EEL 4712C - Digital Design: Lab Report 0

Cole Rottenberg
11062528

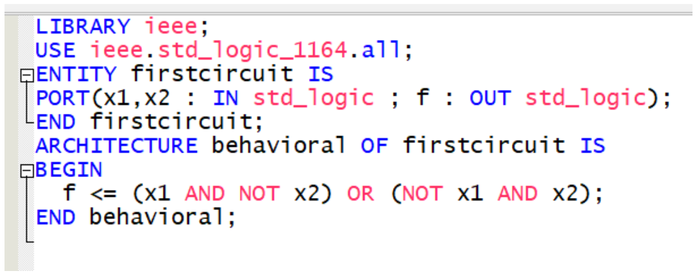
January 28, 2024

Report

Questions

Design and Implementation

firstCircuit Implementation:



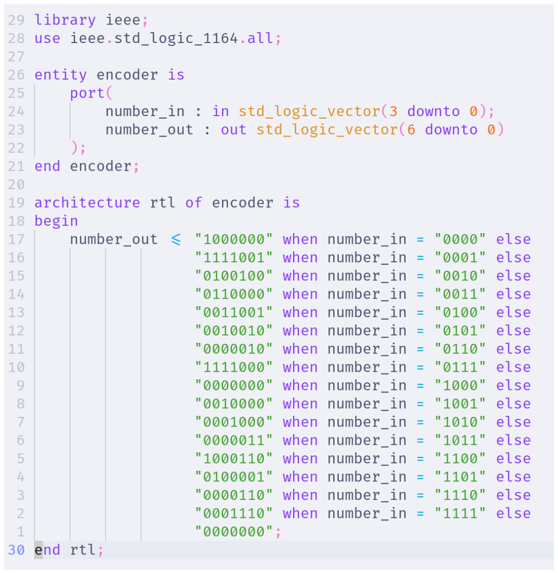
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY firstcircuit IS
PORT(x1,x2 : IN std_logic ; f : OUT std_logic);
END firstcircuit;
ARCHITECTURE behavioral OF firstcircuit IS
BEGIN
  f <= (x1 AND NOT x2) OR (NOT x1 AND x2);
END behavioral;
```

Figure 1: firstCircuit Implementation

Counter Implementation:

The counter circuit is comprised of 4 components interconnected with a top level entity. The components are: an encoder, a clock divider, a counter, and d flip flops. The encoder takes in the 4 bit counter output and outputs a 7 bit value representing the current count in the 7 segment led. The clock divider takes in the 50MHz clock and outputs a 1Hz clock. The counter takes in the 1Hz clock and outputs a 4 bit value representing the current count. The d flip flops take in the 1Hz clock and the 4 bit counter output and output the 4 bit counter output to the encoder. The top level entity takes in the 50MHz clock and outputs the 4 bit counter output and the 7 bit encoder output. The top level entity also connects the 4 bit counter output to the d flip flops and the 1Hz clock to the counter and d flip flops.

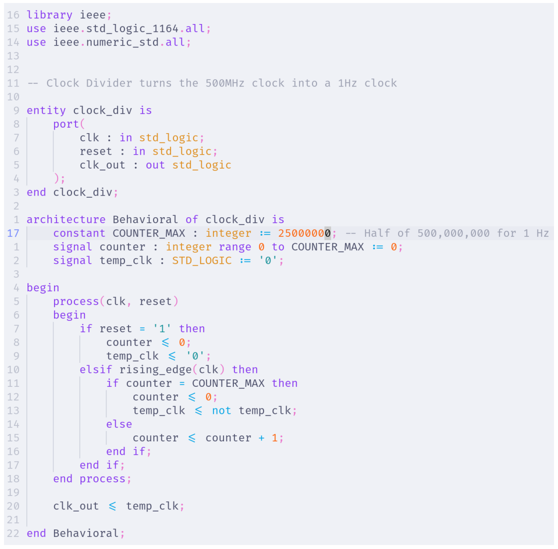
Encoder:



```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity encoder is
    port(
        number_in : in std_logic_vector(3 downto 0);
        number_out : out std_logic_vector(6 downto 0)
    );
end encoder;

architecture rtl of encoder is
begin
    number_out <= "1000000" when number_in = "0000" else
                  "1111001" when number_in = "0001" else
                  "0100100" when number_in = "0010" else
                  "0110000" when number_in = "0011" else
                  "0011001" when number_in = "0100" else
                  "0010010" when number_in = "0101" else
                  "0000010" when number_in = "0110" else
                  "1111000" when number_in = "0111" else
                  "0000000" when number_in = "1000" else
                  "0010000" when number_in = "1001" else
                  "0001000" when number_in = "1010" else
                  "0000011" when number_in = "1011" else
                  "1000110" when number_in = "1100" else
                  "0100001" when number_in = "1101" else
                  "0000110" when number_in = "1110" else
                  "0001110" when number_in = "1111" else
                  "0000000";
end rtl;
```

Figure 2: Encoder Implementation

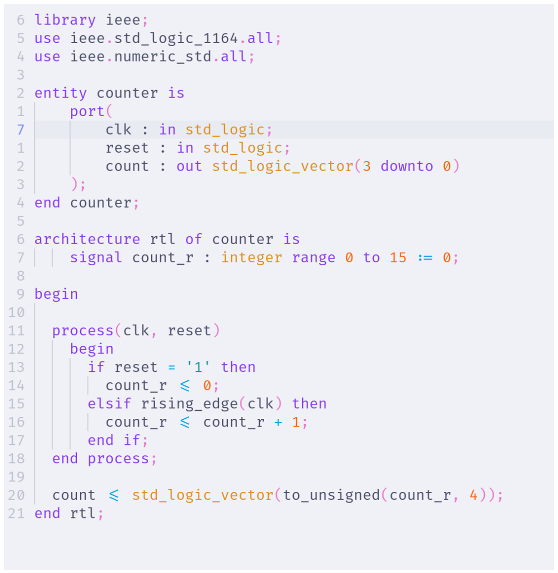
Clock Divider:



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


-- Clock Divider turns the 500MHz clock into a 1Hz clock

entity clock_div is
    port(
        clk : in std_logic;
        reset : in std_logic;
        clk_out : out std_logic
    );
end clock_div;

architecture Behavioral of clock_div is
    constant COUNTER_MAX : integer := 25000000; -- Half of 500,000,000 for 1 Hz
    signal counter : integer range 0 to COUNTER_MAX := 0;
    signal temp_clk : STD_LOGIC := '0';

begin
    process(clk, reset)
    begin
        if reset = '1' then
            counter <= 0;
            temp_clk <= '0';
        elsif rising_edge(clk) then
            if counter = COUNTER_MAX then
                counter <= 0;
                temp_clk <= not temp_clk;
            else
                counter <= counter + 1;
            end if;
        end if;
    end process;

    clk_out <= temp_clk;

end Behavioral;
```

Figure 3: Clock Divider Implementation

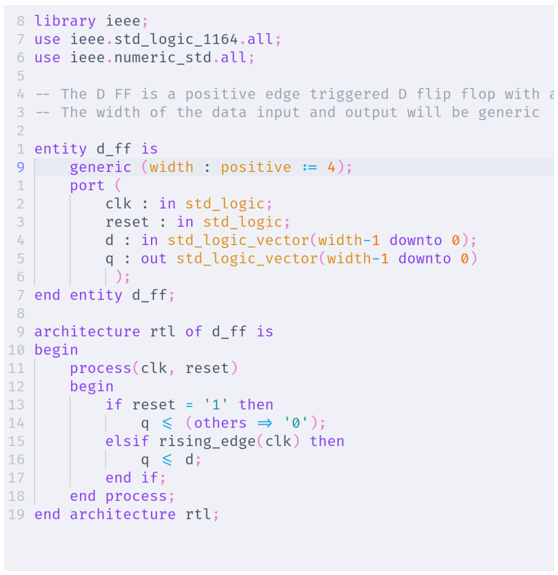
Counter:



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

entity counter is
    port(
        clk : in std_logic;
        reset : in std_logic;
        count : out std_logic_vector(3 downto 0)
    );
end counter;

architecture rtl of counter is
    signal count_r : integer range 0 to 15 := 0;

begin

  process(clk, reset)
    begin
      if reset = '1' then
        count_r <= 0;
      elsif rising_edge(clk) then
        count_r <= count_r + 1;
      end if;
  end process;

  count <= std_logic_vector(to_unsigned(count_r, 4));
end rtl;
```

Figure 4: Counter Implementation

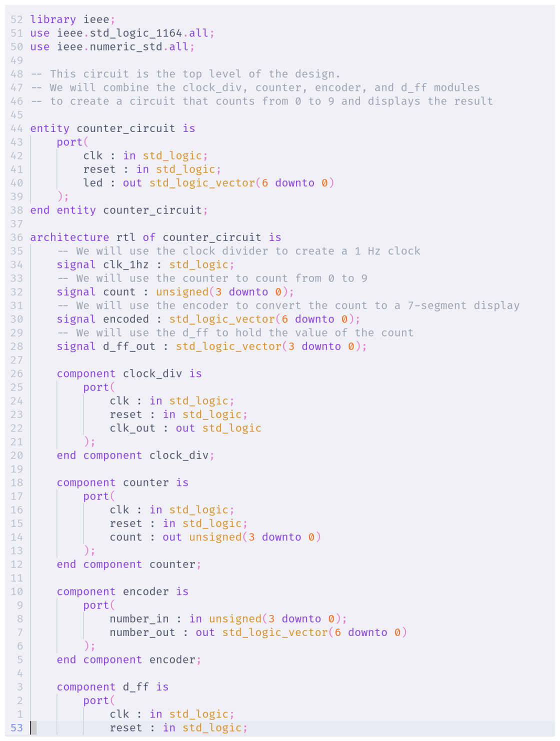
D Flip Flop:



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

-- The D FF is a positive edge triggered D flip flop with a
-- The width of the data input and output will be generic

entity d_ff is
    generic (width : positive := 4);
    port (
        clk : in std_logic;
        reset : in std_logic;
        d : in std_logic_vector(width-1 downto 0);
        q : out std_logic_vector(width-1 downto 0)
        );
end entity d_ff;

architecture rtl of d_ff is
begin
    process(clk, reset)
    begin
        if reset = '1' then
            q <= (others => '0');
        elsif rising_edge(clk) then
            q <= d;
        end if;
    end process;
end architecture rtl;
```

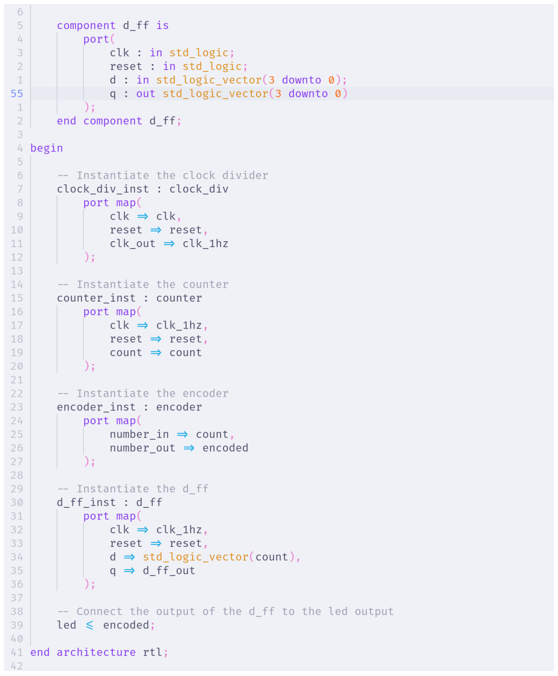
Figure 5: D Flip Flop Implementation

Top Level Entity:



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

-- This circuit is the top level of the design.
-- We will combine the clock_div, counter, encoder, and d_ff modules
-- to create a circuit that counts from 0 to 9 and displays the result

entity counter_circuit is
    port(
        clk : in std_logic;
        reset : in std_logic;
        led : out std_logic_vector(6 downto 0)
    );
end entity counter_circuit;

architecture rtl of counter_circuit is
    -- We will use the clock divider to create a 1 Hz clock
    signal clk_1hz : std_logic;
    -- We will use the counter to count from 0 to 9
    signal count : unsigned(3 downto 0);
    -- We will use the encoder to convert the count to a 7-segment display
    signal encoded : std_logic_vector(6 downto 0);
    -- We will use the d_ff to hold the value of the count
    signal d_ff_out : std_logic_vector(3 downto 0);

    component clock_div is
        port(
            clk : in std_logic;
            reset : in std_logic;
            clk_out : out std_logic
        );
    end component clock_div;

    component counter is
        port(
            clk : in std_logic;
            reset : in std_logic;
            count : out unsigned(3 downto 0)
        );
    end component counter;

    component encoder is
        port(
            number_in : in unsigned(3 downto 0);
            number_out : out std_logic_vector(6 downto 0)
        );
    end component encoder;

    component d_ff is
        port(
            clk : in std_logic;
            reset : in std_logic;
```

Figure 6: Top Level Entity Implementation 1



```vhdl

    component d_ff is
        port(
            clk : in std_logic;
            reset : in std_logic;
            d : in std_logic_vector(3 downto 0);
            q : out std_logic_vector(3 downto 0)
        );
    end component d_ff;

begin

    -- Instantiate the clock divider
    clock_div_inst : clock_div
        port map(
            clk => clk,
            reset => reset,
            clk_out => clk_1hz
        );

    -- Instantiate the counter
    counter_inst : counter
        port map(
            clk => clk_1hz,
            reset => reset,
            count => count
        );

    -- Instantiate the encoder
    encoder_inst : encoder
        port map(
            number_in => count,
            number_out => encoded
        );

    -- Instantiate the d_ff
    d_ff_inst : d_ff
        port map(
            clk => clk_1hz,
            reset => reset,
            d => std_logic_vector(count),
            q => d_ff_out
        );

    -- Connect the output of the d_ff to the led output
    led <= encoded;

end architecture rtl;
```

Figure 7: Top Level Entity Implementation 2

Reflection

The process of teaching myself VHDL has been time-consuming and feels directionless at time, however I managed to find Stitt's GitHub which contains a VHDL tutorial that has been very helpful. Looking back through slides has been helpful as well. I was entirely able to implement the counter. Looking back, I spent about two hours longer than I needed to in order to have a working circuit. In that time, I learned the process of architecture and entity creation, as well as the process of connecting components together. I also learned how to use the 7 segment led. I did not have any issues with the prelab.

Homework

1. **Problem 1:** A_123, A123_, c1__c2, and1
2. **Problem 2:** All are equivalent.
3. **Problem 3:** False.
4. **Problem 4:** Sequential logic happens in a sequence of time, while concurrent logic happens at the same time.
5. **Problem 5:**

```
entity firstCircuit is
  port (a, b, c, d: in std_logic;
        g: out bit);
end firstCircuit;

architecture bhv of firstCircuit is
begin
  e <= a and b;
  f <= c or e;
  g <= d and f;
end bhv;
```

6. **Problem 6:** X_3 must be added to the sensitivity list.

7. **Problem 7:**

X_1	X_2	X_3	C	A	B	D	F
0	0	0	0	1	1	1	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	1	1
0	1	1	1	0	0	0	0
1	0	0	1	1	1	1	1
1	0	1	1	1	0	1	1
1	1	0	1	0	1	1	1
1	1	1	1	0	0	0	0

Appendix