，故 rd address 不會用到可以忽略。**

以下為一些指令的例子：

1. Instruction : 001000-01101-01110-0000000000000100

   Opcode - rs - rt - imm

   From opcode we can know this is the “addi” instruction.

   Note: “addi” instruction destination is rt, not rd. So, rt values is output.

   It means register(01101) = 12, imm = 4 , rt = 12 + 4 = 16.

2. Instruction : 000000-10001-01101-01110-00001-000000

   Opcode- rs - rt- rd - imm- func

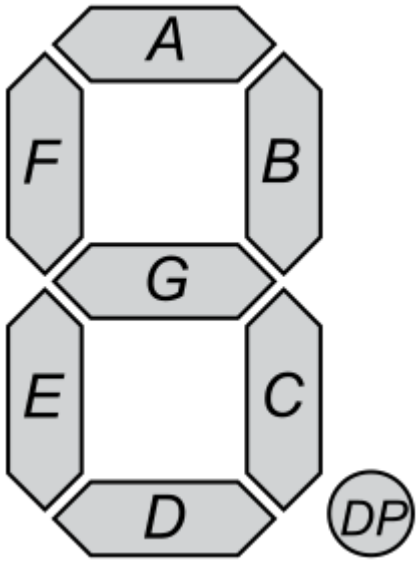
   From opcode & funct we can know this is the “sll” instruction.

   Note: “sll” instruction 是必須移動 rt 的資料而不是 rs.

   It means register(01101) = 12 shift 1-bit left and rd = 24

最後一步驟，你必須將計算完的數值最後 4 個 bits 轉成七字顯示器形式。

七字顯示器的顯示如下：



Digit	Display
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	A
b	b
C	C
d	d
E	E
F	F

在本次作業 MSB(Most Significant Bit)是 A，LSB(Least Significant Bit)是 G, DP 直接忽略。

舉例如下：運算完資料為十進位 19，二進位 10011，後 4bits 為 0011，故七字顯示器顯是 3，故輸出為 1111001。

最後 testbench 測試 pattern 將會在下一個指令進來前 5us 測試這 7-bits output。

Input

Signal name	Number of bit	Description
instruction	**32 bits**	Instruction will be given every 10us

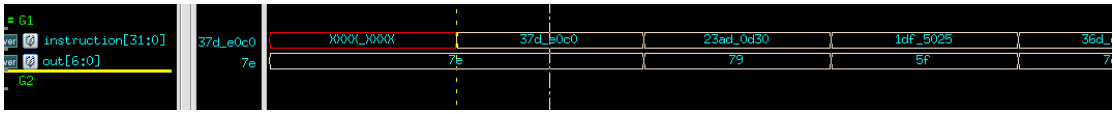
Output

Signal name	Number of	Description
out	**7 bit**	The 7 bit value will be checked 5us before the next instruction.

Specification

1. Top module name : mips (File name: mips.sv)
2. 請用 System verilog 完成你的作業。
3. 請用 Combination circuit 完成你的作業。
4. 請用助教給予你的 register module 拿資料。
5. 02_SYN result 不行有 error 且不能有 latchs。

Example Waveform



白色線條為 **check** 答案時間點

上傳檔案

1. mips_dcsxx.sv (如果需要更新，請直接替換原始舊檔案)
2. report_dcsxx.pdf, **xx is your server account.**
3. 請 **3/22 9:00 am** 之前上傳

Grading Policy

1. Pass the RTL & Synthesis simulation. 70%
2. Area 15%
3. Report. 15%

Note

Template folders and reference commands:

1. 01_RTL/ (RTL simulation) ./01_run
2. 02_SYN/ (Synthesis) ./01_run_dc

報告請簡單且重點撰寫，不超過兩頁 **A4**，並包括以下內容

1. 描述你的設計方法，包含但不限於如何加速(減少 **critical path**)或降低面積。
2. 基於以上，畫出你的架構圖**(Block diagram)**
3. 心得報告，不侷限於此次作業，對於作業或上課內容都可以寫下。