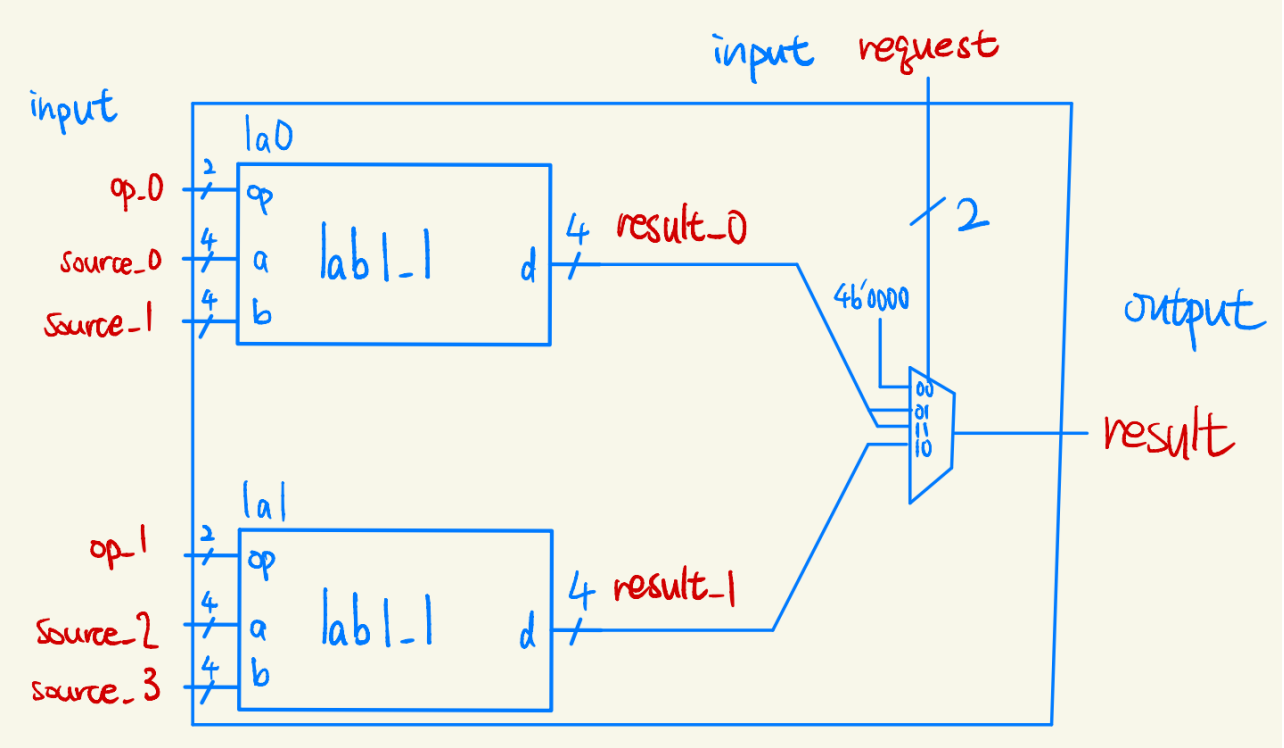
Lab 1

學號: 109021115 姓名: 吳嘉濬

A. Lab Implementation

以下是 module lab1_2 的 block diagram：



我在 module lab1_2 底下 instantiate 兩個 reference 為 lab1_1 的 instance，利用 lab1_1 設計的功能各自算出相對應的 output d，即 result_0 和 result_1，之後再利用 input request 來判斷最終 lab1_2 要 output 的 result 是 result_0 還是 result_1。

```
always @* begin
    d=a>>b;  //default
    if(op==2'b00) d=a&b;
    else if(op==2'b01) d=a<<b;
    else if(op==2'b10) d=a|b;
    else if(op==2'b11) d=a>>b;
end
```

以上是 module lab1_1 在判斷不同 op 值的情況下做出相對應動作的式子。

```
wire [3:0] result_0,result_1;  //cannot use reg
lab1_1 la0 (.op(op_0),.a(source_0),.b(source_1),.d(result_0));
lab1_1 la1 (.op(op_1),.a(source_2),.b(source_3),.d(result_1));
always @* begin
    result=4'b0;  //default
    if(request[0]==1'b1) begin  //op_0 has a higher priority
        result=result_0;
    end else if(request[1]==1'b1) begin
        result=result_1;
    end
end
```

以上是 instantiate 兩個 instance 並各自分別得到 result_0 和 result_1 之後，判斷 request 的值得到正確的 output result。(注意當 request==2'b11 時，op_0 有更高的 priority)

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.

因為輸入完 input 之後，期間需要經過許多電路邏輯閘的運算，才能得到想要的 output，在這過程當中會有相對應的 gate delay，因此我們必須加上#DELAY 等待一些時間，確保 output 的結果出來之後，再去驗證 output 是否正確。

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?

只有當 request==2'b11 時，這裡的改動才會有所影響。我選擇直接把 request 分成 4 種情況：

1.當 request==2'b00，result=4'b0000；

2.當 request==2'b01，result=result_0； //從 op_0 做運算得到的 result

3.當 request==2'b10，result=result_1； //從 op_1 做運算得到的 result

4.當 request==2'b11，if(op_0<=op_1) result=result_0 //從 op_0 做運算得到的 result

else result=result_1 //從 op_1 做運算得到的 result

其中要注意的地方是：當 op_0==op_1 時，op_0 有 higher priority

實作的 code 如下：

```
wire [3:0] result_0,result_1;  //cannot use reg
lab1_1 la0 (.op(op_0),.a(source_0),.b(source_1),.d(result_0));
lab1_1 la1 (.op(op_1),.a(source_2),.b(source_3),.d(result_1));
always @* begin
    result=4'b0000;  //default, i.e. when request==2'b00
    if(request==2'b01) begin
        result=result_0;
    end else if(request==2'b10) begin
        result=result_1;
    end else if(request==2'b11) begin
        if(op_0<=op_1) result=result_0;
        else result=result_1;
    end
end
```

C. Problem Encountered

最一開始在設計 lab1_2.v 時，寫出了以下的 code：

```
always @* begin
    result=4'b0;  //default
    if(request[0]==1'b1) begin  //op_0 has a higher priority
        lab1_1 la0 (.op(op_0),.a(source_0),.b(source_1),.d(result));
    end else if(request[1]==1'b1) begin
        lab1_1 la1 (.op(op_1),.a(source_2),.b(source_3),.d(result));
    end
```

結果出現了以下 error message：

```
Error: Syntax error near "lab1_1".
```

才意識到這個寫法是有問題的，因為 always block 內是會被重複 call 的，但是在 instantiate 完某個 instance 後，該線路就固定住了，所以同一個 instance 是不能一直被重複 instantiate 的。至此，我了解到當我們要 instantiate 一個 instance 時，一定不能寫在 always block 內。

D. Suggestions

有機會的話希望老師可以在課堂上多提供畫 FSM 和 dataflow 的訣竅，總感覺這些比打 Verilog 還要複雜。

笑話：
一個阿姨去醫院打針。
護理師：「先深呼吸。」
阿姨：「我是小姐。」

廢到笑⋯ 😊