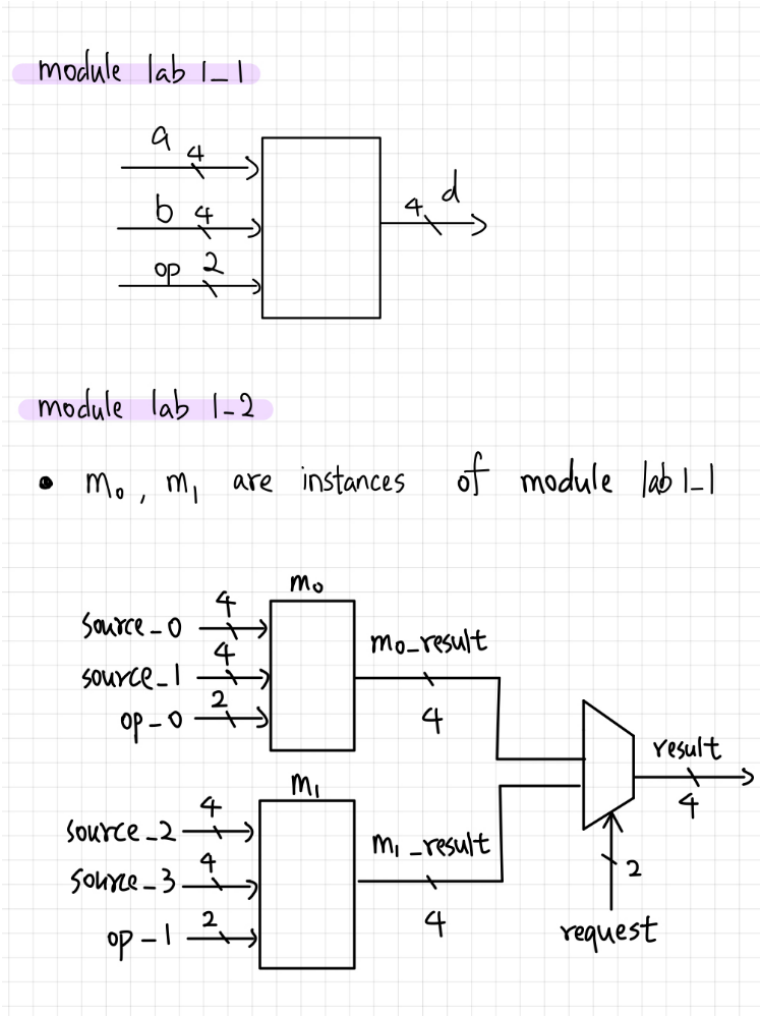
Lab 1

學號: 109062173 姓名: 葉昱揚

A. Lab Implementation

Block diagram :



Module lab1_1:
由 2 個 4 bit input、1 個 2 bit input 、1 個 4 bit output 組成，並檢測當 signal 改變時，根據 op 的值賦予 output d 不同的運算結果。

Module lab1_2:
需要使用 module lab1_1 完成。主體由 2 個 lab1_1 的 instance 組成(如上圖的 m0、m1)，m0、m1 有各自的 output，兩者會被傳送進 2 to 1 的 Mux，最後由 2 bit request 決定輸出 result。

Partial code :

Module lab 1_1 kernel part :

```
always@* begin

    case (op)
        2'b00 : d = a & b;
        2'b01 : d = a << b;
        2'b10 : d = a | b;
        2'b11 : d = a >> b;
        default: d = 0;
    endcase

end
```

always block 的 sensitive list 用*表示當有任何 signal (a,b,op,d) 改變時，根據 op 的值，對 output(d)做不同的運算。

Module lab1_2 kernel part :

```
// Connect modules //

lab1_1 m0(
    .op(op_0),
    .a(source_0),
    .b(source_1),
    .d(m0_result)
);
lab1_1 m1(
    .op(op_1),
    .a(source_2),
    .b(source_3),
    .d(m1_result)
);

////////////////////

always@* begin
    case(request)
        2'b00: result = 4'b0;
        2'b01: result = m0_result;
        2'b10: result = m1_result;
        2'b11: result = m0_result;
        default: result = 4'b0;
    endcase
end
```

Lab1_2 需要用 lab1_1 完成。使用兩個 reference 為 lab1_1 的 instances m0、m1，m0、m1，input port 照著 spec 的敘述接上，output port 拉兩條 wire 接上，分別為 m0_result、m1_result，最後根據 request 賦予 result 不同的值。

B. Questions and Discussions

(A) In the testbench lab1_1_t.v, please explain why we place #DELAY between input assignment and output verification. Hint: Gate delay.

依據 Module lab1_1 的設計，當 op、a、b 三者數值改變時，其會計算 output d。testbench lab1_1_t.v 中的#DELAY 是為了模擬硬體接收 input signal 時，用於運算的底層 gates 需要一段時間計算 output，此處的#DELAY 即是模擬硬體實際運算時花費的時間。

若將 #DELAY 放在 {op,a,b} assigning 之前或是 if-else block 之後，d 和 golden_d 計算用的 input signal {op,a,b} 將不同。

假設在 **time = T** 的時候 {op,a,b} 遞增，在 **time = T + #DELAY** 時才能得到正確的 output d，但 testbench 會在 **time = T** 將尚未計算的 output d 與計算完成的 golden_d 比較，必然得到錯誤的比較結果。

(B) If we want to let the 2'b00 operation of op_0 and op_1 have the highest priority, 2'b01 have the 2nd highest priority, and so on. When op_0 and op_1 has same operation, op_0 still has higher priority. How would you modify the code?

根據題意上述題意和討論區助教的解釋，在 request=2'b11 的時候比較 op_0 和 op_1 的大小，數值越小優先度越高，當數值相等的時候優先做 op_o operation。

舉例來說，如果 op_0 小於等於 op_1 (e.g. op_0 = 2'b00、op_1 = 2'b01)，則做 op_0 operation，反之。因此可以把 module lab1_2 的 always block 用三元運算子改成下列模樣 ：

```
always@* begin
    result = 4'b0;
    case(request)
        2'b00: result = 4'b0;
        2'b01: result = m0_result;
        2'b10: result = m1_result;
        2'b11: result = (op_0 <= op_1) ? m0_result : m1_result;
        default: result = 4'b0;
    endcase
end
```

原本在 request=2’b11 時，result 等於 op_0 operation 的運算結果，現在更改成比較 op_0、op_1 的大小後再決定 result 的值。

C. Problem Encountered

對於 Question A 的 gate delay 想了很久，起初單看程式碼無法判斷出為什麼要加#DELAY。

後來把#DELAY 加在各個程式區域，並用 vivado + testbench 每一個都跑一遍，觀察電路圖，最重要的是 vivado 內的 step 功能可以一步一步看程式正在執行哪一條指令，看了很久發現執行到#DELAY 時會跳到 module lab1_1 計算 Output，跳回 testbench 的時候 Output 計算完且時間往前了 5ns，我才成功從這邊反推出 gate delay 的功能 。詳細的 gate delay 功能已寫在 Question A 裡面。

D. Suggestions

非常感謝老師和助教們在討論區的幫忙，lab1 我問了幾個問題助教們非常迅速地回答，真的很感謝，沒有大家的友善幫助 Question 和實作過程會卡住很久，謝謝 ！

附上笑話一則。