Erich Viebrock

ECPE 174

**Lab Report #1**

Problem Summary:

An old car has tail lights controlled by 6 LEDs. In order to understand which LED is which, the names LA, LB, LC, RA, RB, and RC are used to represent each LED, from left to right. When the car is in IDLE (no switches used), all LEDs should be off. For the switch LEFT, the LEDs flash in a particular order: LB, LA and LC, all left LEDs on, and all off. For the switch RIGHT, the LEDs flash in a particular order: RB, RA and RC, all right LEDs on, and all off. In order to signify HAZARD lights are on, all LEDs flash on, and all LEDs flash off. When LEFT and RIGHT are both turned on, LEDs will go to IDLE. When more than one switch is used and HAZARD is involved, the sequence for HAZARD will begin. For all situations, LEDs will repeat their respected loop until the switch is changed. A clock divider is used in order to display LEDs at a rate conceivable to the human eye. The design is done in a Moore FSM.

Design Approach:

Our design is simple; by VHDL we were able to create a Moore FSM to replicate the behavior of an old car’s tail lights. Loops allowed us to create the desired output for all different scenarios.

Verification Procedure:

Our design did not run into error, but confusion arose after seeing the Moore, One-Hot, and Mealy diagrams of our code. Surprisingly, each diagram appeared the same. To our knowledge, this is almost never the case. Because of this, we believed there was error in our VHDL. However, this was not the problem, which ensued to further confusion and frustration. Then, the professor approved the similar diagrams, as they were desired output.

Post-Lab Questions:

* What, if any, difference is there between the two State Machine Viewer outputs? How do they compare to the one you designed in pre-lab step 1?
  + We found no difference in the State Machine Viewer outputs. This is because both methods will produce the simplest design possible.
  + In terms of the difference between Quartus’s State Machine Viewer and my personal state machine, there was slight difference. I was unaware an IDLE state was needed in the middle of the transition from LEFT to RIGHT, or RIGHT to LEFT. Therefore, this was the only difference.
* What equations did the synthesis tool use to implement the design? How does this compare with your results and ideas outlined in pre-lab step 3?
  + The synthesis tool uses hundreds of equations to implement the design. Some typical equations used, for example, are:
    - B1L41 is clockdiv:clockstage|Add0~40
    - B1L41 = (B1\_Count[20] & (B1L40 $ (GND))) # (!B1\_Count[20] & (!B1L40 & VCC));
  + Or:
    - B1L1 is clockdiv:clockstage|Add0~0
    - B1L1 = B1\_Count[0] $ (VCC);
  + The equations implement the same design as VHDL, but with far more statements. Therefore, the synthesis tool is considered to be have far higher cost.
* Outline the results of the fitter resource usage summary: how much space on the FPGA does the design require? What types of elements were used?
  + The design required minimal space on the FPGA. 3 switches, and 6 output LEDs were used to test for desired output.
* How fast could you clock the design according to Quartus? What happens if you attempt to clock it at that speed? Faster than that speed?
  + According to Quartus, the design can be clocked at 27 MHz.
  + If you attempt to clock it at that speed, the system will be able to produce output, but will be too fast for the human eye to recognize.
  + If you attempt to clock faster than that speed, the system will not be able to keep up and skip clock cycles.

You need to explain more in your postlab questions and write them in paragraph form. Also, you did not properly analyze the design in terms of the compilation report. Make sure to discuss this with your lab partner and/or myself if you do not understand where to find the information requested.

Rubric:  
Organization = 2  
Quality = 1  
Amount = 1  
Technical Content = 1.5

Appendix:

**VHDL Main:**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

ENTITY tail\_light IS

PORT ( L, R, H, Clock :IN STD\_LOGIC;

Lights :buffer STD\_LOGIC\_VECTOR (1 to 6));

END tail\_light;

ARCHITECTURE Behavior OF tail\_light IS

TYPE State\_type IS (off, LA, LB, LC, RA, RB, RC, LR);

SIGNAL y: State\_type;

BEGIN

PROCESS (Clock, H, L, R)

BEGIN

IF H = '1' THEN

y <= off;

ELSIF clock = '0' THEN

y <= off;

ELSIF (Clock'EVENT AND Clock = '1') THEN

CASE y IS

WHEN off =>

IF H = '1' THEN y <= LR;

ELSIF L = '1' THEN y <= LA;

ELSIF R = '1' THEN Y <= RA;

ELSE y <= off;

END IF;

WHEN LA =>

IF H = '1' then y <= LR;

ELSE y <= LB;

END IF;

WHEN LB =>

IF H = '1' then y <= LR;

ELSE y <= LC;

END IF;

WHEN LC =>

y <= off;

WHEN RA =>

IF H = '1' then y <= LR;

ELSE y <= RB;

END IF;

WHEN RB =>

IF H = '1' then y <= LR;

ELSE y <= RC;

END IF;

WHEN RC =>

y <= off;

WHEN LR =>

y <= off;

END CASE;

END IF;

END PROCESS;

PROCESS (Lights, y)

BEGIN

IF

y = off

THEN Lights <= "000000";

ElSIF

y = LR

THEN Lights <= "111111";

ElSIF

y = LA

THEN Lights <= "010000";

ElSIF

y = LB

THEN Lights <= "101000";

ElSIF

y = LC

THEN Lights <= "111000";

ElSIF

y = RA

THEN Lights <= "000010";

ElSIF

y = RB

THEN Lights <= "000101";

ElSIF

y = RC

THEN Lights <= "000111";

END IF;

END PROCESS;

END Behavior;

**Clock Divider:**

LIBRARY ieee ;

USE ieee.std\_logic\_1164.all ;

ENTITY masterclk IS

PORT ( iclk : IN STD\_LOGIC;

oclk : OUT STD\_LOGIC);

END masterclk;

ARCHITECTURE Behavior OF masterclk IS

CONSTANT halfcount : POSITIVE := 13500000;

SIGNAL Count: INTEGER RANGE 0 TO halfcount-1;

SIGNAL clkstate : STD\_LOGIC;

BEGIN

PROCESS ( iclk )

BEGIN

IF (iclk'EVENT AND iclk = '1') THEN

IF Count = halfcount-1 THEN

Count <= 0;

clkstate <= not clkstate;

ELSE

Count <= Count + 1;

END IF;

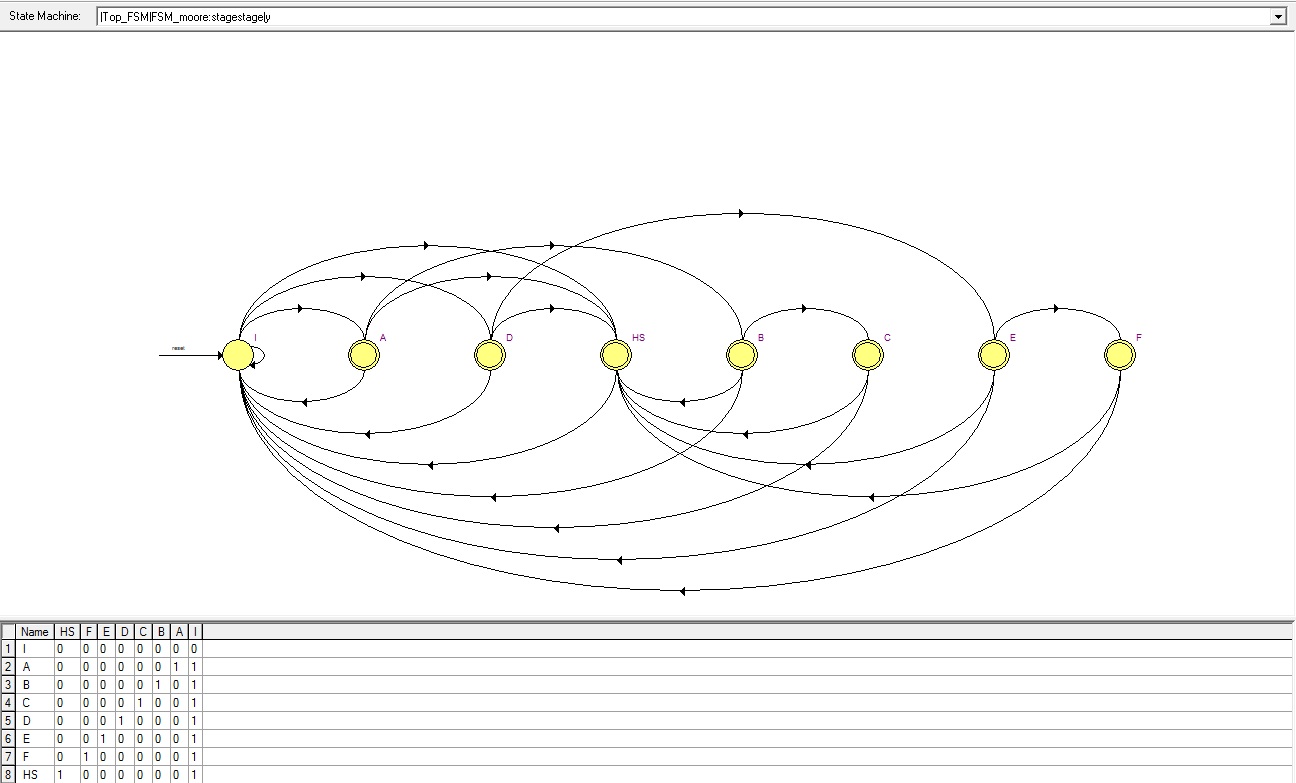
END IF;

END PROCESS;

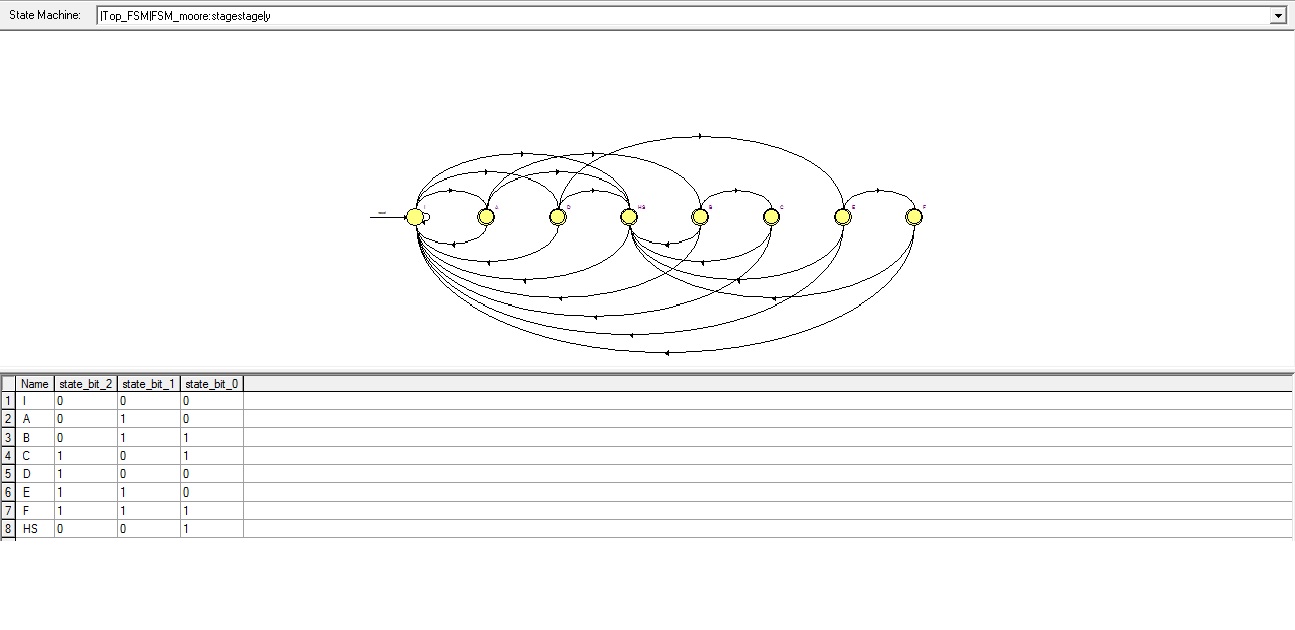
oclk <= clkstate;

END Behavior;

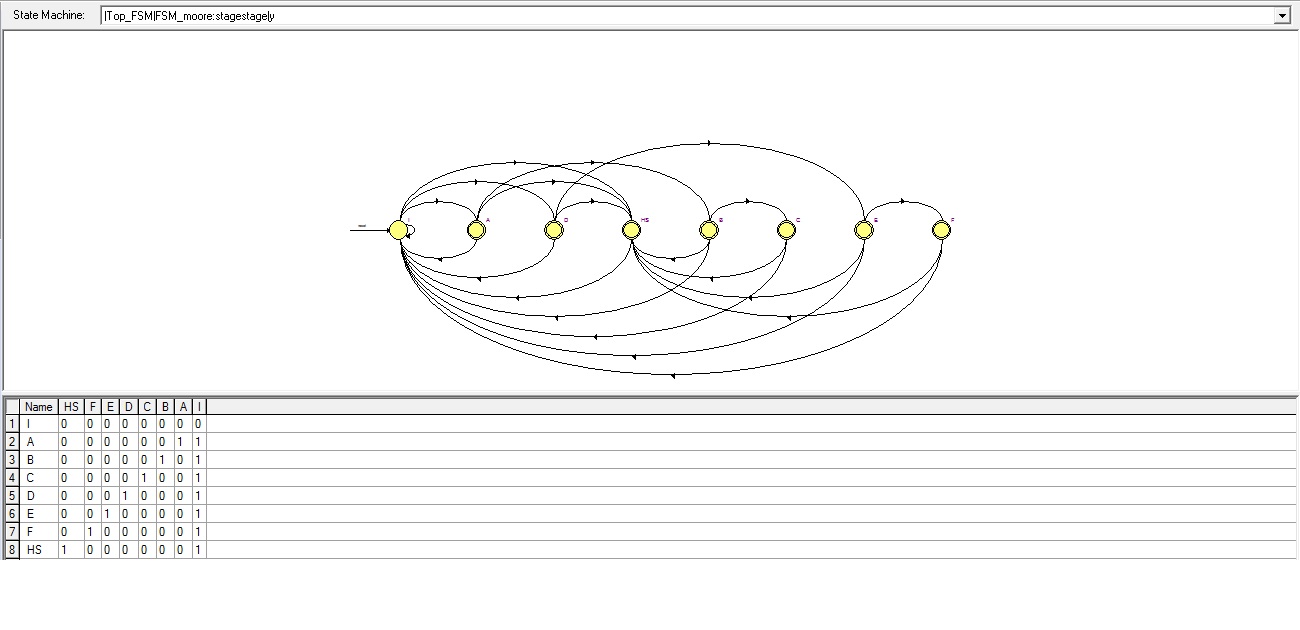
**State Machine Auto**

****

**State Machine Minimal Bit**

****

**State Machine One Hot**

****