ECEN 429: Introduction to Digital Systems Design Laboratory

North Carolina Agricultural and Technical State University

Department of Electrical and Computer Engineering

Ian Parker (Reporter)

Tayanna Lee (Lab Partner)

March 14, 2019

Lab #6

Introduction:

For lab 6, we had to first figure out to manipulate the speed of the clock. We then, continued with the use of components - implementing the clock divider with the Moore, and then Mealy Machine. My lab partner and I became familiar with how to develop Moore and Mealy Machine VHDL code.

Part 1: Clock Divider

Truth Table (Dr. Doss said not to include one)**

Schematic(Dr. Doss said not to include one)**

Pin Assignment

Variables	Input/Output	Pin Assignments
clk	in	W5
FastClock	out	U19
led0	out	L1
MediumClock	out	E19
SlowClock	out	U19
start_timer	in	V16

Results

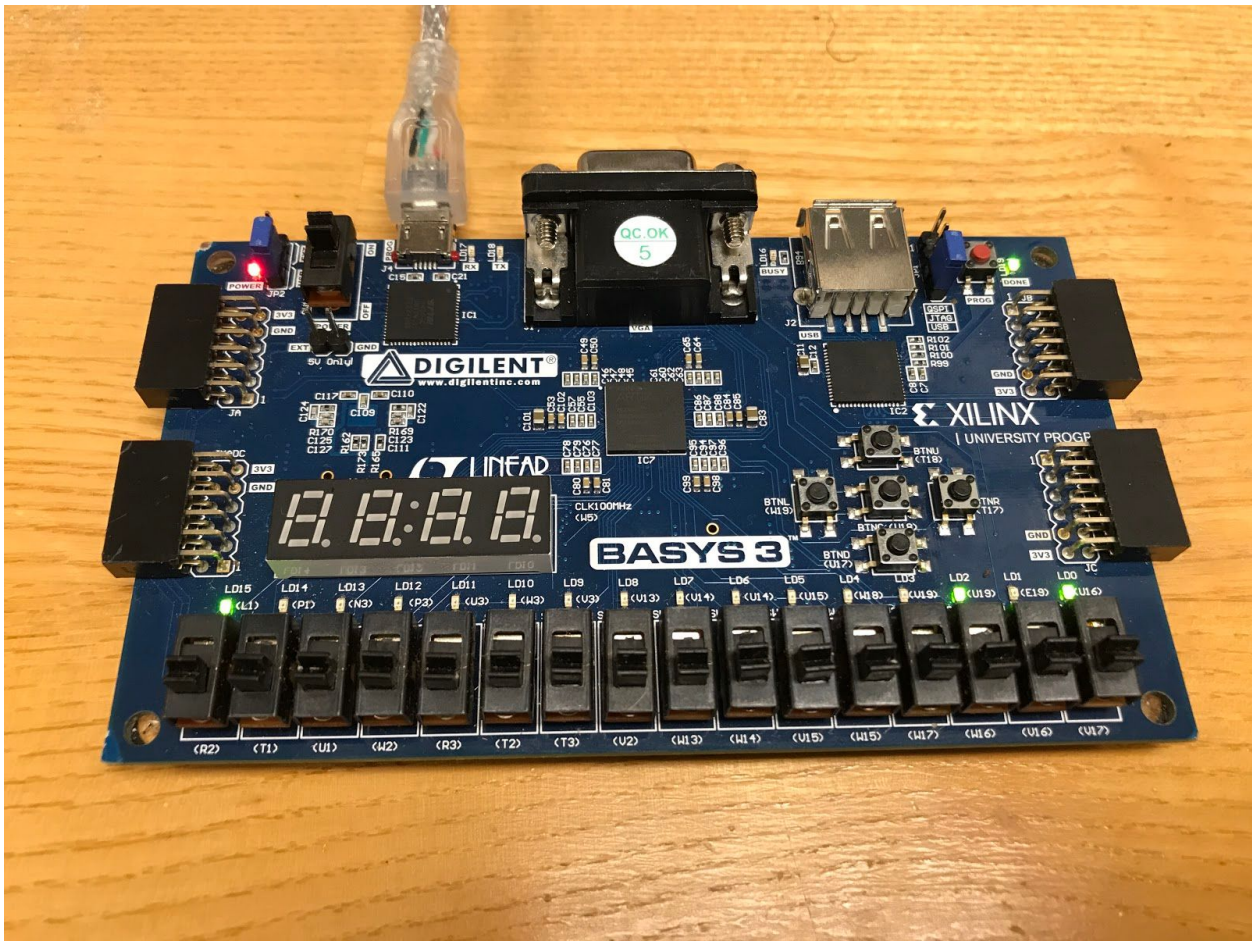
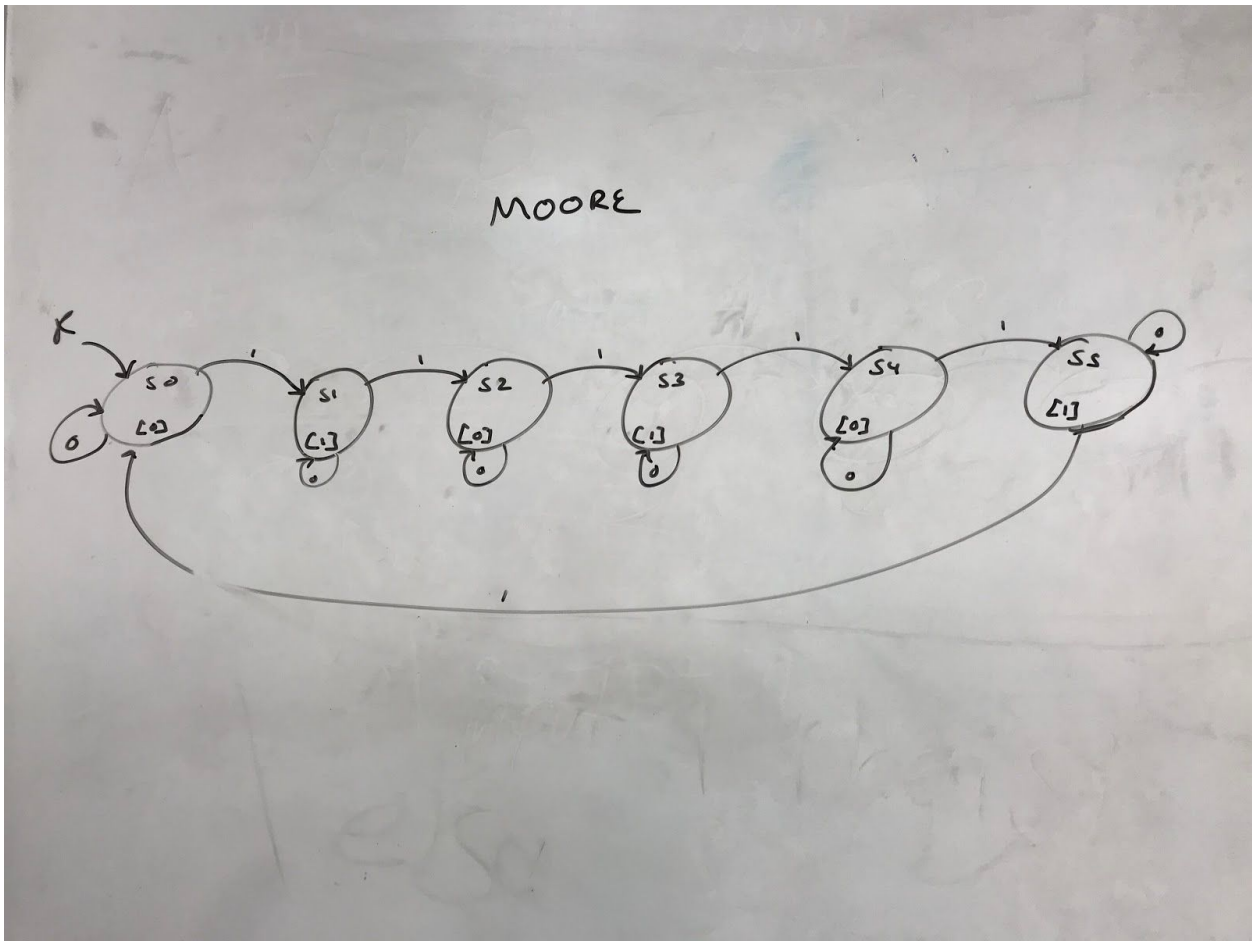


Figure 1.1) Clock Divider with Fast, Medium, and Slow Clocks (flashing)

Part 2: Moore State Machine

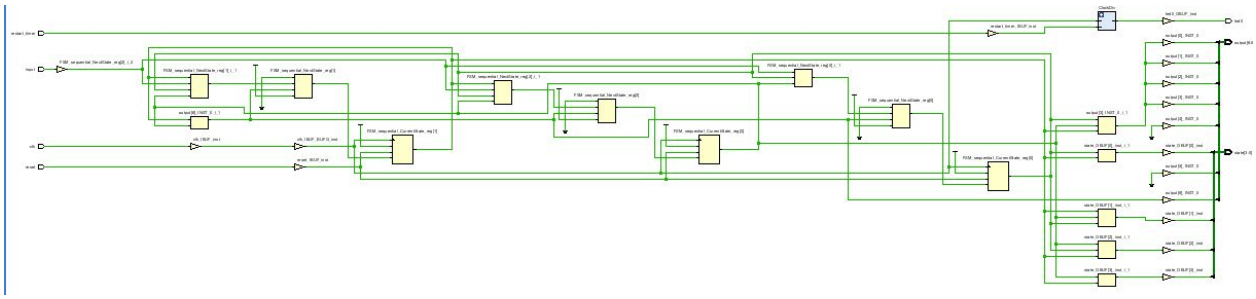
Diagram



Pin Assignments

Variables	Input/Output	Pin Assignments
output(6)	out	U7
output(5)	out	W6
output(4)	out	U8
output(3)	out	V8
output(2)	out	U5
output(1)	out	V5
output(0)	out	W7
state(3)	out	V19
state(2)	out	U19
state(1)	out	E19
state(0)	out	U16
clk	in	W5
input	in	R2
led0	out	L1
reset	in	V16
restart_timer	in	V17

Schematic



Results

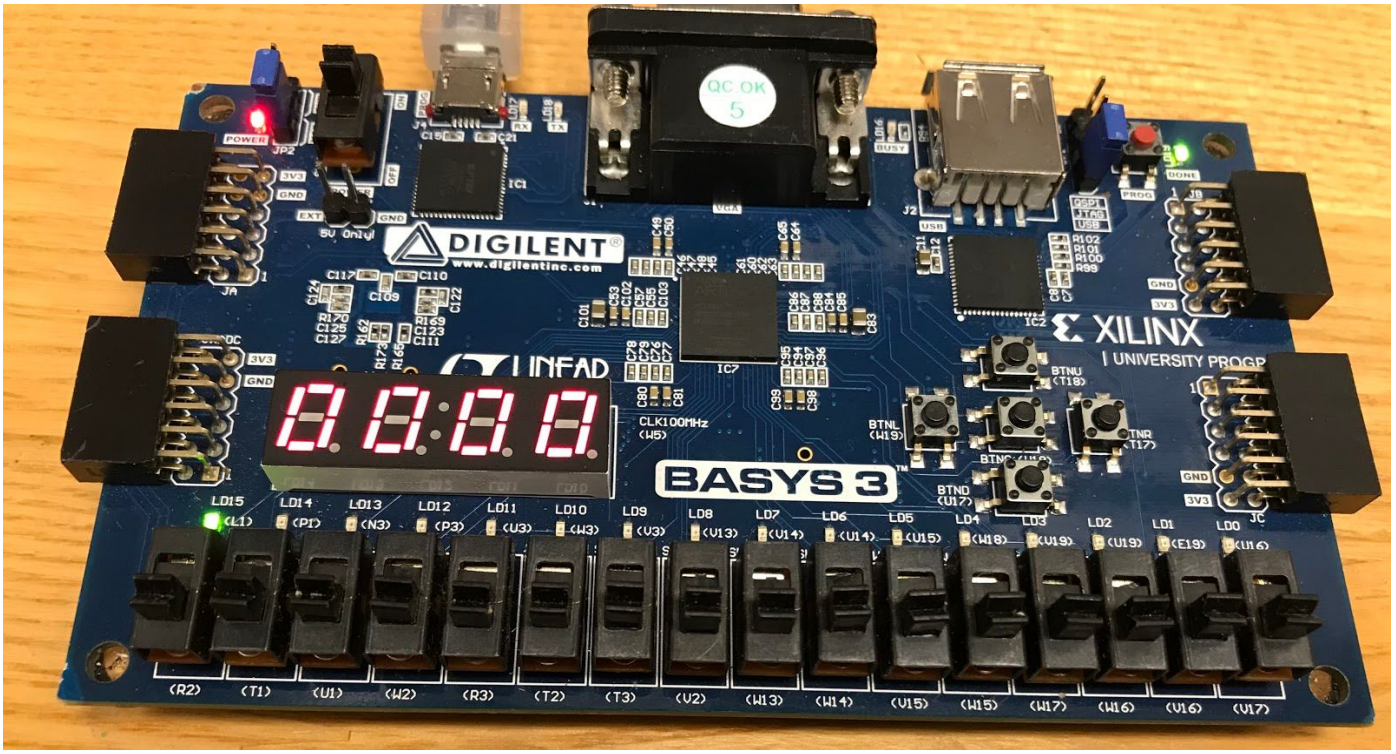


Figure 2.1) At State 0 with an input of 0

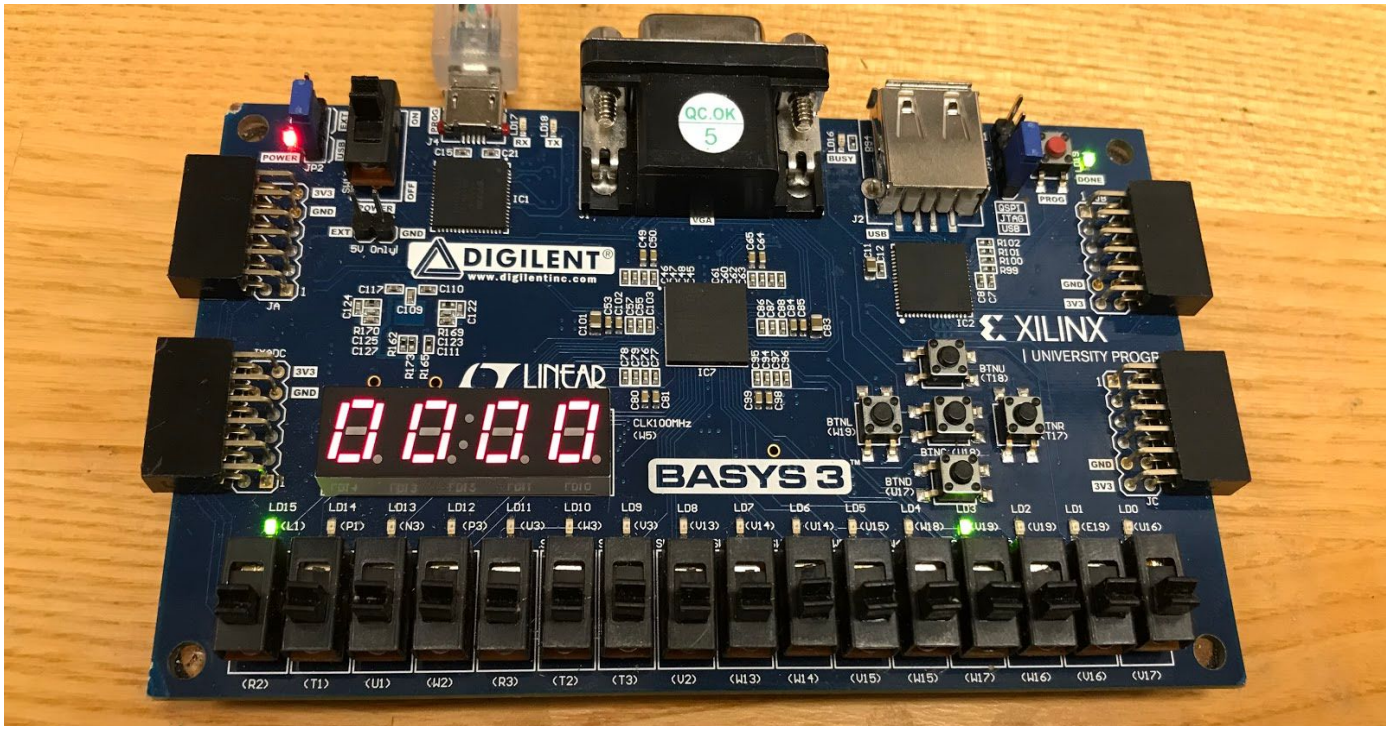
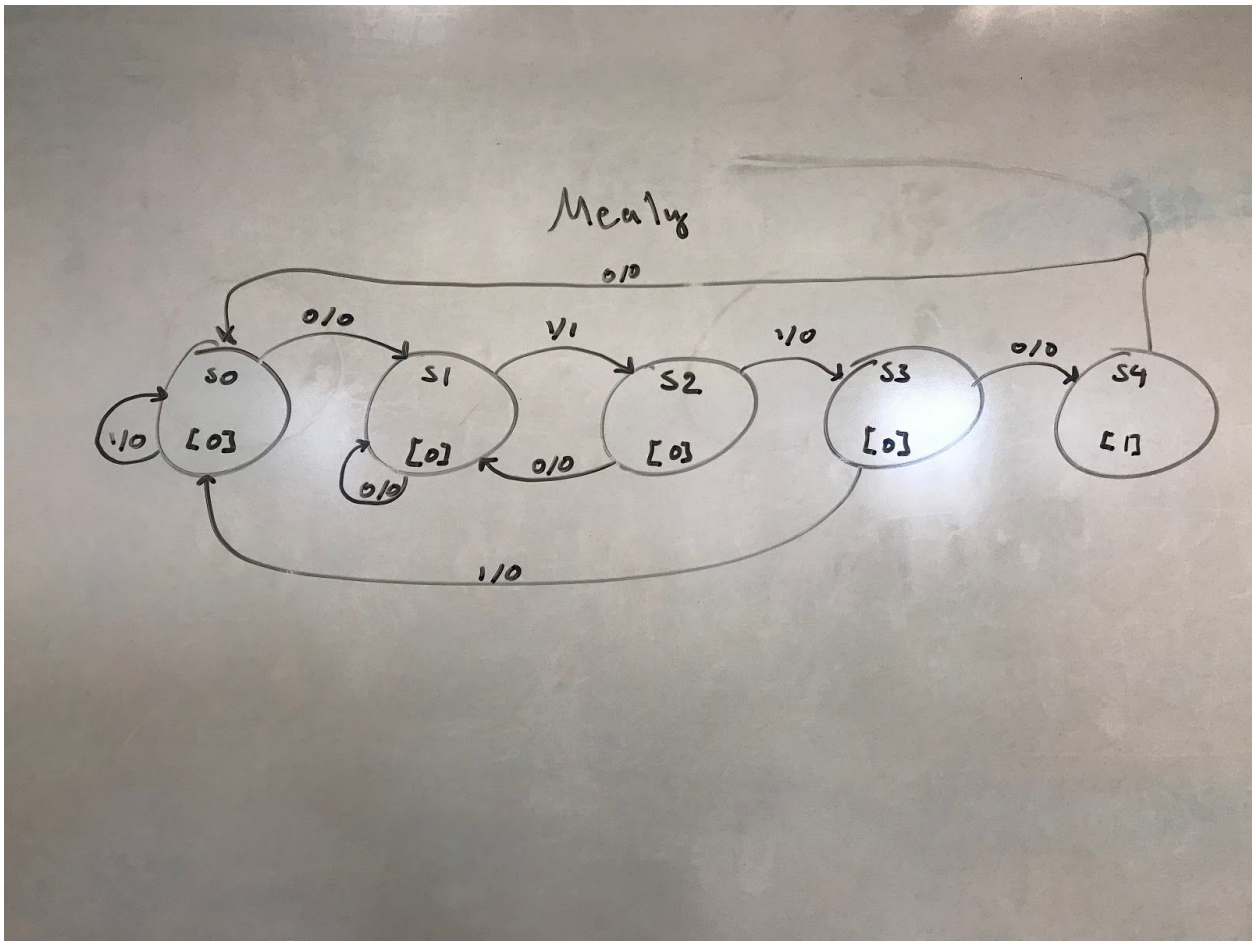


Figure 2.2) At State 4 with input of 0

Part 3: Mealy State Machine

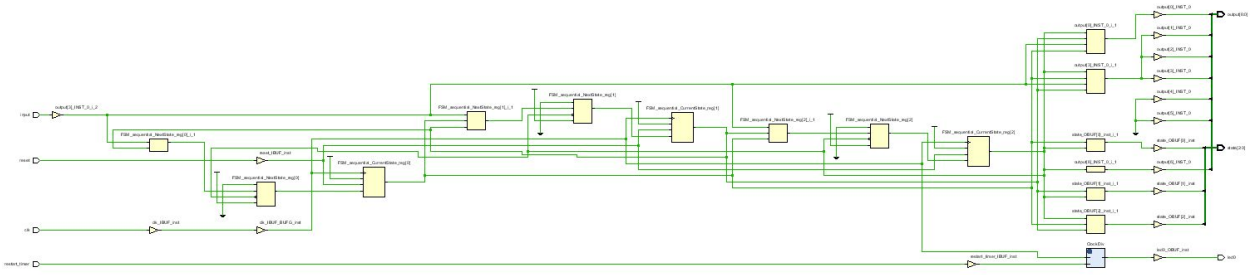
Diagram



Pin Assignments

output (7)	OUT					☑	34	LVCMOS33*
output[6]	OUT				U7	☑	34	LVCMOS33*
output[5]	OUT				W6	☑	34	LVCMOS33*
output[4]	OUT				U8	☑	34	LVCMOS33*
output[3]	OUT				V8	☑	34	LVCMOS33*
output[2]	OUT				U5	☑	34	LVCMOS33*
output[1]	OUT				V5	☑	34	LVCMOS33*
output[0]	OUT				W7	☑	34	LVCMOS33*
state (3)	OUT					☑	14	LVCMOS33*
state[2]	OUT				U19	☑	14	LVCMOS33*
state[1]	OUT				E19	☑	14	LVCMOS33*
state[0]	OUT				U16	☑	14	LVCMOS33*
Scalar ports (5)								
clk	IN				W5	☑	34	LVCMOS33*
input	IN				R2	☑	34	LVCMOS33*
led0	OUT				L1	☑	35	LVCMOS33*
reset	IN				V16	☑	14	LVCMOS33*
restart_timer	IN				V17	☑	14	LVCMOS33*

Schematic



Results

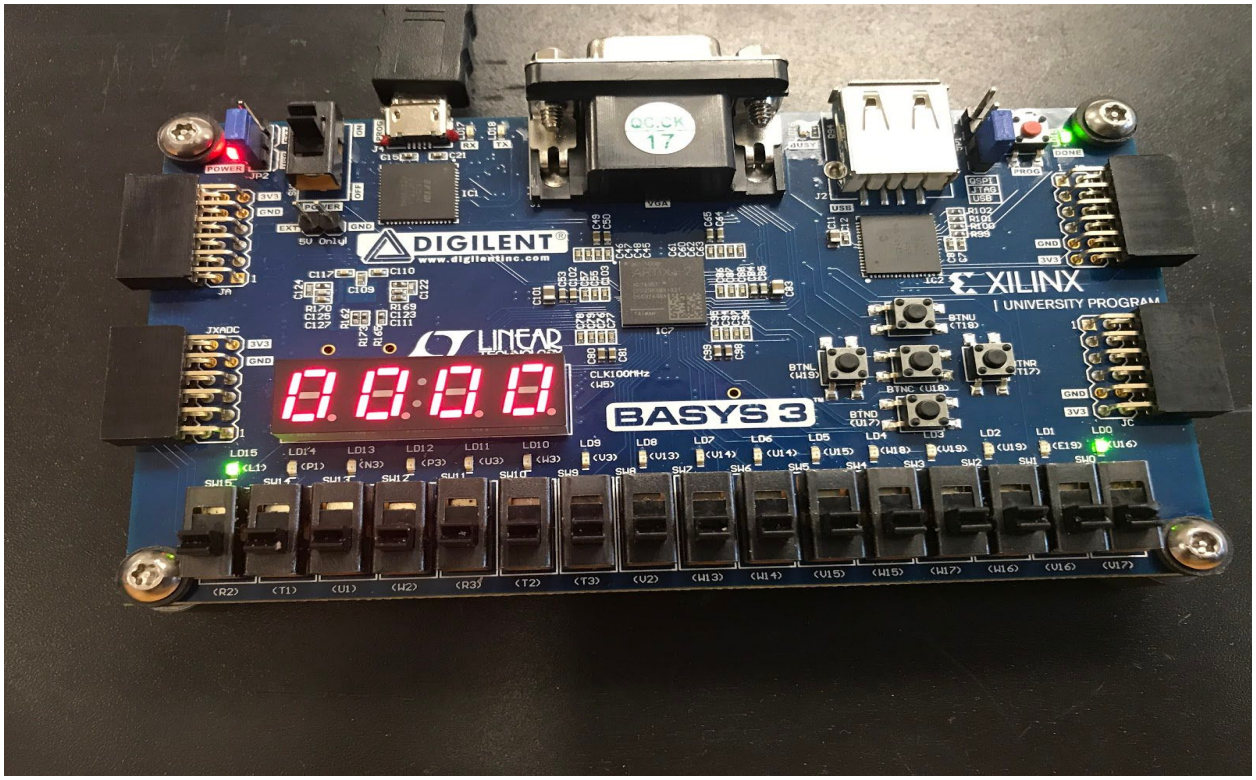


Figure 2.1) At state 0 with input 0 and output 0

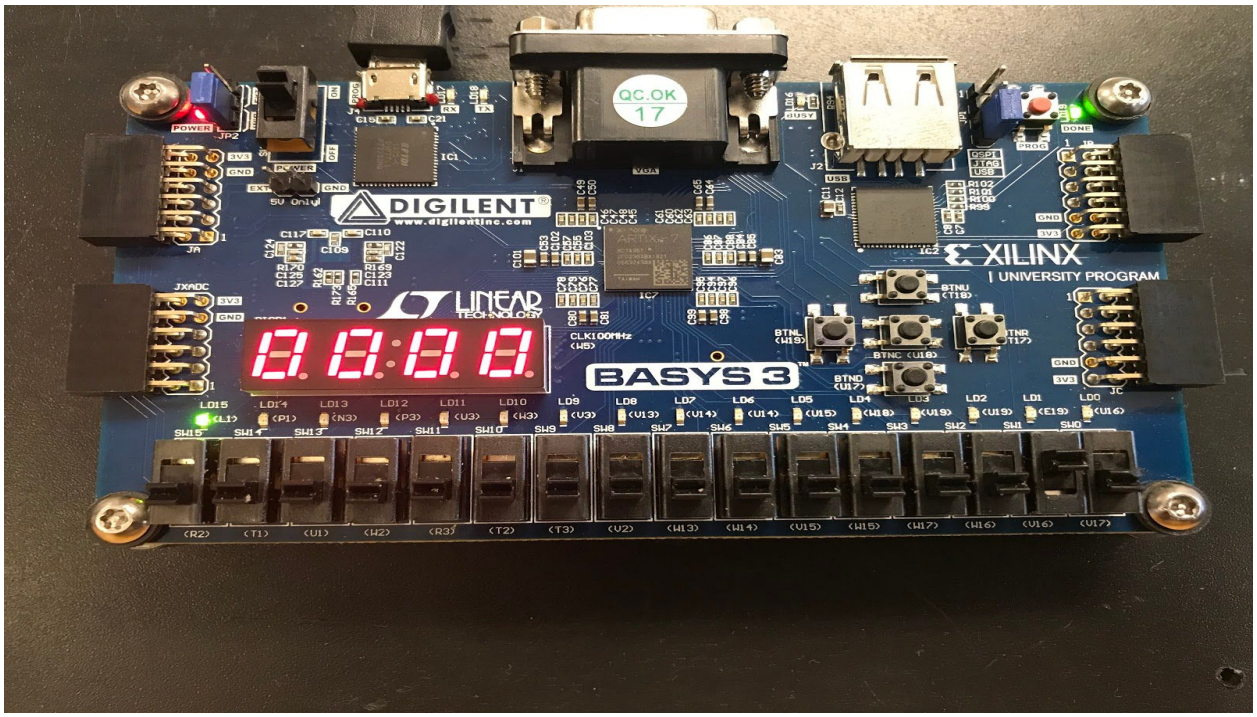


Figure 2.1) The reset is active, sending us back to state 0

Conclusion:

In conclusion, when learning how to control the clock - we ended up creating three different speeds: fast, medium, and slow. We then designed a component for the clock divider, to be implemented into both the Moore and Mealy Machine scenarios. This lab helped us review how differently both Finite State Machines are coded, especially when it comes to displaying the outputs. A Moore Machine output values are determined by its current state, while a Mealy Machine output values are determined by its current state and by the values of its inputs. We displayed those outputs using the Seven Segment Displays.

Appendices:

Part 1 VHDL CODE : Clock Divider

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

entity ClockDivider is
    port( clk : in STD_LOGIC ;
          start_timer : in STD_LOGIC ;
          FastClock , MediumClock , SlowClock , led0 : out STD_LOGIC );
end Clockdivider ;

architecture beh of Clockdivider is
    --signals for clock
     signal slowClock_sig : STD_LOGIC;
        begin
        process

        variable cnt : STD_LOGIC_VECTOR (26 downto 0):= "00000000000000000000000000"
; --Counter. When this variable reaches the Most Signifigant Bit approximatley one second (1
Hz) will have passed

        begin
        wait until (( clk 'EVENT) AND ( clk = '1'));--Wait until clock is high

            if ( start_timer = '1') then
                --Reset the Counter
                cnt := "00000000000000000000000000" ;
            else
                --Start Counting by incrementing by 1
                cnt := STD_LOGIC_VECTOR( unsigned ( cnt ) + 1);
            end if ;

                --Clock Settings
                FastClock     <= cnt (22);         --FastClock == 2*MediumClock Frequency
                MediumClock   <= cnt (24);         --MediumClock = 2*SlowClock Frequency
```

```
            SlowClock     <= cnt (26);        --SlowClock = 1Hz
            slowClock_sig <= cnt (26);        -- same as slow clock

        --Display the clock information to the LED
        if ( slowClock_sig = '1') then
           led0 <= '1';
        else
           led0 <= '0';
        end if ;

   end process;
end beh;
```

Part 2 : Moore Machine Constraints

```
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports {output[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[0]}]
set_property PACKAGE_PIN U16 [get_ports {state[0]}]
set_property PACKAGE_PIN E19 [get_ports {state[1]}]
set_property PACKAGE_PIN U19 [get_ports {state[2]}]
set_property PACKAGE_PIN V19 [get_ports {state[3]}]
set_property PACKAGE_PIN W7 [get_ports {output[0]}]
set_property PACKAGE_PIN V5 [get_ports {output[1]}]
set_property PACKAGE_PIN U5 [get_ports {output[2]}]
set_property PACKAGE_PIN V8 [get_ports {output[3]}]
set_property PACKAGE_PIN U8 [get_ports {output[4]}]
set_property PACKAGE_PIN W6 [get_ports {output[5]}]
set_property PACKAGE_PIN U7 [get_ports {output[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {state[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {state[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {state[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {state[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports input]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports led0]
set_property IOSTANDARD LVCMOS33 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports restart_timer]
set_property PACKAGE_PIN R2 [get_ports input]
set_property PACKAGE_PIN L1 [get_ports led0]
set_property PACKAGE_PIN V16 [get_ports reset]
set_property PACKAGE_PIN V17 [get_ports restart_timer]
```

Part 2 : Moore Machine VHDL Code

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

--A State Machine that outputs a 1 only if the state is Odd. The output of the state is displayed on the 7-Segment Display. The actual state number is displayed on the LED.
entity MooreStateMachine is
    Port ( input, clk, reset, restart_timer : in STD_LOGIC;
           led0  : out STD_LOGIC;--led representing SlowClock
           output: out STD_LOGIC_VECTOR(6 downto 0);--Output of the state displayed on 7 Segment Display
           state : out STD_LOGIC_VECTOR(3 downto 0));--Current State of State Machine displayed on LEDs;
end MooreStateMachine;

architecture Behavioral of MooreStateMachine is

           --Component Declaration for Clockdivider
           component Clockdivider is Port(clk, start_timer : in STD_LOGIC;
                                          FastClock, MediumClock,SlowClock,led0 : out STD_LOGIC);
           end component;
           --Component Declaration for Seven Segment Display
           component SevenSegmentDisplay is Port(BCD : in  STD_LOGIC_VECTOR (6 downto 0);
                                                 Display : out STD_LOGIC_VECTOR (6 downto 0));
           end component;
```

```
        --Signal for Moore Machine Output
        signal outTemp : STD_LOGIC_VECTOR (6 downto 0):= "0000000";

        --Signals For Clocks
        signal SlowClock : STD_LOGIC;
        signal MediumClock : STD_LOGIC;
        signal FastClock : STD_LOGIC;
        --Display Signal for Seven Segment Display
        signal Display : STD_LOGIC_VECTOR (6 downto 0):= "0000000";
        --Signal for Current and Next State
        signal CurrentState : STD_LOGIC_VECTOR(3 downto 0) := "0000";
        signal NextState : STD_LOGIC_VECTOR(3 downto 0) :="0000";

    begin
        --Component Instantiation for ClockDivider
        ClockDiv : ClockDivider port map(clk, restart_timer, FastClock,
MediumClock,SlowClock,led0);
        --Component Instantiation for Seven Segment Display
        SevenSegDisplay : SevenSegmentDisplay port map(outTemp,Display);

            --Process for SlowClock Reset
            process(clk, reset)
                begin
                    --Reset sends back to State 0
                    if (reset = '1') then
                        CurrentState <= "0000";
                    --otherwise move onto next state
                    elsif(clk'event AND (clk = '1')) then
                        CurrentState <= NextState;
                    end if;
            end process;


            process(CurrentState)
                begin
                    case CurrentState is
                        --when current state is 0 the output is 0 displayed on 7 seg
                        when "0000" => outTemp <="1000000";--0
```

```
    if(input = '0') then
        NextState <= "0000";--remain in state 0
    elsif(input = '1') then
        NextState <= "0001";--move onto state 1
    end if;
--when current state is 1 the output is 1 displayed on 7 seg
when "0001" =>outTemp <= "1001111"; --1
    if(input = '0') then
        NextState <= "0001";--remain in state 1
    elsif(input = '1') then
        NextState <= "0010";--move onto state 2
    end if;
--when current state is 2 the output is 2 displayed on 7 seg
when "0010" =>outTemp <= "1000000";
    if(input = '0') then
        NextState <= "0010";--remain in state 2
    elsif(input = '1') then
        NextState <= "0100";--move onto state 3
    end if;
--when current state is 3 the output is 3 displayed on 7 seg
when "0100" =>outTemp <= "1001111";
    if(input = '0') then
        NextState <= "0100";--remain in state 3
    elsif(input = '1') then
        NextState <= "1000";--move onto state 4
    end if;
--when current state is 4 the output is 4 displayed on 7 seg
when "1000" =>outTemp <= "1000000";
    if(input = '0') then
        NextState <= "1000";--remain in state 4
    elsif(input = '1') then
        NextState <= "1001";--move onto state 5
    end if;
--when current state is 5 the output is 5 displayed on 7 seg
when "1001" =>outTemp <= "1001111";
    if(input = '0') then
        NextState <= "1000";--remain in state 5
    elsif(input = '1') then
        NextState <= "0000";--move onto state 0
```

```
                end if;
            when others => outTemp <= "0000001";
        end case;
    end process;

    state <= CurrentState;
    output <=outTemp;

end Behavioral;
```

Part 3 : Mealy Machine Constraints

```
set_property IOSTANDARD LVCMOS33 [get_ports {output[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {output[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {state[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {state[1]}]
set_property PACKAGE_PIN U7 [get_ports {output[6]}]
set_property PACKAGE_PIN W6 [get_ports {output[5]}]
set_property PACKAGE_PIN U8 [get_ports {output[4]}]
set_property PACKAGE_PIN V8 [get_ports {output[3]}]
set_property PACKAGE_PIN U5 [get_ports {output[2]}]
set_property PACKAGE_PIN V5 [get_ports {output[1]}]
set_property PACKAGE_PIN W7 [get_ports {output[0]}]
set_property PACKAGE_PIN U16 [get_ports {state[0]}]
set_property PACKAGE_PIN E19 [get_ports {state[1]}]
set_property PACKAGE_PIN U19 [get_ports {state[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports input]
set_property IOSTANDARD LVCMOS33 [get_ports led0]
set_property IOSTANDARD LVCMOS33 [get_ports {state[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports restart_timer]
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN R2 [get_ports input]
```

```
set_property PACKAGE_PIN L1 [get_ports led0]
set_property PACKAGE_PIN V16 [get_ports reset]
set_property PACKAGE_PIN V17 [get_ports restart_timer]
```

Part 3 : Mealy Machine VHDL Code

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

--A state machine that takes a single input and detects whether a 0-1-1-0 sequence has been
received. There should be 2 different outputs, one that would turn on if the 0-1 in the sequence
has been detected and another one that is used to be turned on only when the full sequence has
been detected. Your results should be displayed on LEDs

entity MealyStateMachine is
    Port ( input, clk, reset, restart_timer : in STD_LOGIC;
           led0  : out STD_LOGIC;--led representing SlowClock
           output: out STD_LOGIC_VECTOR(6 downto 0);--Output of the state displayed on 7
Segment Display
           state : out STD_LOGIC_VECTOR(2 downto 0));--Current State of State Machine
displayed on LEDs;
end MealyStateMachine;

architecture Behavioral of MealyStateMachine is

           --Component Declaration for Clockdivider
           component Clockdivider is Port(clk, start_timer : in STD_LOGIC;
                                 FastClock, MediumClock,SlowClock,led0 : out STD_LOGIC);
           end component;
           --Component Declaration for Seven Segment Display
           --component SevenSegmentDisplay is Port(BCD : in  STD_LOGIC_VECTOR (6 downto
0);
                                       --Display : out STD_LOGIC_VECTOR (6 downto 0));
           --end component;

           --Signal for Mealy Machine Output
           signal outTemp : STD_LOGIC_VECTOR (6 downto 0):= "0000000";
```

```
        --Signals For Clocks
        signal SlowClock : STD_LOGIC;
        signal MediumClock : STD_LOGIC;
        signal FastClock : STD_LOGIC;
        --Display Signal for Seven Segment Display
        signal Display : STD_LOGIC_VECTOR (6 downto 0):= "0000000";
        --Signal for Current and Next State
        signal CurrentState : STD_LOGIC_VECTOR(2 downto 0) := "000";
        signal NextState : STD_LOGIC_VECTOR(2 downto 0) :="000";

    begin
        --Component Instantiation for ClockDivider
        ClockDiv : ClockDivider port map(clk, restart_timer, FastClock,
MediumClock,SlowClock,led0);
        --Component Instantiation for Seven Segment Display
        --SevenSegDisplay : SevenSegmentDisplay port map(outTemp,Display);

            --Process for SlowClock Reset
            process(clk, reset)
                begin
                    --Reset sends back to State 0
                    if (reset = '1') then
                        CurrentState <= "000";
                    --otherwise move onto next state
                    elsif(clk'event AND (clk = '1')) then
                        CurrentState <= NextState;
                    end if;
            end process;

            --NOTE: NEED TO ADD CORRECT 7SEG VALUES FOR OUTPUTS
            process(CurrentState)
                begin
                    case CurrentState is
                        --when current state is 0 the output is 0 displayed on 7 seg
                        when "000" => outTemp <="1000000";-- *0
                            if(input = '0') then
                                NextState <= "001";--move to state 1
                            elsif(input = '1') then
                                NextState <= "000";--stay at state 0
```

```
                end if;
            --when current state is 1 the output is 0 displayed on 7 seg
            when "001" =>
                if(input = '0') then
                    NextState <= "001";--remain in state 1
                    outTemp <= "1000000"; --*0
                elsif(input = '1') then
                    NextState <= "010";--move onto state 2
                    outTemp <= "1001111";  -- *1
                end if;
            --when current state is 2 the output is 2 displayed on 7 seg
            when "010" =>
                if(input = '0') then
                    NextState <= "010";--stay in state 2
                    outTemp <= "1000000"; --*0
                elsif(input = '1') then
                    NextState <= "011";--move onto state 3
                    outTemp <= "1000000"; --*0
                end if;
            --when current state is 3 the output is 3 displayed on 7 seg
            when "011" =>
                if(input = '0') then
                    NextState <= "100";--move to state 4
                    outTemp <= "1001111";  -- *1
                elsif(input = '1') then
                    NextState <= "011";--stay in state 3
                    outTemp <= "1000000"; --*0
                end if;
                        when others => outTemp <= "0000001";
        end case;
    end process;

    state <= CurrentState;
    output <=outTemp;

end Behavioral;
```