AHMET FARUK ÇOLAK-22102104

1 Purpose

The purpose of this lab session is to delve into the functionality of the seven-segment display on the BASYS3 board and to address the challenge of displaying 4-digit hexadecimal numbers due to the board's anode design limitation. Additionally, the lab aims to explore the concept of persistence of vision and its application in displaying multiple digits simultaneously. Furthermore, the lab will focus on understanding clock division on the BASYS3 board to generate slower clock signals for various applications.

2 Questions

A. What is the internal clock frequency of BASYS3?:
The internal clock frequency of BASYS3 is 100 MHz.

B. How can you create a slower clock signal from this one?
Clock division techniques can be employed, such as using counters to divide the frequency by a specified value or employing vector manipulation to achieve the desired frequency.

C. Can you create a clock with any arbitrary frequency lower than that of the internal clock? If not, which frequencies can you create?
It is not feasible to create a clock with any arbitrary frequency lower than that of the internal clock. The achievable frequencies are limited to those of the form 100 MHz / 2^n due to the constraints of clock division methods.

3 Methodolgy

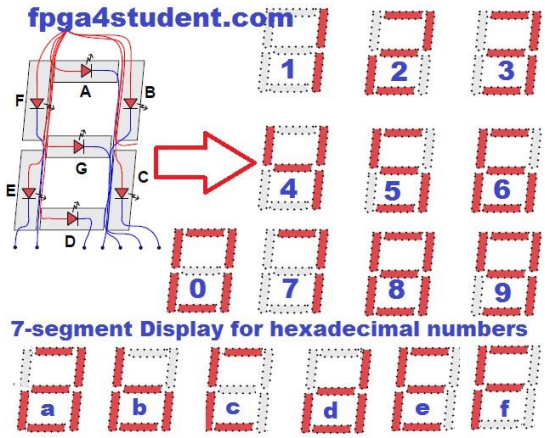

Figure1-BASYS3 7 segment display

1. **Understanding Seven-Segment Display:**

   - I studied the configuration and operation of the seven-segment display, focusing on the anode and cathode connections.
   - I analyzed the limitations imposed by the BASYS3 internal structure, which allowed either all digits to display the same input or only one digit to be displayed at a time.

2. **Exploring Persistence of Vision:**
    - I investigated the concept of persistence of vision and its relevance in creating the illusion of simultaneous digit display.
    - I experimented with different clock frequencies to achieve a blinking speed above the threshold of human perception.
3. **Implementing Clock Division:**
    - I developed VHDL code to implement clock division techniques, such as using counters or vector manipulation, to generate slower clock signals.
    - I designed multiplexers and decoders to control the display of digits based on the divided clock signals.

4 Results

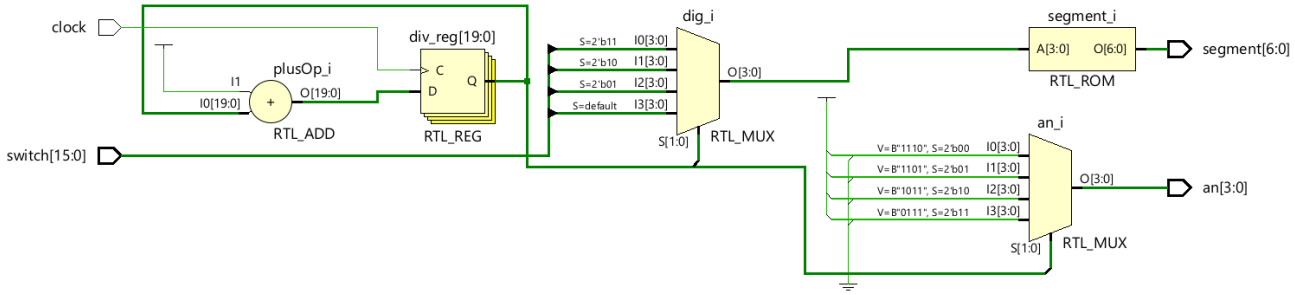

Figure2-RTL Schematic

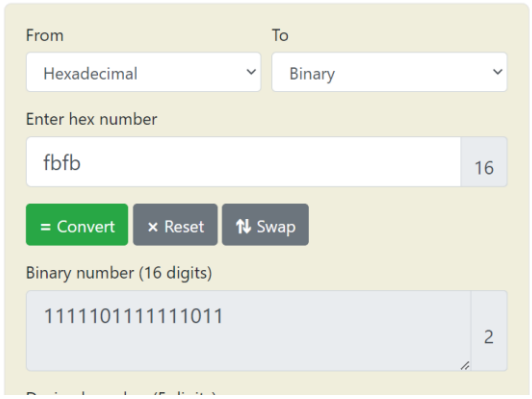

Figure-3 "fbfb" Hexadecimal to Binary

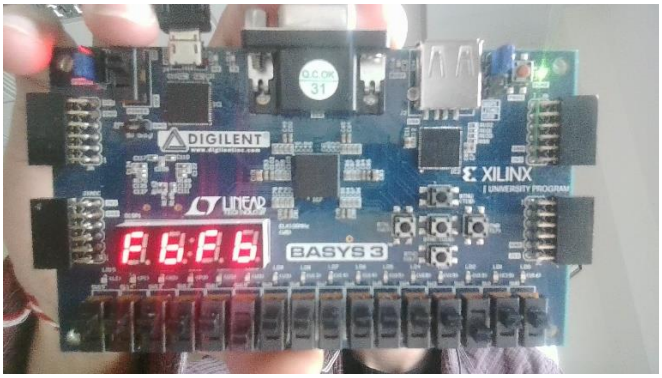

Figure-4 "FbFb"

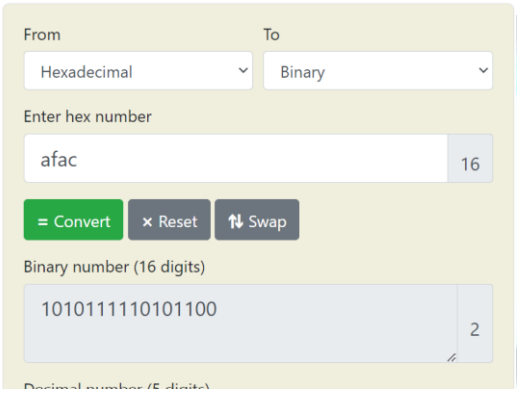

Figure-5 "afac" Hexadecimal to Binary

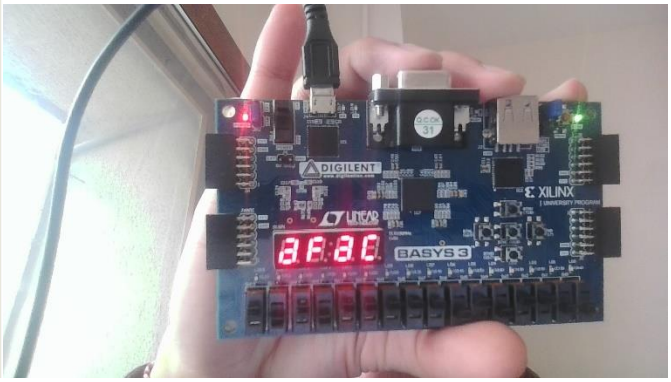

Figure-6 "aFaC"

5 Conclusion

In conclusion, this lab session provided valuable insights into the operation of the seven-segment display and clock division techniques on the BASYS3 board. Through practical experimentation and theoretical exploration, I gained a deeper understanding of how to effectively utilize these components in digital circuit design.

Appendix

6 Main Function

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 25.03.2024 10:12:08
-- Design Name:
-- Module Name: main_function - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
```

```
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity main_function is
   Port ( clock : in STD_LOGIC;
          switch : in STD_LOGIC_VECTOR (15 downto 0);
          an : out STD_LOGIC_VECTOR (3 downto 0);
          segment : out STD_LOGIC_VECTOR (6 downto 0));
end main_function;

architecture Behavioral of main_function is

signal div: STD_LOGIC_VECTOR (19 downto 0) := "00000000000000000000" ;
signal dig: STD_LOGIC_VECTOR (3 downto 0) := "0000";

begin

-- Process to increment the 'div' signal on every rising edge of the clock1

process(clock)
begin
if(rising_edge(clock)) then

div <= div + 1;

end if;
end process;
```

```
-- Segment decoder mapping for displaying digits on a 7-segment display


with dig select
segment <=
 "0000001" when "0000",
 "1001111" when "0001",
 "0010010" when "0010",
 "0000110" when "0011",
 "1001100" when "0100",
 "0100100" when "0101",
 "0100000" when "0110",
 "0001101" when "0111",
 "0000000" when "1000",
 "0000100" when "1001",
 "0000010" when "1010",
 "1100000" when "1011",
 "0110001" when "1100",
 "1000010" when "1101",
 "0110000" when "1110",
 "0111000" when "1111",
 "1111111" when others;


-- Multiplexer to select the appropriate digit to display based on 'div'

with div (19 downto 18) select
dig <= switch (15 downto 12) when "11",
 switch (11 downto 8) when "10",
 switch (7 downto 4) when "01",
```

```
 switch (3 downto 0) when others;



 -- Process to select the active digit based on the value of 'div'


process(div (19 downto 18))
begin
case div (19 downto 18) is
when "00" => an <= "1110";
when "01" => an <= "1101";
when "10" => an <= "1011";
when "11" => an <= "0111";
when others => an <= "1111";
end case;
end process;


end Behavioral;
```

7 Test Bench

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

entity test_bench is
end test_bench;

architecture Behavioral of test_bench is
COMPONENT main_function

Port(
an: out std_logic_vector(3 downto 0);
segment : out std_logic_vector(6 downto 0);
switch : in std_logic_vector(15 downto 0);
clock: in std_logic );

end component;
signal an : std_logic_vector(3 downto 0);
signal segment : std_logic_vector(6 downto 0);
signal switch : std_logic_vector(15 downto 0);
signal clock: std_logic := '0';
constant period : time := 0.1ns;

begin
UUT: main_function port map(
an => an,
segment => segment,
switch => switch,
clock => clock
);
```

```
clock2 : process
begin
wait for period /2;
clock <= '1';
wait for period/2;
clock <= '0';
end process;
tb : Process
begin
switch <= "0011000100110001";
wait for 0.5 ms;
switch <= "1010101111001101";
wait for 0.5 ms;
switch <= "0010001000010000";
wait for 0.5 ms;
switch <= "0010100000111000";
wait for 0.5 ms ;
end process;
end Behavioral;
```