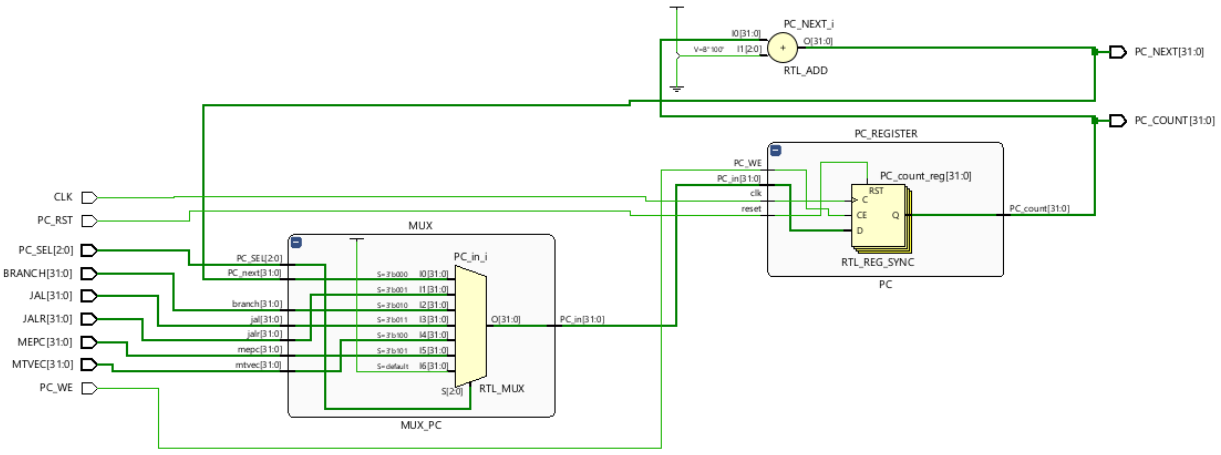
CPE 233 HW 2

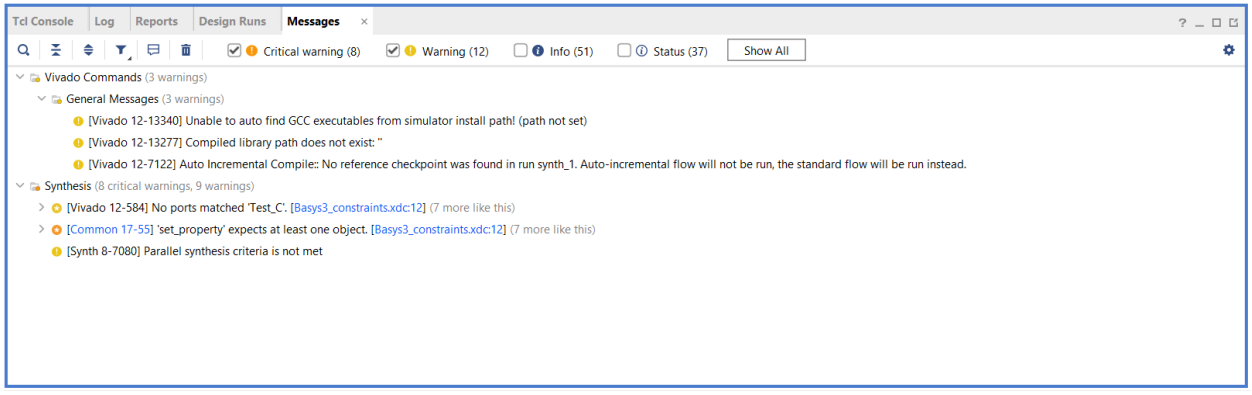
Wyatt Tack

1. Behavior Description: The following design is used to select the specific address in the memory of the Program Memory as to which line of machine code stored will be executed. The usual behavior of the device is to increment by 4 each clock cycle, outputting the next address of the next “word” of code. For other linking/branching commands, a multiplexor is added to the input of the register that stores the current memory address of the line being executed, as some commands would stop the register from outputting the next line (being +4), and outputs a separate word.

2. Structural Design:



3. Synthesis Warnings Listing:



4. Verification:

Test Case Input	Test Case Output	Test Case Reasoning
PC_SEL = 0	PC_COUNT increments: 0x0000_0000, 0x0000_0004	Standard operation case to make sure counter can count by 4 under selected condition
PC_SEL = 1 JALR = 0xa123a123	PC_COUNT = 0xa123a123	Makes sure JALR is outputted when MUX is selected to JALR
PC_SEL = 1 JALR = 0x321a321a	PC_COUNT = 0x321a321a	Makes sure when JALR is changed and selected register updates on CLK
PC_SEL = 2 BRANCH = 0xb234b234	PC_COUNT = 0xb234b234	Makes sure BRANCH is outputted when MUX is selected to BRANCH
PC_SEL = 3 JAL = 0xc345c345	PC_COUNT = 0xc345c345	Makes sure JAL is outputted when MUX is selected to JAL
PC_SEL = 4 MTVEC = 0xd456d456	PC_COUNT = 0xd456d456	Makes sure MTVEC is outputted when MUX is selected to MTVEC
PC_SEL = 5 MEPC = 0xf567f567	PC_COUNT = 0xf567f567	Makes sure MEPC is outputted when MUX is selected to MEPC
PC_SEL = 6 And PC_SEL = 7	PC_COUNT = 0xffffffff	Tests 2 unused MUX selector cases, all high output expected as default
PC_RST = 0	PC_COUNT = 0x00000000	Tests Reset input
PC_SEL = 1 JAL = 0xfffffff8 (next CLK) PC_SEL = 0	PC_COUNT increments: 0xfffffff8, 0xfffffffc, 0x00000000	Tests overflow adding
PC_WE = 0 PC_SEL = 1, 6 (Previous test case stopped at PC_COUNT =	PC_COUNT =	Tests turning off enable, should stop counting and constant output for different MUX SELECTIONS
PC_RST = 1	PC_COUNT = 0x00000000	Tests that reset works when enable is off (register is “paused”)

Simulation:

5. System Verilog Source Code:

For PC_REG:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////
// Company: Cal Poly SLO
// Engineer: Wyatt Tack
//
// Create Date: 01/16/2024
// Design Name: PC Register
// Project Name: OTTER MCU
// Target Devices: Basys 3 Board
// Description: 32 Bit Register for Program Counter
//              on OTTER MCU
//
//////////////////////////////////////////////////////////////
module PC(
input[31:0] PC_in,
input reset,
input PC_WE,
input clk,
output logic[31:0] PC_count
    );
always_ff@(posedge clk)
//at clock edge if enable, flip register, if reset, reset register
    begin
    if (reset == 1)
        PC_count <=0;
    else if (PC_WE == 1)
        PC_count <=PC_in;
    end
endmodule
```

For Multiplexor:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Company: Cal Poly SLO
// Engineer: Wyatt Tack
//
// Create Date: 01/16/2024
// Design Name: PC Multiplexor
// Project Name: OTTER MCU
// Target Devices: Basys 3 Board
// Description: MUX for program adress for PC
//              on OTTER MCU
//
//////////////////////////////////////////////////////////////////

module MUX_PC(
input [2:0] PC_SEL, //initialize select, output, and inputs
input [31:0] PC_next, jalr, branch, jal, mtvec, mepc,
output logic [31:0] PC_in
    );
    always_comb
    begin
    case(PC_SEL) //select on multiplexor for inputs
        0: PC_in = PC_next;
        1: PC_in = jalr;
        2: PC_in = branch;
        3: PC_in = jal;
        4: PC_in = mtvec;
        5: PC_in = mepc;
        default: PC_in = 32'hffffffff;
    endcase
    end
endmodule
```

Top Module:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Company: Cal Poly SLO
// Engineer: Wyatt Tack
//
// Create Date: 01/16/2024
// Design Name: HW_2
// Project Name: OTTER MCU
// Target Devices: Basys 3 Board
// Description: Hardware assignment #2. Used to show
//              working program counter
//
//////////////////////////////////////////////////////////////////

module HW_2(
input PC_RST, PC_WE, CLK, //initialize all inputs and outputs
input [31:0] JALR, BRANCH, JAL, MTVEC, MEPC,
input [2:0] PC_SEL,
output logic [31:0] PC_COUNT, PC_NEXT
    );
logic [31:0] PC_DIN;
assign PC_NEXT = PC_COUNT + 4;  //add 4 to PC_Count
//Link up lower level modules
MUX_PC MUX (.PC_SEL(PC_SEL), .PC_next(PC_NEXT), .jalr(JALR),
          .branch(BRANCH), .jal(JAL), .mtvec(MTVEC),
          .mepc(MEPC), .PC_in(PC_DIN));
PC PC_REGISTER (.PC_in(PC_DIN), .reset(PC_RST), .PC_WE(PC_WE),
          .clk(CLK), .PC_count(PC_COUNT));
endmodule
```

Simulation Test Bench:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Company: Cal Poly SLO
// Engineer: Wyatt Tack
//
// Create Date: 01/16/2024
// Design Name: HW_2_TB
// Project Name: OTTER MCU
// Target Devices: Basys 3 Board
// Description: Hardware assignment #2 Test Bench. Used to show
//              working program counter in simulation
//
//////////////////////////////////////////////////////////////////

module HW_2_TB();
logic PC_RST, PC_WE, CLK = 0;
logic[31:0] JALR, BRANCH, JAL, MTVEC, MEPC;
logic [2:0] PC_SEL;
logic [31:0] PC_COUNT, PC_NEXT;

HW_2 UUT (.PC_RST(PC_RST), .PC_WE(PC_WE), .CLK(CLK), .JALR(JALR),
.BRANCH(BRANCH), .JAL(JAL), .MTVEC(MTVEC), .MEPC(MEPC), .PC_SEL(PC_SEL),
.PC_COUNT(PC_COUNT), .PC_NEXT(PC_NEXT));

always begin
#5
CLK = ~CLK;
end

always begin
PC_RST = 1; //reset counter to start program at 0
#13; //offset to clk
//set reset to 0, enable to 1, and unique adresses for multiplexor inputs
PC_RST = 0;
PC_WE = 1;
JALR = 32'ha123a123;
BRANCH = 32'hb234b234;
JAL = 32'hc345c345;
MTVEC = 32'hd456d456;
MEPC = 32'he567e567;
PC_SEL = 0;
#30 //A few clock cycles of PC incrementing
PC_SEL = 1; //select jalr
#10 JALR = 32'h321a321; //change jalr while selected
#10 PC_SEL = 2; //select branch
#10 PC_SEL = 3; //select jal
#10 PC_SEL = 4; //select mtvec
#10 PC_SEL = 5; //select mepc
#10 PC_SEL = 6; //select non option
#10 PC_SEL = 7; //select non option
#10 PC_RST = 1; //reset
#10 PC_RST = 0;
    PC_SEL = 1; //start counting at high end
    JALR = 32'hfffffff8;
#10 PC_SEL = 0;      //overflow count
#40 PC_WE = 0; //pause register
#10 PC_SEL = 1;
#10 PC_SEL = 6;
#10 PC_RST = 1;
#10;
end
endmodule
```