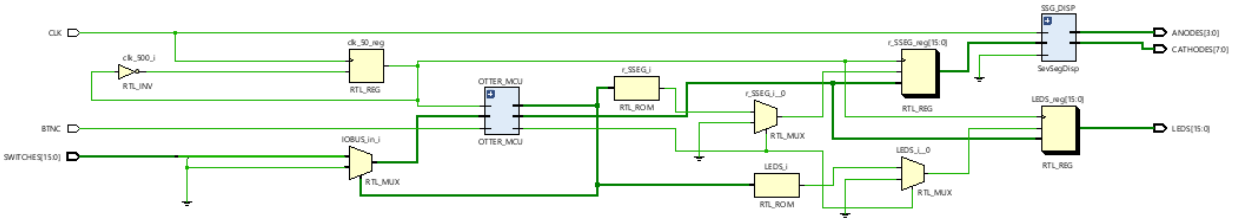
CPE 233 HW 7

Wyatt Tack

1. Behavior Description: The OTTER Wrapper functions as the device that connects the OTTER MCU to the various In and Out that wished to be read or written to for the program to work. For example, it's the wrapper's job to funnel in all the drivers to and from external inputs (such as the switches on the basys 3 board) to the IOBUS_IN input on the MCU, to be read as memory. The same applies for the outputs, in which if data is wished to be written to the 7 segment display on the board, then the wrapper is responsible for connecting the IOBUS_OUT to the specific seven segment display driver and leds, along with the address select and write enable.

2. Structural Design for whole OTTER:



3. Synthesis Warnings Listing:

4. Verification:

Test File uploaded to board and to be shown to Professor in class

5. System Verilog Source Code:

For Control Unit Decoder:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Company: Cal Poly SLO
// Engineer: Wyatt Tack
//
// Create Date: 02/01/2024
// Design Name: Control Unit Decoder
// Project Name: OTTER MCU
// Target Devices: Basys 3 Board
// Description: Sends non timing dependent selection signals
//              for data manipulation in otter MCU
//
//////////////////////////////////////////////////////////////////

module CU_DCDR(
input [31:0] ir,
input br_eq, br_lt, br_ltu,
//input int_taken
output logic [3:0] ALU_FUN,
output logic [1:0] srcA_SEL,
output logic [2:0] srcB_SEL,
output logic [2:0] PC_SEL,
output logic [1:0] RF_SEL
    );
always_comb begin
    ALU_FUN = 0;
    srcA_SEL = 0;
    srcB_SEL = 0;
    PC_SEL = 0;
    RF_SEL = 0;
    case (ir[6:0]) //op-codes
    7'b0110011: //All R-Type
    begin
        ALU_FUN = {ir[30], ir[14:12]};
        srcA_SEL = 0;
        srcB_SEL = 0;
        PC_SEL = 0;
        RF_SEL = 3;
    end
    7'b0010011: //1st set of I-Type Instructions
    begin
        if (ir[14:12] == 3'b101) ALU_FUN = {ir[30], ir[14:12]};
        else ALU_FUN = {1'b0, ir[14:12]};
        srcA_SEL = 0;
        srcB_SEL = 1;
        PC_SEL = 0;
        RF_SEL = 3;
    end
    7'b0000011: //2nd set of I-Type Instructions
    begin
        ALU_FUN = 4'b0000;
        srcA_SEL = 0;
        srcB_SEL = 1;
        PC_SEL = 0;
        RF_SEL = 2;
    end
    7'b1100111: //Last set of I-Type (for jalr)
    begin
        PC_SEL = 1;
        RF_SEL = 0;
    end
    7'b0100011: //All S-Type Instructions
    begin
        ALU_FUN = 4'b0000;
        srcA_SEL = 0;
        srcB_SEL = 2;
        PC_SEL = 0;
    end
    7'b1100011: //All B-Type Instructions (include conditions)
    begin
        case(ir[14:13])
        2'b00: //equal
        begin
        PC_SEL = {1'b0,(ir[12]^br_eq),1'b0};
        end
        2'b10: //less than
        begin
        PC_SEL = {1'b0,(ir[12]^br_lt),1'b0};
        end
        2'b11: //less than unsigned
        begin
        PC_SEL = {1'b0,(ir[12]^br_ltu),1'b0};
        end
        default:
        begin
        PC_SEL = 0;
        end
        endcase
    end
    7'b0110111: //1st set of U-Type (lui)
    begin
        ALU_FUN = 4'b1001;
        srcA_SEL = 1;
        PC_SEL = 0;
        RF_SEL = 3;
    end
    7'b0010111: //2nd set of U-Type (auipc)
    begin     //auipc**********************************auipc
        ALU_FUN = 4'b0000;
        srcA_SEL = 1;
        srcB_SEL = 3;
        PC_SEL = 0;
        RF_SEL = 3;
    end
    7'b1101111: //All J-Type instructions (jal)
    begin
        PC_SEL = 3;
        RF_SEL = 0;
    end
    default:
    begin
    ALU_FUN = 0;
    srcA_SEL = 0;
    srcB_SEL = 0;
    PC_SEL = 0;
    RF_SEL = 0;
    end
    endcase
end
endmodule
```

For Control Unit FSM:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////
// Company: Cal Poly SLO
// Engineer: Wyatt Tack
//
// Create Date: 02/01/2024
// Design Name: Control Unit FSM
// Project Name: OTTER MCU
// Target Devices: Basys 3 Board
// Description: Sends selection signals for timing specific
//              operations in memory of OTTER MCU
//
//////////////////////////////////////////////////////////////


module CU_FSM(
input RST, clk,
//input INTR,
input [31:0] ir,
output logic PC_WE, RF_WE, memWE2, memRDEN1, memRDEN2, reset
//output logic csr_WE, int_taken, mret_exec
    );
typedef enum {ST_INIT, ST_FETCH, ST_EXEC, ST_WRITE} state_type;
state_type NS, PS;
always_ff@(posedge clk) begin //state register
    if(RST == 1) PS<=ST_INIT;
    else PS<=NS;
end
always_comb begin //input/output logic
    PC_WE = 0;
    RF_WE = 0;
    memWE2 = 0;
    memRDEN1 = 0;
    memRDEN2 = 0;
    reset = 0;
//  csr_WE = 0;
//  int_taken = 0;
//  mret_exec = 0;
case(PS)
    ST_INIT: begin
    reset = 1'b1;
    NS = ST_FETCH;
    end
    ST_FETCH: begin
    memRDEN1 = 1'b1;
    NS = ST_EXEC;
    end
    ST_EXEC: begin
        NS = ST_FETCH;
        case (ir[6:0]) //op-codes
        7'b0110011: //All R-Type
        begin
            PC_WE = 1;
            RF_WE = 1;
        end
        7'b0010011: //1st set of I-Type Instructions
        begin
            PC_WE = 1;
            RF_WE = 1;
        end
        7'b0000011: //2nd set of I-Type Instructions
        begin
            NS = ST_WRITE;
            memRDEN2 = 1;
        end
        7'b1100111: //Last set of I-Type (for jalr)
        begin
            PC_WE = 1;
            RF_WE = 1;
        end
        7'b0100011: //All S-Type Instructions
        begin
            PC_WE = 1;
            memWE2 = 1;
        end
        7'b1100011: //All B-Type Instructions (include conditions)
        begin
            PC_WE = 1;
        end
        7'b0110111: //1st set of U-Type (lui)
        begin
            PC_WE = 1;
            RF_WE = 1;
        end
        7'b0010111: //2nd set of U-Type (auipc)
        begin
            PC_WE = 1;
            RF_WE = 1;
        end
        7'b1101111: //All J-Type instructions (jal)
        begin
            PC_WE = 1;
            RF_WE = 1;
        end
        default:
        begin

        end
        endcase
    end
    ST_WRITE: begin
        NS = ST_FETCH;
        PC_WE = 1;
        RF_WE = 1;
    end
    default: begin
    NS = ST_INIT;
    end
endcase
end
endmodule
```