Digital Clock

19ECE383 VLSI Design Lab

B. Tech. ECE

Batch: A1

2023-2024 Even Semester

CB.EN.U4ECE21009	U. Deepika
CB.EN.U4ECE21011	D. Shaahid
CB.EN.U4ECE21012	D. Santhosh

Faculty Incharges:

Dr. S. R. Ramesh

Dr. Bala Tripura Sundari

Signature of the Faculty:

Department of Electronics and Communication Engineering

Amrita School of Engineering

Amrita Vishwa Vidyapeetham

Amritanagar, Coimbatore – 641112

Objective:

1. Design and Implement a Digital Clock Using VHDL: The primary objective is to design a digital clock using VHDL that can display hours, minutes, and seconds in two digits on a 7-segment LED display.
2. Develop Three Key Modules: Develop the three key modules that make up the digital clock: the frequency division module, the counting module, and the decoding display module.
3. Frequency Division Module: Design a frequency division module that can divide a 50 MHz input signal to obtain a 1 Hz clock signal.
4. Counting Time Module: Develop a counting time module that can count and adjust the time of the clock, minutes, and seconds.
5. Decoding Display Module: Implement a decoding display module that can take the output from the counting time module and display it on the FPGA development board.
6. Simulation and Testing: Simulate the digital clock using appropriate tools and methodologies to ensure it functions as expected.
7. Implementation on FPGA: Implement the digital clock design on an FPGA development board and validate its operation.

Software hardware used:

Simulation Software: Modelsim - Intel FPGA Starter Edition 10.5b

(Quartus Prime 18.1)

Synthesis Software: VIVADO 2023.2

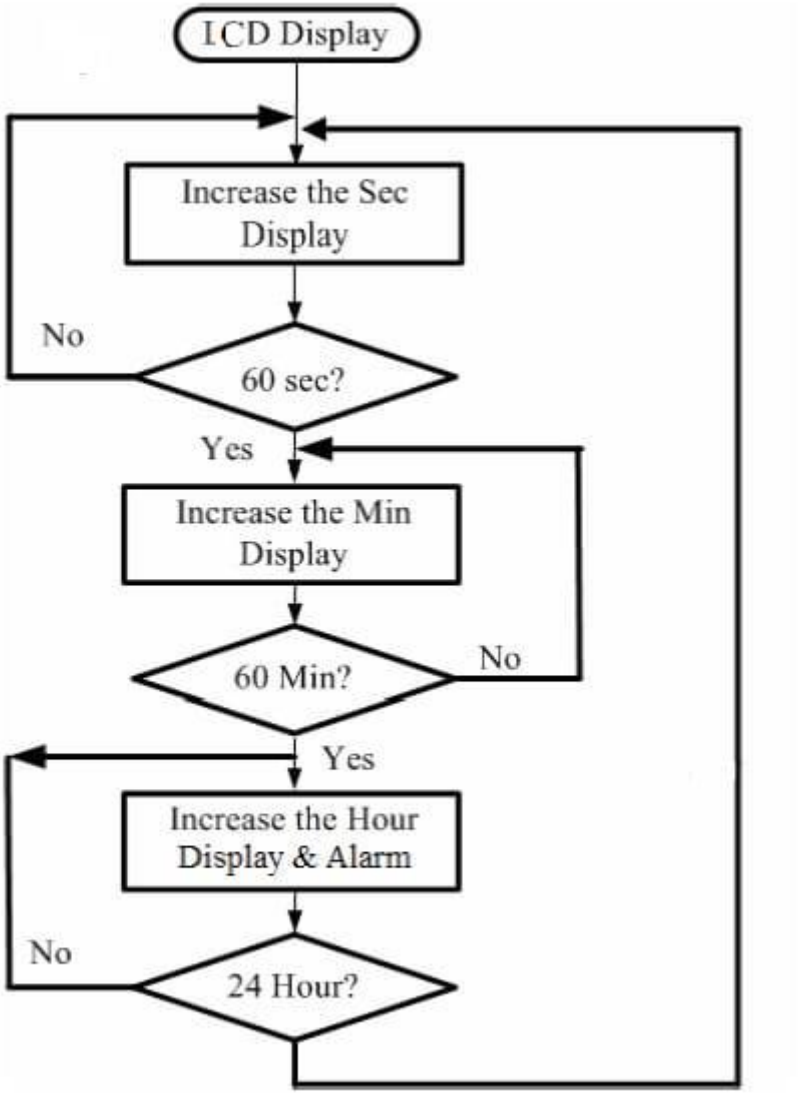
Hardware Used:

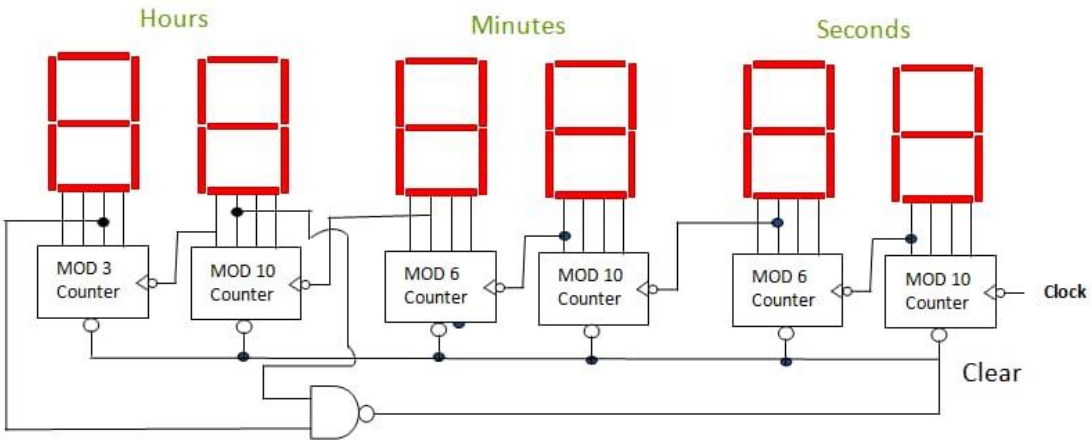
1. **Nexys 4 Artix-7 FPGA Trainer Board**

Overview of the system:

A digital clock implemented in a VLSI design project is a complex yet fascinating application of Very Large Scale Integration (VLSI) technology. It involves the design and development of a fully implementable Verilog code of a working digital alarm clock synthesizable for FPGA. The clock is capable of displaying hours, minutes, and seconds up to two significant digits. It includes features enabling resetting the clock, setting up an alarm for a specific time, and stopping the alarm when it goes off. The project also involves rigorous testing and synthesis processes to ensure the clock functions as intended.

Block diagram:





Introduction:

A digital clock implemented in a VLSI design project is an intricate blend of digital electronics and VLSI design principles. The project involves creating a digital clock using Verilog, a hardware description language, and implementing it on an FPGA, a programmable integrated circuit. The clock displays time in hours, minutes, and seconds with precision up to two significant digits. It also includes an alarm feature that can be set to a specific time and stopped when it rings. The project encompasses various stages including design, testing, and synthesis, each of which plays a crucial role in ensuring the successful implementation of the digital clock. This project serves as a practical application of VLSI design, demonstrating its potential in creating complex digital systems.

Complete Code with comments:

```
--VHDL code for digital clock:

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.numeric_std.all;


entity digital_clock is

port (

 clk: in std_logic;

 -- clock 50 MHz

 rst_n: in std_logic;
```

```
-- Active low reset pulse, to set the time to the input hour and minute (as
-- defined by the H_in1, H_in0, M_in1, and M_in0 inputs) and the second to 00.
-- For normal operation, this input pin should be 1.
H_in1: in std_logic_vector(1 downto 0);
-- 2-bit input used to set the most significant hour digit of the clock
-- Valid values are 0 to 2.
H_in0: in std_logic_vector(3 downto 0);
-- 4-bit input used to set the least significant hour digit of the clock
-- Valid values are 0 to 9.
M_in1: in std_logic_vector(3 downto 0);
-- 4-bit input used to set the most significant minute digit of the clock
-- Valid values are 0 to 9.
M_in0: in std_logic_vector(3 downto 0);
-- 4-bit input used to set the least significant minute digit of the clock
-- Valid values are 0 to 9.
H_out1: out std_logic_vector(6 downto 0);
-- The most significant digit of the hour. Valid values are 0 to 2 ( Hexadecimal value on 7-segment LED)
H_out0: out std_logic_vector(6 downto 0);
-- The most significant digit of the hour. Valid values are 0 to 9 ( Hexadecimal value on 7-segment LED)
M_out1: out std_logic_vector(6 downto 0);
-- The most significant digit of the minute. Valid values are 0 to 9 ( Hexadecimal value on 7-segment LED)
M_out0: out std_logic_vector(6 downto 0)
-- The most significant digit of the minute. Valid values are 0 to 9 ( Hexadecimal value on 7-segment LED)
);
end digital_clock;
architecture Behavioral of digital_clock is
component bin2hex
port (
```

```
 Bin: in std_logic_vector(3 downto 0);
 Hout: out std_logic_vector(6 downto 0)
 );
end component;
component clk_div
port (
 clk_50: in std_logic;
 clk_1s: out std_logic
    );
end component;

signal clk_1s: std_logic; -- 1-s clock
signal counter_hour, counter_minute, counter_second: integer;
-- counter using for create time
signal H_out1_bin: std_logic_vector(3 downto 0); --The most significant digit of the hour
signal H_out0_bin: std_logic_vector(3 downto 0);--The least significant digit of the hour
signal M_out1_bin: std_logic_vector(3 downto 0);--The most significant digit of the minute
signal M_out0_bin: std_logic_vector(3 downto 0);--The least significant digit of the minute
begin
-- create 1-s clock --|
create_1s_clock: clk_div port map (clk_50 => clk, clk_1s => clk_1s);
-- clock operation ---|
process(clk_1s,rst_n) begin

 if(rst_n = '0') then
 counter_hour <= to_integer(unsigned(H_in1))*10 + to_integer(unsigned(H_in0));
 counter_minute <= to_integer(unsigned(M_in1))*10 + to_integer(unsigned(M_in0));
 counter_second <= 0;
```

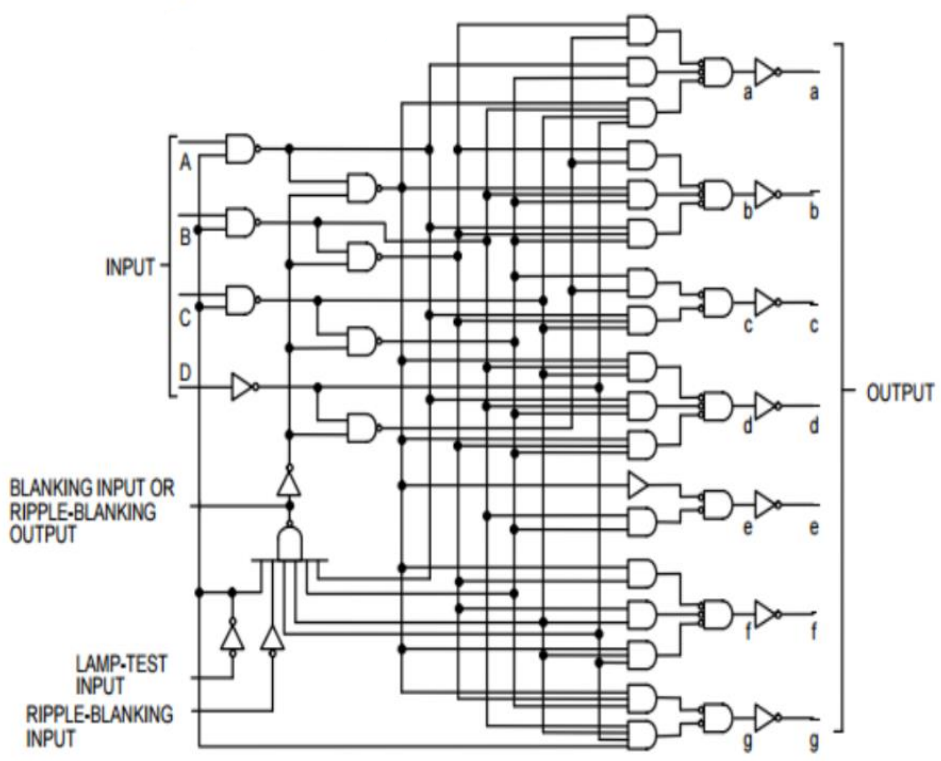
```
 elsif(rising_edge(clk_1s)) then
 counter_second <= counter_second + 1;
 if(counter_second >=59) then -- second > 59 then minute increases
 counter_minute <= counter_minute + 1;
 counter_second <= 0;
 if(counter_minute >=59) then -- minute > 59 then hour increases
 counter_minute <= 0;
 counter_hour <= counter_hour + 1;
 if(counter_hour >= 24) then -- hour > 24 then set hour to 0
 counter_hour <= 0;
 end if;
 end if;
 end if;
 end if;
end process;

-- Conversion time ---

-- H_out1 binary value
 H_out1_bin <= x"2" when counter_hour >=20 else
 x"1" when counter_hour >=10 else
 x"0";
-- 7-Segment LED display of H_out1
convert_hex_H_out1: bin2hex port map (Bin => H_out1_bin, Hout => H_out1);
-- H_out0 binary value
 H_out0_bin <= std_logic_vector(to_unsigned((counter_hour -
to_integer(unsigned(H_out1_bin))*10),4));
-- 7-Segment LED display of H_out0
convert_hex_H_out0: bin2hex port map (Bin => H_out0_bin, Hout => H_out0);
-- M_out1 binary value
 M_out1_bin <= x"5" when counter_minute >=50 else
```

```
 x"4" when counter_minute >=40 else
 x"3" when counter_minute >=30 else
 x"2" when counter_minute >=20 else
 x"1" when counter_minute >=10 else
 x"0";
-- 7-Segment LED display of M_out1
convert_hex_M_out1: bin2hex port map (Bin => M_out1_bin, Hout => M_out1);
-- M_out0 binary value
 M_out0_bin <= std_logic_vector(to_unsigned((counter_minute -
to_integer(unsigned(M_out1_bin))*10),4));
-- 7-Segment LED display of M_out0
convert_hex_M_out0: bin2hex port map (Bin => M_out0_bin, Hout => M_out0);
end Behavioral;



-- BCD to HEX For 7-segment LEDs display
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity bin2hex is
port (
 Bin: in std_logic_vector(3 downto 0);
 Hout: out std_logic_vector(6 downto 0)
);
end bin2hex;
architecture Behavioral of bin2hex is
begin
 process(Bin)
 begin
  case(Bin) is
   when "0000" =>  Hout <= "1000000"; --0--
   when "0001" =>  Hout <= "1111001"; --1--
```

```
   when "0010" =>  Hout <= "0100100"; --2--
   when "0011" =>  Hout <= "0110000"; --3--
   when "0100" =>  Hout <= "0011001"; --4--
   when "0101" =>  Hout <= "0010010"; --5--
   when "0110" =>  Hout <= "0000010"; --6--
   when "0111" =>  Hout <= "1111000"; --7--
   when "1000" =>  Hout <= "0000000"; --8--
   when "1001" =>  Hout <= "0010000"; --9--
   when "1010" =>  Hout <= "0001000"; --a--
   when "1011" =>  Hout <= "0000011"; --b--
   when "1100" =>  Hout <= "1000110"; --c--
   when "1101" =>  Hout <= "0100001"; --d--
   when "1110" =>  Hout <= "0000110"; --e--
   when others =>  Hout <= "0001110";
   end case;
 end process;
end Behavioral;

-- Clock divider module to get 1 second clock
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
entity clk_div is
port (
   clk_50: in std_logic;
   clk_1s: out std_logic
  );
end clk_div;
architecture Behavioral of clk_div is
signal counter: std_logic_vector(27 downto 0):=(others =>'0');
```

```
begin
 process(clk_50)
 begin
  if(rising_edge(clk_50)) then
   counter <= counter + x"0000001";
   if(counter>=x"2FAF080") then -- for running on FPGA -- comment when running simulation
   --if(counter_slow>=x"0000001") then -- for running simulation -- comment when running on FPGA
    counter <= x"0000000";
   end if;
  end if;
 end process;
 clk_1s <= '0' when counter < x"17D7840" else '1';
end Behavioral;
```

RTL:

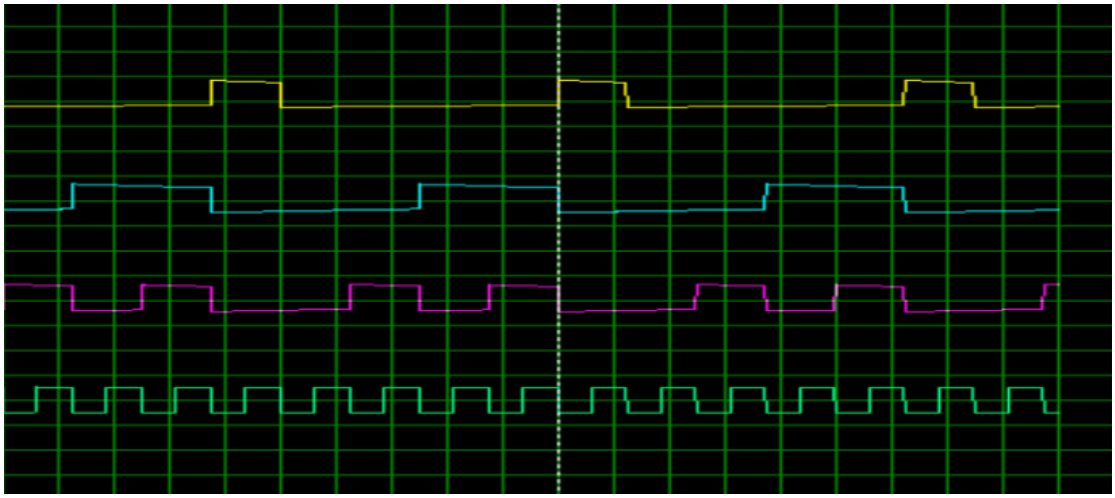


Resource Utilization summary table:

S.no	Module name	Module description	Status
1	Binary to BCD	To generate clock pulse to the counter	completed
2	Counting module	To count hours, minutes and seconds	completed
3	Frequency module	Binary to BCD conversion	completed

Modules	Status
Nexys 4 Artix-7 FPGA Trainer Board	Partially Utilized

Output Waveform:



Hardware Images: