CpE Lab

Lab 9: Introduction to Sequential Logic-II

Prepared By:

Olujide Jacobs

Prepared For:

Matt Grubb

October 29, 2013

**Introduction**

The main purpose of this lab is to introduce students to sequential logic, a type of logic circuit whose output depends not only on the present input but also on the history of the input. This is in contrast to combinational logic, whose output is a function of, and only of, the present input.

**Experiment**

**Part 1 – Counter design using VHDL**

The goal of this part of the lab is to the up-down counter on a DE2 board using the equations obtained in the previous lab

**Methodology**

In order to do this, a new VHDL file was created and an entity named clock1 was defined to have inputs X and CLK and inout q defined as a vector. Component ‘slow’ (the D Flip Flop created in the previous lab) was instantiated three times and the inputs (‘CLK’ and ‘D’) and outputs (‘q’ and ‘qnot’) of each D Flip Flop was port mapped accordingly. Before port mapping the inputs and outputs of the D Flip Flop, a signal D was defined as a vector from 2 down to 0, and each signal D0, D1 and D2, was defined using the equations obtained in the previous lab. The vhdl code for this experiment can be seen below.

**VHDL Code**

**LIBRARY ieee;  
USE ieee.std\_logic\_1164.all;  
  
entity counter1 is   
PORT(  
X: IN STD\_LOGIC;  
CLK: IN STD\_LOGIC;  
q: INOUT STD\_LOGIC\_VECTOR(2 downto 0)  
);  
end counter1;  
   
 architecture behavior of counter1 is  
 signal  D :  STD\_LOGIC\_VECTOR(2 downto 0);  
 signal qnot: STD\_LOGIC\_VECTOR(2 downto 0);  
   
 component slow  
PORT(  
clock1, D : in STD\_logic;  
Q, Q0 : out STD\_logic  
);  
end component;  
  
begin  
  
D(0)<=(not q(0));  
D(1)<=((X and not q(1) and not q(0)) or (not X and not q(1) and q(0)) or (X and q(1) and q(0)) or (not X and q(1) and not q(0)));  
D(2)<=((not X and not q(2) and q(1) and q(0)) or (not X and q(2) and not q(1)) or (X and q(2) and q(0)) or (q(2) and q(1) and not q(0)) or (X and not q(2) and not q(1) and not q(0)));  
  
comp1: slow port map(CLK,D(0),q(0),qnot(0));  
comp2: slow port map(CLK,D(1),q(1),qnot(1));  
comp3: slow port map(CLK,D(2),q(2),qnot(2));  
end behavior;**

The code was compiled, and the input ‘CLK’ was assigned to pin N2, the internal clock of the DE2 board, while input X was assigned to a switch and the outputs ‘q’ were assigned to LEDs.

**Result**

The counter functioned properly as it counted up when input X had a value of 0 and down when it had a value of 1.



**Part II**

The assignment here was to design a binary to 7- segment decoder for the counter and drive the outputs to display the number on a 7- segment display.

**Methodology**

To begin with the VHDL code for a 7 segment display was created it had 4-inputs and 7-outputs. The Boolean equation for each output was stated and the code was compiled and programmed unto the FPGA. The inputs were connected to switched while the outputs were connected to the hexadecimal display on the DE2 board.

**VHDL code for the 7 segment display**

**LIBRARY ieee;**

**USE ieee.std\_logic\_1164.all;**

**ENTITY lab9 IS**

**PORT (**

**W : IN STD\_LOGIC;**

**X : IN STD\_LOGIC;**

**Y : IN STD\_LOGIC;**

**Z : IN STD\_LOGIC;**

**a : OUT STD\_LOGIC;**

**b : OUT STD\_LOGIC;**

**c : OUT STD\_LOGIC;**

**d : OUT STD\_LOGIC;**

**e : OUT STD\_LOGIC;**

**f : OUT STD\_LOGIC;**

**g : OUT STD\_LOGIC**

**);**

**END lab9;**

**ARCHITECTURE Behavior of lab9 IS**

**begin**

**a <= ((NOT W and NOT X and NOT Y and Z) or (NOT W and X and NOT Y and NOT Z) or ( W and NOT X and Y and Z) or (W and X and NOT Y and Z));**

**b <= ((not W AND X AND not Y AND Z) OR (not W AND X AND Y AND not Z) OR ( W AND not X AND Y AND Z) OR ( W AND X AND not Y AND not Z) OR ( W AND X AND Y AND not Z) OR ( W AND X AND Y AND Z));**

**c <= ((not W AND not X AND Y AND not Z) OR ( W AND X AND not Y AND not Z) OR ( W AND X AND Y AND not Z) OR (W and X and Y AND Z));**

**d <= ((not W AND not X AND not Y AND Z) OR (not W AND X AND not Y AND not Z) OR (not W AND X AND Y AND Z) OR (W AND not X AND not Y AND Z) OR ( W AND not X AND Y AND not Z) OR (W AND X AND Y AND Z));**

**e <= ((not W AND not X AND not Y AND Z) OR (not W AND not X AND Y AND Z) OR (not W AND X AND not Y AND not Z) OR (not W AND X AND not Y AND Z) OR (not W AND X AND Y AND Z) OR (W AND not X AND not Y AND Z));**

**f <= ((not W AND not X AND not Y AND Z) OR (not W AND not X AND Y AND not Z) OR (not W AND not X AND Y AND Z) OR (not W AND X AND Y AND Z) OR ( W AND X AND not Y AND Z));**

**g <= ((not W AND not X AND not Y AND not Z) OR (not W AND not X AND not Y AND Z) OR (not W AND X AND Y AND Z) OR ( W AND X AND not Y AND not Z));**

**END Behavior;**

Next a modular approach was used to couple the counter created in part 1 and the new created 7 segment display. This was done by creating a new VHDL file and defining an entity which had the same inputs as the counter, outputs as the 7 segment display and inout K from 3 down to 0. The component ‘counter1’ and ‘lab9’ were instantiated once and they were combined by port mapping the corresponding inputs and outputs. Also 0 was made the most significant bit by driving k(3) to 0.

**The VHDL code**

**LIBRARY ieee;**

**USE ieee.std\_logic\_1164.all;**

**ENTITY COUNT7 is**

**PORT(**

**x: in STD\_LOGIC;**

**clk: in STD\_LOGIC;**

**K: inout STD\_LOGIC\_VECTOR(3 downto 0);**

**a: out STD\_LOGIC;**

**b: out STD\_LOGIC;**

**c: out STD\_LOGIC;**

**d: out STD\_LOGIC;**

**e: out STD\_LOGIC;**

**f: out STD\_LOGIC;**

**g: out STD\_LOGIC**

**);**

**end COUNT7;**

**architecture behavior of COUNT7 is**

**component counter1**

**PORT(**

**X: IN STD\_LOGIC;**

**CLK: IN STD\_LOGIC;**

**q: INOUT STD\_LOGIC\_VECTOR(2 downto 0)**

**);**

**end component;**

**component lab9**

**PORT (**

**W : IN STD\_LOGIC;**

**X : IN STD\_LOGIC;**

**Y : IN STD\_LOGIC;**

**Z : IN STD\_LOGIC;**

**a : OUT STD\_LOGIC;**

**b : OUT STD\_LOGIC;**

**c : OUT STD\_LOGIC;**

**d : OUT STD\_LOGIC;**

**e : OUT STD\_LOGIC;**

**f : OUT STD\_LOGIC;**

**g : OUT STD\_LOGIC**

**);**

**END component;**

**begin**

**K(3)<='0';**

**count1: counter1 port map(x, clk, K(2 downto 0));**

**end behavior;**

Finally the code was compiled and the inputs and outputs were assigned to switches and the hex display. Input ‘CLK’ was assigned to pin N2, and the code was downloaded unto the FPGA.

**Result**

The binary to 7-segment display was a success as it displayed the binary numbers on the 7 segment display of the DE2 board.

**Conclusion**

Although the group encountered some problem along the way, this was still a successful lab. I was able to brush up my modular design understanding, and at the end I have a better understanding of sequential logic circuits and how they function.

Post Lab Questions

1. **LIBRARY ieee;**

**USE ieee.std\_logic\_1164.all;**

**entity slow\_clock is**

**PORT(**

**clock\_in : in std\_logic;**

**clock\_out : out std\_logic**

**);**

**end slow\_clock;**

**architecture behavior of slow\_clock is**

**signal clock\_tmp : std\_logic;**

**begin**

**process(clock\_in)**

**variable x : integer := 0;**

**begin**

**if(clock\_in'event and clock\_in='0') then**

**x:=x+1;**

**if x = 25000000 then**

**x:=0;**

**clock\_tmp <= not clock\_tmp;**

**clock\_out <= clock\_tmp;**

**end if;**

**end if;**

**end process;**

**end behavior;**

1. The J-K Flip Flop has two inputs labeled J and K. If J and K are different then the output Q takes the value of J at the next clock edge. If J and K are both low then no change occurs. If J and K are both high at the clock edge then the output will toggle from one state to the other. The outputs feed back to the enabling NAND gates, this is what gives the toggling action when J=K=1.

**Pre-Lab Questions**

In the Moore model output depends only on the present state but in the Mealy model the output depends on the both present state and the input.