7 RTL Verilog Modeling & FPGA

7.1 實驗目的

- 練習 RTL Verilog modeling 語法，並在 Nexys4 上完成加減乘計算機。

7.2 實驗器材與元件

名稱	說明
 Nexys4	Nexys4是以Xilinx Artix7為核心之FPGA開發實驗板。整合switch、7-segment、溫度感測器、加速度器等多樣化的週邊，省去各位同學在麵包板上接線，或是焊接電路板的繁瑣步驟。Nexys4詳細的週邊資訊與板上的連線請參閱線上datasheet：https://reference.digilentinc.com/_media/nexys4-ddr:nexys4ddr_rm.pdf
ISE DESIGN SUITE ISE	ISE 是由一家叫做 Xilinx 的公司設計的開發工具，被用在 Verilog 語言的撰寫與將撰寫好的 Verilog 程式燒錄到 Xilinx 製作的 FPGA 開發板上執行。 下載網址: http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html

7.3 實驗內容

同學應在先前 Lab 了解 Structural Verilog Modeling 語法與 Nexys4 開發板的使用，本次實驗則是要讓同學練習 RTL Verilog modeling 語法，RTL 是 Register Transfer Level 的縮寫，也就是暫存器轉換語言，這種寫法與 C、Java 等高階語言非常相似。本次實驗為使用 RTL Verilog modeling 在 Nexys4 實作加法計算機。

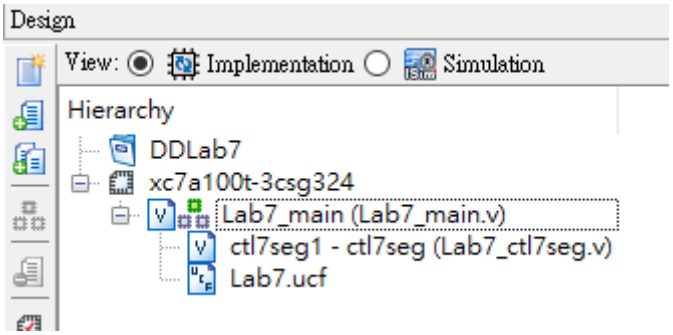
詳細的實驗內容如下

實驗

步驟一：

打開 DDLab7 資料夾，點擊兩下DDLab7.xise開啟專案

開啟專案後，在左上方 Hierarchy 欄位可以看到專案中的檔案

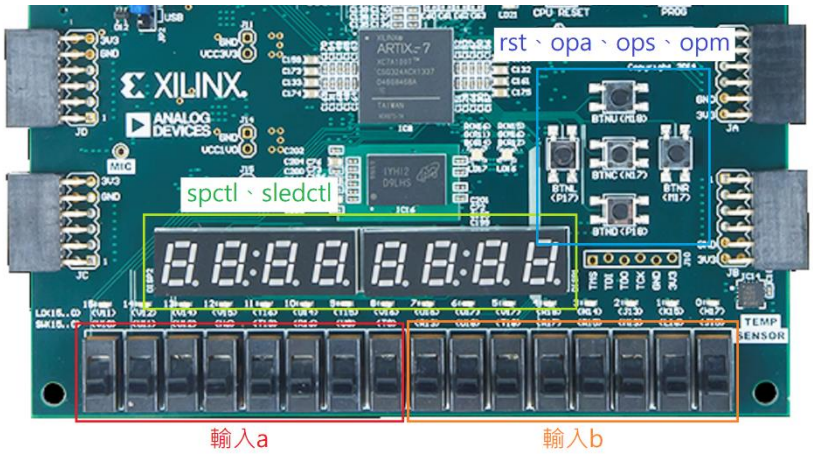


步驟二：

專案中主要的 Verilog 檔案有兩個：

第一個是 Lab7_main.v，為主要控制開關輸入、計算結果與七段顯示器輸出用

```
1  `timescale 1ns / 1ps
2
3  module Lab7_main(
4      input [7:0] a, b,
5      input rst, opa, ops, opm, clk,
6
7      output [6:0] sledctl,
8      output [7:0] spctl
9      );
10
11     reg signed [15:0] result;
12     reg [1:0]op;
13
14     ctl7seg ctl7seg1(.clk(clk), .a(a), .b(b), .result(result), .op(op), .sledctl(sledctl), .spctl(spctl));
15
16     always@(posedge clk)
17     begin
18         // Write your design here
19     end
20 endmodule
```



spctl：控制哪個位置的七段顯示器要亮 (可參考 Lab2 之講義)

sledctl：控制一個七段顯示器中的那些 LED 要亮 (可參考 Lab2 之講義)

中間按鈕是 rst　　為告訴計算機不做任何運算

上面按鈕為 opa　　為告訴計算機要做加法

下面按鈕為 ops　　為告訴計算機要做減法

左邊按鈕為 opm　　為告訴計算機要做乘法

右邊按鈕　　本次實驗未使用

註解 Write your design here 處為這次實驗課要修改的地方

第二個是 Lab7_ctl7seg.v、為將輸入 a、b 與運算結果轉換成七段顯示器輸出的格式用

```
1   `timescale 1ns / 1ps
2
3   module ctl7seg(
4       input clk,
5       input [7:0] a, b,          a、b 為一開始之輸入值
6       input [15:0] result,       result 為計算結果
7       input [1:0] op,            op 為是否有做計算 0 為沒有 非 0 則有
8
9       output [7:0] spctl,        spctl：控制哪個位置的七段顯示器要亮 (可參考 Lab2 之講義)
10      output [6:0] sledctl       sledctl：控制一個七段顯示器中的那些 LED 要亮 (可參考 Lab2 之講義)
11      );
12
13      reg [7:0] spctl;
14      reg [6:0] sledctl;
15      reg [11:0] count;          count：下面計算輔助用
16      reg [4:0] tmpin;           tmpin：分離位數數字用
17      reg dr;
18
19      always@(posedge clk)
20      begin
21          case (count)
22              12'b111111111111 : count <= 12'b0;
23              default : count <= count +1;
24          endcase
25      end
26
27
28      always@(posedge clk)
29      begin                      將 a、b 與計算結果的每一位數的十進位是多少分離出來
30          if(op==0)
31          begin
32              dr=0;
33              case (count[11:9])
34              3'b000 : tmpin = b % 10;
35              3'b001 : tmpin = b / 10 % 10;
36              3'b010 : tmpin = b / 100;
37              3'b100 : tmpin = a % 10;
38              3'b101 : tmpin = a / 10 % 10;
39              3'b110 : tmpin = a / 100;
40              default : tmpin = 4'b1111;
41              endcase
42
43          end
44          else
45          begin
46              dr=1;
47              case (count[11:9])
48              3'b000 : tmpin = result % 10;
49              3'b001 : tmpin = result / 10 % 10;
50              3'b010 : tmpin = result / 100 % 10;
51              3'b011 : tmpin = result / 1000 % 10;
52              3'b100 : tmpin = result / 10000;
53              default : tmpin = 4'b1111;
54              endcase
55          end
56      end
```

```
57
58      always@(posedge clk)
59      begin                                   將 tmpin 之數值轉換為七段顯示器之格式
60          case(tmpin)                         (格式可參考 Lab2 之講義)
61          0 : sledctl <= 7'b1000000;
62          1 : sledctl <= 7'b1111001;
63          2 : sledctl <= 7'b0100100;
64          3 : sledctl <= 7'b0110000;
65          4 : sledctl <= 7'b0011001;
66          5 : sledctl <= 7'b0010010;
67          6 : sledctl <= 7'b0000010;
68          7 : sledctl <= 7'b1111000;
69          8 : sledctl <= 7'b0000000;
70          9 : sledctl <= 7'b0010000;
71          default : sledctl <= 7'b1111111;
72      endcase
73
74      case({dr,count[11:9]})
75          4'b1000 : spctl <= 8'b11111110;     控制哪一個七段顯示器要亮
76          4'b1001 : spctl <= 8'b11111101;     (如何控制可參考 Lab2 之講義)
77          4'b1010 : spctl <= 8'b11111011;
78          4'b1011 : spctl <= 8'b11110111;
79          4'b1100 : spctl <= 8'b11101111;
80          4'b0000 : spctl <= 8'b11111110;
81          4'b0001 : spctl <= 8'b11111101;
82          4'b0010 : spctl <= 8'b11111011;
83          4'b0100 : spctl <= 8'b11101111;
84          4'b0101 : spctl <= 8'b11011111;
85          4'b0110 : spctl <= 8'b10111111;
86          default : spctl <= 8'b11111111;
87      endcase
88  end
89  endmodule
```

這邊可以看到 Verilog 與其他語言的 Switch-Case 寫法不同（以下以 C 為例子）：

C 的 Switch-Case	Verilog 的 Switch-Case

switch (input)
{
case A:
//do something
break;
case B:
//do something
break;
default:
//do something
break;
}
	case (input)
	A:
	//do something
	B:
	//do something
	default:
	//do something
	endcase

Verilog 的 Case 不用加 break

步驟三：

在 Lab7_main.v 的 Write your design here 處填入以下程式碼

```
case({rst, opa})
        2'b10 : op=0;
        2'b01 : op=1;
endcase


case (op)
        2'b01 : result=a+b;
        default : result = 16'b0;
endcase
```

將 rst 與 opa 合併成一個 2-bit 一起判斷
若只有 rst 為 1 將 op 設定為 0
若只有 opa 為 1 則將 op 設定為 1
其餘情況不更改 op

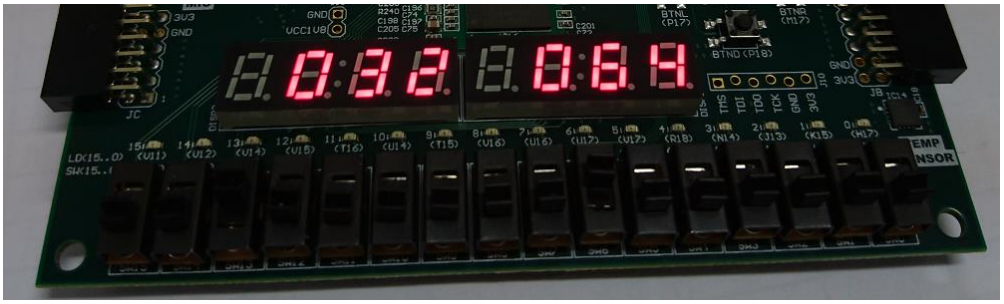
看 op 數值決定計算
若 op=1 時在 result 中放去加法之結果
其他情況則在 result 中放 0 代表沒有要做計算

步驟四：

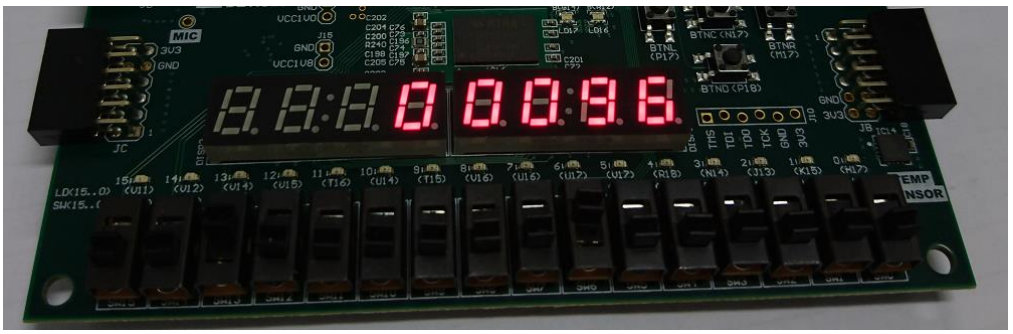
參考 Lab2 與 Lab4 之步驟合成並載入結果到 Nexys4 開發板中

結果：

一開始會分別將 a、b 顯示出來



按下"上"按鈕之後會顯示加法之結果 (32+64=96)



按下中間按鈕則會回到上面顯示尚未運算之結果

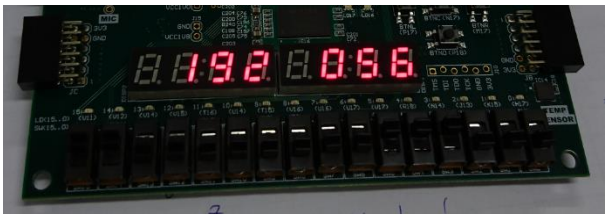
7.4 練習題

利用與實驗一樣的架構實作出可執行加法、減法與乘法的計算機，按鈕要求就如實驗內容所示：

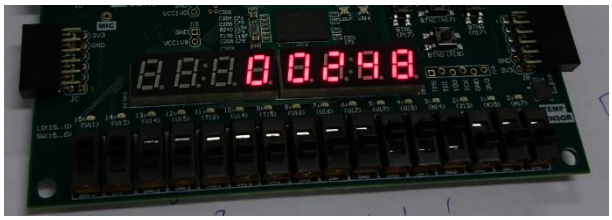
中間按鈕是 rst	為告訴計算機不做任何運算
上面按鈕為 opa	為告訴計算機要做加法
下面按鈕為 ops	為告訴計算機要做減法
左邊按鈕為 opm	為告訴計算機要做乘法
右邊按鈕	本次實驗未使用

預期結果如以下：

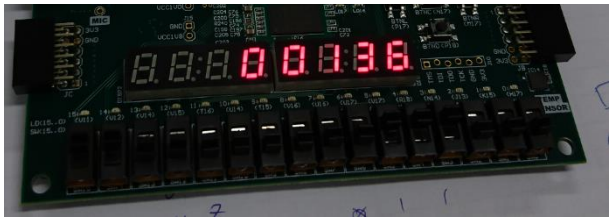
一開始顯示出 a、b 的數值：



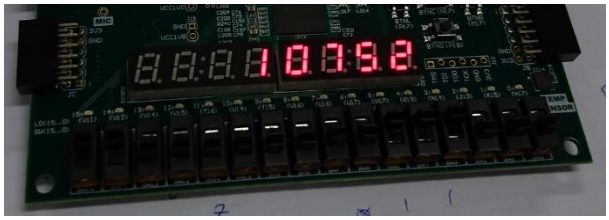
按了”上”按鈕之後顯示加法的結果 (192+56=248)



按了”下”按鈕之後顯示減法的結果 (192-56=136)



按下”左”按鈕之後顯示乘法的結果 (192x56=10752)



按下”中間”按鈕之後顯示尚未計算的 a、b 值 (如圖片一)