Documentation

Team name

Date

Project Report for FPGA based Alarm Clock with PCB Design

1 Team members

- Arsal Abbasi
- Ahmed Helmy
- Jires Donfack Voufo

2 Introduction

Our project is a prototype of an alarm clock implemented with VHDL and a PCB design corresponding to the model. We used VHDL to program our code as that allowed us to describe the system behavior and simulate it, and then used synthesis tool in order to translate the software model into a real hardware model. We then used Xilinx Vivado to implement our ModelSim design of alarm clock on our FPGA board Nexys A7-100T by generating bitstream and uploading the code to our board Nexys A7.

VHDL is a language that is used to describe the behavior and structure of digital systems. In our project, VHDL allowed us to describe the functionality of alarm clock, which includes timekeeping, display on alarm and alarm set time, alarm triggering, and other features like LED and button control. By using VHDL we were able to simulate the system's behavior. VHDL basically provided us with high-level abstraction that helped us with the overall design process.

Use of VHDL ModelSim allowed us to simulate the behavior of the alarm clock design, which helped us in the design process before implementing it on Xilinx Vivado. With the help of testbench on ModelSim we were able to test different scenerios, validate the functionality, and identify errors in the design and timing issues.

FPGA provided us with a platform to realize the design of alarm clock on physical hardware. By synthesizing the VHDL code on Xilinx Vivado, we were able to convert the behavioral description into hardware implementation specifically tailored for our board Nexys A7.

3 Concept description

The concept of our project is described as follows;

We have divided the seven-segment display of FPGA board Nexys A7 into two parts. On the one side we are showing the running clock, on the other side we are having our alarm clock set. User can use the buttons to set the alarm time on the board. When the running clock time reaches the alarm set time, the LEDs on the board will go on. In our first idea, we planned to represent alarm by a buzzing sound on the headset using Audio Out, but due to the time constraint and the complexity involved we dropped the idea of that and decided to represent alarm using LEDs present on the board.

The concept of the alarm clock is represented in the block diagram shown in figure 1.

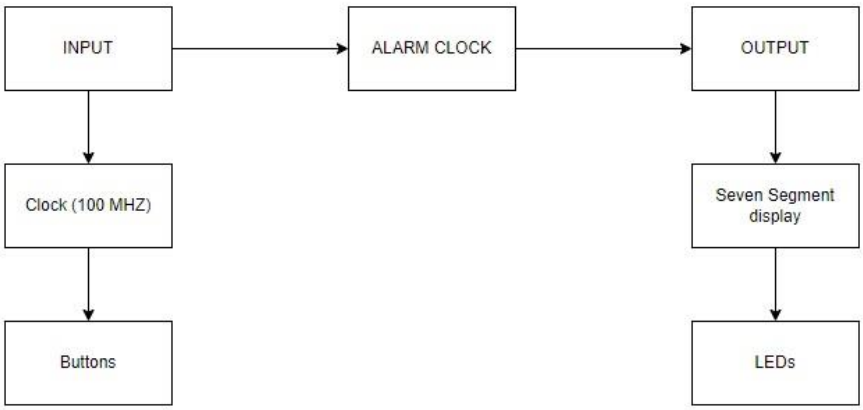


Fig. 1 System block diagram

As it can be seen from the diagram, we are using system clock MHZ, with the help of buttons present on FPGA board Nexys A7 user will be able to input the alarm set time. Clock and alarm set time will be output using seven segment display, and when the alarm time reaches the running clock time then

What is the main application for your prototype?

4 Project/Team management

5 Technologies

Describe the technological approaches you will use to implement your project.

The approaches we used are divided into three categories;

- *VHDL*
- *FPGA*
- *KiCAD*

As discussed earlier we first started with VHDL design on ModelSim . The figure below represents our inputs and outputs. In this code, we have defined an entity called DigitalAlarmClock. It has various input and output ports. The inputs include the clock signal (clk), reset signal (reset), set time enable signal (set_time), time input in Binary-Coded

Decimal (BCD) format (time_in), set alarm enable signal (set_alarm), and alarm input in BCD format (alarm_in). The output is the alarm signal (alarm).

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

entity DigitalAlarmClock is
    Port (
        clk       : in  STD_LOGIC;          -- Clock input
        reset     : in  STD_LOGIC;          -- Reset input
        set_time  : in  STD_LOGIC;          -- Set time enable input
        time_in   : in  STD_LOGIC_VECTOR(3 downto 0);  -- Time input (BCD format)
        set_alarm : in  STD_LOGIC;          -- Set alarm enable input
        alarm_in  : in  STD_LOGIC_VECTOR(3 downto 0); -- Alarm input (BCD format)
        alarm     : out STD_LOGIC           -- Alarm output
    );
end DigitalAlarmClock;
```

Fig. 2 DigitalAlarmClock code snippet ModelSim

Inside the Behavioral architecture, we declare two signals: current_time and current_alarm. These signals are used to store the current time and alarm values, respectively. They are initialized to all zeros ("0000") as default values.
In he process statement, we define the behavior of the alarm clock. The process is sensitive to changes in clk and reset. If the reset signal is asserted ('1'), indicating a reset condition, we reset the current_time and current_alarm signals to "0000" and set the alarm output to '0' to ensure the alarm is not triggered. On a rising edge of the clk signal (indicating a new clock cycle), we check the status of the set_time and set_alarm signals. If set_time is asserted ('1'), we update the current_time signal with the new time value provided through time_in. Similarly, if set_alarm is asserted ('1'), we update the current_alarm signal with the new alarm value provided through alarm_in. Next, we compare the current_time and current_alarm values. If they are equal, we set the alarm output to '1', triggering the alarm. Otherwise, we set the alarm output to '0', resetting the alarm.
For our testbench we we declare the signals that will be used in the testbench. These signals correspond to the ports of the DigitalAlarmClock entity. We initialize some of the signals with default values. The CLOCK_PERIOD constant represents the desired clock period for simulation purposes.

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

entity DigitalAlarmClock_TB is
end DigitalAlarmClock_TB;

architecture Behavioral of DigitalAlarmClock_TB is

    signal clk         : std_logic := '0';
    signal reset       : std_logic := '1';
    signal set_time    : std_logic := '0';
    signal time_in     : std_logic_vector(3 downto 0) := "0000";
    signal set_alarm   : std_logic := '0';
    signal alarm_in    : std_logic_vector(3 downto 0) := "0000";
    signal alarm       : std_logic;

    constant CLOCK_PERIOD : time := 10 ns; -- Clock period of 10 ns
```

Fig. 3 Testbench snippet

Then, we instantiate the unit under test (DigitalAlarmClock) using the uut label. We connect the signals declared in the testbench to the corresponding ports of the DigitalAlarmClock entity using the port map statement.
The process is responsible for generating the clock signal (clk) in the testbench. It runs for a simulation time of 100 ns. Inside the loop, we toggle the clk signal between '0' and '1' with a half-clock period delay.
Our testbench sets up the necessary signals, generates the clock signal, and provides stimulus to the digital alarm clock design to test its functionality and behavior under different input conditions.
The result of our design simulation on ModelSim is shown below.

6 VHDL and FPGA Implementation

7 PCB Design

A PCB is the backbone of all the modern days electronic device. In our case it helped us prototyped our work. The following electronics components were used in our design: 7-segments display, an oscillator, resistors, a capacitor, a buzzer, push buttons ,3.3V supply source and a controller.
We used the Software KIcad to design the PCB for our alarm clock. The different designing steps are as follows.

1) Adding libraries

Here we start by adding all the necessary libraries that will be used for our project. In this case that was the Sparkfun, Kicad and an FPGA file provided by the Professor. We also added the corresponding footprint libraries too.

2)Schematic
This is the circuit diagram for all the electronics components that are needed to implement our design. We first selected and imported all symbols that represent the components that we need. After putting all symbols together, we refer to the datasheet to connect and labelled them with wires in an appropriate way. We try to group the components in a way the connections will look as clean as possible

3)ERC
After all the connections we did an ERC (electrical rules check) to find any hazardous issues to the circuit according to the electronics design rules. Some errors occur related to the power supply and open connections which we fixed by using a (PWR_FLAG) symbols and putting a cross on all open connections to tell Kicad they are not being used.

4)Creating the footprint
After the ERC was completed, we used the footprint library from Sparkfun, Kicad library and from the Professor to assign the correct footprint to each of the components. The footprint represents the physical shape of the schematic symbols T5) Generate the Net List

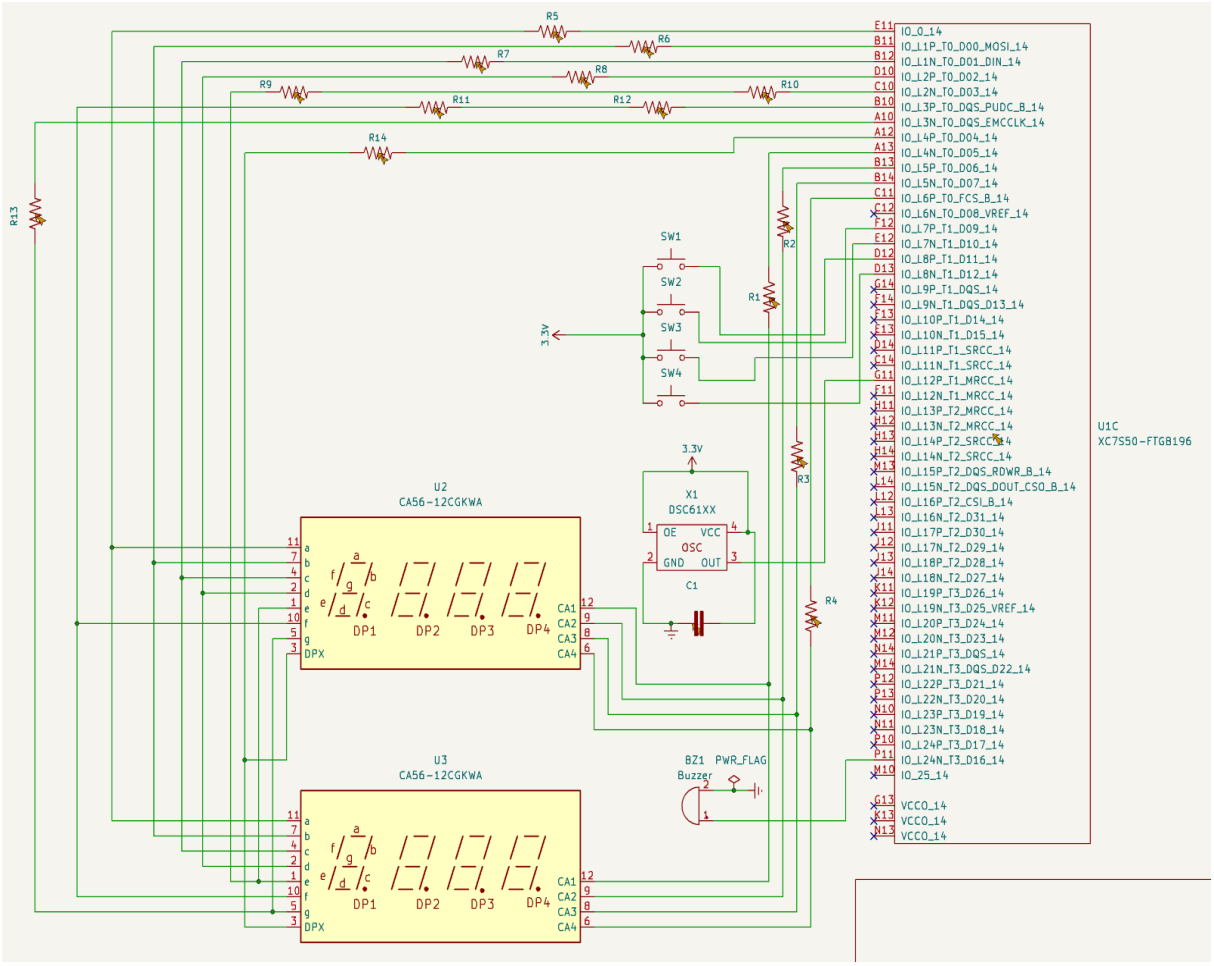
5)Creating the Netlist
Here we generated the Netlist file which is used to tell Kicad how the different components terminals are connected.

6)BOM (Bill of Materials)
The bill of materials shows all used components with their reference, description and quantities

7)PCB Layout
Here we had to decide how the components will be placed on the PCB taking in account trace spacing and components interactions.

8)Cost and Size

The following pictures illustrate respectively the Schematic and the Bill of materials



	A	B	C
1	Reference	Description	Qty
2	BZ1	Buzzer	1
3	C1	0.1UF-0402-16V-10%	1
4	R1	330KOHM-0603-1_10W-1%	1
5	R2	330KOHM-0603-1_10W-1%	1
6	R3	330KOHM-0603-1_10W-1%	1
7	R4	330KOHM-0603-1_10W-1%	1
8	R5	330KOHM-0603-1_10W-1%	1
9	R6	330KOHM-0603-1_10W-1%	1
10	R7	330KOHM-0603-1_10W-1%	1
11	R8	330KOHM-0603-1_10W-1%	1
12	R9	330KOHM-0603-1_10W-1%	1
13	R10	330KOHM-0603-1_10W-1%	1
14	R11	330KOHM-0603-1_10W-1%	1
15	R12	330KOHM-0603-1_10W-1%	1
16	R13	330KOHM-0603-1_10W-1%	1
17	R14	330KOHM-0603-1_10W-1%	1
18	SW1	SW_Push	1
19	SW2	SW_Push	1
20	SW3	SW_Push	1
21	SW4	SW_Push	1
22	U1	XC7S50-FTGB196	1
23	U2	CA56-12CGKWA	1
24	U3	CA56-12CGKWA	1
25	X1	DSC61XX	1
26			

8 Sources/References

Provide the sources on the technologies and algorithms you used in your project (Github).

Team Name