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 28, 2019

Lab #7

Introduction:

For lab 7, we had to create an ALU with four different functions and a FSM. We then took these two separate programs and produced components from both, in order to develop a top-level design that performs a different function based upon the current state. We also implemented the seven segment display to show the results of those functions being executed.

Part 1: ALU

TRUTH TABLE

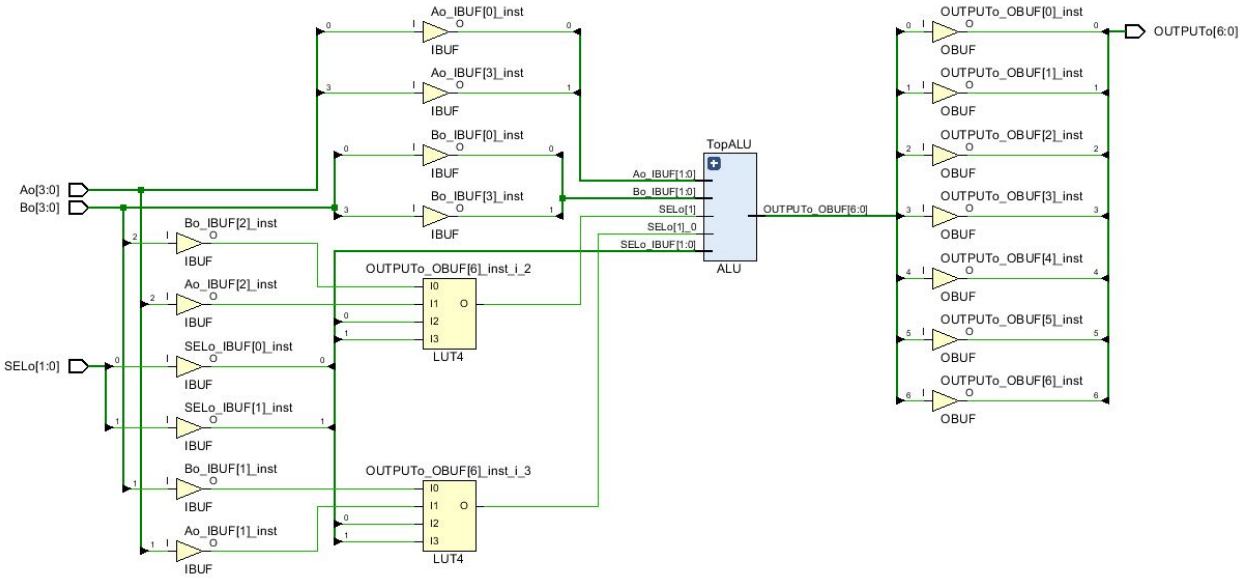
For two inputs A and B

SELECT(CASE)	OUTPUT (OPERATION)
00	XOR
01	NAND
10	ADD
11	SUBTRACT

PIN ASSIGNMENTS

All ports (17)								
Ao (4)	IN					✓	14	LVCMOS33*
Ao[3]	IN				W17	✓	14	LVCMOS33*
Ao[2]	IN				W16	✓	14	LVCMOS33*
Ao[1]	IN				V16	✓	14	LVCMOS33*
Ao[0]	IN				V17	✓	14	LVCMOS33*
Bo (4)	IN					✓	34	LVCMOS33*
Bo[3]	IN				R2	✓	34	LVCMOS33*
Bo[2]	IN				T1	✓	34	LVCMOS33*
Bo[1]	IN				U1	✓	34	LVCMOS33*
Bo[0]	IN				W2	✓	34	LVCMOS33*
OUTPUTo (7)	OUT					✓	34	LVCMOS33*
OUTPUTo[6]	OUT				U7	✓	34	LVCMOS33*
OUTPUTo[5]	OUT				V5	✓	34	LVCMOS33*
OUTPUTo[4]	OUT				U5	✓	34	LVCMOS33*
OUTPUTo[3]	OUT				V8	✓	34	LVCMOS33*
OUTPUTo[2]	OUT				U8	✓	34	LVCMOS33*
OUTPUTo[1]	OUT				W6	✓	34	LVCMOS33*
OUTPUTo[0]	OUT				W7	✓	34	LVCMOS33*
SELo (2)	IN					✓	14	LVCMOS33*
SELo[1]	IN				W13	✓	14	LVCMOS33*
SELo[0]	IN				W14	✓	14	LVCMOS33*

SCHEMATIC


Ao[3:0]
Bo[3:0]
SELo[1:0]
Ao_IBUF[0]_inst
Ao_IBUF[3]_inst
Bo_IBUF[0]_inst
Bo_IBUF[3]_inst
Bo_IBUF[2]_inst
Ao_IBUF[2]_inst
SELo_IBUF[0]_inst
SELo_IBUF[1]_inst
Bo_IBUF[1]_inst
Ao_IBUF[1]_inst
IBUF
OUTPUTo_OBUF[6]_inst_i_2
OUTPUTo_OBUF[6]_inst_i_3
I0
I1
I2
I3
O
LUT4
TopALU
Ao_IBUF[1:0]
Bo_IBUF[1:0]
SELo[1]
SELo[1]_0
SELo_IBUF[1:0]
OUTPUTo_OBUF[6:0]
ALU
OUTPUTo_OBUF[0]_inst
OUTPUTo_OBUF[1]_inst
OUTPUTo_OBUF[2]_inst
OUTPUTo_OBUF[3]_inst
OUTPUTo_OBUF[4]_inst
OUTPUTo_OBUF[5]_inst
OUTPUTo_OBUF[6]_inst
OBUF
OUTPUTo[6:0]

RESULT

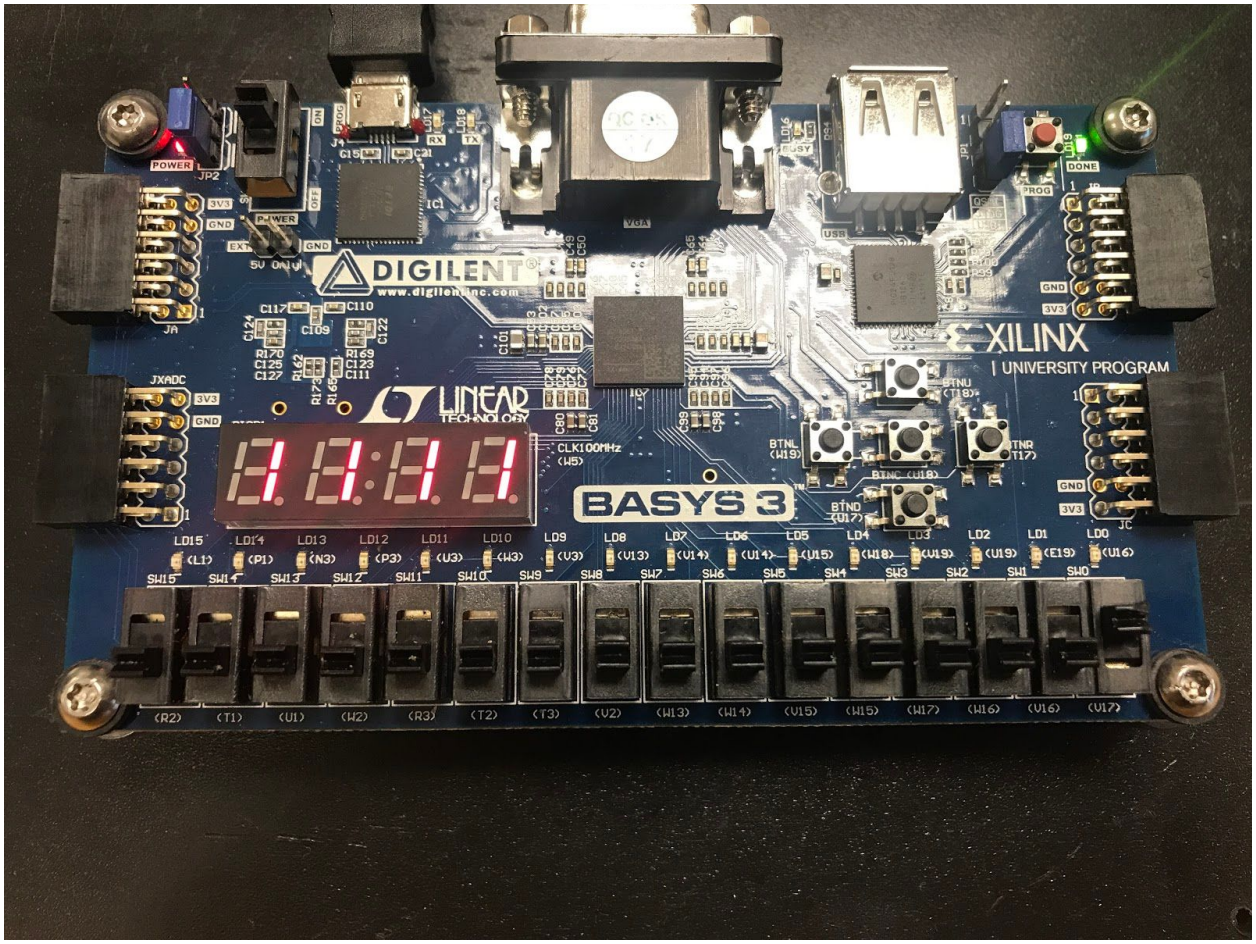


Figure 1.1) B(0000) XOR A(0001) = (0001) or 1 in HEX

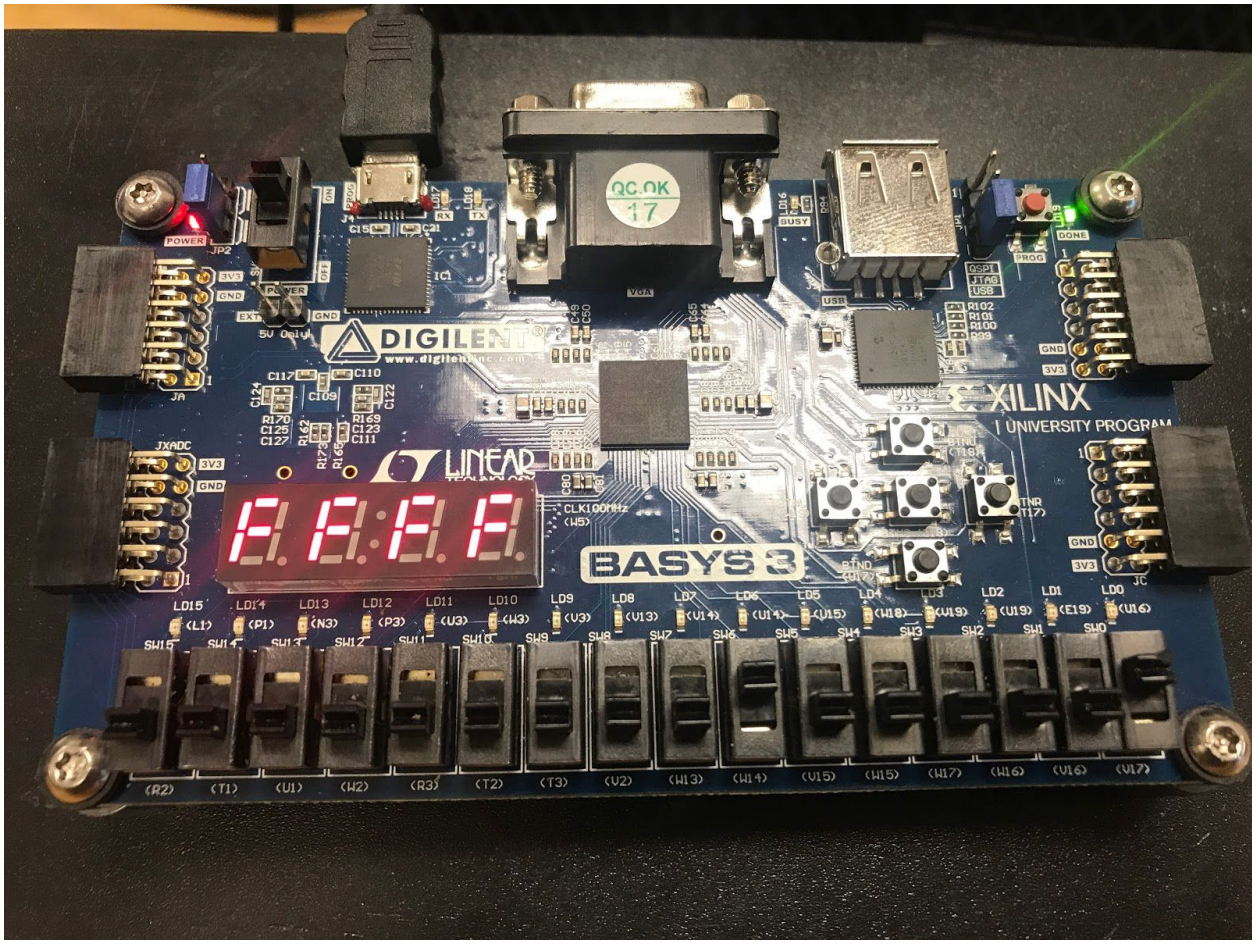
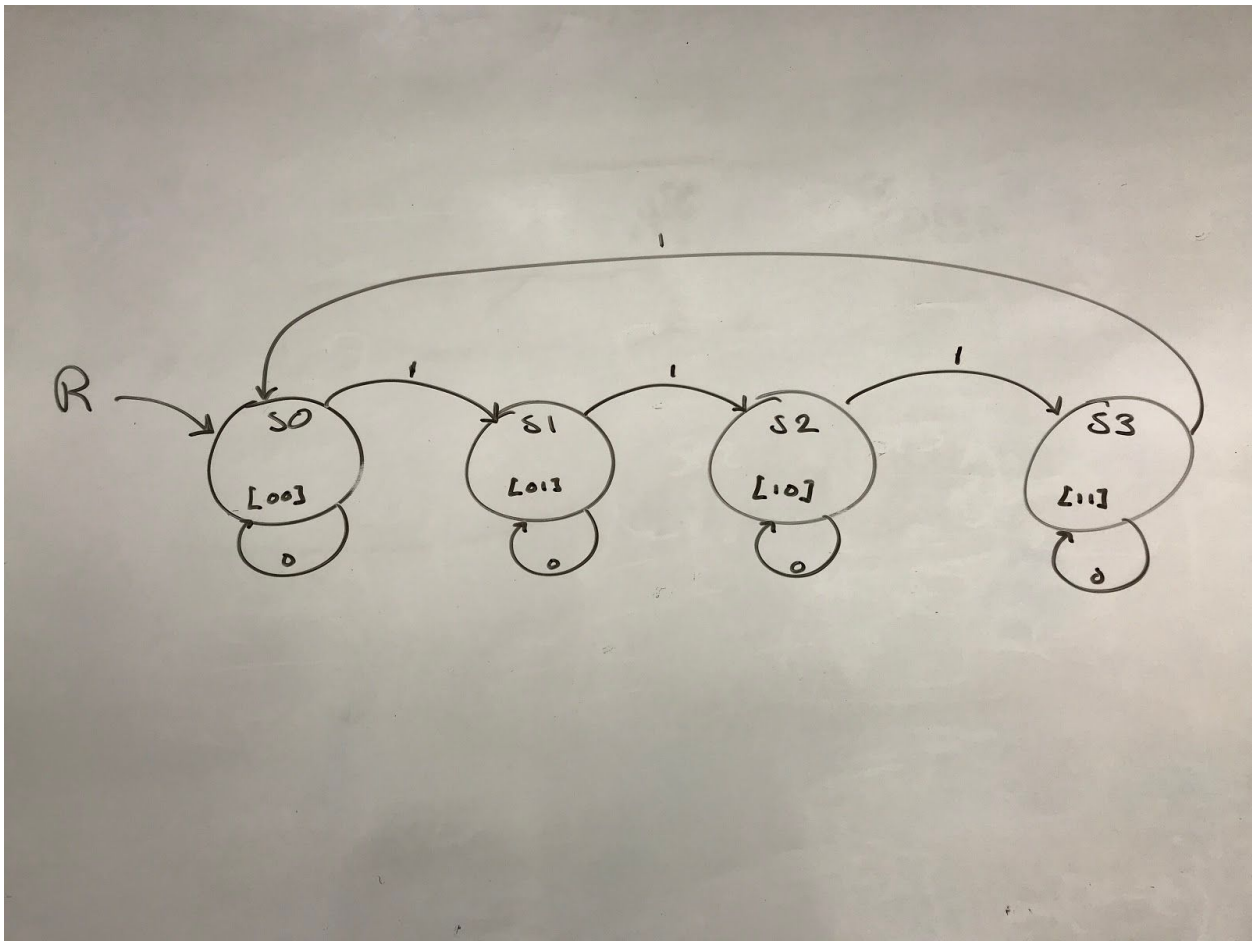


Figure 1.2) B(0000) NAND A(0001) = (1111) or F in HEX

PART 2: FSM

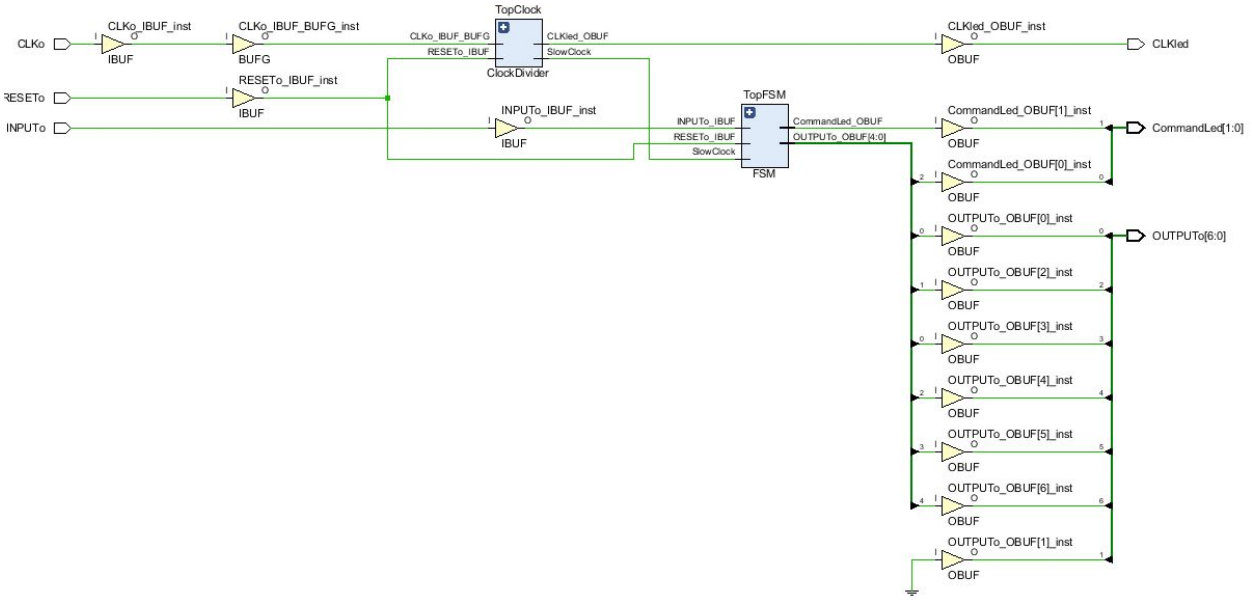
STATE DIAGRAM



PIN ASSIGNMENTS

All ports (13)								
CommandLed (2)	OUT					✓	35	LVCMOS33*
CommandLed[1]	OUT				L1	✓	35	LVCMOS33*
CommandLed[0]	OUT				P1	✓	35	LVCMOS33*
OUTPUTo (7)	OUT					✓	34	LVCMOS33*
OUTPUTo[6]	OUT				U7	✓	34	LVCMOS33*
OUTPUTo[5]	OUT				V5	✓	34	LVCMOS33*
OUTPUTo[4]	OUT				U5	✓	34	LVCMOS33*
OUTPUTo[3]	OUT				V8	✓	34	LVCMOS33*
OUTPUTo[2]	OUT				U8	✓	34	LVCMOS33*
OUTPUTo[1]	OUT				W6	✓	34	LVCMOS33*
OUTPUTo[0]	OUT				W7	✓	34	LVCMOS33*
Scalar ports (4)								
CLKled	OUT				V3	✓	34	LVCMOS33*
CLKo	IN				W5	✓	34	LVCMOS33*
INPUTo	IN				R2	✓	34	LVCMOS33*
RESETo	IN				T1	✓	34	LVCMOS33*

SCHEMATIC



RESULTS

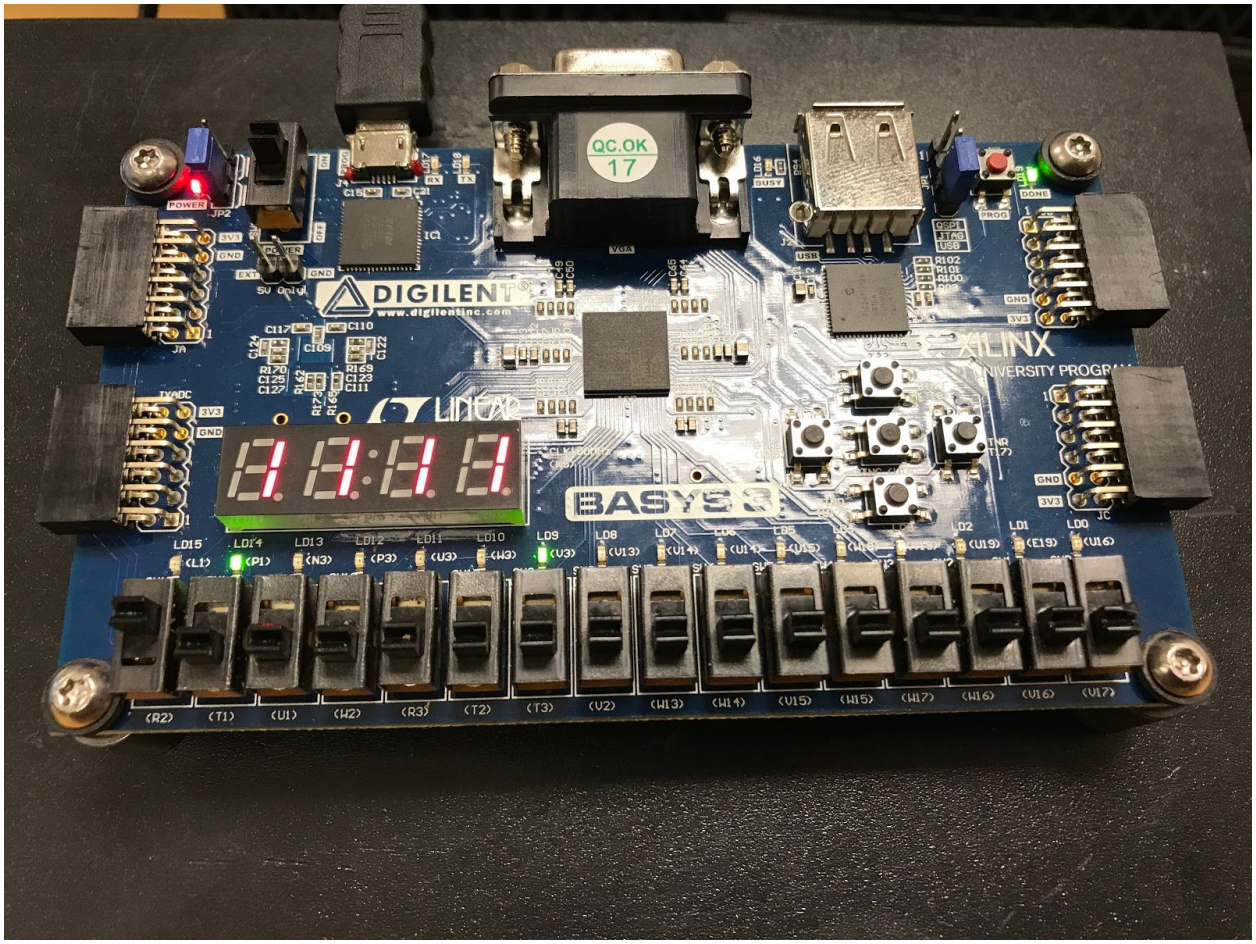


Figure 2.1) INPUT = 1 and CLOCK is high. Here we see that the Command issued is (01) and the State is (1)

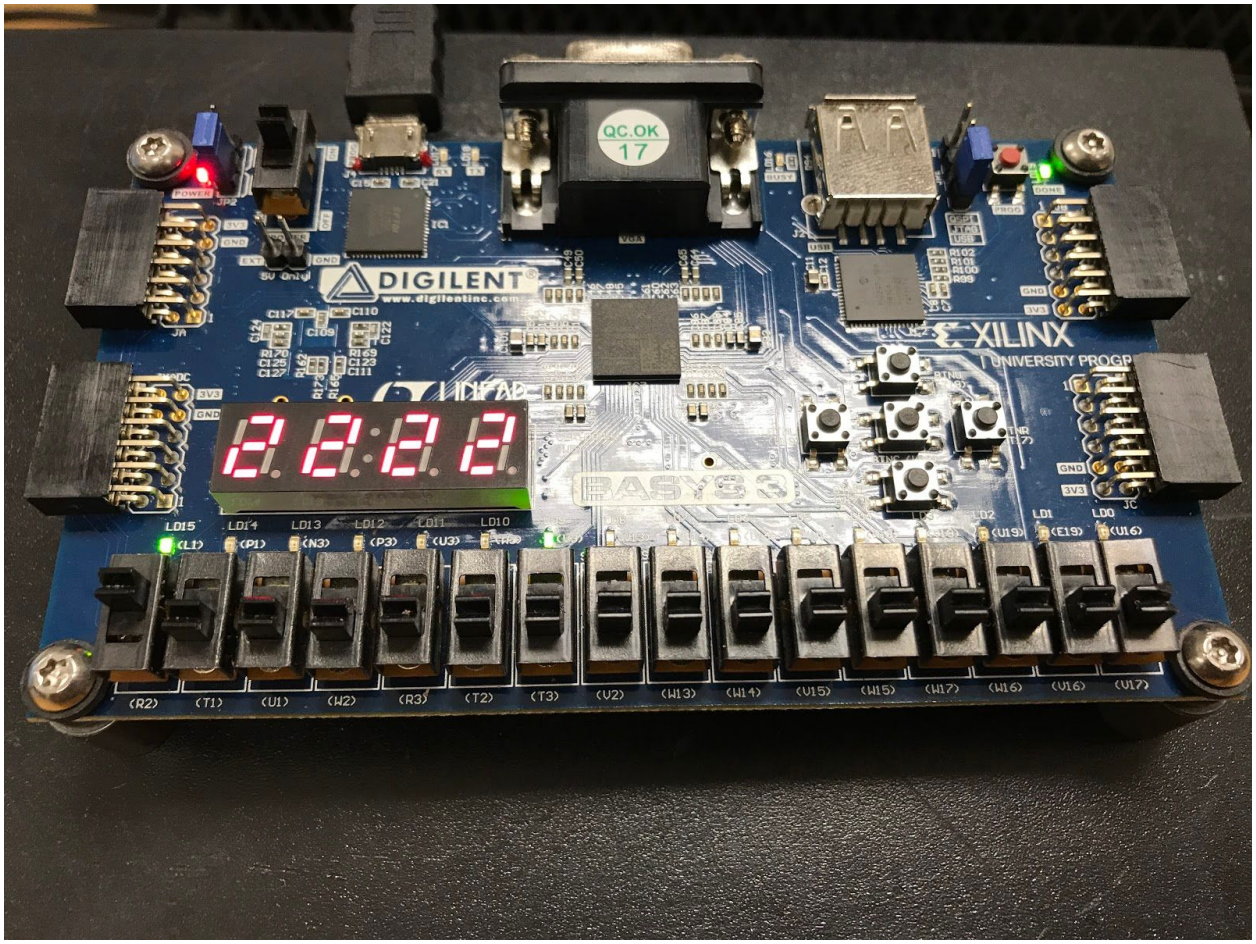


Figure 2.2) INPUT = 1 and CLOCK is high. Here we see that the Command issued is (10) and the State is (2)

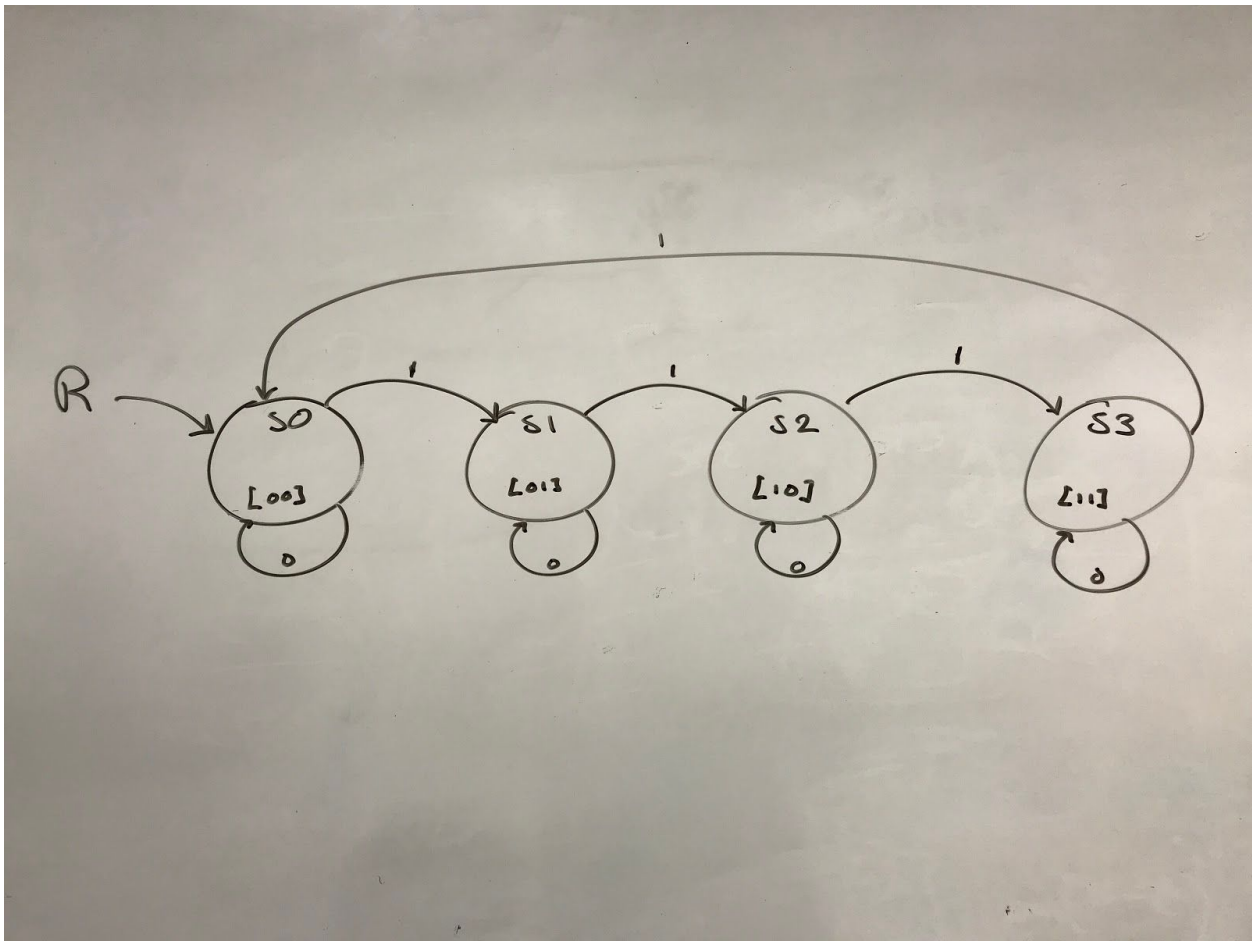
PART 3: ALU-FSM Combined

TRUTH TABLE for ALU

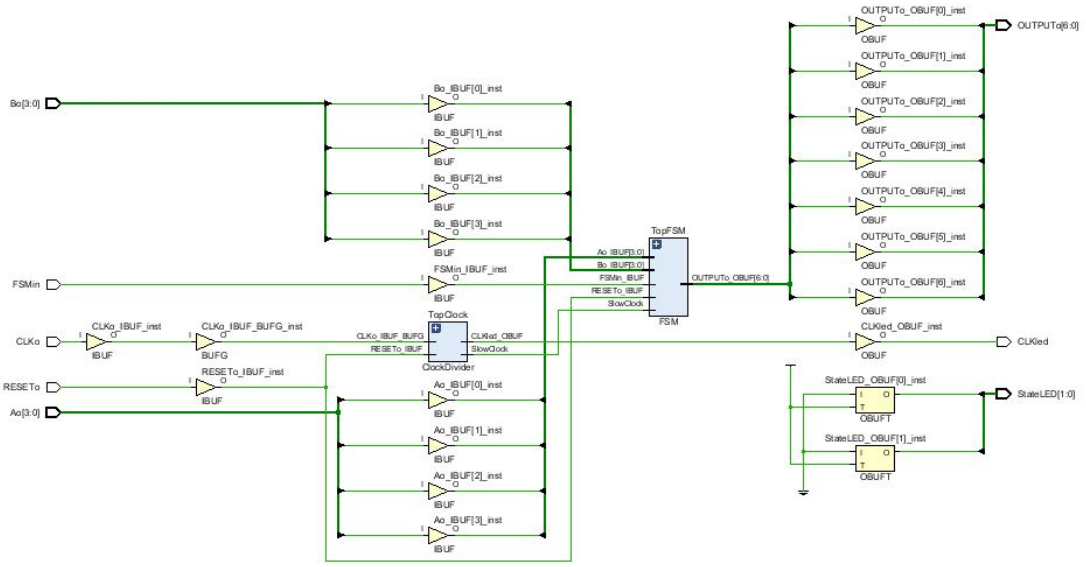
For two inputs A and B

SELECT(CASE)	OUTPUT (OPERATION)
00	XOR
01	NAND
10	ADD
11	SUBTRACT

STATE DIAGRAM



Schematic



PIN ASSIGNMENTS

All ports (21)					
Ao (4)	IN				
Ao[3]	IN				W17
Ao[2]	IN				W16
Ao[1]	IN				V16
Ao[0]	IN				V17
Bo (4)	IN				
Bo[3]	IN				R2
Bo[2]	IN				T1
Bo[1]	IN				U1
Bo[0]	IN				W2
OUTPUTo (7)	OUT				
OUTPUTo[6]	OUT				U7
OUTPUTo[5]	OUT				V5
OUTPUTo[4]	OUT				U5
OUTPUTo[3]	OUT				V8
OUTPUTo[2]	OUT				U8
OUTPUTo[1]	OUT				W6
OUTPUTo[0]	OUT				W7
StateLED (2)	OUT				
StateLED[1]	OUT				E19
StateLED[0]	OUT				U16
Scalar ports (4)					
CLKled	OUT				V3
CLKo	IN				W5
FSMin	IN				T3
RESETo	IN				V2

RESULTS

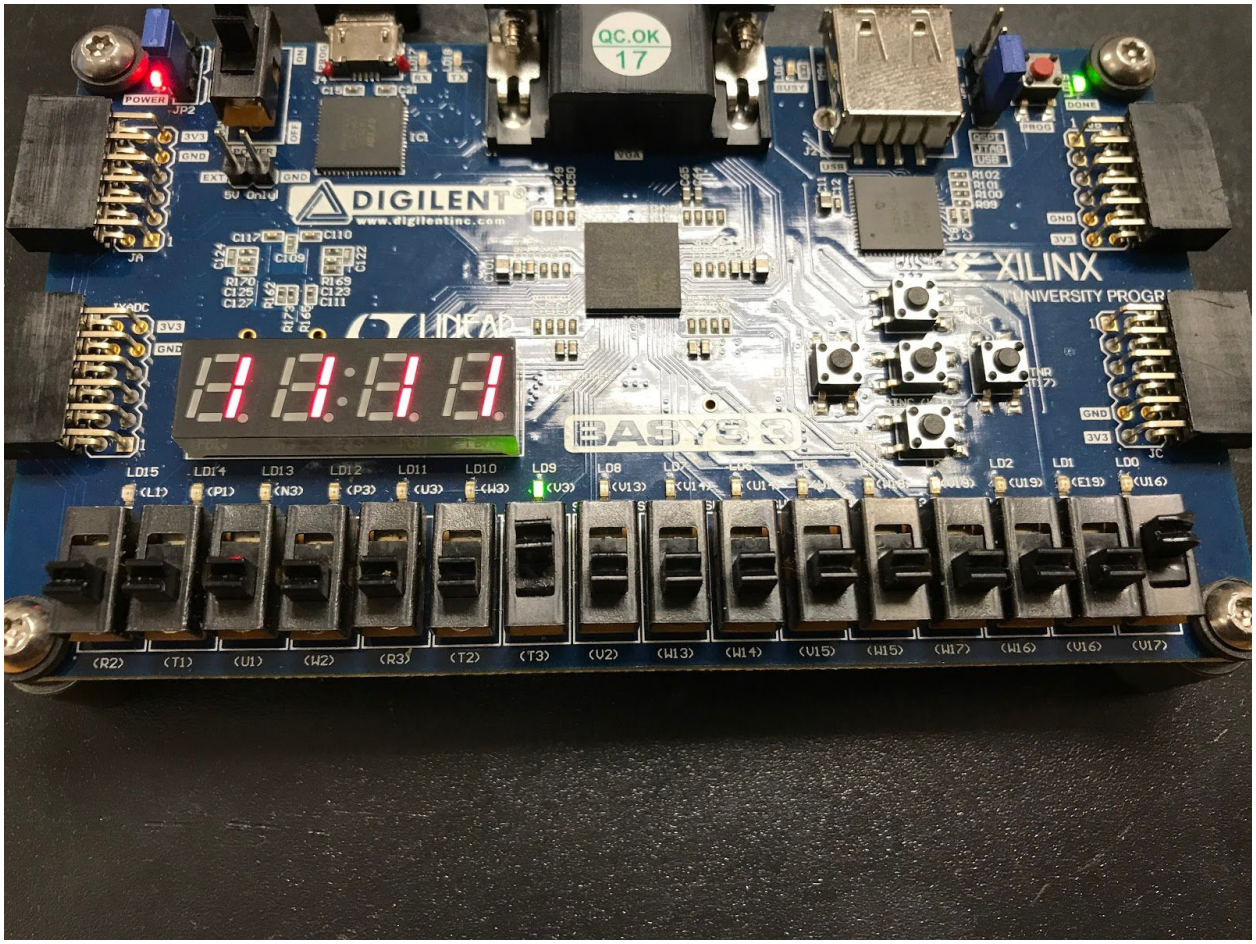


Figure 3.1) INPUT = 1 and CLOCK is high. State(00). B(0000) XOR A(0001) = (0001) or 1 in HEX.

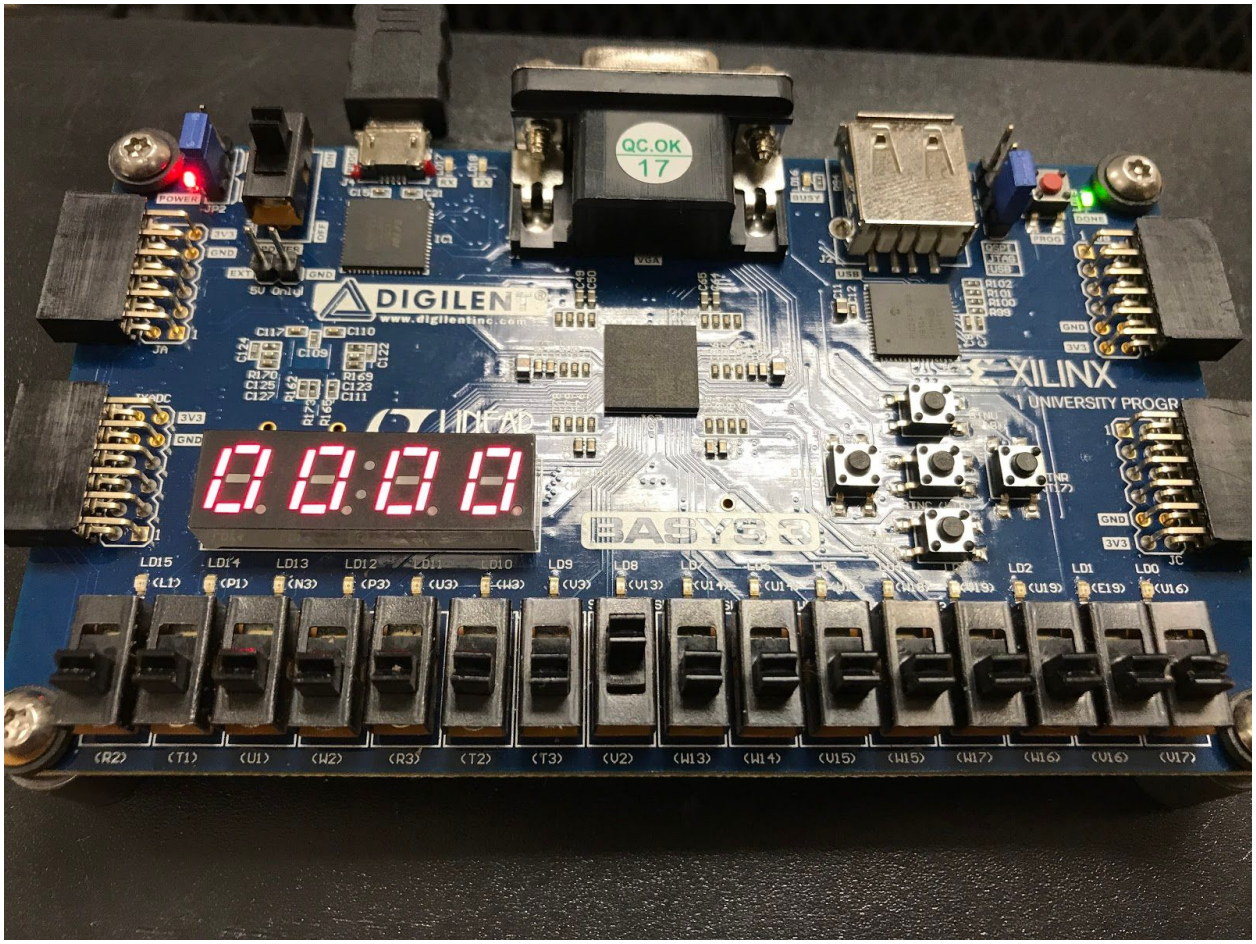


Figure 3.2) INPUT = 0 and CLOCK is high but RESET = 1 State(00). B(0000) XOR A(0000) = (0000) or 0 in HEX.

Conclusion:

In conclusion, this continues to strengthen our use of components and understanding of\ FSM, and how they can be used as a control unit (in this case the ALU). This very difficult lab should give us a cushion for the labs to come, when it comes to implementing FSMs.

Appendices:

PART1

ALU

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;


entity ALU is

  port(A,B : in std_logic_vector(3 downto 0);
       SEL : in std_logic_vector(1 downto 0);
          OUTPUT: out std_logic_vector (3 downto 0));
end;

architecture Behavioral of ALU is
  signal OutTemp, XOROutTemp : std_logic_vector(3 downto 0); --holds value for output of A
and B XORed, which will be shifted. Also a regular output temp signal
  begin
  process(SEL)
  begin
       if(SEL = "00") then
                OutTemp <= A XOR B; --XOR
                XOROutTemp <= OutTemp;
       elsif(SEL = "01") then
                OutTemp <= A NAND B; --NAND
       elsif(SEL = "10") then
          OutTemp <= (A NAND B) + "0100"; --ADD 4 to the inputs NANDed
       elsif(SEL = "11") then
          OutTemp <= (A NAND B) - "0010"; --SUBTRACT 2 to the inputs NANDed
       end if;
  end process;
  OUTPUT <= OutTemp;
end Behavioral;
```

DISPLAY

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

entity Display is
    port(Input :in std_logic_vector(3 downto 0);
         Output :out std_logic_vector(6 downto 0));
end Display;

architecture Behavioral of Display is
   begin
   process (Input)
   begin
   case Input is
   --Individual Segments
  when "0000" =>  Output <= "1000000"; --0
   when "0001" => Output <= "1111001"; --1
   when "0010" => Output <= "0100100";--2
   when "0011" => Output <= "0110000"; --3
   when "0100" => Output <= "0011001"; --4
   when "0101" => Output <= "0010010";--5
   when "0110" => Output <= "0000010";--6
   when "0111" => Output <= "1111000";--7
   when "1000" => Output <= "0000000";--8
   when "1001" => Output <= "0010000";--9
   when "1010" => Output <= "0100000";--10 A
   when "1011" => Output <= "0000011";--11 B
   when "1100" => Output <= "1000110";--12 C
   when "1101" => Output <= "0100001";--13 D
   when "1110" => Output <= "0000110";--14 E
   when "1111" => Output <= "0001110";--15 F
   end case;
   end process;
   end Behavioral;
```

TOPLEVEL

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

entity TopLevel is
   port(Ao,Bo : in std_logic_vector(3 downto 0);
        SELo : in std_logic_vector(1 downto 0);
      OUTPUTo: out std_logic_vector (6 downto 0));--for display
end TopLevel;

architecture Behavioral of TopLevel is
   signal ALU_OUT_DISPLAY_IN : std_logic_vector(3 downto 0);--signal for the output of the
ALU which is also the input for the Display

   --ALU component
   component ALU port(A,B : in std_logic_vector(3 downto 0);
                      SEL : in std_logic_vector(1 downto 0);
                      OUTPUT: out std_logic_vector (3 downto 0));
   end component;

   component Display port(Input :in std_logic_vector(3 downto 0);
                          Output :out std_logic_vector(6 downto 0));
   end component;


begin

   TopALU : ALU port
map(A=>Ao,B=>Bo,SEL=>SELo,OUTPUT=>ALU_OUT_DISPLAY_IN);
   TopDisplay : Display port map(Input=>ALU_OUT_DISPLAY_IN,Output=>OUTPUTo);


end Behavioral;
```

CONSTRAINT FILES

```
set_property IOSTANDARD LVCMOS33 [get_ports {Ao[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Ao[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Ao[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Ao[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Bo[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Bo[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Bo[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Bo[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SELo[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SELo[0]}]
set_property PACKAGE_PIN W17 [get_ports {Ao[3]}]
set_property PACKAGE_PIN W16 [get_ports {Ao[2]}]
set_property PACKAGE_PIN V16 [get_ports {Ao[1]}]
set_property PACKAGE_PIN V17 [get_ports {Ao[0]}]
set_property PACKAGE_PIN R2 [get_ports {Bo[3]}]
set_property PACKAGE_PIN T1 [get_ports {Bo[2]}]
set_property PACKAGE_PIN U1 [get_ports {Bo[1]}]
set_property PACKAGE_PIN W2 [get_ports {Bo[0]}]
set_property PACKAGE_PIN U7 [get_ports {OUTPUTo[6]}]
set_property PACKAGE_PIN V5 [get_ports {OUTPUTo[5]}]
set_property PACKAGE_PIN U5 [get_ports {OUTPUTo[4]}]
set_property PACKAGE_PIN V8 [get_ports {OUTPUTo[3]}]
set_property PACKAGE_PIN U8 [get_ports {OUTPUTo[2]}]
set_property PACKAGE_PIN W6 [get_ports {OUTPUTo[1]}]
set_property PACKAGE_PIN W7 [get_ports {OUTPUTo[0]}]
set_property PACKAGE_PIN W13 [get_ports {SELo[1]}]
set_property PACKAGE_PIN W14 [get_ports {SELo[0]}]
```

PART 2

FSM

```
library ieee;
use ieee. std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee. std_logic_unsigned.all;


Entity FSM is--With Feedback
   Port(RESET, CLK, INPUT: in std_logic;
            OUTPUT: out std_logic_vector(1 downto 0));
End;
Architecture Behavioral of FSM is
   Signal CS,NS: std_logic_vector(1 downto 0); --Current and Next State

   Begin
   Process(RESET,CLK)
   Begin
         if (RESET = '1')then-- When RESET is active
                  CS<= "00";--Reset State Machine
         elsif(CLK='1' and CLK'EVENT) then
                  CS<=NS;--When Clock is active, move to next state
         end if;
   end process;

   process (CS,INPUT)
   begin
   case CS is
   when ("00")=>--State 0
      if (INPUT = '1') then
         NS<= "01";
      else
         NS <= CS; --Stay in State
      end if;
   When ("01")=>--State 1
      if (INPUT = '1') then
         NS<= "10";
      else
         NS <= CS;  --Stay in State
```

```
        end if;
    When ("10")=>--State 2
        if (INPUT = '1') then
            NS<= "11";
        else
            NS <= CS; --Stay in State
        end if;
    When ("11")=>--State 3
        if (INPUT = '1') then
            NS<= "00";
        else
            NS <= CS; --Stay in State
        end if;
    end case;
    end process;
    OUTPUT<= CS;
end Behavioral;
```

CLOCK DIVIDER

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

entity ClockDivider is
    port( clk : in STD_LOGIC ;
          RESET : in STD_LOGIC ;
          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 ( RESET = '1') then
                --Reset the Counter
                cnt := "00000000000000000000000000" ;
            else
                --Start Counting by incrementing by 1
                cnt := STD_LOGIC_VECTOR( unsigned ( cnt ) + 1);
            end if ;


                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;
```

DISPLAY

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

entity Display is
    port(Input :in std_logic_vector(1 downto 0);
         Output :out std_logic_vector(6 downto 0));
end Display;

architecture Behavioral of Display is
   begin
   process (Input)
   begin
   case Input is
   --Individual Segments
   when "00" =>  Output <= "1000000"; --0
   when "01" => Output <= "1111001"; --1
   when "10" => Output <= "0100100";--2
   when "11" => Output <= "0110000"; --3
   end case;
   end process;
   end Behavioral;
```

TOPLEVEL

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

entity TopLevel is
   port(CLKo,RESETo,INPUTo : in std_logic;
       CommandLed : out std_logic_vector(1 downto 0);--Shows Command on State Machine
(for ALU even though not used in part 2)
       CLKled : out std_logic;--Shows Clock
       OUTPUTo: out std_logic_vector (6 downto 0));--Shows State
end TopLevel;

architecture Behavioral of TopLevel is
   signal CommandTemp : std_logic_vector(1 downto 0);
   signal SlowClockTemp : std_logic;

   --FSM component
   component FSM is port(RESET, CLK, INPUT: in std_logic;
                  OUTPUT: out std_logic_vector(1 downto 0));
   end component;
   --Clock component
   component ClockDivider is port( clk : in STD_LOGIC ;
                        RESET : in STD_LOGIC ;
                        SlowClock , led0 : out STD_LOGIC );
   end component;
   --Display component
   component Display port(Input :in std_logic_vector(1 downto 0);
                  Output :out std_logic_vector(6 downto 0));
   end component;

begin

   TopFSM : FSM port map(RESET=>RESETo, CLK=> SlowClockTemp, INPUT=> INPUTo,
OUTPUT=>CommandTemp);
   TopClock : ClockDivider port map(CLK=>CLKo, RESET=> RESETo,SlowClock =>
SlowClockTemp, led0=>CLKled );
   TopDisplay : Display port map(Input=>CommandTemp, Output=>OUTPUTo);
```

```
    CommandLed <= CommandTemp; --Sends the value of the command (output of FSM) to the
LED


end Behavioral;
```

CONSTRAINT FILE

```
set_property IOSTANDARD LVCMOS33 [get_ports {CommandLed[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {CommandLed[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports CLKled]
set_property IOSTANDARD LVCMOS33 [get_ports CLKo]
set_property IOSTANDARD LVCMOS33 [get_ports INPUTo]
set_property IOSTANDARD LVCMOS33 [get_ports RESETo]
set_property PACKAGE_PIN L1 [get_ports {CommandLed[1]}]
set_property PACKAGE_PIN P1 [get_ports {CommandLed[0]}]
set_property PACKAGE_PIN U7 [get_ports {OUTPUTo[6]}]
set_property PACKAGE_PIN V5 [get_ports {OUTPUTo[5]}]
set_property PACKAGE_PIN U5 [get_ports {OUTPUTo[4]}]
set_property PACKAGE_PIN V8 [get_ports {OUTPUTo[3]}]
set_property PACKAGE_PIN U8 [get_ports {OUTPUTo[2]}]
set_property PACKAGE_PIN W6 [get_ports {OUTPUTo[1]}]
set_property PACKAGE_PIN W7 [get_ports {OUTPUTo[0]}]
set_property PACKAGE_PIN V3 [get_ports CLKled]
set_property PACKAGE_PIN W5 [get_ports CLKo]
set_property PACKAGE_PIN R2 [get_ports INPUTo]
set_property PACKAGE_PIN T1 [get_ports RESETo]
```

PART 3

ALU

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity ALU is
   generic ( constant X: natural := 3); --for shifts and roatates, number of bits is 3
   port(A,B : in std_logic_vector(3 downto 0);
         SEL : in std_logic_vector(1 downto 0);
            OUTPUT: out std_logic_vector (3 downto 0));
end;

architecture Behavioral of ALU is
   signal OutTemp, XOROutTemp : std_logic_vector(3 downto 0); --holds value for output of A
and B XORed, which will be shifted. Also a regular output temp signal
   begin
   process(SEL)
   begin
         if(SEL = "00") then
                  OutTemp <= A XOR B; --XOR
                  XOROutTemp <= OutTemp;
         elsif(SEL = "01") then
                  OutTemp <= A NAND B; --NAND
         elsif(SEL = "10") then
            OutTemp <= (A NAND B) + "0100"; --ADD 4 to the inputs NANDed
                  --OutTemp <= std_logic_vector(unsigned(XOROutTemp) sll X);--Shift Left
         elsif(SEL = "11") then
            OutTemp <= (A NAND B) - "0010"; --SUBTRACT 2 to the inputs NANDed
                  --OutTemp <= std_logic_vector(unsigned(XOROutTemp) srl X);--Shift Right
         end if;
   end process;
   OUTPUT <= OutTemp;
end Behavioral;
```

FSM

```
library ieee;
use ieee. std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee. std_logic_unsigned.all;

Entity FSM is
   Port(RESET, CLK, INPUT: in std_logic;
            OUTPUT: out std_logic_vector(1 downto 0));
End;
Architecture Behavioral of FSM is
   Signal CS,NS: std_logic_vector(1 downto 0); --Current and Next State

   Begin
   Process(RESET,CLK)
   Begin
         if (RESET = '1')then-- When RESET is active
                  CS<= "00";--Reset State Machine
         elsif(CLK='1' and CLK'EVENT) then
                  CS<=NS;--When Clock is active, move to next state
         end if;
   end process;

   process (CS,INPUT)
   begin
   case CS is
   when ("00")=>--State 0
      if (INPUT = '1') then
         NS<= "01";
      else
         NS <= CS; --Stay in State
      end if;
   When ("01")=>--State 1
      if (INPUT = '1') then
         NS<= "10";
      else
         NS <= CS;  --Stay in State
      end if;
   When ("10")=>--State 2
```

```
        if (INPUT = '1') then
            NS<= "11";
        else
            NS <= CS; --Stay in State
        end if;
    When ("11")=>--State 3
        if (INPUT = '1') then
            NS<= "00";
        else
            NS <= CS; --Stay in State
        end if;
    end case;
    end process;
    OUTPUT<= CS;
end Behavioral;
```

CLOCK DIVIDER

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

entity ClockDivider is
    port( clk : in STD_LOGIC ;
          RESET : in STD_LOGIC ;
          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 ( RESET = '1') then
                --Reset the Counter
                cnt := "00000000000000000000000000" ;
            else
                --Start Counting by incrementing by 1
                cnt := STD_LOGIC_VECTOR( unsigned ( cnt ) + 1);
            end if ;


                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;
```

DISPLAY

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

entity Display is
    port(Input :in std_logic_vector(3 downto 0);
         Output :out std_logic_vector(6 downto 0));
end Display;

architecture Behavioral of Display is
   begin
   process (Input)
   begin
   case Input is
   --Individual Segments
  when "0000" =>  Output <= "1000000"; --0
   when "0001" => Output <= "1111001"; --1
   when "0010" => Output <= "0100100";--2
   when "0011" => Output <= "0110000"; --3
   when "0100" => Output <= "0011001"; --4
   when "0101" => Output <= "0010010";--5
   when "0110" => Output <= "0000010";--6
   when "0111" => Output <= "1111000";--7
   when "1000" => Output <= "0000000";--8
   when "1001" => Output <= "0010000";--9
   when "1010" => Output <= "0100000";--10 A
   when "1011" => Output <= "0000011";--11 B
   when "1100" => Output <= "1000110";--12 C
   when "1101" => Output <= "0100001";--13 D
   when "1110" => Output <= "0000110";--14 E
   when "1111" => Output <= "0001110";--15 F
   end case;
   end process;
   end Behavioral;
```

TOP LEVEL

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

entity TopLevel is
   port(Ao,Bo : in std_logic_vector(3 downto 0);--Inputs for ALU
        FSMin : in std_logic;--Input for FSM
        CLKo,RESETo : in std_logic;
        CLKled : out std_logic;--Shows Clock
        StateLED : out std_logic_vector(1 downto 0); --Shows the current State, the output of the
FSM
        OUTPUTo: out std_logic_vector (6 downto 0));--for display, shows result of ALU
end TopLevel;

architecture Behavioral of TopLevel is
   signal FSMoutTemp : std_logic_vector(1 downto 0); --The command of the FSM which is
used as the select bit of the ALU
   signal ALUoutTemp : std_logic_vector(3 downto 0); --The output of the ALU which will be
used as the input of the 7SegDisplay
   signal SlowClockTemp : std_logic; --For the SlowClock

   --ALU component
   component ALU port(A,B : in std_logic_vector(3 downto 0);
                SEL : in std_logic_vector(1 downto 0);
                OUTPUT: out std_logic_vector (3 downto 0));
   end component;
   --Display Component
   component Display port(Input :in std_logic_vector(3 downto 0);
                    Output :out std_logic_vector(6 downto 0));
   end component;
   --FSM component
   component FSM is port(RESET, CLK, INPUT: in std_logic;
                    OUTPUT: out std_logic_vector(1 downto 0));
   end component;
   --Clock component
   component ClockDivider is port( clk : in STD_LOGIC ;
                           RESET : in STD_LOGIC ;
                           SlowClock , led0 : out STD_LOGIC );
```

```
    end component;


begin

    TopALU : ALU port map(A=>Ao, B=>Bo, SEL=> FSMoutTemp, OUTPUT=>ALUoutTemp
);
    TopDisplay : Display port map(Input=> ALUoutTemp, Output=>OUTPUTo);
    TopFSM : FSM port map(RESET=>RESETo, CLK=>SlowClockTemp, INPUT=> FSMin,
OUTPUT=>FSMoutTemp);
    TopClock : ClockDivider port map(clk=>CLKo, RESET=>RESETo,SlowClock =>
SlowClockTemp, led0=>CLKled );

    StateLED <= FSMoutTemp;
end Behavioral;
```

CONSTRAINT FILES

```
set_property IOSTANDARD LVCMOS33 [get_ports {Ao[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Ao[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Ao[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Ao[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Bo[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Bo[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Bo[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Bo[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {OUTPUTo[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {StateLED[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {StateLED[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports CLKled]
set_property IOSTANDARD LVCMOS33 [get_ports CLKo]
set_property IOSTANDARD LVCMOS33 [get_ports FSMin]
set_property IOSTANDARD LVCMOS33 [get_ports RESETo]
set_property PACKAGE_PIN W17 [get_ports {Ao[3]}]
set_property PACKAGE_PIN W16 [get_ports {Ao[2]}]
set_property PACKAGE_PIN V16 [get_ports {Ao[1]}]
set_property PACKAGE_PIN V17 [get_ports {Ao[0]}]
set_property PACKAGE_PIN R2 [get_ports {Bo[3]}]
set_property PACKAGE_PIN T1 [get_ports {Bo[2]}]
set_property PACKAGE_PIN U1 [get_ports {Bo[1]}]
set_property PACKAGE_PIN W2 [get_ports {Bo[0]}]
set_property PACKAGE_PIN U7 [get_ports {OUTPUTo[6]}]
set_property PACKAGE_PIN V5 [get_ports {OUTPUTo[5]}]
set_property PACKAGE_PIN U5 [get_ports {OUTPUTo[4]}]
set_property PACKAGE_PIN V8 [get_ports {OUTPUTo[3]}]
set_property PACKAGE_PIN U8 [get_ports {OUTPUTo[2]}]
set_property PACKAGE_PIN W6 [get_ports {OUTPUTo[1]}]
set_property PACKAGE_PIN W7 [get_ports {OUTPUTo[0]}]
set_property PACKAGE_PIN E19 [get_ports {StateLED[1]}]
set_property PACKAGE_PIN U16 [get_ports {StateLED[0]}]
```

```
set_property PACKAGE_PIN V3 [get_ports CLKled]
set_property PACKAGE_PIN W5 [get_ports CLKo]
set_property PACKAGE_PIN T3 [get_ports FSMin]
set_property PACKAGE_PIN V2 [get_ports RESETo]
```