Assignment - 2 Report

Tanish Gupta , Divyanshu Agarwal

28 April 2022

1 Introduction

The objective of this assignment is to display hexadecimal digits using 7-segment displays. The code is written in VHDL, simulated on edaplayground, synthesized using Vivado and displayed on BASYS3 FPGA Board.

2 Strategy and Logic

Firstly, the code was written in VHDL and simulated on EDAPlayground. A testbench was written to test the output. The generated EP Wave was then analysed for each LED. After this, we used Vivado to firstly synthesis the code. We then ran the implementation, and generated the BitStream. This BitStream was then programmed on the BASYS3 Board.
For the VHDL code part, first we wrote the logic and made a truthtable for each LED. The expressions for each LED were then minimized using K-Maps. This final logic was then written in DesignFile,vhd.
The constraints file was taken from moodle, and only the necessary parts were uncommented. The 4 switches on the rightmost side on the BASYS3 board were used.

3 Code and Working

The main directory contains the DesignFile.vhd file, which has the VHDL code logic. The directory also has the constraint file and the generated bitstream.

```
entity DesignFile is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C : in STD_LOGIC;
           D : in STD_LOGIC;

           LED : out std_logic_vector(6 downto 0);
           anode : out STD_LOGIC_Vector(3 downto 0));
end DesignFile;

architecture Design_arch of DesignFile is
    signal nA , nB , nC , nD : std_logic;

begin
        anode <= "1110";
        nA<=not(A);
        nB<=not(B);
        nC<=not(C);
        nD<=not(D);
```

```
    LED(0) <= not((A and nD) or
                  (A and nB and nC) or
                  (nA and B and D) or
                  (nA and C) or
                  (B and C) or
                  (nB and nD));

    LED(1) <= not((nA and nB) or
                     (nB and nD) or
                     (nA and nC and nD) or
                     (nA and C and D) or
                     (A and nC and D)) ;

    LED(2) <=  not((nA and nC) or
                       (nA and D) or
                       (nC and D) or
                       (nA and B) or
                       (A and nB)) ;

    LED(3) <= not((A and nC) or
                     (nA and nB and nD) or
                     (nB and C and D) or
                     (B and nC and D) or
                     (B and C and nD));

    LED(4) <= not((nB and nD) or
                     (C and nD) or
                     (A and C) or
                     (A and B)) ;

    LED(5) <= not((nA and B and nC) or
                     (B and nD) or
                     (A and nB) or
                     (A and C) or
                     (nC and nD)) ;

    LED(6) <= not((nB and C) or
                     (C and nD) or
                     (A and nB) or
                     (A and D) or
                     (nA and B and nC));

end Design_arch;
```

The working of the BASYS3 board can be found here - Video Link

4 Simulation

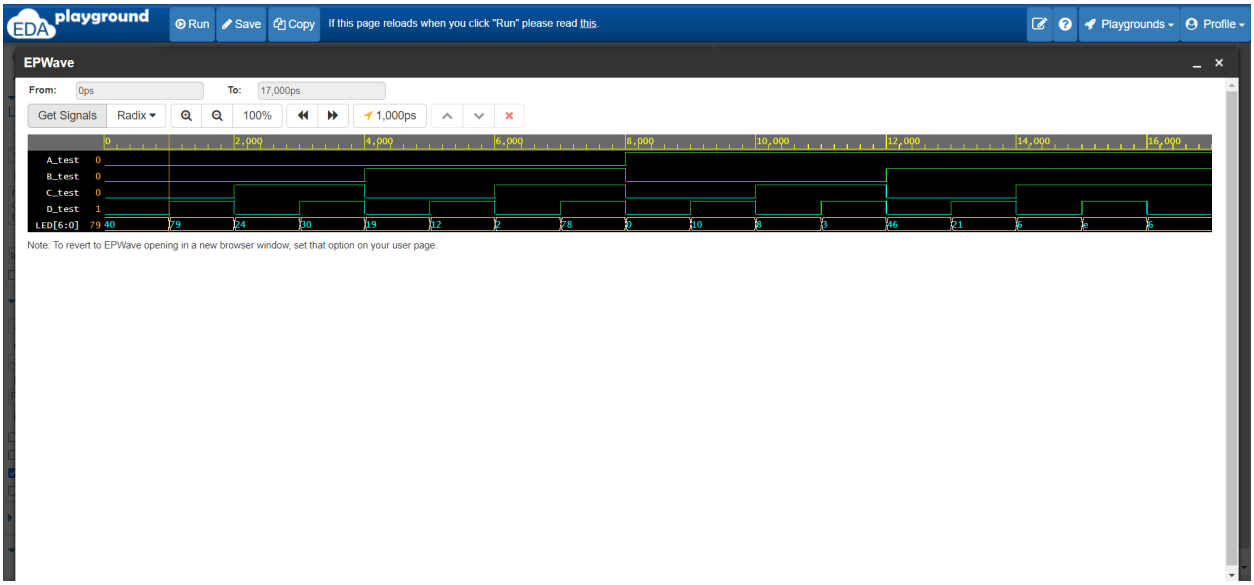

Figure 1: Generated EP Wave

This EPWave correctly produces the output. The output produced can be explained as follows:
LED is a vector of 7 elements, { LEDg, LEDf, LEDe, LEDd, LEDc, LEDb, LEDa }. Thus, when inputs are 0, the output is 40 (this is in hexadecimal, which in decimal is 64), whose binary equivalent is 1000000, which means that LEDg is '1' and all others LED signals are '0'. This is the exact negatation of expected value. This is because we need the current to flow in opposite direction, as given in the BASYS pdf uploaded on moodle.
Similar logics follow for other results.

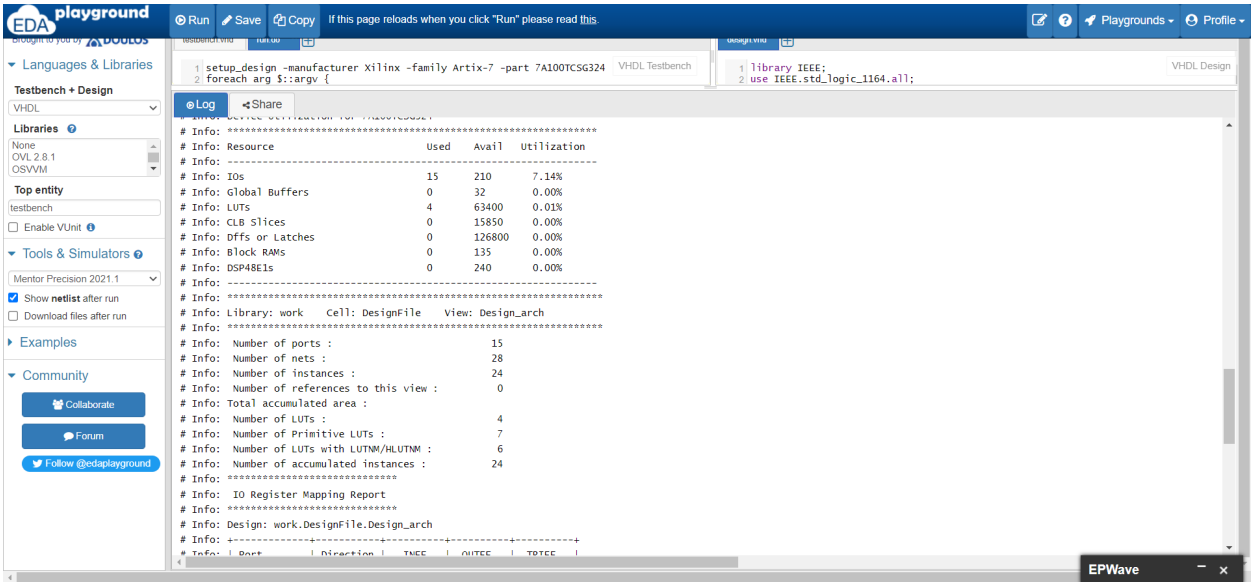

Figure 2: Generated EP Wave

This is the synthesis report generated on EDAPlayground. Our logic consists of 15 IO ports (4 BCD, 7 LEDs and 4 anodes). 7 Primitive LUTs have been used.