Comp E 475

Microprocessors

Lab : HW 5

Student: Davit Barbakadze

Red ID #: 822164584

10.12.2020s

Contents

Task Description

In Assignment 5 we had to use previous work (HW4) and add some instructions. We had to add two 3bit outputs mem_instr_type, jmp_instr_type and had to control those outputs with specific logic:

- **mem_instr_type**, 2 bits, values from 0 to 2
  - 1 if given Memory instruction type is of Immediate
  - 2 if it's "Register shifted by value" type
  - 0 if it's not identifiable
- **jmp_instr_type**, 2 bits , values from 0 to 2
  - 1 if given Jump instruction type is of Branch only
  - 2 if it's Branch and Link
  - 0 if it's not identifiable

Solution

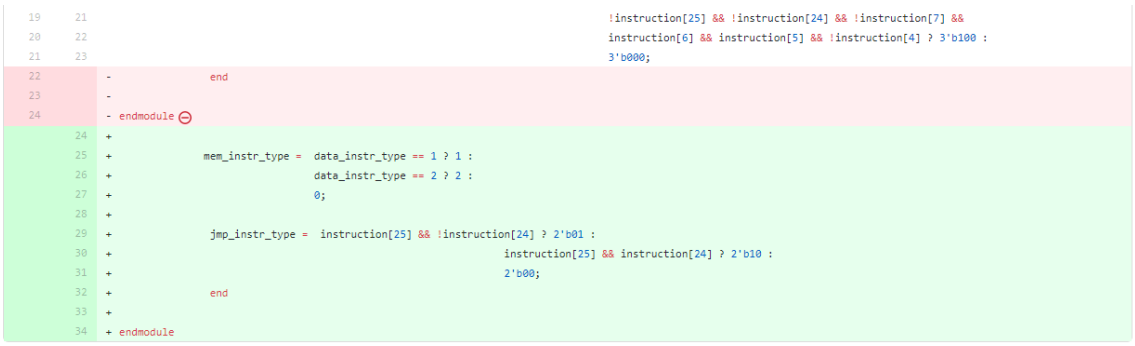
I added two short conditional statements two determine the value of both outputs. I could remove clock and use just combinational circuit but the code also works inside always block so I left it as it was.

```
1
2  module INSTR_DATA(
3     input clk,
4     input wire[31:0] instruction,
5     output reg[2:0] instr_type,
6     output reg[3:0] data_instr_type,
7     output reg[2:0] mem_instr_type,
8     output reg[2:0] jmp_instr_type
9
10     );
11
12    always@(posedge clk) begin
13        instr_type = instruction[27] && !instruction[26] ? 2'b11 :
14                            !instruction[27] && instruction[26] ? 2'b10 :
15                            !instruction[27] && !instruction[26] ? 2'b01:
16                            2'b00;
17
18         data_instr_type = instruction[25] ? 3'b001 :
19                                  !instruction[25] && !instruction[4] ? 3'b010 :
20                                  instruction[25] && !instruction[7] && instruction[4] ? 3'b011 :
21                                  !instruction[25] && !instruction[24] && !instruction[7] &&
22                                  instruction[6] && instruction[5] && !instruction[4] ? 3'b100 :
23                                  3'b000;
24
25        mem_instr_type =  data_instr_type == 1 ? 1 :
26                            data_instr_type == 2 ? 2 :
27                            0;
28
29       | jmp_instr_type =  instruction[25] && !instruction[24] ? 2'b01 :
30                            instruction[25] && instruction[24] ? 2'b10 :
31                            2'b00;
32
33    end
34 endmodule
35
36
37
```

Figure 1 verilog code

In figure 1 we see what our new code looks like.

Simulation & Verification



```
19  21                                                  !instruction[25] && !instruction[24] && !instruction[7] &&
20  22                                                  instruction[6] && instruction[5] && !instruction[4] ? 3'b100 :
21  23                                                  3'b000;
22      -               end
23      -
24      - endmodule
        24 +
        25 +               mem_instr_type =  data_instr_type == 1 ? 1 :
        26 +                                 data_instr_type == 2 ? 2 :
        27 +                                 0;
        28 +
        29 +                jmp_instr_type =  instruction[25] && !instruction[24] ? 2'b01 :
        30 +                                                    instruction[25] && instruction[24] ? 2'b10 :
        31 +                                                    2'b00;
        32 +               end
        33 +
        34 + endmodule
```

Figure 2 git commit output

Last two assignment operations are new for this lab, which can also be seen in my git commit in green color.

Comparison

As we have committed on the last assignment, we only changed some part of our code, so we see what exactly is changed on the code compared to previous submission of our repository file.

Conclusion

I learned how to commit on git and how to make difference between old and new code of the repository.

Git repository for this task: https://github.com/nikanika0221/Instruction-data