Computer Architecture Final Report

Students: b07901010 范詠為、b07901016 朱哲廣

Snapshots

no latch

```
Inferred memory devices in process
        in routine RISCV line 189 in file
                '/home/raid7_2/userb07/b07016/ken/Final_Project_v2/RISCV.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|   mem_addr_I_reg    | Flip-flop |  30   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================

Statistics for case statements in always block at line 268 in file
```

Timing report

```
  clock CLK (rise edge)                                   6.00       6.00
  clock network delay (ideal)                             0.50       6.50
  clock uncertainty                                      -0.10       6.40
  register/register_r_reg_4__10_/CK (DFFRX1)              0.00       6.40 r
  library setup time                                     -0.13       6.27
  data required time                                                 6.27
  -------------------------------------------------------------------------
  data required time                                                 6.27
  data arrival time                                                 -6.27
  -------------------------------------------------------------------------
  slack (MET)                                                        0.00


1
dc_shell>
```

Area report

```
Number of ports:                         1772
Number of nets:                         13079
Number of cells:                        11306
Number of combinational cells:           9278
Number of sequential cells:              2017
Number of macros/black boxes:               0
Number of buf/inv:                       1826
Number of references:                      79

Combinational area:              103444.647097
Buf/Inv area:                     15235.862378
Noncombinational area:            65309.160217
Macro/Black Box area:                 0.000000
Net Interconnect area:          1512754.454193

Total cell area:                  168753.807314
Total area:                     1681508.261507
1
dc_shell>
```

Describe design

- RISCV (top)

  Top module 的部分，always block 掌管的是 PC，負責告訴 instruction memory 要拿的 instruction 的位置，其他部分的 output 都是從下列 submodule 拿出來的值，只要線路照圖接好就沒問題。至於 PC 的 always block 要做的事是去分是哪類的指令，預設是直接 PC+4，比較特別的是 BEQ, BNE, JAL, JALR 四個，PC 要照 RISCV 的規定跳到正確的位置。

- Decoder

  基本上就是使用 HW3 和 HW4 寫好的 code，為了省面積，我們把原本的 type 和 format 拿掉。這邊比較特別的是 srai 這個指令的 immediate 沒有到那麼多位，如果直接拿整個的話會拿到他的 function7。

- Register

  原本最初不知道怎麼寫 register，但看了 memory.v 裡面的寫法後豁然開朗，於是也用類似的方式開了一個陣列，更新時即判斷是不是要寫入的 register 編號再決定是否給值或保留舊值。另外還有限制 RegWrite 信號要拉起的狀態和寫入的 register 不能是 x0。

```
always @(*) begin
    for (i=0; i<REGSIZE; i=i+1) begin
        register_w[i] = (RegWrite && (i == write_reg) && write_reg != 0) ?
write_data : register_r[i];
    end
end
```

- ALU

  輸入 ALU_ctrl 信號去判斷要做甚麼運算，再多輸出一個 zero 信號即可。至於 Branch 跟 Store/Load，則是在最外面的 RISCV module 判斷。Branch 給 SUB 的 ctrl，Store/Load 則是給 ADD。

```
always@(*) begin
    case(ctrl)
        ADD: res = (data1 + data2);
        SUB: res = (data1 - data2);
        SLL: res = (data1 << data2);
        SLT: res = (data1 < data2);
        XOR: res = (data1 ^ data2);
        SRL: res = (data1 >> data2);
        SRA: res = (data1 >>> data2);
        OR:  res = (data1 | data2);
        AND: res = (data1 & data2);
        default: res = 64'b0;
    endcase
end
```

- LE_CONV

  這是將 little endian 轉成 big endian 的 module，任何經過 data memory 的資料不管進來或出去都要經過轉換，而因為這個轉換是對稱的，就不需要再寫 big 轉 little 的 module。至於 instruction memory 的部分因為 instruction size 是 32bit，跟 data 是 64bit 不太一樣，加上最一開始沒注意到，我直接用硬轉的方式處理。

```
assign out[7:0]   = in[63:56];
assign out[15:8]  = in[55:48];
assign out[23:16] = in[47:40];
assign out[31:24] = in[39:32];
assign out[39:32] = in[31:24];
assign out[47:40] = in[23:16];
assign out[55:48] = in[15:8];
assign out[63:56] = in[7:0];
```

- Some improvement on A*T

  為了讓 Area 變小，我們將 Decoder 不需要的部分拿掉，只留下 ctrl_signal 和 immediate。除此之外，我們試過將 decoder 的 switch case 改成都用 assign 來處理，但在時間的表現上沒有變好。面積還反而增加了一些。

- Problems encountered

1. 一開始是跑 rtl 時，tb2,3 都過了但 tb1 一直過不了，而且還是卡在 simulation，也沒有顯示 timeout 訊息，也因為這樣 nWave 也看不出為什麼。原本以為是 jal 的問題，但後來直接改 tb 內的指令後發現是只要 read/write 是同一個 register 就會跑到一半卡住，推論是 Register module 寫錯了，結果還真的是因為register_r 寫成 register_w。只能說 verilog 的 debug 真的需要很多經驗。
2. 在跑 dc_shell 時有 latch，上網查原因發現是有些 switch case 的 default 沒寫造成的。

Work Distribution

b07901010 范詠為: Register, ALU, LE_CONV, RISCV, Report

b07901016 朱哲廣: Decoder, Debug, Optimization, Report