Comp E 475

Digital Systems

HW5 :instr Mem, Jmp

Student: Vano Mazashvili

Red ID #: 822059245

10.12.2020

Contents

Task Description

In this assignment, I extended the previous 'skeleton' for the ARM instruction decoder module. As we know, ARM instruction set encodes data in 32 bits (as we are learning). With this assignment, we were to write the code which had an input as a 32-bit instruction and determined instruction type, data instruction type, memory instruction type and jump instruction type. The following code is uploaded on a Github repository (https://github.com/vmazashvili).

The code should include

- **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

Besides that, old code has following statements:

- you have inputs:
  - **instruction**, 32 bits
  - **clk**
- outputs:
  - **instr_type**, 2 bits, values from 0 to 3
    - 1 if given 32 bit instruction is Data Processing instruction
    - 2 if it's memory type instruction
    - 3 if it's branch instruction
    - 0 if not identifiable
  - **data_instr_type**, 3 bits, values from 0 to 4
    - 1 if given Data Processing instruction is "Immediate" type
    - 2 if it's "Register shifted by value" type
    - 3 if it's "Register shifted by register" type
    - 4 if it's "Multiplication" type
    - 0 if not identifiable

Solution

I changed my previous code, which had an error in it. The code became more compact as I implemented combinational logic, without clock.

```
module HW4_instr_Data(
    input wire[31:0] instruction,
    output reg[2:0] instr_type, //2 bits, values from 0 to 3
    output reg[3:0] data_instr_type, //3 bits, values from 0 to 4
    output reg[1:0] mem_instr_type, //two bits, values from 0 to 2
    output reg[1:0] jmp_instr_type //two bits, values from 0 to 2
);
    always @(*) begin
        if(!instruction[27] && !instruction[26]) begin //1 if given 32 bit instruction is Data Processing instruction
            instr_type=2'b01;
            if(!instruction[25] && !instruction[4]) begin
                data_instr_type = 3'b010; //2 if it's "Register shifted by value" type
            end

            else if(instruction[25] && !instruction[7] && instruction[4]) begin
                data_instr_type = 3'b011; //3 if it's "Register shifted by register" type
            end

            else if(!instruction[25] && !instruction[24] && !instruction[7] && instruction[6] && instruction[5] && !instruction[4]) begin
                data_instr_type = 3'b100; //4 if it's "Multiplication" type
            end

            else begin
                data_instr_type = 3'b000; //0 if not identifiable
            end
        end

        else if(!instruction[27] && instruction[26]) begin //2 if it's memory type instruction
            instr_type=2'b10;
            assign mem_instr_type = instruction[25] ? 2'b01 : //1 if given Memory instruction type is of Immediate
            instruction[25] && !instruction[4] ? 2'b10 : //2 if it's "Register shifted by value" type
            2'b00; //0 if not identifiable
        end

        else if(instruction[27] && !instruction[26]) begin //3 if it's branch instruction
            instr_type=2'b11;
            assign jmp_instr_type = instruction[25] && !instruction[24] ? 2'b01 : //1 if given Jump instruction type is of Branch only
            instruction[25] && instruction[24] ? 2'b10 : //2 if it's Branch and Link
            2'b00; //0 if not identifiable
        end

        else begin
            instr_type=2'b00; //0 if not identifiable
        end
    end
endmodule
```

Simulation & Verification

The code simulated without errors. The warnings were about some instruction bits not being used.

HW4_instr_Data Project Status (10/15/2020 - 00:09:23)			
Project File:	HW4_instr_Data.xise	**Parser Errors:**	No Errors
Module Name:	HW4_instr_Data	**Implementation State:**	Synthesized
Target Device:	xc3s100e-4cp132	• **Errors:**	No Errors
Product Version:	ISE 14.5	• **Warnings:**	11 Warnings (0 new)
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Conclusion

In this assignment, I was able to implement an ARM decoder which was able to decode and distinguish jump and memory instruction types. For this, I used combinational, rather that more straight-forward (if/else) approach.