Taylor Cowley

EE 220

May 26 2016

Lab 08: 4x7 Segment Controller

**Mod4 Verilog Code**

module mod4(out, incr, reset, clk);

input incr, reset, clk;

output[1:0] out;

reg[1:0] out\_ness = 2'b00;

always@(posedge clk)

begin

if(reset)

out\_ness = 2'b00;

else if(incr)

out\_ness = out\_ness + 2'b01;

else

out\_ness = out\_ness;

end

assign out = out\_ness;

endmodule

**Mod4 tcl file**

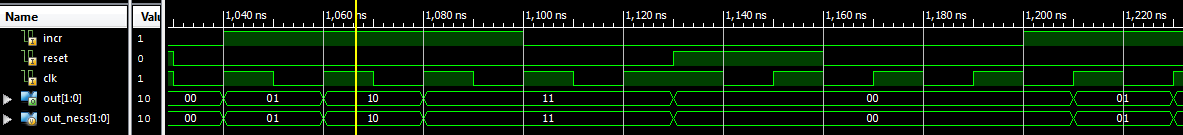
isim force add clk 1 -time 0 -value 0 -time 10ns -repeat 20ns

isim force add reset 1 -time 0 -value 0 -time 30ns

isim force add incr 0 -time 0 -value 1 -time 70ns -value 0 -time 250ns

run 300ns

**Mod4 Simulation waveform**



**Programmable Timer Verilog code**

module test\_timer(clk, out\_tp, out\_zero);

output out\_zero, out\_tp;

input clk;

prog\_timer yolo(clk, 0, 1, 24'd250000, counter, out\_zero, out\_tp);

endmodule

**Programmable Timer UCF file**

## clock pin for Nexys 2 Board

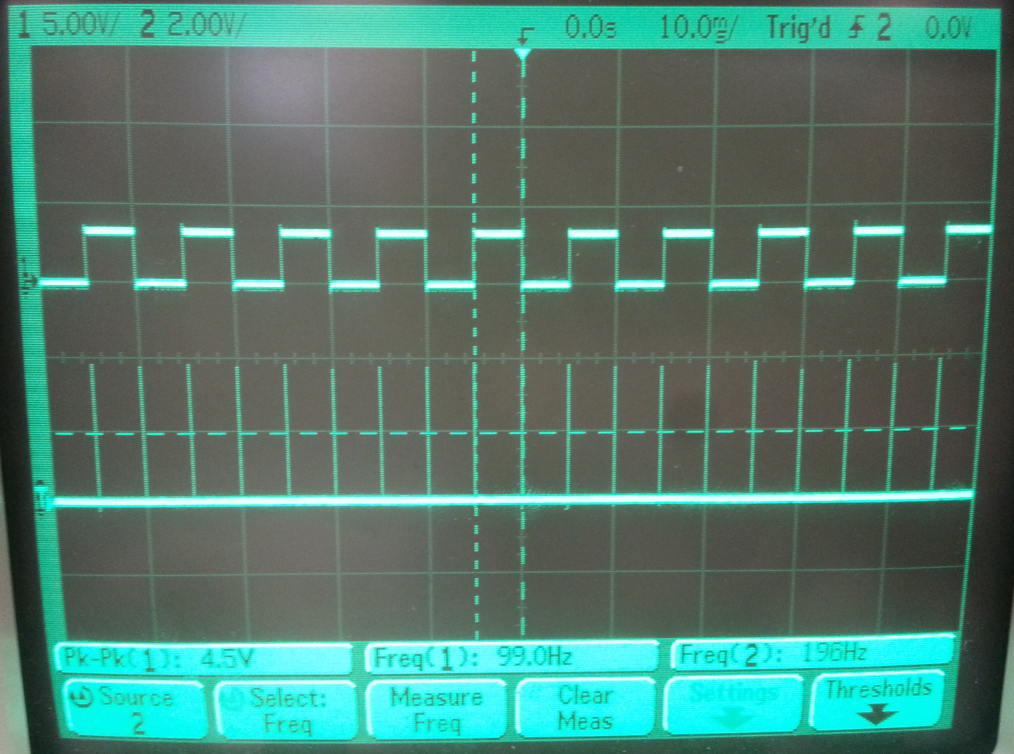
NET "clk" LOC = "B8";

## FX2 connector

NET "out\_tp" LOC = "B4";

NET "out\_zero" LOC = "A4";

**Programmable Timer Screenshot**



**4x7 Segment Verilog Code**

module seg4x7(seg, an, dp, dp\_in, in, clk, reset, tp);

input[15:0] in; // The raw data. full 16 bits

input clk;

input[3:0] dp\_in;

input reset;

output dp;

output [3:0] an;

output[6:0] seg;

output tp;

wire[3:0] seg\_data; // In between the 16-4 mux and the decoder

wire[1:0] an\_encoded;// Comes out of mod4

wire[3:0] an\_not\_inverted;

wire[3:0] dp\_inverted;

wire zero, tp; // Will be connected to mod4's incr signal

assign dp\_inverted = ~dp\_in;

prog\_timer timer(clk, reset, 1, 24'd250000, , zero, tp);

// produces an\_encoded from zero

mod4 mod(an\_encoded, zero, reset, clk);

// Gives us a one-hot version of an

decoder\_2\_4 decode(an\_encoded, an\_not\_inverted);

assign an = ~an\_not\_inverted;

mux\_16\_4 right\_an(seg\_data, an\_encoded, in[15:12], in[11:8], in[7:4], in[3:0]);

// Changes dp\_in (4-bit) to dp (1 bit) based on the an\_encoded

mux\_4 a\_mux(dp, an\_encoded, dp\_inverted);

// Adding our seven-seg decoder schematic

seven\_seg2 seven\_seg\_decoder(.a(seg[0]), .b(seg[1]), .c(seg[2]), .d(seg[3]), .e(seg[4]), .f(seg[5]), .g(seg[6]), .n0(seg\_data[0]), .n1(seg\_data[1]), .n2(seg\_data[2]), .n3(seg\_data[3]));

endmodule

**Testbench Verilog Code**

module test(seg, an, dp, Led, sw, clk, btn, tp);

output dp;

output[3:0] an;

output[6:0] seg;

output[7:0] Led;

input clk;

input btn;

input[7:0] sw;

output tp;

wire[15:0] go\_in;

assign Led = sw;

assign go\_in[15:12] = sw[7] ? 4'b1011 : 4'b0001;

assign go\_in[11:8] = sw[6] ? 4'b1010 : 4'b1010;

assign go\_in[7:4] = sw[5] ? 4'b1011 : 4'b1011;

assign go\_in[3:0] = sw[4] ? 4'b1110 : 4'b1000;

seg4x7 poop(seg, an, dp, sw[3:0], go\_in, clk, btn, tp);

endmodule

**Testbench UCF file**

## clock pin for Nexys 2 BoardNET "clk" LOC = "B8"; # Bank = 0, Pin name = IP\_L13P\_0/GCLK8, Type = GCLK, Sch name = GCLK0## Leds

NET "Led<0>" LOC = "J14";

NET "Led<1>" LOC = "J15";

NET "Led<2>" LOC = "K15";

NET "Led<3>" LOC = "K14";

NET "Led<4>" LOC = "E17";

NET "Led<5>" LOC = "P15";

NET "Led<6>" LOC = "F4";

NET "Led<7>" LOC = "R4";

## Switches

NET "sw<0>" LOC = "G18";

NET "sw<1>" LOC = "H18";

NET "sw<2>" LOC = "K18";

NET "sw<3>" LOC = "K17";

NET "sw<4>" LOC = "L14";

NET "sw<5>" LOC = "L13";

NET "sw<6>" LOC = "N17";

NET "sw<7>" LOC = "R17";

## Buttons

NET "btn" LOC = "B18";

#NET "btn<1>" LOC = "D18";

#NET "btn<2>" LOC = "E18";

#NET "btn<3>" LOC = "H13";

## 7 segment displayNET "seg<0>" LOC = "L18";

NET "seg<1>" LOC = "F18";

NET "seg<2>" LOC = "D17";

NET "seg<3>" LOC = "D16";

NET "seg<4>" LOC = "G14";

NET "seg<5>" LOC = "J17";

NET "seg<6>" LOC = "H14";

NET "dp" LOC = "C17";

NET "an<0>" LOC = "F17";

NET "an<1>" LOC = "H17";

NET "an<2>" LOC = "C18";

NET "an<3>" LOC = "F15";

NET "tp" LOC = "A4"; # Bank = 0, Pin name = IO\_L24P\_0, Type = I/O, Sch name = R-IO2

**Anomalies**

So I had my thing all coded up, but it was giving garbage values. Took me about 4.5 hours to figure out that I had things mixed up- the 2 bit an\_encoded variable and the 4-bit an variable. As well as needing to invert signals for the 7 segment display that I forgot to do.