1. Problem/Objective

For Lab 2 we were asked to code an FSMD which is a known Finite State Machine with a datapath. We used VHDL to simulate the possible outputs and applied it to an FPGA using Xilinx and the Digilent S3 board. We were given a shell of the code and needed to code the state registers as well as the next-state logic and the Moore output logic. After completing this we were able to create the test bench and see the waveform. Part B of the lab was to implement the code on the Digilent S3 board, but we included a clock to prevent the LEDs from flashing too quickly.

2. Methodology

http://ieeexplore.ieee.org.lib-proxy.fullerton.edu/stamp/stamp.jsp?tp=&arnumber=5548689

In this article it talks about FSMD partitioning, which they say is an effective tool to isolate circuit components. They use the Plackett and Burman experiment design to help simulate annealing to help partition the FSMD. The single components can be power gated and/or clock gated to save a tremendous amount of power. The Finite State Machine with Datapath partitioning technique can be used to confine the logic within hardware architecture. They then go on to talk about the procedure about how to optimize the model better.

They go in to more depth about the FSMD partitioning technique. They describe that the at the behavioral level the technique is functional before synthesis. They talk about acquiring a competent partition which allows them to solve a non-linear programming problem, by using an algorithm. Their sole purpose is to reduce the number of components shared between partitions. Also, to minimize the amount of transitions that occur between each partition.

3. Question(s)

a. Consider the following Mealy state machine with A as the initial state.

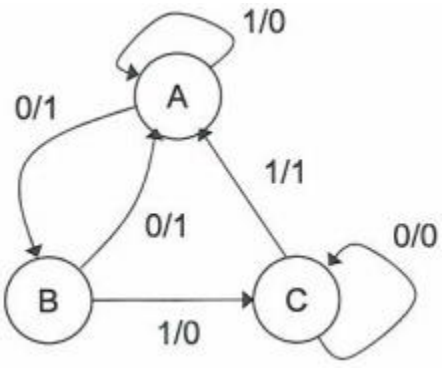

Figure 1: Mealy machine.

If the input sequence is 011, then the corresponding output sequence, $A \xrightarrow{0/1} B \xrightarrow{1/0} C \xrightarrow{1/1} A$, would be 101.

i. What would be the output if the input is fixed to a constant 1 for all cycles?
A 1/0 -> A -> 1/0 A-> 1/0 , output 000.

ii. What would be the output if the input is fixed to a constant 0 for all cycles?

A 0/1-> B 0/1 -> A 0/1 -> B 0/1, output 111.

b. For Part B, you were asked to cause a state transition to occur once a second. What was the value of the "count" variable that you used to assure this? Show your work to receive full credit.

Our count variable that we used for part B was 50000000 because the Digilent S3 board used a 50MHz oscillator clock. This means that there are 50000000 cycles per second, this helps when we test the code on the board so that the LEDs do not cycle quickly.

$50\ {}^{MHz}/_{X} = 1\ sec.$

$X = 50{,}000{,}000$

4. Program Code

Copy your code here. Please provide comments in your code. This will help me analyze your code and remove any ambiguity. **Provide your code as text, not as a screenshot/image**.

Part A:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity PartA is
   port(
      clk, reset : in std_logic;
      r1_reg, r2_reg : inout std_logic_vector (7 downto 0) := "00000000");
end PartA;

architecture Behavioral of PartA is
   type eg_state_type is (s0, s1, s2, s3);
   signal state_reg, state_next: eg_state_type;
   signal r1_next: std_logic_vector (7 downto 0);
begin
   --State Register
   process(clk, reset)
   begin
   --Set values at reset
      if (reset = '1') then
         r1_reg <= "00000000";
         r2_reg <= "00000001";
         state_reg <= s0;
      --Update reg on rising edge
      elsif (clk'event and clk = '1') then
         r1_reg <= r1_next;
         state_reg <= state_next;
      end if;
   end process;

   --Next State Logic
```

```
   process(state_reg)
   begin
   -- Cycle states to do manipulations on buffer
      case state_reg is
         when s0 =>
            r1_next <= "00001000";
            state_next <= s1;
         when s1 =>
            r1_next <= r1_reg + r2_reg;
            state_next <= s2;
         when s2 =>
            r1_next <= (r1_reg(5 downto 0) & "00");
            state_next <= s3;
         when s3 =>
            r1_next <= r1_reg;
            state_next <= s0;
      end case;
   end process;
end Behavioral;
```

Part B:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity PartB is
   port(
      clk, reset : in std_logic;
      r1_reg, r2_reg : inout std_logic_vector (7 downto 0));
end PartB;

architecture Behavioral of PartB is
   type eg_state_type is (s0, s1, s2, s3);
   signal state_reg, state_next: eg_state_type;
   signal r1_next: std_logic_vector (7 downto 0);
begin
   --State Register
   process(clk, reset)
      variable count : integer := 0;
   begin
   --Set values at reset
      if (reset = '1') then
         r1_reg <= "00000000";
         r2_reg <= "00000001";
         count := 0;
         state_reg <= s0;
      --Update reg on rising edge & inc count to slow output cycles
```

```
    elsif (clk'event and clk = '1') then
      if   (count = 50000000) then
        r1_reg <= r1_next;
        state_reg <= state_next;
        count := 0;
      else
        count := count + 1;
      end if;
    end if;
  end process;

  --Next State Logic
  process(state_reg)
  begin
  -- Cycle states to do manipulations on buffer
    case state_reg is
      when s0 =>
        r1_next <= "00001000";
        state_next <= s1;
      when s1 =>
        r1_next <= r1_reg + r2_reg;
        state_next <= s2;
      when s2 =>
        r1_next <= (r1_reg(5 downto 0) & "00");
        state_next <= s3;
      when s3 =>
        r1_next <= r1_reg;
        state_next <= s0;
    end case;
  end process;
end Behavioral;
```

5. Test Bench

Copy your test bench code here. Again, please provide comments in your code. This will help me analyze your code and remove any ambiguity. **Provide your test bench code as text, not as a screenshot/image**.

Test Bench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY parta_test_vhd_tb IS
END parta_test_vhd_tb;

ARCHITECTURE behavior OF parta_test_vhd_tb IS

  COMPONENT parta
  PORT(
```

```
      clk : IN std_logic;
      reset : IN std_logic;
      r1_reg : INOUT std_logic_vector(7 downto 0);
      r2_reg : INOUT std_logic_vector(7 downto 0));
   END COMPONENT;

   SIGNAL clk :  std_logic := '0';
   SIGNAL reset :  std_logic := '0';
   SIGNAL r1_reg :  std_logic_vector(7 downto 0);
   SIGNAL r2_reg :  std_logic_vector(7 downto 0);

BEGIN

   uut: parta PORT MAP(
      clk => clk,
      reset => reset,
      r1_reg => r1_reg,
      r2_reg => r2_reg);


-- *** Test Bench - User Defined Section ***
   clk_proc : PROCESS
  BEGIN
   --Simulate clk by using inverter
   clk <= not clk after 20 ns;
   wait for 10 ns;
  END PROCESS;

   stim_proc : PROCESS
  BEGIN
   --Reset for 1 clk cycle
   reset <= '1';
   wait for 20 ns;
   --Clear reset and let clock run
   reset <= '0';
   wait for 10000 ns;

   --Notify user when finished
   assert false report "End of Simulation" severity failure;
  END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```