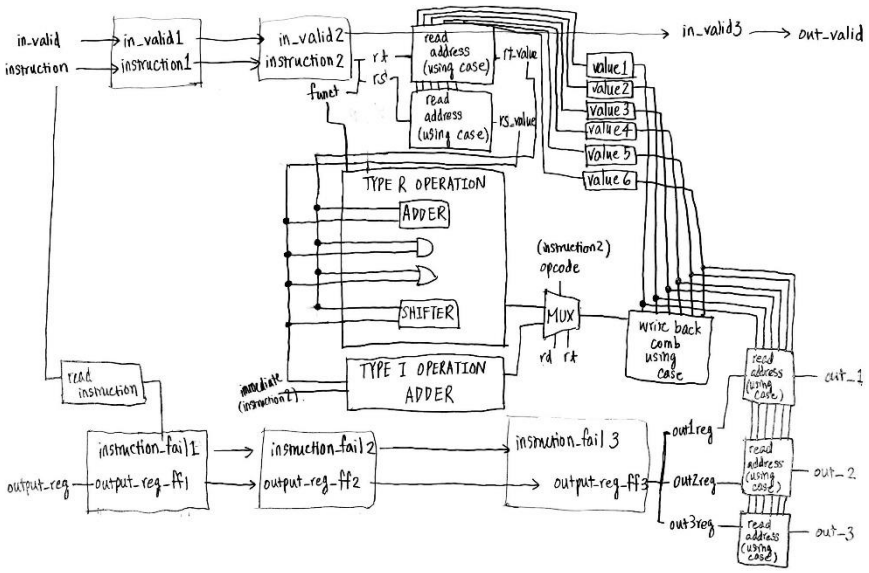
Block Diagram



設計過程錯誤和 debug

1. **判斷的 in_valid 依據錯誤**

   可以由上方的 block diagram 看出，本次為了達成 pipeline 的效果，所以創了三個 in_valid 的 register 做傳遞 (in_valid1, in_valid2, in_valid3)，而同時 instruction、output_reg 也要進來並且傳遞，而每一層 instruction、output_reg 的 flip flop 要不要傳到下一個 flip flop 也必需看 in_valid 是否為 1，不是的話就要停止傳遞。但是每一層要判斷的 in_valid register 是不一樣的，當初犯的錯誤就是把所有的傳遞都寫在 in_valid 的判斷裡面，導致 in_valid 一清零，中間傳遞到一半的值就停滯不出來了。所以應該要依據「前」一層級的 in_valid 做判斷(因為 in_valid 是用 non_blocking 傳的)

   如下圖:

```
151
152         if(in_valid == 0) begin
153             instruction1 <= 0;
154             output_reg_ff1 <= 0;
155         end
156         else begin
157         //first blue flip flop
158             output_reg_ff1 <= output_reg;
159             instruction1 <= instruction;
160         end
161         //second blue flip flop
162         if(in_valid1 != 0) begin
163             output_reg_ff2 <= output_reg_ff1;
164             instruction2 <= instruction1;
165         end
166         //third blue flip flop
167         if(in_valid2 != 0) begin
168
169             output_reg_ff3 <= output_reg_ff2;
170
171             if(instruction_fail2 == 0) begin
172                 if(instruction2[29] == 0) begin
```

2. **output_reg 的傳遞沒有寫入 in_valid 的判斷條件中**

   若沒有寫入 in_valid 的 if-else 裡面，會在 instruction 給完後繼續讀取 output_reg 的 address，然後依然去取值，這樣輸出當然就會是錯的。所以必需將 output_reg 的 pipeline 寫在 in_valid 的判斷裏頭如下圖。

```
122         ((instruction[31:26] == 6'b000000) &&
123         ((instruction[5:0] == 6'b100100)||(instruction[5:0] == 6'b100101)||
124         (instruction[5:0] == 6'b100111)||(instruction[5:0] == 6'b000000)||
125         (instruction[5:0] == 6'b000010)||(instruction[5:0] == 6'b100000)))||
126         (instruction[31:26] == 6'b001000)
127         )
128         instruction_fail1 <= 0;
129         else instruction_fail1 <= 1;
130
131         intruction_fail <= instruction_fail3;
132         instruction_fail3 <= instruction_fail2;
133         instruction_fail2 <= instruction_fail1;
134
135         output_reg_ff1 <= output_reg;
136         output_reg_ff2 <= output_reg_ff1;
137         output_reg_ff3 <= output_reg_ff2;
138
139
140         if(in_valid == 0) instruction1 <= 0;
141         else begin
142         //first blue flip flop
143             output_reg_ff1 <= output_reg;
144             instruction1 <= instruction;
145         //second blue flip flop
146             rs_value <= rs_value1;
147             rt_value <= rt_value1;
148             opcode <= instruction1[29];
149             rd <= rd1;
150             rt <= rt1;
151             shamt <= shamt1;
```

程式、時間和面積的優化過程

這次的修改主要分成兩部分:

A. 模組的廢除

B. 在第二個 flip flop 才做 decode 的動作

在最初的設計中，我讀 address 這件事是另用一個模組來實現，原本的想法是因為很多動作，像是 rs, rt, 讀取值、output_reg 為 out1, out2, out3 選取輸出，都是在做同一件事，所以用一個模組，寫一次 case 再接上六個變數就可以重複宣告，可以少寫很多行 code。但是，code 行數的多寡不見得代表面積和 slack 的增減!

我依序嘗試了以下更改組合:

1. 一開始我在還保留模組的設計下先做 B.，結果面積變大、slack 變小；
2. 再來我把模組的寫法改成多個 always_comb 寫在主模組裡面，也就是不另宣告模組，但是 decode 依然在第一個 flip flop 後做，雖然 code 的行數變多，但結果果真有好一些
3. 把模組廢除，寫多個 always_comb、decode 到第二個 flip flop 後做，面積小最多，但 slack 比 2. 小了些

我最終保留 3. 的設計，原因是減掉幾千的面積感覺還是比少了零點零幾的 slack 來的優質，以下是各次跑 synthesis 的截圖:

原始設計 (沒做 A、沒做 B)

```
Net interconnect area:      undefined  (No wire

Total cell area:              67748.790117
Total area:                 undefined
1
report_timing
```

```
509   ---------------------------------------------------------
510   data required time                                 4.71
511   data arrival time                                 -4.69
512   ---------------------------------------------------------
513   slack (MET)                                        0.02
514
```

1. 做 B 沒做 A

```
417
418 Total cell area:          68570.410843
419 Total area:               undefined
420 1
421 report_timing
```

```
477   ---------------------------------------------------------
478   data required time                                 4.67
479   data arrival time                                 -4.67
480   ---------------------------------------------------------
481   slack (MET)                                        0.00
482
```

2. 做 A 沒做 B

```
446
447 Total cell area:          67718.852511
448 Total area:               undefined
449 1
450 report_timing
```

```
507   ---------------------------------------------------------
508   data required time                                 4.72
509   data arrival time                                 -4.67
510   ---------------------------------------------------------
511   slack (MET)                                        0.04
512
```

3. 做 A 做 B (最終保留的設計)

```
425 Net Interconnect area:    undefined  (No wire load specified)
426
427 Total cell area:          65703.053808
428 Total area:               undefined
429 1
430 report_timing
```

```
490   ---------------------------------------------------------
491   slack (MET)                                        0.01
492
```