Comp E 475

# MICROPROCESSORS

# Instruction Data

Student: Vano Mazashvili

Red ID #: 822059245

10.11.2020

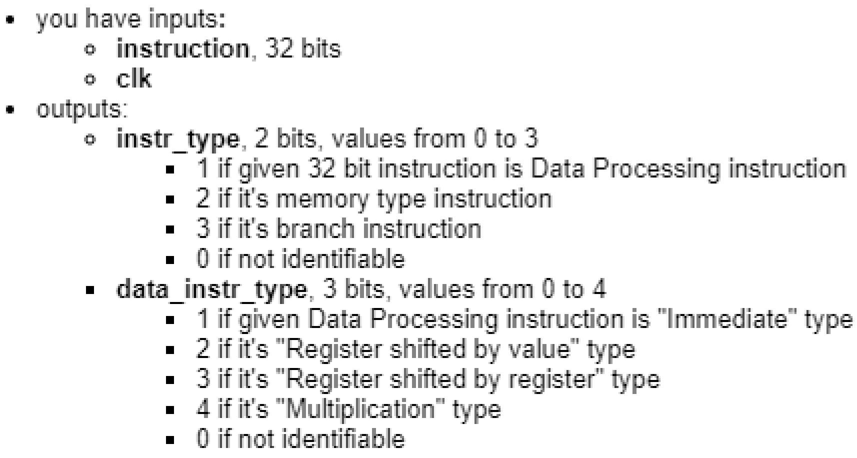
# Contents

# Task Description

The task in this homework was rather easy. As it was mainly to develop Git skills. For this assignment, we had to write a ‘skeleton’ for the ARM instruction decoder module.

The logic behind it would be an ability to decode arm instructions on a bit level. As we know, ARM instructions can be represented in 32 bits, so we can differentiate them from each other.

In this case, we should write a code for the Data processing instruction, memory type instruction, branch instruction, and several data instruction types: immediate, register shifted by value, register shifted by register,  and multiplication. The code should output instructed results:

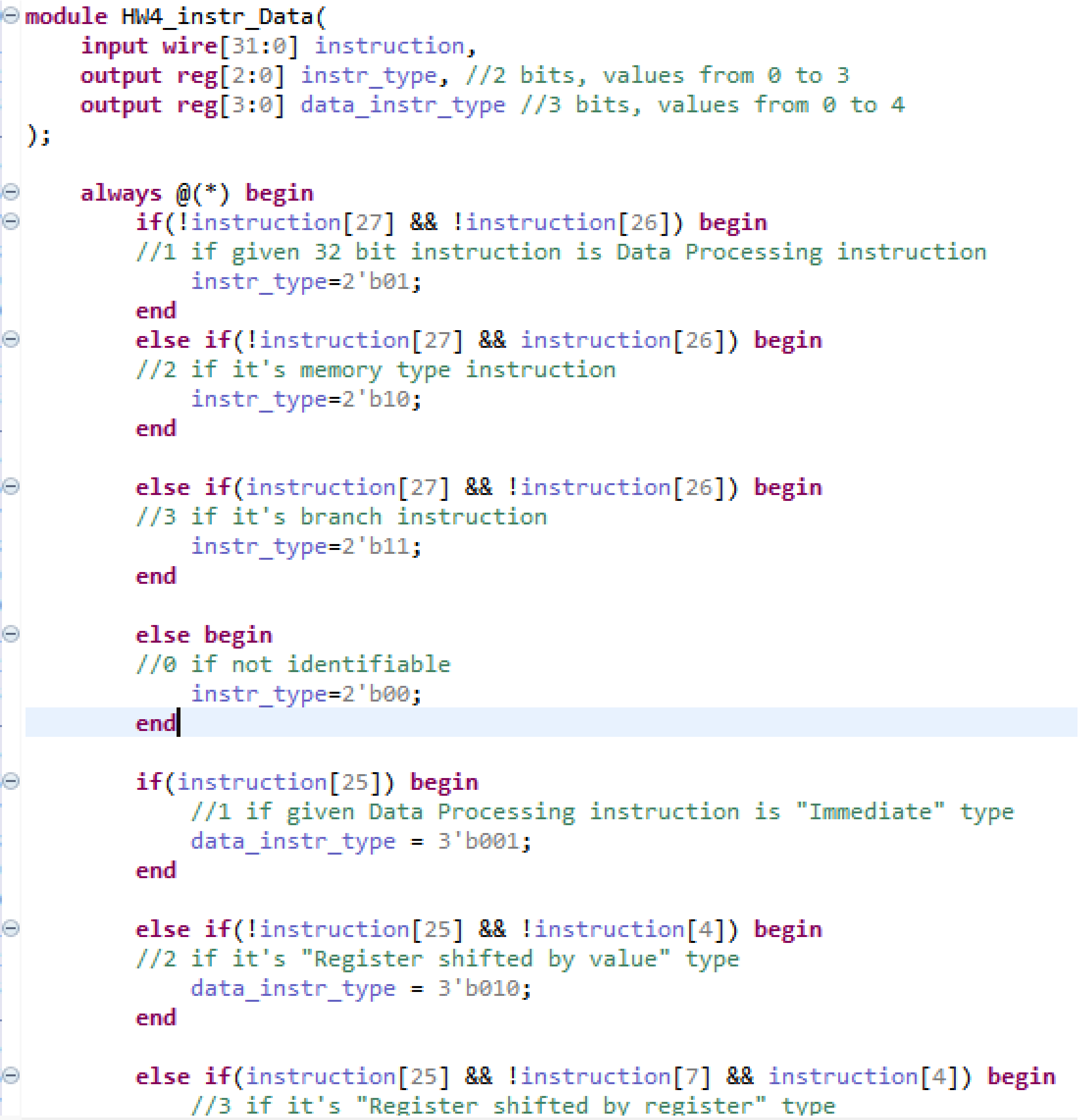


- you have inputs:
  - **instruction**, 32 bits
  - **clk**
- outputs:
  - **instr_type**, 2 bits, values from 0 to 3
    - 1 if given 32 bit instruction is Data Processing instruction
    - 2 if it's memory type instruction
    - 3 if it's branch instruction
    - 0 if not identifiable
  - **data_instr_type**, 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

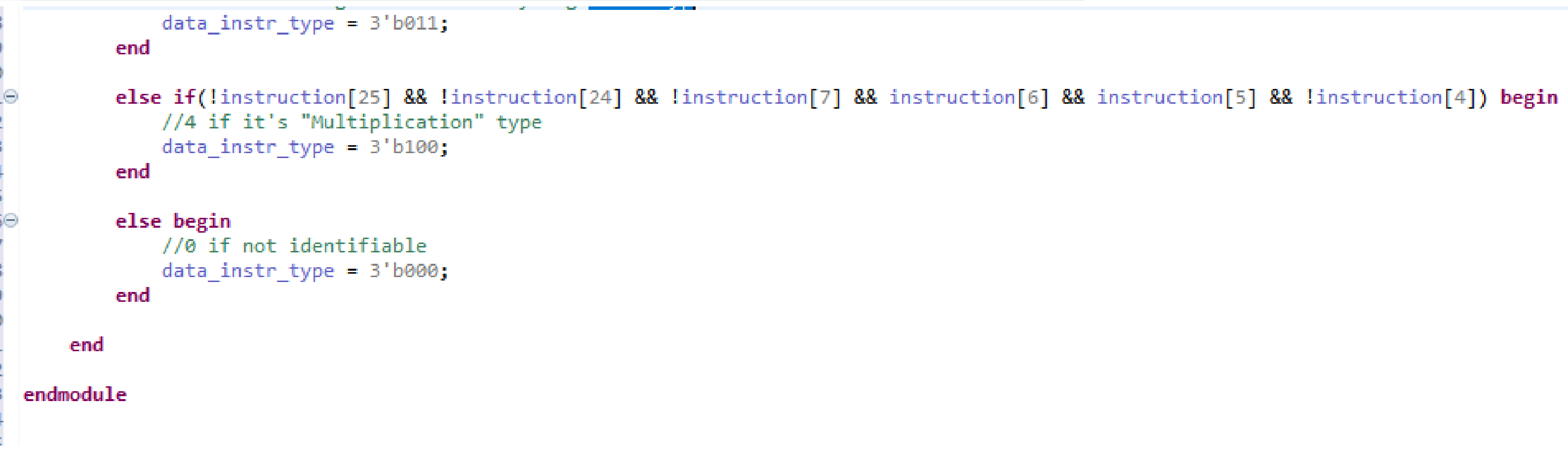
# Solution

For solution, I chose the most basic and straight-forward approach – if-else statements. And removed clock, which wasn’t necessary in this circumstance. The code is uploaded on Github, with several commits and one merge.

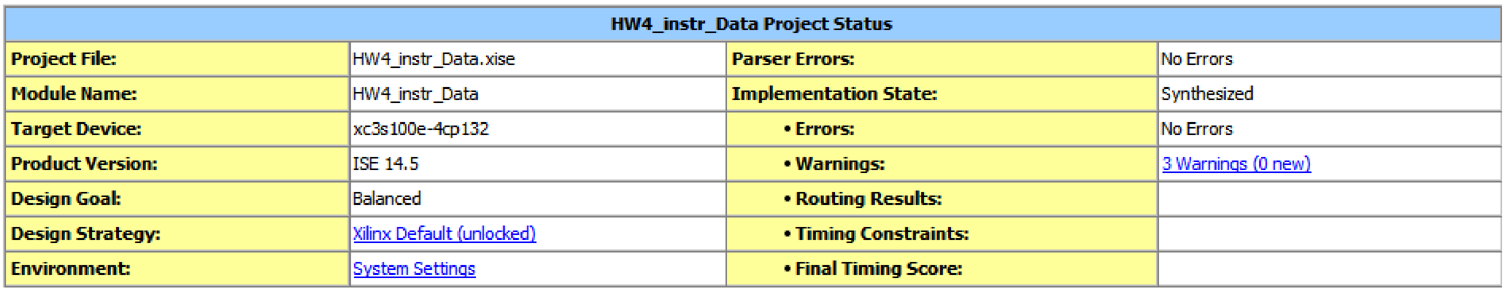
## Simulation & Verification



```
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
);

    always @(*) begin
        if(!instruction[27] && !instruction[26]) begin
        //1 if given 32 bit instruction is Data Processing instruction
            instr_type=2'b01;
        end
        else if(!instruction[27] && instruction[26]) begin
        //2 if it's memory type instruction
            instr_type=2'b10;
        end

        else if(instruction[27] && !instruction[26]) begin
        //3 if it's branch instruction
            instr_type=2'b11;
        end

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

        if(instruction[25]) begin
            //1 if given Data Processing instruction is "Immediate" type
            data_instr_type = 3'b001;
        end

        else if(!instruction[25] && !instruction[4]) begin
        //2 if it's "Register shifted by value" type
            data_instr_type = 3'b010;
        end

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

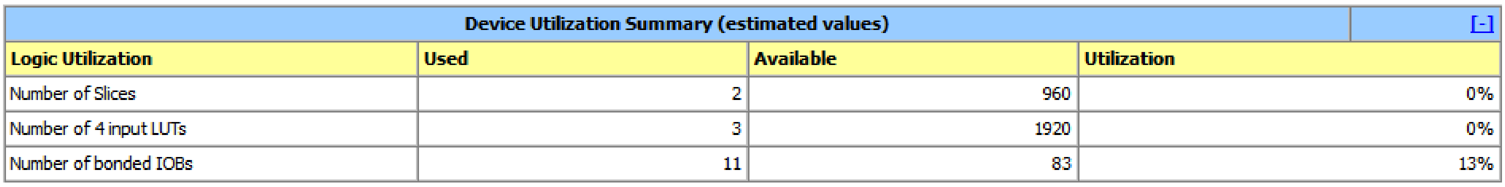


```
            data_instr_type = 3'b011;
        end

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

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

    end

endmodule
```



**HW4_instr_Data Project Status**

| | | | |
|---|---|---|---|
| **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:** | 3 Warnings (0 new) |
| **Design Goal:** | Balanced | • **Routing Results:** | |
| **Design Strategy:** | Xilinx Default (unlocked) | • **Timing Constraints:** | |
| **Environment:** | System Settings | • **Final Timing Score:** | |



**Device Utilization Summary (estimated values)** [-]

| **Logic Utilization** | **Used** | **Available** | **Utilization** |
|---|---|---|---|
| Number of Slices | 2 | 960 | 0% |
| Number of 4 input LUTs | 3 | 1920 | 0% |
| Number of bonded IOBs | 11 | 83 | 13% |

The code synthesized successfully with no errors, but 3 warnings. Which were due to the fact that we weren't able to implement <instruction <31:28>>, <25:5> and <3:0> because of the requirements.

## Comparison

The assignment could be done with case statements, which I think, would be more efficient, but for this application, it wasn't needed

## Conclusion

In this assignment we got familiar with Git source control, Git Bash, Git GUI and Github. The code is uploaded there with commits and merges.

On the other hand, I got introduced with an ARM programming in Verilog.