Assignment - 4 Report

Tanish Gupta , Divyanshu Agarwal

8th May 2022

1 Introduction

Primarily, there are 3 main objectives of this assignment:

1. Display the 4 BCD digits based on the 16 slider switches with highest brightness on the left most display and lowest on the rightmost display.
2. The digits continuously "rotate" giving an impression that when the digit moves to the left it starts becoming brighter.
3. If the input from slider switches change register at a suitable time so as to avoid looking like an abrupt change.

We successfully managed to fulfill all the objectives in this assignment :)

2 Strategy and Logic

Firstly, to control brightness, we decided to change the duty cycle for each digit. To rotate the digits, we introduced a new signal position, which will decided which LED to light corresponding to which set of 4 switches.
We faced a very fundamental issue in our code. We did not know that in VHDL, the mod operator does not work for all values. Specifically, the mod operator returns correct value for a mod b, iff b is a power of 2. Upon correction of this mistake, we could fix the abrupt changes of values on changing switches.

If we change the value through switches, we only change it after one cycle, so that we do not observe an abrupt change.

3 Code and Working

The main directory contains the design.vhd file, which has the VHDL code logic and the singleDisplay.vhd, which is the same file as submitted in assignment 2. The directory also has the constraint file and the generated bitstream.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_Std.all;

entity lightDisplay is
        Port (
            Number : in std_logic_vector(15 downto 0);
            Clk : in std_logic;
            LED : out std_logic_vector(6 downto 0);
            anode : out STD_LOGIC_Vector(3 downto 0));
```

```
end lightDisplay;

architecture structure of lightDisplay is
    component singleDisplay
        Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C : in STD_LOGIC;
           D : in STD_LOGIC;

           light_num : in integer;
           LED : out std_logic_vector(6 downto 0);
           anode : out STD_LOGIC_Vector(3 downto 0));

end component singleDisplay;

    signal light_num        : integer;
    signal digit_A          : std_logic;
    signal digit_B          : std_logic;
    signal digit_C          : std_logic;
    signal digit_D          : std_logic;
    signal counter          : integer :=0;
    signal counter_new      : integer := 0 ;
    signal counter_2        : integer := 0 ;
    signal Sampled_number   : std_logic_vector(15 downto 0);


    begin
        process(clk)
            variable position : integer := 0;
            variable state2   : integer := 0;
                    begin

                    if(Clk'event  and rising_Edge(clk)) then
                        counter<= ((counter+1)) ;

                        counter_new<= ((counter) / 400000) mod 4  ;
                        state2 := ((counter)/400000) mod 1024 ;
                        counter_2 <= ((counter/10000)mod 32) ;
                    end if ;

                    if(state2 = 0) then Sampled_number<=Number ; end if;

                    position := (state2/256) mod 4 ;

                    case (counter_new) is  -- this loop tells which digit to display
                        when 0 =>
                        digit_A <= Sampled_Number(15);
                        digit_B <= Sampled_Number(14);
                        digit_C <= Sampled_Number(13);
                        digit_D <= Sampled_Number(12);

                        when 1 =>
                        digit_A <= Sampled_Number(11);
                        digit_B <= Sampled_Number(10);
```

```
        digit_C <= Sampled_Number(9);
        digit_D <= Sampled_Number(8);

        when 2 =>
        digit_A <= Sampled_Number(7);
        digit_B <= Sampled_Number(6);
        digit_C <= Sampled_Number(5);
        digit_D <= Sampled_Number(4);

        when others =>
        digit_A <= Sampled_Number(3);
        digit_B <= Sampled_Number(2);
        digit_C <= Sampled_Number(1);
        digit_D <= Sampled_Number(0);

    end case;

    case(position) is
        -- this loop tells where to display and for how much time

        when 0 =>
            -- display starting from 1st light
            -- (leftmost digit in display is the 1st digit)

            case (counter_new) is

                when 0 =>
                    light_num <= 0 ;  -- display for maxtime

                when 1 =>
                    if(counter_2 < 10) then light_num <=1 ;
                    else light_num <= 4 ;
                    end if ;

                when 2 =>
                    if(counter_2 < 5) then light_num <=2 ;
                    else light_num <= 4 ;
                    end if ;

                when others =>
                    if(counter_2 < 2) then light_num <=3 ;
                    else light_num <= 4 ;
                    end if ; -- for mintime

            end case;

        when 1 =>
            -- display starting from 2nd light
            -- (leftmost digit in display is the last digit)

            case (counter_new) is

                when 0 =>
                    if(counter_2 < 2) then light_num <=3 ;
```

```
                else light_num <= 4 ;
                end if ;

            when 1 =>
                light_num <= 0 ;

            when 2 =>
                if(counter_2 < 10) then light_num <= 1 ;
                else light_num <= 4 ;
                end if ;
                -- mintime is with 3rd digit of number

            when others =>
                if(counter_2 < 5) then light_num <= 2 ;
                else light_num <= 4 ;
                end if ;
                -- max time now is with the 4th digit in the number
                -- as it will be at first place

        end case;

    when 2 =>
        -- display starting from 3rd light
        -- (leftmost digit in display is 3rd digit)

        case (counter_new) is

            when 0 =>
                if(counter_2 < 5) then light_num <= 2 ;
                else light_num <= 4 ;
                end if ;

            when 1 =>
                if(counter_2 < 2) then light_num <=3 ;
                else light_num <= 4 ;
                end if ;

            when 2 =>
                light_num <= 0 ;
                -- max time is with the 3rd digit

            when others =>
                if(counter_2 < 10) then light_num <= 1 ;
                else light_num <= 4 ;
                end if ;

        end case;

    when others =>
    -- display starting from 4th light
    -- (leftmost digit in display is 2nd digit)

        case (counter_new) is
```

```
                        when 0 =>
                            if(counter_2 < 10) then light_num <= 1 ;
                            else light_num <= 4 ;
                            end if ;
                            -- min time is with first digit

                        when 1 =>
                            if(counter_2 < 5) then light_num <= 2 ;
                            else light_num <= 4 ;
                            end if ;
                            -- max time is with second digit

                        when 2 =>
                            if(counter_2 < 2) then light_num <= 3 ;
                            else light_num <= 4 ;
                            end if ;

                        when others => light_num <= 0 ;

                    end case;

                end case;

        end process;




    Single_display: entity work.singleDisplay(Design_arch) port map(
        digit_A ,digit_B ,digit_C ,digit_D ,light_num ,LED ,anode
        );


end structure;
```

4 Simulation

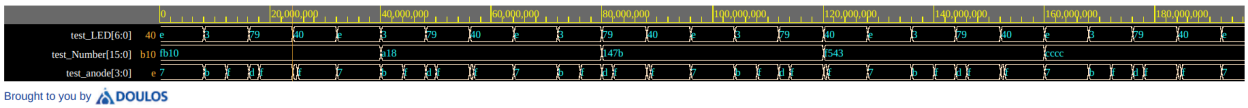

Figure 1: Generated EP Wave

For the purposes of generating EP Wave, we changed the refresh rate to 4 microsecond, instead of 4 millisecond, so that we can see all the transitions in a single frame. The synthesis has been correctly implemented, though.

The report generated on Vivado is as follows:

```
Copyright 1986-2019 Xilinx, Inc. All Rights Reserved.
----------------------------------------------------------------------------------------------------
| Tool Version : Vivado v.2019.1 (lin64) Build 2552052 Fri May 24 14:47:09 MDT 2019
| Date         : Sun May  8 21:42:36 2022
| Host         : divyanshu-HP-ENVY-x360-Convertible-13-bd0xxx
                 running 64-bit Ubuntu 20.04.2 LTS
| Command      : report_utilization -file lightDisplay_utilization_synth.rpt
                 -pb lightDisplay_utilization_synth.pb
| Design       : lightDisplay
| Device       : 7a35tcpg236-2
| Design State : Synthesized
----------------------------------------------------------------------------------------------------

Utilization Design Information

Table of Contents
-----------------

1. Slice Logic
1.1 Summary of Registers by Type
2. Memory
3. DSP
4. IO and GT Specific
5. Clocking
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists

1. Slice Logic
--------------

+----------------------------+------+-------+-----------+-------+
|          Site Type         | Used | Fixed | Available | Util% |
+----------------------------+------+-------+-----------+-------+
| Slice LUTs*                |   91 |     0 |     20800 |  0.44 |
|   LUT as Logic             |   91 |     0 |     20800 |  0.44 |
|   LUT as Memory            |    0 |     0 |      9600 |  0.00 |
| Slice Registers            |   63 |     0 |     41600 |  0.15 |
|   Register as Flip Flop    |   47 |     0 |     41600 |  0.11 |
```

Register as Latch	16	0	41600	0.04
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	-	-	-
0	-	-	Set
0	-	-	Reset
0	-	Set	-
0	-	Reset	-
0	Yes	-	-
0	Yes	-	Set
16	Yes	-	Reset
0	Yes	Set	-
47	Yes	Reset	-

2. Memory

Site Type	Used	Fixed	Available	Util%
Block RAM Tile	0	0	50	0.00
RAMB36/FIFO*	0	0	50	0.00
RAMB18	0	0	100	0.00

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

3. DSP

Site Type	Used	Fixed	Available	Util%
DSPs	0	0	90	0.00

4. IO and GT Specific

Site Type	Used	Fixed	Available	Util%
Bonded IOB	28	0	106	26.42
Bonded IPADs	0	0	10	0.00
Bonded OPADs	0	0	4	0.00
PHY_CONTROL	0	0	5	0.00
PHASER_REF	0	0	5	0.00
OUT_FIFO	0	0	20	0.00
IN_FIFO	0	0	20	0.00
IDELAYCTRL	0	0	5	0.00
IBUFDS	0	0	104	0.00
GTPE2_CHANNEL	0	0	2	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	20	0.00
PHASER_IN/PHASER_IN_PHY	0	0	20	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	250	0.00
IBUFDS_GTE2	0	0	2	0.00
ILOGIC	0	0	106	0.00
OLOGIC	0	0	106	0.00

5. Clocking

Site Type	Used	Fixed	Available	Util%
BUFGCTRL	1	0	32	3.13
BUFIO	0	0	20	0.00
MMCME2_ADV	0	0	5	0.00
PLLE2_ADV	0	0	5	0.00
BUFMRCE	0	0	10	0.00
BUFHCE	0	0	72	0.00
BUFR	0	0	20	0.00

6. Specific Feature

Site Type	Used	Fixed	Available	Util%
BSCANE2	0	0	4	0.00
CAPTUREE2	0	0	1	0.00
DNA_PORT	0	0	1	0.00
EFUSE_USR	0	0	1	0.00
FRAME_ECCE2	0	0	1	0.00
ICAPE2	0	0	2	0.00
PCIE_2_1	0	0	1	0.00
STARTUPE2	0	0	1	0.00
XADC	0	0	1	0.00

7. Primitives

Ref Name	Used	Functional Category
FDRE	47	Flop & Latch
LUT2	40	LUT
LUT1	36	LUT
CARRY4	22	CarryLogic
IBUF	17	IO
LUT5	16	LUT
LDCE	16	Flop & Latch
LUT6	14	LUT
OBUF	11	IO
LUT3	11	LUT
LUT4	8	LUT
BUFG	1	Clock

8. Black Boxes

Ref Name	Used

9. Instantiated Netlists

Ref Name	Used