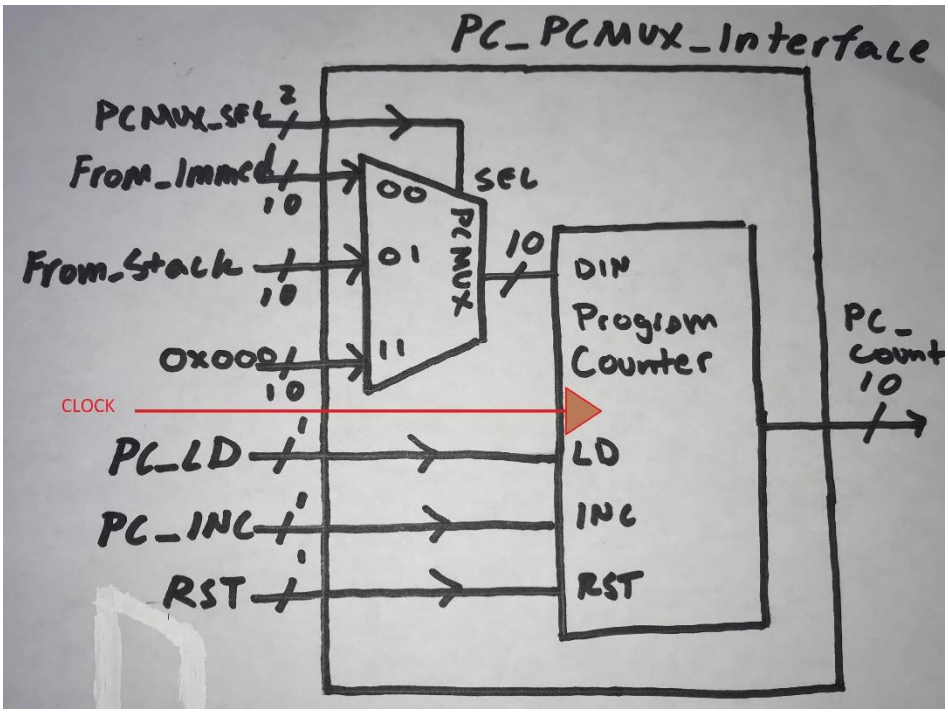
CPE 233: RAT assignment 2

Prof. Bridget Benson

Luis Gomez, Jared Rocha

BBD



Behavior

In this assignment, we built 3 modules in System Verilog to support a Program Counter. The module behaviors can be described as following:

Program Counter (PC): A synchronous 10-bit counter, that provides the user with the ability to load a 10-bit value into its registers, the ability to increment the stored value according to a rising clock edge signal, and to reset the registers to 0. The counter outputs the stored 10-bit signal for use in the RAT computer we are building in this class.

PC Multiplexer: A 4:2 multiplexer, currently only utilizing ¾ input signals. The output signal of the PC Mux is a 10-but value called DIN, which is determined by a 2-bit selector signal. The PC Mux input signals are 10-bit values that provides input values from the following sources:

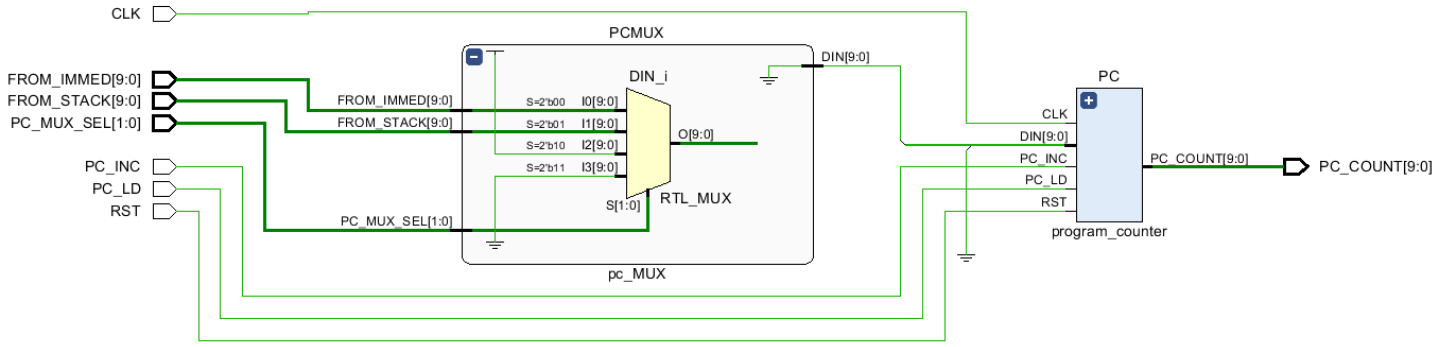
FROM_IMMED- 10-bit value provided explicitly in the Assembly instruction. (SEL = 0'b00)

FROM_STACK- 10-bit value provided from the SCR module (to be completed later) (SEL = 0'b01)

Constant value Zero- currently in place because we wanted to account for unused selector value (SEL = 0'b1X)

PC to PC MUX interface: This module provides an interface between PC MUX and the Program Counter. It provides no new input or output, but it does obscure the DIN signal as an internal logic signal. It effectively behaves like the Black Box found above.

Structural Design



Verification

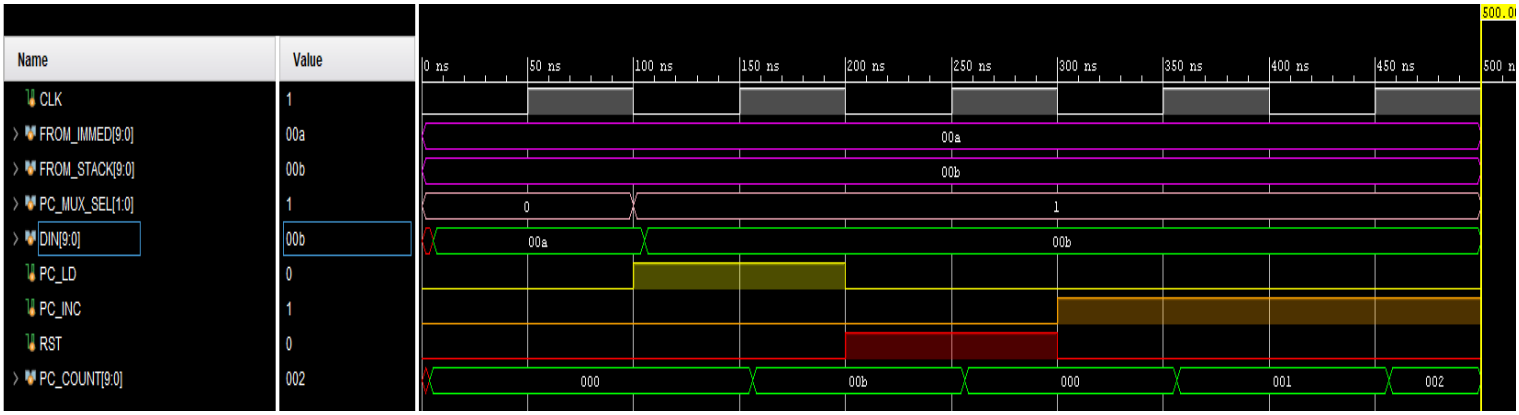
To verify the behavior of our Program Counter and PC MUX, we designed four test cases to demonstrate the functionality of the modules.

Test 1: We simulate unique values FROM_IMMED and FROM_STACK, and disable PC_LD, INC, and RST. PC_MUX_SEL is set to 0'b01, which outputs FROM_IMMED to DIN. We began this test with a simple test of the ability of the mux to output the selected signal while demonstrating that a disabled counter will not load DIN or increment its stored value.

Test 2: We now set PC_MUX_SEL to 0'b01, outputting FROM_STACK to DIN. We also set PC_LD = 1, so the counter loads the new value output from DIN to its registers. We did this test to again show that the mux can switch between input signals according to its selector input. Furthermore, we wanted to demonstrate that the counter loads the input at the correct edge trigger but does not increment nor reset.

Test 3: We now disable PC_LD and enable RST. We did this to demonstrate that the counter can switch between modes and reset its stored value to zero.

Test 4: We now disable RST and enable INC (for several clock cycles), to show that the counter can increment its stored value.



TESTBENCH

```
module PC_and_PCMUX_tb(); // Normally we would define input/output inside here,
instead our inputs/outputs are internal logic signals found below
    // clock signal
    logic CLK;
    // PC_MUX interface
    logic [9:0] FROM_IMMED, FROM_STACK;
    logic [1:0] PC_MUX_SEL;
    // Program Counter interface
    logic PC_LD, PC_INC, RST;
    logic [9:0] PC_COUNT;
    logic [9:0] DIN;

    PC_PCMUX_interface DUT_interface(.*);

    always // Sim Clock signal
    begin
        CLK = 0; #50; // 5 nanosecs
        CLK = 1; #50;
    end

    initial begin
        // Test 1: NO LOAD, DIN = FROM_IMMED
        FROM_IMMED = 10'hA; FROM_STACK = 10'hB;
        PC_MUX_SEL = 0; PC_LD = 0; PC_INC = 0; RST = 0;
        #100; // Test 2: : LOAD, DIN = FROM_STACK
        PC_LD = 1; PC_MUX_SEL = 1;
        #100; // Test 3: RESET
        RST = 1; PC_LD = 0;
        #100; // Test 4: : INCREMENT
        PC_INC = 1; RST = 0;
        #200;
        $finish;
    end
endmodule
```

Source Code

PC MUX INTERFACE

```
module PC_PCMUX_interface(
    // clock signal
    input CLK,
    // PC_MUX interface
    input [9:0] FROM_IMMED, FROM_STACK,
    input [1:0] PC_MUX_SEL,
    // output logic [9:0] DIN
    // Program Counter interface
    input PC_LD, PC_INC, RST,
    output logic [9:0] PC_COUNT
    // output logic [9:0] DIN // ENABLE DURING TEST BENCH TO SEE SIGNAL
    );
    logic [9:0] DIN;

    pc_MUX PCMUX(FROM_IMMED, FROM_STACK, PC_MUX_SEL, DIN);
    program_counter PC(DIN, PC_LD, PC_INC, RST, CLK, PC_COUNT);

endmodule
```

PROGRAM COUNTER

```
module program_counter(
    input [9:0] DIN,
    input PC_LD, PC_INC, RST, CLK,
    output logic [9:0] PC_COUNT
    );

    always_ff @ (posedge CLK) begin
        if (RST == 1) begin // RESET
           PC_COUNT <= 0;
        end
        else if (PC_LD == 1) begin // LOAD
            PC_COUNT <= DIN;
        end
        else if (PC_INC == 1) begin // INCREMENT
            case(PC_COUNT)
                10'h3FF: begin
                         PC_COUNT <= 0; // OVERFLOW
                         end
               default: begin
                         PC_COUNT++;
                         end
            endcase
        end
    end
endmodule
```

PC MUX

```
module pc_MUX(
    input [9:0] FROM_IMMED, FROM_STACK,
    input [1:0] PC_MUX_SEL,
    output logic [9:0] DIN = 0
    );

    parameter [9:0] SIZE = 10'h3FF;

    always_comb begin
        case (PC_MUX_SEL)
            2'b00:  begin
                    DIN <= FROM_IMMED;
                    end
            2'b01:  begin
                    DIN <= FROM_STACK;
                    end
            2'b10:  begin
                    DIN <= 10'h3FF;
                    end
            2'b11:  begin
                    DIN <= 0;
                    end
        endcase
    end
endmodule
```