Digital Design Lab #9:

Charlie Coleman
Lab Partner: Luke Taylor
2016 November 03

Objective: The objective of this lab is to design and analyze a synchronous counter, and gain experience building state diagrams.

Design:

Part One: The goal of part one was to derive a state table for a sequential circuit based off of a schematic, then to convert the state table into a state diagram and to recreate the schematic in Xilinx. After the counter was successfully created, the seven segment decoder from the previous labs was connected to the outputs of the counter and the circuit was tested on the board.

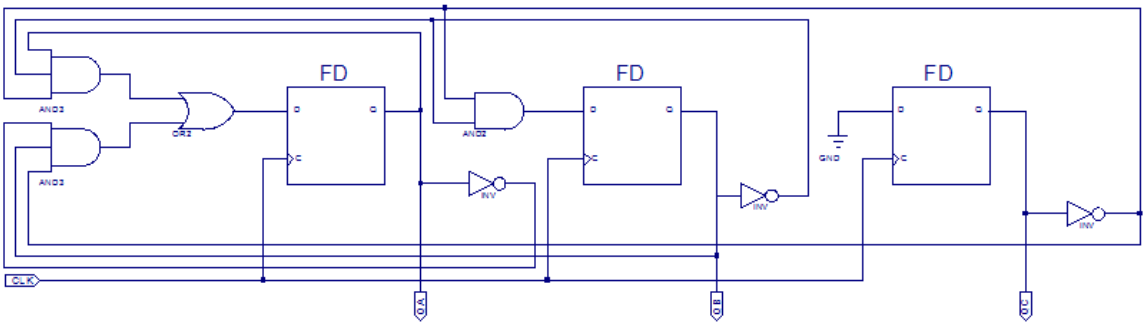
State Table:

Present State			Next State (t+1)					
Q2	Q1	Q0	D2	D1	D0	Q2	Q1	Q0
0	0	0	0	1	0	0	1	0
0	0	1	0	1	1	0	1	1
0	1	0	0	0	1	0	0	1
0	1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

State Diagram:

1XX → 000 → 010 → 001 → 011 → 000

Schematic:



Part Two: The goal of part two was to create a HDL code for a 4 bit odd counter that went up or down depending on an input. This code was tested with the seven segment decoder on the board aswell.

State Table:

Present State					Next State			
UD	Q3	Q2	Q1	QØ	Q3	Q2	Q1	QØ
Ø	Ø	Ø	Ø	1	1	1	1	1
Ø	Ø	Ø	1	1	Ø	Ø	Ø	1
Ø	Ø	1	Ø	1	Ø	Ø	1	1
Ø	Ø	1	1	1	Ø	1	Ø	1
Ø	1	Ø	Ø	1	Ø	1	1	1
Ø	1	Ø	1	1	1	Ø	Ø	1
Ø	1	1	Ø	1	1	Ø	1	1
Ø	1	1	1	1	1	1	Ø	1
1	Ø	Ø	Ø	1	Ø	Ø	1	1
1	Ø	Ø	1	1	Ø	1	Ø	1
1	Ø	1	Ø	1	Ø	1	1	1
1	Ø	1	1	1	1	Ø	Ø	1
1	1	Ø	Ø	1	1	Ø	1	1
1	1	Ø	1	1	1	1	Ø	1
1	1	1	Ø	1	1	1	1	1
1	1	1	1	1	Ø	Ø	Ø	1
X	X	X	X	Ø	Ø	Ø	Ø	1

Code:

```
entity UpDownCounter is
    Port ( CLK : in STD_LOGIC;
                      UpDown : in  STD_LOGIC;
                      Q : inout STD_LOGIC_VECTOR(3 downto Ø));
end UpDownCounter;
architecture Behavioral of UpDownCounter is
signal D : STD_LOGIC_VECTOR(3 downto Ø);
begin
       process(CLK)
       begin
              if(CLK' event and CLK = '1') then
                     case UpDown is
                            when 'Ø' =>
                                   case Q is
                                          when "ØØØ1" => D <= "1111";
                                          when "ØØ11" => D <= "ØØØ1";
                                          when "Ø1Ø1" => D <= "ØØ11";
                                          when "Ø111" => D <= "Ø1Ø1";
                                          when "1ØØ1" => D <= "Ø111";
                                          when "1Ø11" => D <= "1ØØ1";
                                          when "11Ø1" => D <= "1Ø11";
                                          when "1111" => D <= "11Ø1";
                                          when others => D <= "ØØØ1";
                                   end case;
                            when '1' =>
                                   case Q is
                                          when "ØØØ1" => D <= "ØØ11";
                                          when "ØØ11" => D <= "Ø1Ø1";
                                          when "Ø1Ø1" => D <= "Ø111";
                                          when "Ø111" => D <= "1ØØ1";
                                          when "1ØØ1" => D <= "1Ø11";
                                          when "1Ø11" => D <= "11Ø1";
                                          when "11Ø1" => D <= "1111";
                                          when "1111" => D <= "ØØØ1";
```

```
                                        when others => D <= "0001";
                                    end case;
                                when others => D <= "XXXX";
                            end case;
                            Q <= D;
                        end if;
                end process;
        end Behavioral;
```

Procedure: To begin the lab, the diagram below was analyzed and a state table and diagram were derived from it. The diagram was then created in Xilinx, compiled, and simulated. The seven segment decoder was then brought in and turned into a symbol. With the seven segment decoder symbol, the output of the flip-flops were connected to the lowest bits of the decoder. The seven segment decoder flip flop was then compiled, simulated, and downloaded to a board and tested. Next, a VHDL code was written for a self starting 4 bit sequence odd number counter. The code was compiled and simulated. The seven segment decoder was then brought in again and the outputs of the counter were assigned to the inputs of the 7 segment decoder using signals. This design was then synthesized and downloaded to the board and tested.

Data: See attached Xilinx simulation results.

Data Analysis: The simulation results for part one and two of the lab matched the state tables found in the design for both parts. Each design was also tested on the boards, and the designs performed as was expected, with the display showing that the output was incremented by two each time the clock went from low to high. In part one, the counter cycled from Ø up to 6 and then back to Ø. In part two, the counter cycled from Ø up to F when the up-down input was set to 1 and from F down to Ø when the up-down input was set to Ø. This matched the given design description in the lab manual and matched our state table.

Conclusion: The objective of this lab was to design sequential circuits using schematics and HDL code. These skills are important in designing more complex sequenctial circuits in the future and to be able to implement counters into designs to control when certain things are triggered. One difficulty in this lab was the derivation of the state table from the schematic given in part one of this lab. Many of the lines were overlapping, making reading the schematic very difficult. It helped to zoom in and follow a trace from its start to finish.