Isai Mercado

Lab 9

Nov 12, 2014

Programmable timer calculations for 10Hz signal (2)

50MHz/10Hz = 5 million

SR latch Verilog module, simulation, and TCL file (4)

```
//////////////////////////////////////////////////
module SRlatch (OUTq, OUTqnot, INset, INreset);
output OUTq, OUTqnot;
input INset, INreset;
nor(OUTq, OUTqnot, INreset);
nor(OUTqnot, OUTq, INset);
endmodule
```

```
wave add / -radix hex

#define how the data input signals will behave when you run the simulation
#the command "isim force add (signal) 0 -time 0 -value 1 -time 20ns -repeat 40ns" means that (signal)
#will have a value of 0 from time 0, then change to 1 at time 20ns, and then repeat that cycle every 40ns
isim force add INset 0 -time 0 -value 1 -time 10ns -repeat 20ns
isim force add INreset 0 -time 0 -value 1 -time 20ns -repeat 40ns

#Nothing will change in the waveform viewer until you run the simulation for some period of time.
run 320ns
```

MOD6 Verilog module (4)

```
//////////////////////////////////////////////////////////////////////////////////
module MOD6(inc,rst,clk,curS);
    input inc, rst, clk;
    output[2:0] curS;
    wire[2:0] nxtS;

    // next state logic
    //assign nxtS = (curS == 2'd3)? 0 : curS + 2'd1;
    assign nxtS = (inc != 1'd1)? curS : (curS < 3'd6)? curS + 3'd1 : 3'd0;


    FF flip0(curS[0],clk,rst,nxtS[0]);
    FF flip1(curS[1],clk,rst,nxtS[1]);
    FF flip2(curS[2],clk,rst,nxtS[2]);

endmodule



module FF(q, clk, clr, d);
     input clk, clr, d;
     output reg q;

     always @(posedge clk)
         if (clr) q <= 0;
         else q <= d;
endmodule
```

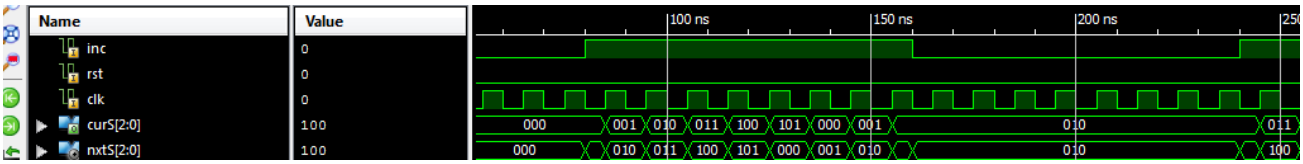
MOD6 TCL file (2)

```
File  Edit  Format  View  Help
 #add all signals to the waveform viewer
wave add / -radix hex

#define how the data input signals will behave when you run the simulation
#the command "isim force add (signal) 0 -time 0 -value 1 -time 20ns -repeat 40ns" means that (signal)
#will have a value of 0 from time 0, then change to 1 at time 20ns, and then repeat that cycle every 40ns
isim force add clk 0 -time 0 -value 1 -time 5ns -repeat 10ns
isim force add rst 1 -time 0 -value 0 -time 40ns
isim force add inc 0 -time 0 -value 1 -time 80ns -repeat 160ns


#Nothing will change in the waveform viewer until you run the simulation for some period of time.
run 320ns
```

MOD6 simulation waveform (3)



MOD10 Verilog module (4)

```
20  //////////////////////////////////////////////////////////////////////////////////
21  module MOD10(inc,rst,clk,curS);
22      input inc, rst, clk;
23      output[3:0] curS;
24      wire[3:0] nxtS;
25
26      // next state logic
27      //assign nxtS = (curS == 2'd3)? 0 : curS + 2'd1;
28      assign nxtS = (inc != 1'd1)? curS : (curS < 4'h9)? curS + 4'd1 : 4'd0;
29
30
31      FF flip0(curS[0],clk,rst,nxtS[0]);
32      FF flip1(curS[1],clk,rst,nxtS[1]);
33      FF flip2(curS[2],clk,rst,nxtS[2]);
34      FF flip3(curS[3],clk,rst,nxtS[3]);
35
36  endmodule
37
```

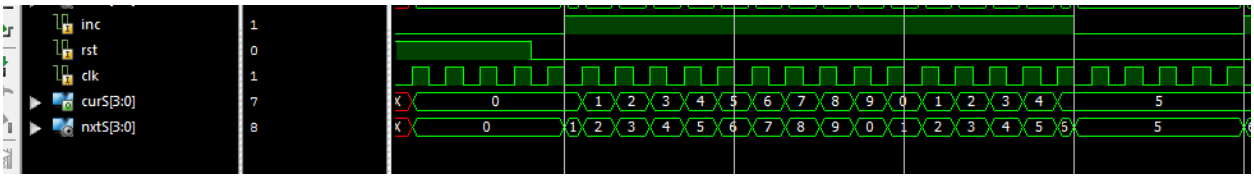
MOD10 TCL file (2)

```
File  Edit  Format  View  Help
 #add all signals to the waveform viewer
wave add / -radix hex

#define how the data input signals will behave when you run the simulation
#the command "isim force add (signal) 0 -time 0 -value 1 -time 20ns -repeat 40ns" means that (signal)
#will have a value of 0 from time 0, then change to 1 at time 20ns, and then repeat that cycle every 40ns
isim force add clk 0 -time 0 -value 1 -time 5ns -repeat 10ns
isim force add rst 1 -time 0 -value 0 -time 40ns
isim force add inc 0 -time 0 -value 1 -time 50ns -repeat 200ns

#Nothing will change in the waveform viewer until you run the simulation for some period of time.
run 320ns
```

MOD10 simulation waveform (4)



Counter block Verilog module (4)

```
20  //////////////////////////////////////////////////////////////////////////////////
21  module counterBlock(INinc,INrst,INclk,OUToneMin,OUTtenSec,OUToneSec,OUToneTenth);
22
23     input INinc,INrst,INclk;
24     output[3:0] OUToneMin,OUTtenSec,OUToneSec,OUToneTenth;
25     wire[2:0] modOut;
26     wire a,b,c;
27
28
29     MOD10 tenths (INinc,INrst,INclk,OUToneTenth); //MOD10(inc,rst,clk,curS);
30     assign a = (OUToneTenth == 4'b1001)? 1'b1 : 1'b0;
31
32
33     MOD10 oneSec (a,INrst,INclk,OUToneSec);
34     assign b = (OUToneSec == 4'b1001 && a == 1'b1)? 1'b1 : 1'b0;
35
36
37     MOD6  tenSec (b,INrst,INclk,modOut);
38     assign OUTtenSec = {1'b0,modOut};
39     assign c = (modOut == 3'b110 && b == 1'b1)? 1'b1 : 1'b0;
40
41
42     MOD10 oneMin (c,INrst,INclk,OUToneMin);
43
44  endmodule
45
```

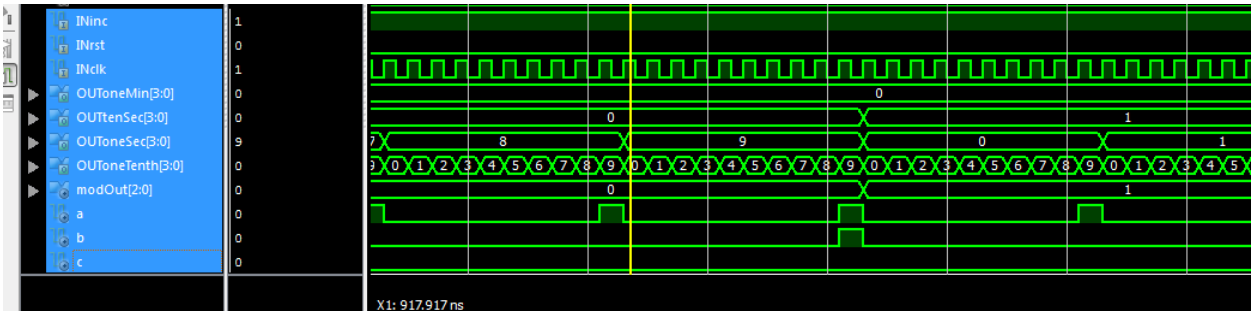
Counter block TCL file (2)

```
File  Edit  Format  View  Help
 #add all signals to the waveform viewer
wave add / -radix hex

#define how the data input signals will behave when you run the simulation
#the command "isim force add (signal) 0 -time 0 -value 1 -time 20ns -repeat 40ns" means that (signal)
#will have a value of 0 from time 0, then change to 1 at time 20ns, and then repeat that cycle every 40ns
isim force add INinc 1 -time 0 #-value 1 -time 5ns -repeat 10ns
isim force add INrst 1 -time 0 -value 0 -time 20ns
isim force add INclk 0 -time 0 -value 1 -time 5ns -repeat 10ns

#Nothing will change in the waveform viewer until you run the simulation for some period of time.
run 10000ns
```

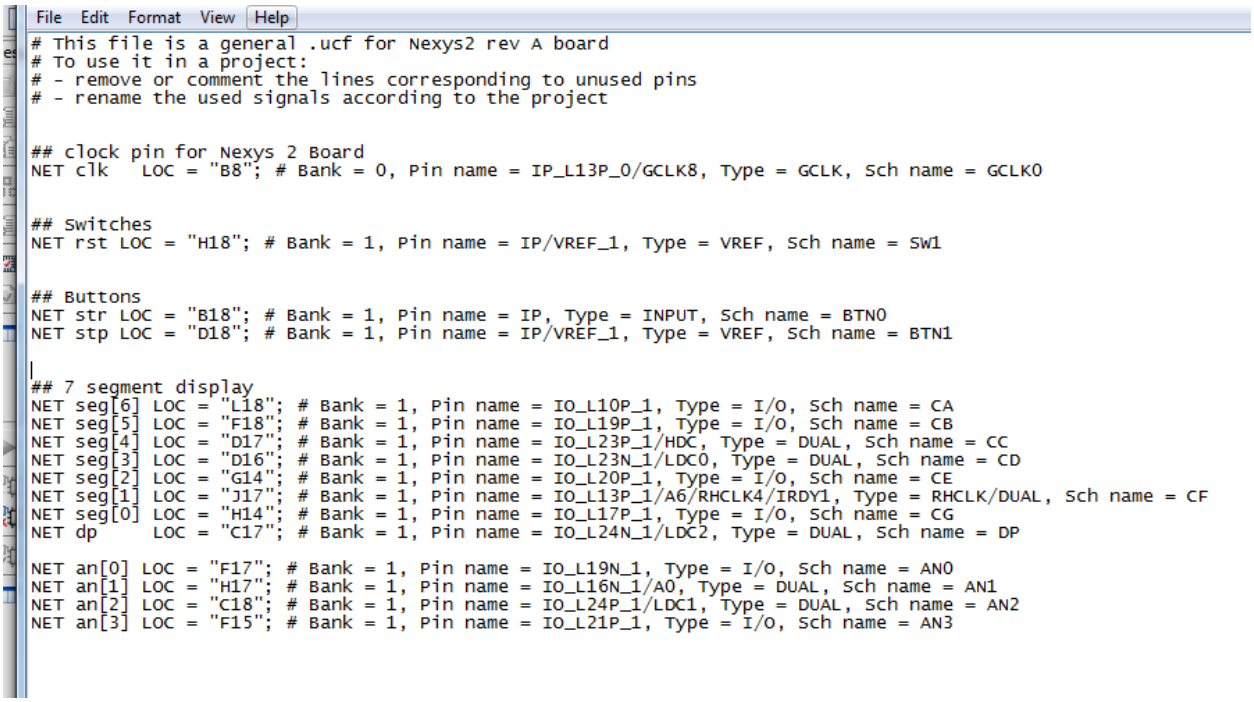
Counter block simulation waveform (4)



TestBench Verilog code (4)

```
20  //////////////////////////////////////////////////////////////////////////////////
21  module testBenchComplete( clk , seg , an , dp , rst , str , stp );
22
23      input clk , rst , str , stp;
24
25      output[6:0] seg;
26      output[3:0] an;
27      output dp;
28
29      wire[3:0] oneTenth , oneSec , tenSec , oneMin;
30      wire newClk, q , notq;
31
32
33      testBench fsm ( oneMin , tenSec , oneSec , oneTenth , 1'b1 , 1'b0 , 1'b1 , 1'b0 , clk , 0 , seg , an , , , dp ); //
34
35
36      counterBlock counter ( 1'b1 , rst , newClk , oneMin , tenSec , oneSec , oneTenth ); //counterBlock( INinc , INrst ,
37
38
39      SRlatch latch ( q , notq , str , stp ); //SRlatch (OUTq, OUTqnot, INset, INreset);
40
41
42      prog_timer timer ( clk , 0'b0 , q , 24'd4999999 , , newClk , ); // prog_timer (clk, reset, clken, load_number, cour
43                                          // prog_timer (clk_in,1'b0,1'b1,24'd250000, ,zero_out,tp_out);
44
45  endmodule
46
```

TestBench UCF file (1)



```
File  Edit  Format  View  Help
# This file is a general .ucf for Nexys2 rev A board
# To use it in a project:
# - remove or comment the lines corresponding to unused pins
# - rename the used signals according to the project


## clock pin for Nexys 2 Board
NET clk   LOC = "B8"; # Bank = 0, Pin name = IP_L13P_0/GCLK8, Type = GCLK, Sch name = GCLK0


## Switches
NET rst LOC = "H18"; # Bank = 1, Pin name = IP/VREF_1, Type = VREF, Sch name = SW1


## Buttons
NET str LOC = "B18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = BTN0
NET stp LOC = "D18"; # Bank = 1, Pin name = IP/VREF_1, Type = VREF, Sch name = BTN1

## 7 segment display
NET seg[6] LOC = "L18"; # Bank = 1, Pin name = IO_L10P_1, Type = I/O, Sch name = CA
NET seg[5] LOC = "F18"; # Bank = 1, Pin name = IO_L19P_1, Type = I/O, Sch name = CB
NET seg[4] LOC = "D17"; # Bank = 1, Pin name = IO_L23P_1/HDC, Type = DUAL, Sch name = CC
NET seg[3] LOC = "D16"; # Bank = 1, Pin name = IO_L23N_1/LDC0, Type = DUAL, Sch name = CD
NET seg[2] LOC = "G14"; # Bank = 1, Pin name = IO_L20P_1, Type = I/O, Sch name = CE
NET seg[1] LOC = "J17"; # Bank = 1, Pin name = IO_L13P_1/A6/RHCLK4/IRDY1, Type = RHCLK/DUAL, Sch name = CF
NET seg[0] LOC = "H14"; # Bank = 1, Pin name = IO_L17P_1, Type = I/O, Sch name = CG
NET dp     LOC = "C17"; # Bank = 1, Pin name = IO_L24N_1/LDC2, Type = DUAL, Sch name = DP

NET an[0] LOC = "F17"; # Bank = 1, Pin name = IO_L19N_1, Type = I/O, Sch name = AN0
NET an[1] LOC = "H17"; # Bank = 1, Pin name = IO_L16N_1/A0, Type = DUAL, Sch name = AN1
NET an[2] LOC = "C18"; # Bank = 1, Pin name = IO_L24P_1/LDC1, Type = DUAL, Sch name = AN2
NET an[3] LOC = "F15"; # Bank = 1, Pin name = IO_L21P_1, Type = I/O, Sch name = AN3
```

Anomalies:

Fun lab.. it feels good to see the timer working!!!... ta are very helpful