

Laboratory 8
CSE 3381
Combinational Building Blocks - Cadence Verilog

This experiment will show how combinational building blocks like a multiplexer and a decoder can be used to implement combinational logic functions. It will also increase your Verilog skills. You should print out this experiment before going to the lab so that you can obtain the lab instructor's initials where needed.

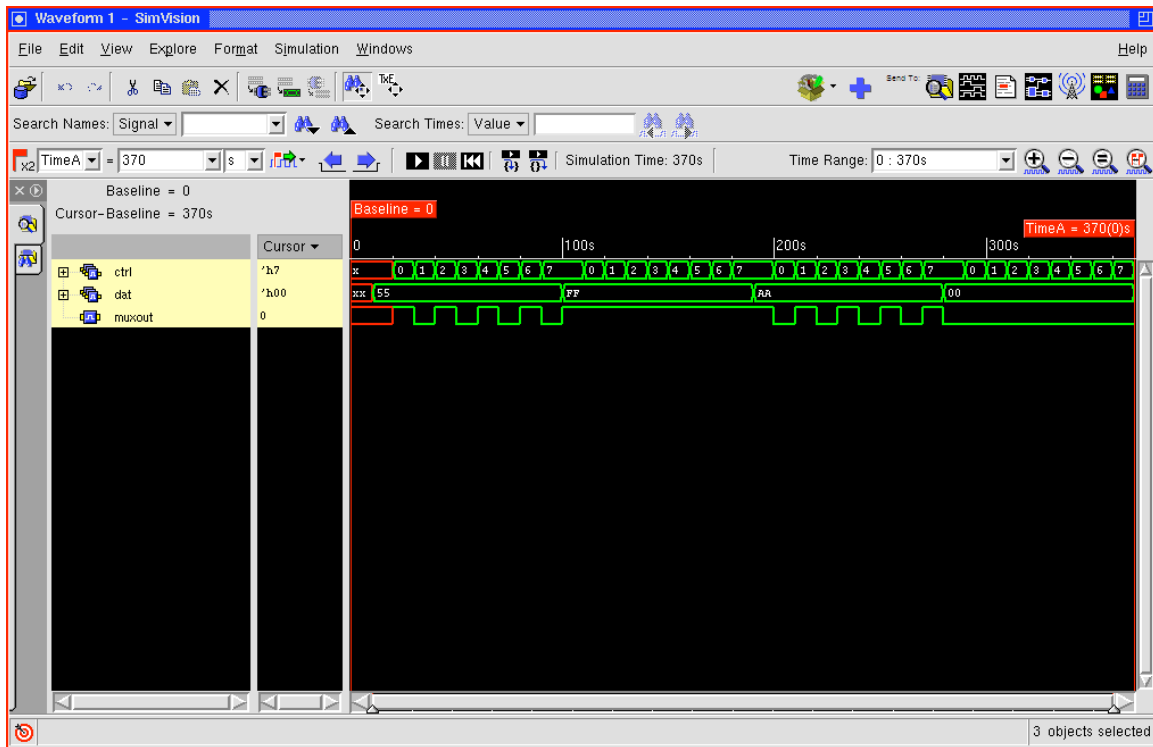
PART 1: Create a Verilog module named `mux81` that implements the function of an 8:1 multiplexer. You should implement your module using Boolean expressions and the `assign` keyword – do not use UDPs. Test your design with the testbench code `testmux.v`. A portion of the `testmux.v` file is shown below. The entire file can be copied from the class schedule webpage.

```
// Testbench for 8:1 MUX
module testmux;
    reg [7:0] dat;
    reg [2:0] ctrl;
    wire muxout;
    mux81 u1 (muxout, dat, ctrl);
    initial
        begin
            $stop;
            #10;
            dat = 8'h55;
            #10;
            ctrl = 3'o0;
            #10;
            ctrl = 3'o1;
            #10;
            ctrl = 3'o2;
            #10;
```

Additional lines of input vectors here

```
            ctrl = 3'o1;
            #10;
            ctrl = 3'o2;
            #10;
            ctrl = 3'o3;
            #10;
            ctrl = 3'o4;
            #10;
            ctrl = 3'o5;
            #10;
            ctrl = 3'o6;
            #10;
            ctrl = 3'o7;
            #10;
            $stop;
        end
endmodule
```

Your output waveform should match the following:



Once you have this window, get a screen capture of it (alt-prtscr in Windows) and save it to a file for your final report. Also, have the lab instructor look at it and initial below.

LAB INSTRUCTOR'S INITIALS: _____

PART 2: Create a Verilog module named `dec416` that implements the function of an 4:16 decoder. You should implement your module using Boolean expressions and the `assign` keyword – do not use UDPs. Test your design with the testbench code `testdec.v`. A portion of the `testdec.v` file is shown below. The entire file can be copied from the class schedule webpage.

```
// Testbench for 4:16 Decoder
module testdec;
  reg [3:0] enc;
  wire [15:0] dec;
  dec416 u1 (dec, enc);
  initial
  begin
    $stop;
    #10;
    enc = 4'h0;
    #10;
    enc = 4'h1;
    #10;
    enc = 4'h2;
```

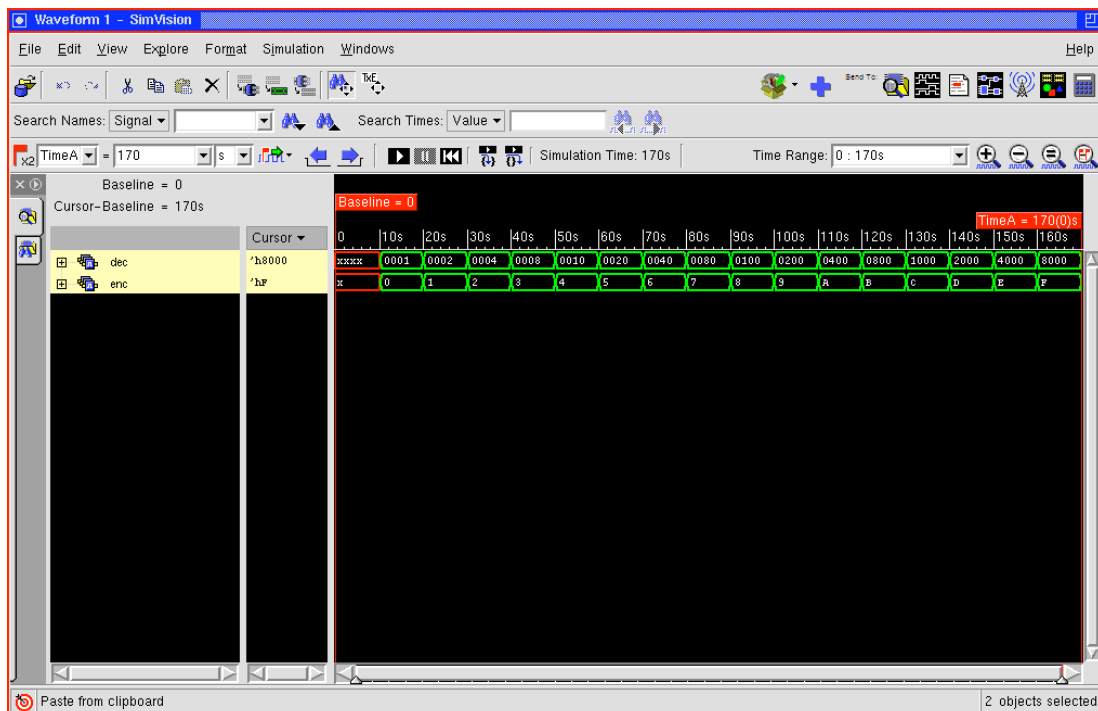
```

#10
enc = 4'h3;
#10
enc = 4'h4;
#10
enc = 4'h5;
#10
enc = 4'h6;
#10
enc = 4'h7;
#10
enc = 4'h8;
#10
enc = 4'h9;
#10
enc = 4'ha;
#10
enc = 4'hb;
#10
enc = 4'hc;
#10
enc = 4'hd;
#10
enc = 4'he;
#10
enc = 4'hf;
#10
$stop;

end
endmodule

```

Your output waveform should match the following:



Once you have this window, obtain a screen capture of it (alt-prtscr in Windows) and save it to a file for your final report. Also, ask the lab instructor look at it and initial below.

LAB INSTRUCTOR'S INITIALS: _____

PRELAB:

You are to build two 4-bit odd-prime number detector circuits. This is the combinational logic function given in canonical sum-of-minterms form as:

$$f = \sum(1,3,5,7,11,13)$$

In the first circuit, you are to use an 8:1 MUX and possibly some inverters. You should draw the circuit diagram and present it for PRELAB credit. In the second circuit, you are to implement it using 4:16 decoder and an OR gate. You should also draw this circuit and present it for PRELAB credit.

LAB INSTRUCTOR'S INITIALS (for MUX Circuit): _____

LAB INSTRUCTOR'S INITIALS (for Decoder Circuit): _____

PART 3: Create a Verilog module named `decprm` that instantiates your `dec416` module and also an `or` primitive that implements the prime number detector. **DO NOT** modify your `dec416` module. Use the testbench `testdecprm.v` which is shown below. The entire file can be copied from the class schedule webpage.

```
// Testbench for 4:16 Decoder
module testdecprm;
  reg [3:0] enc;
  wire encout;
  decprm u1 (encout, enc);
  initial
  begin
    $stop;
    #10;
    enc = 4'h0;
    #10;
    enc = 4'h1;
    #10;
    enc = 4'h2;
    #10;
    enc = 4'h3;
    #10;
    enc = 4'h4;
    #10;
    enc = 4'h5;
    #10;
    enc = 4'h6;
    #10;
    enc = 4'h7;
  end
endmodule
```


FINAL REPORT: Your final report is due the following class period and should include a discussion of any problems you had with the lab and how they were resolved. It should also contain the three waveform screen captures described above, a listing of your Verilog source code, and your design information (i.e. the maps, logic equations, circuits if drawn, etc.). It should also include the original hardcopy of this sheet with the lab instructors' initials and your PRELAB work.