Gillian Kearney

Lab Report 5

ECE 2031 L09

24 February 2022

Graphical user interface

Description automatically generated**Figure 1.**  Functional simulation waveform of Red, Green, Blue finite state machine with 4 inputs and 4 outputs. The input vector covers all possible state transitions to prove the correctness of the circuit.

A screenshot of a computer

Description automatically generated

**Figure 2.** Signal Tap configuration for specifications needed to read from the state machine implemented on the DE10-Standard Board.

A picture containing timeline

Description automatically generated

**Figure 3.** Signal Tap acquisition from a 3 output state machine implemented on the DE10 DE10-Standard Board. All possible state transition are achieved through variations of the input vector.

Appendix A

VHDL Code Implementing Moore State Machine

-- StateMachine.vhd

-- Four-State Moore State Machine

-- Gillian Kearney

-- ECE 2031 L09

-- 2/24/22

-- A Moore machine's outputs are dependent only on the current state.

-- The output is written only when the state changes. (State

-- transitions are synchronous.)

library ieee;

use ieee.std\_logic\_1164.all;

entity StateMachine is

port(

clk : in std\_logic;

reset : in std\_logic;

cool : in std\_logic;

xmas : in std\_logic;

output: out std\_logic\_vector(1 downto 0);

r\_en : out std\_logic;

g\_en : out std\_logic;

b\_en : out std\_logic

);

end entity;

architecture rtl of StateMachine is

-- Build an enumerated type for the state machine

type state\_type is (red, green, blue);

-- Register to hold the current state

signal state : state\_type;

begin

-- Logic to advance to the next state

process (clk, reset)

begin

if reset = '1' then

state <= red;

elsif (rising\_edge(clk)) then

case state is

when red=>

state <= green;

when green=>

if xmas = '1' and cool = '0' then

state <= red;

else

state <= blue;

end if;

when blue=>

if cool = '0' then

state <= red;

else

state <= blue;

end if;

end case;

end if;

end process;

-- Output depends solely on the current state

process (state)

begin

case state is

when red =>

output <= "00";

r\_en <= '1';

g\_en <= '0';

b\_en <= '0';

when green =>

output <= "01";

r\_en <= '0';

g\_en <= '1';

b\_en <= '0';

when blue =>

output <= "10";

r\_en <= '0';

g\_en <= '0';

b\_en <= '1';

end case.

end process;

end rtl;