Scott Sargent and Ezra Davis
12/15/2013

# Lab #5 - Using Sequential Logic Circuits
## A Digital Stopwatch

Introduction:

The purpose of this lab was to program the Nexys FPGA board so that it could function as a stopwatch. It was to be designed so that it could count from 0 to 10 minutes with an accuracy of 0.1 seconds. The lab was to be completely implemented in Verilog and use the 4-digit to seven segment display driver as well.

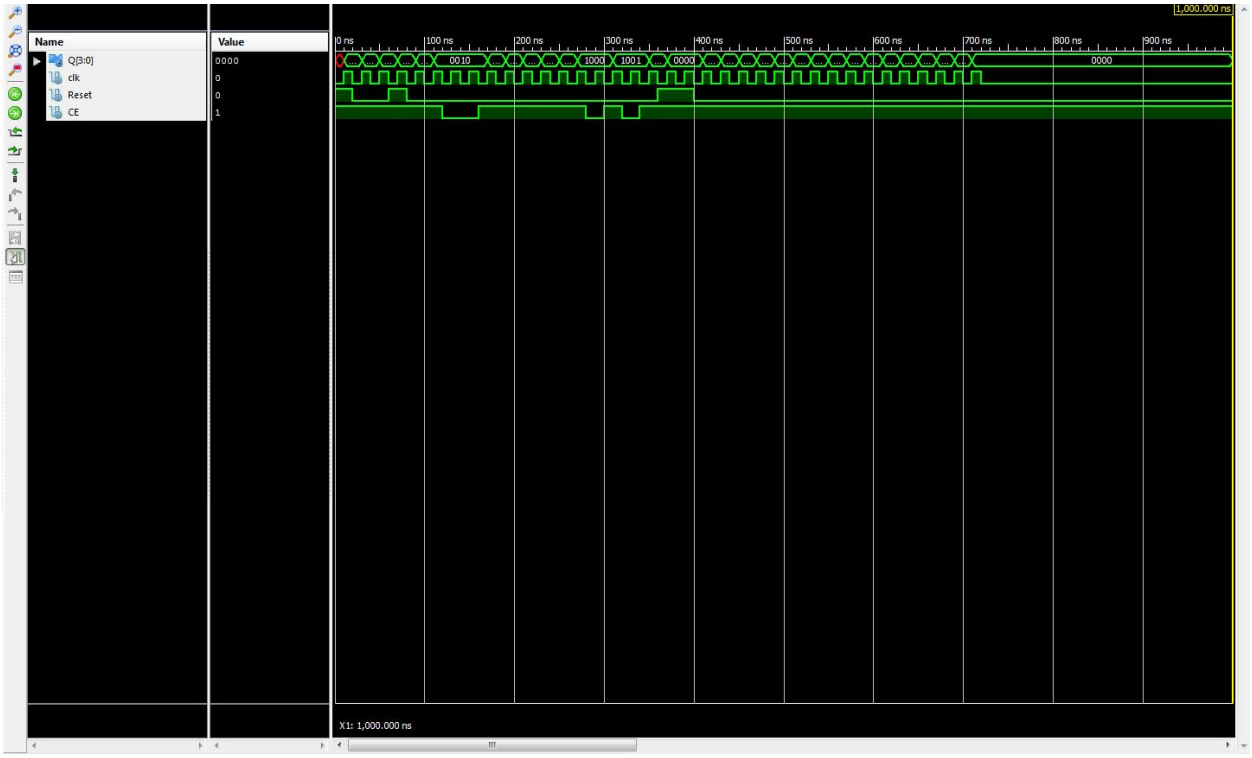
Discussion:

1 We added the clkDiv10K module from the class website to a new project in Xilinx. Using the clkDiv10K as a model, we created a module (clkDiv1K) that divides the input clock speed by 1000 (Code in Appendix B).

2 Next we created a tenHzFlash module that uses the system clock and the clock divider modules created above to flash the output once every tenth of a second (Appendix J).

3 The next step was to implement this module with an implementation constraints file (Appendix K) to test the clock divider. This was connected to the 100MHz clock built into the FPGA and one of the output LEDs. When programmed to the FPGA, it blinked at approximately the right speed.

4 We implemented a 4 bit binary counter using the model in the prelab. We then tested it using a Verilog testbench (Appendix H) and iSim's behavioral testing. After numerous corrections to the binary counter, the results looked like this:




5 Next we imported the fourDigitDisplay code and it's dependencies that were developed for lab 4 (with a few modifications, see Appendix E though Appendix G). The next step only uses the Hex2Seg module, but the rest will be used later.

6 The next module created was mod10cnt (Appendix C). It uses code provided in the lab description document and, using the 4 bit binary counter module (with reset), it counts from 0 to 9.

7 We copied the code for tenthsDisplay from the lab description document, and its corresponding .ucf file. We modified it to use our Hex2Seg (Appendix G) module. We then implemented, generated the programming file, and tested it on the FPGA. The tenthsDisplay module displays the digits 0 to 9 sequentially, once a second. It also has reset and chip enable inputs.

8 Then we created a new module called secondsDisplay with the code provided in the lab description document. It shows the number of seconds and tenths of seconds that have passed (0.0 to 9.9) while the chip was enabled since the the last time it was reset. We also added the constraints file from the lab description and implemented it on the board.

9 We created a mod6cnt module using mod10cnt as a guide (Code in Appendix D).

10 We then added a stopWatch module, made it our top module. It uses the code provided for it in the lab description document, customized to our implementation of the modulo 6 counter

and the fourDigitDisplay. Code in Appendix I. We also added an implementation constraints file. This was implemented onto the FPGA and worked as expected. The leftmost digit displays minutes, the next two display seconds, and the rightmost 7-segment display shows tenths of seconds.

Conclusion:

All the tests were successful. We learned how to use and implement binary counters, write sequential circuits in verilog, and got more practice displaying the results of a module to the FPGA's 7-segment displays.

Appendices:

Appendix A:

```
module clkDiv10K(
       input clk_in,
       output div_clk
       );
// This module divides down the input clock by 10000. That means
// if the input clock is the 100MHz Nexys 3 system clock then the
// output clock is a 10KHz clock.
// If the input clock to this module is 10KHz then the output
// clock is a 1Hz clock
 reg clk_out;
 reg [13:0] div_count;
 always @ (posedge clk_in)
  if (div_count < 5000)
   begin
    div_count <= div_count + 14'b1;
    clk_out <= 1'b0;
   end
  else if (div_count < 10000)
   begin
    div_count <= div_count + 14'b1;
    clk_out <= 1'b1;
   end
  else begin
   div_count <= 1'b0;
  end

 assign div_clk = clk_out;
endmodule
```

Appendix B:

```
module clkDiv1K(
    input clk_in,
```

```
    output div_clk
   );
   // This module divides down the input clock by 1000. That means
   // if the input clock is the 100MHz Nexys 3 system clock then the
   // output clock is a 100KHz clock.
   // If the input clock to this module is 10KHz then the output
   // clock is a 10Hz clock
    reg clk_out;
    reg [13:0] div_count;
    always @ (posedge clk_in)
    if (div_count < 14'd500)
     begin
      div_count <= div_count + 14'b1;
      clk_out <= 1'b0;
     end
    else if (div_count < 14'd1000)
     begin
      div_count <= div_count + 14'b1;
      clk_out <= 1'b1;
     end
    else begin
     div_count <= 1'b0;
    end
    assign div_clk = clk_out;
endmodule
```

## Appendix C:

```
module mod10cnt(
        input CLK,
        input R,
        input CE,
        output [3:0] Q,
        output R_out
        );
 assign R_out = (R | ((Q=='b1001) & CE)) ? 1'b1:1'b0;

 bin_cnt4 U1(CLK,R_out,CE,Q);
endmodule
```

## Appendix D:

```
module mod6cnt(
        input CLK,
        input R,
        input CE,
        output [3:0] Q,
        output R_out
        );
 assign R_out = (R | ((Q=='b0101) & CE)) ? 1'b1:1'b0;

 bin_cnt4 U1(CLK,R_out,CE,Q);
endmodule
```

## Appendix E:

```
module fourDigitDisplay(
       input [3:0] W,
       input [3:0] X,
       input [3:0] Y,
       input [3:0] Z,
       input [3:0] decPts,
       input [3:0] signs,
       input CLK,
       output [6:0] segs,
       output DP,
       output [3:0] anodes
       );

 wire clk_10K;
 wire [1:0] Q;
 wire [3:0] inputCurrent;
 wire decPt, sgn;// The currently displayed value of decPts and signs

 // Divide the system clock to 10KHz
 clkDiv10K U1(CLK, clk_10K);

  //Chooses which digit to display
 digitChooser U2(W,X, Y, Z, decPts, signs, Q, inputCurrent, decPt, sgn, anodes);

 Hex2Seg U3(inputCurrent, sgn, decPt, segs, DP);

 // --- State memory ---
 // 2-bit binary counter driving the 2-to-4 anode decoder
 //Reset is 0
 bin_cnt2 U4(clk_10K, 1'b0, Q);

endmodule
```

## Appendix F:

```
module digitChooser(
       input [3:0] W,
       input [3:0] X,
       input [3:0] Y,
       input [3:0] Z,
       input [3:0] decPts,
       input [3:0] signs,
  input [1:0] Q,
       output [3:0] D, //D = data
       output DP,
       output sign,
  output [3:0]anodes
       );
  //W is most significant digit
```

```
 //Q 00 corresponds to W also decPts[3] & signs[3] & anode == 4'b0111
 //  01 -> X
 //    anodes == 4'b1011
 //  10 -> Y
 //    anodes == 4'b1101
 //  11 -> Z
 //    anodes == 4'b1110
 mux4to1 U1(W, X, Y, Z, Q, D);
 mux1bit4to1 U2(decPts,Q,DP);

 mux1bit4to1 U3(signs, Q, sign);

 //decodeAnode U4(1'b1, ~Q[1], ~Q[0], anodes);
 assign anodes = (Q==2'b00) ? 4'b0111 : (
       (Q==2'b01) ? 4'b1011 : (
       (Q==2'b10) ? 4'b1101 : 4'b1110
       ));

endmodule
```

## Appendix G:

```
module Hex2Seg(
       input [3:0] D,
  input sgn,
  input decPt,
       output [6:0] segs,
  output DP
       );

 //All of these are active low

 assign segs =  (sgn==1'b1) ?7'b1111110 : (//Choosing to display just the negative sign
      (D==4'b0000)?7'b0000001 : ( //0
      (D==4'b0001)?7'b1001111 : ( //1
      (D==4'b0010)?7'b0010010 : ( //2
      (D==4'b0011)?7'b0000110 : ( //3
      (D==4'b0100)?7'b1001100 : ( //4
      (D==4'b0101)?7'b0100100 : ( //5
      (D==4'b0110)?7'b0100000 : ( //6
      (D==4'b0111)?7'b0001111 : ( //7
      (D==4'b1000)?7'b0000000 : ( //8
      (D==4'b1001)?7'b0001100 : ( //9
      (D==4'b1010)?7'b0001000 : ( //A
      (D==4'b1011)?7'b1100000 : ( //B
      (D==4'b1100)?7'b0110001 : ( //C
      (D==4'b1101)?7'b1000010 : ( //D
      (D==4'b1110)?7'b0110000 : 7'b0111000 //E F
      ))))))))))))));

 assign DP = (sgn==1'b1) ? 1'b1 : ~decPt; //Decimal point

endmodule
```

## Appendix H:

```
module bin_cnt4Test;

	// Inputs
	reg clk;
	reg Reset;
	reg CE;

	// Outputs
	wire [3:0] Q;

	// Itantiate the Unit Under Test (UUT)
	bin_cnt4 uut (
		.clk(clk),
		.Reset(Reset),
		.CE(CE),
		.Q(Q)
	);

	initial begin
		// Initialize Inputs
		clk = 0;
		Reset = 1;
		CE = 1;

		// Wait 10  for global reset to finish
		#10;clk = 1;#10;clk = 0;
		Reset = 0; //Reset = 0

		#10;clk = 1;#10;clk = 0;

		#10;clk = 1;#10;clk = 0;
		Reset = 1; //Reset = 1

		#10;clk = 1;
		#10;clk = 0;
		Reset = 0; //Reset = 0
//Clock cycle 4
		#10;clk = 1;
		#10;clk = 0;

		#10;clk = 1;
		#10;clk = 0;
		CE = 0; //CE = 0

		#10;clk = 1;
		#10;clk = 0;

		#10;clk = 1;
		#10;clk = 0;
		CE = 1; //CE = 1
//Clock cycle 8
		#10;clk = 1;
		#10;clk = 0;

		#10;clk = 1;
		#10;clk = 0;
```

```
                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;
//Clock cycle 12
                #10;clk = 1;
                #10;clk = 0;


                #10;clk = 1;
                #10;clk = 0;
                CE = 0; //CE = 0

                #10;clk = 1;
                #10;clk = 0;
                CE = 1;//CE = 1

                #10;clk = 1;
                #10;clk = 0;
                CE = 0; //CE = 0
//Clock cycle 16
                #10;clk = 1;
                #10;clk = 0;
                CE = 1;//CE = 1

                #10;clk = 1;
                #10;clk = 0;
                Reset = 1;

                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;
                Reset = 0;
//Clock cycle 20
                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;
//Clock cycle 24
                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
```

```
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;
//Clock cycle 28
                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;
//Clock cycle 32
                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;

                #10;clk = 1;
                #10;clk = 0;
        end

endmodule
```

## Appendix I:

```
module stopWatch(
    input CLK,
    input R,
    input CE,
    output [6:0] segs,
    output DP,
    output [3:0] anodes
    );

        wire clk_10K, clk_10Hz, R_out1, R_out2, R_out3, R_out4;
        wire [3:0] Qth, Qsec, Qsec10, Qmin;

        //Divide the system clock to 10KHz
        clkDiv10K U1(CLK,clk_10K);

        //Divide the 10KHz clock to 10Hz
        clkDiv1K U2(clk_10K,clk_10Hz);

        //Modulo 10 counter - for 10ths digit
        mod10cnt U3(clk_10Hz, R, CE, Qth, R_out1);

        // Modulo 10 counter - for seconds digit
        mod10cnt U4(clk_10Hz, R, R_out1, Qsec, R_out2);
```

```
        // Modulo 6 counter - for 10s of seconds digit
        mod6cnt U5(clk_10Hz, R, R_out2, Qsec10, R_out3);

        //Add modulo 10 counter for minutes (0-9)
        mod10cnt U6(clk_10Hz, R, R_out3, Qmin, R_out4);

        //Display output of 2 digit  counter with decimal point
        //Use the system CLK as the input clock and turn on the
        //     decimal point on the seconds digit
        fourDigitDisplay U7(Qmin, Qsec10, Qsec, Qth,4'b0010, 4'b0000, CLK, segs, DP,
anodes);
        //Argument order:
        //(inputs) W X Y Z decPts signs CLK (outputs) segs DP anodes


endmodule
```

## Appendix J:

```
module tenHzFlash(
    input CLK,
    output clk10Hz
    );
        wire tenKHz;

        clkDiv10K U1(CLK, tenKHz);

        clkDiv1K U2(tenKHz, clk10Hz);

endmodule
```

## Appendix K:

```
NET "CLK" LOC = "V10";
NET "clk10Hz" LOC = "V16";
```