Documentation

Team name

Date

Important Notes:

- ***The descriptions in italics in this document (except for some section headings) are exemplary and explanatory and must be removed from the completed report.***
- ***Identify which section of this report was created by which team member***
- ***Your documentation should have ca. 8 pages (content! Without cover sheet, references, appendix etc.).***

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.

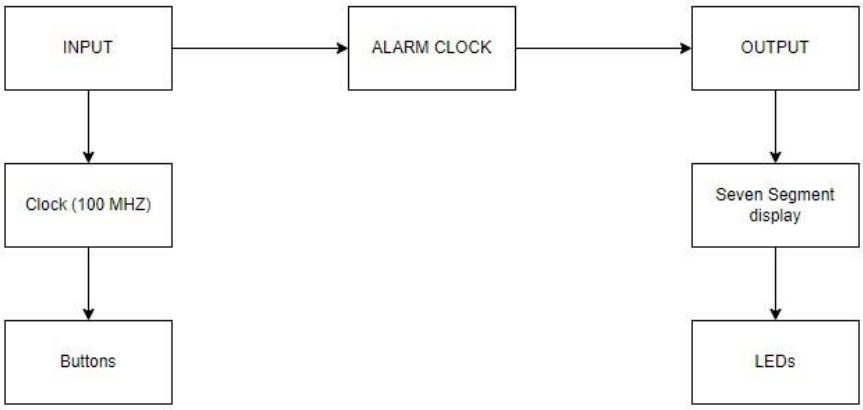


Fig. 1 System block diagram

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

Which project methods you used in your project?
Breakdown: How you managed your tasks?
What are the different tasks/roles of the team members in the project?
Describe which team member did which tasks.

5 Technologies

The approaches we used are divided into three categories;

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

6 VHDL and FPGA Implementation

VHDL Implementation

As discussed earlier we first started with VHDL design on ModelSim . The figure below represents our inputs and outputs. This is the top file of our design Temp_on_7_Seg. This represents and how will be used. The alarm clock can be set to any number by using buttons, and the clock can also be changed to set any time. Switch 0 can be turned HIGH in order to change hours, when it is LOW user can change the minutes.

```
5
6   entity Temp_on_7_Seg is
7       port(
8           clk_i : in std_logic;
9           rstn_i : in std_logic;
10          seg_o : out std_logic_vector(7 downto 0);
11          an_o  : out std_logic_vector(7 downto 0);
12          LED   : out std_logic_vector(15 downto 0);
13          alram_down : in STD_LOGIC;
14          alram_up : in STD_LOGIC;
15          clock_down : in STD_LOGIC;
16          clock_up : in STD_LOGIC;
17          hrs_min : in STD_LOGIC
18      );
19  end Temp_on_7_Seg;
20
```

Fig. 2 Temp_on7Seg code snippet ModelSim

In our top file, the entity Temp_on_7_Seg defines the interface and connectivity of the system, including input and output signals for clock, reset, seven-segment display, LEDs, and control signals.
Inside the architecture block Behavioral, various reusable components are declared using component declarations. These components include Seven_Segment (for driving the seven-segment display), seg_decode (for converting hexadecimal values to corresponding segment patterns), One_sec (for generating a one-second pulse), Alarm_Time (for handling alarm time settings), and Clock_Time (for tracking and updating the current time).
The architecture also declares a set of signals that store intermediate values. These signals include dispVal (concatenated segment outputs), hex_in_0 to hex_in_7 (input values for the segment decoder), seg_out_0 to seg_out_7 (segment decoder outputs), and various other signals related to time and alarm settings.
The code instantiates the components declared earlier using component instantiations. These instantiations connect the input and output ports of the components to the corresponding signals or ports in the architecture. This allows for the modular reuse of functionality throughout the code.
In the main part of the architecture, there is a process called flash_process. This process is sensitive to the clock and reset signals and controls the flashing behavior of the LEDs. When the reset signal is active, the LED counter is reset to zero. On a rising clock edge, the process checks if the current time matches the alarm time. If they match and the sec_plus signal is active (indicating a one-second pulse), the LED pattern is toggled between x"0000" and x"FFFF". If the clock and alarm time do not match, the LEDs are turned off (LED_int set to x"0000").

Finally, the process assigns the value of LED_int to the output signal LED, allowing the LED pattern to be displayed externally.
To verify our system and simulate the alarm clock and LEDs, we wrote testbench for our Temp_on7Seg component.
In the Behavioral architecture, various signals are declared to interface with the UUT and control the simulation. These signals include clk_i (clock input), rstn_i (reset input), seg_o (seven-segment output), an_o (anode output), LED (LED output), alram_down (alarm decrease control input), alram_up (alarm increase control input), clock_down (clock decrease control input), clock_up (clock increase control input), and hrs_min (hours/minutes control input).
The UUT component Temp_on_7_Seg is instantiated with the declared signals connected to its corresponding ports. This allows for the simulation and testing of the module's functionality.
The testbench includes two processes: clk_process and stimulus_process.
The clk_process is responsible for generating the clock signal. It alternates the value of clk_i between '0' and '1' every half of the specified CLK_PERIOD time.
The stimulus_process is the main test stimulus process. It defines the sequence of events and inputs to be applied to the UUT for testing purposes. The process begins by holding the rstn_i signal low for 10 clock cycles to perform a reset. Afterward, it sets the initial values of the control signals and waits for a specified period before performing a series of operations.
In this specific test scenario, the testbench simulates setting the alarm time to 7:30. It sets the alram_up signal high for a certain duration, followed by setting the hrs_min signal high to switch to the minutes setting mode. After waiting for a specific duration, the alarm time is incremented by setting alram_up high again. This process simulates the user interface actions to set the alarm time.
*Finally, the testbench waits for a simulated period of 24 hours (24 * 60 * CLK_PERIOD) to observe the behavior of the system over an extended period of time.*
The wait statement at the end of the stimulus_process process ensures that the simulation continues indefinitely, allowing for observation and analysis of the UUT's behavior.
The result of the simulation of our design is shown below. It can be seen how the LEDs toggle once the time reaches the set time 7:30.

Fig. 3 Simulation results ModelSim

FPGA Implementation

We implemented our alarm clock on FPGA Board Nexys A7 100T using Xilinx Vivado.
The top file of our design is Temp_on_7_Seg, which is the top file of our design. Our design has various input and output ports as represented in the ModelSim snippet fig.2. It provides various controls like (alram_down, alram_up, clock_down, clock_up, hrs_min).
We have dofferent components such as Seven_Segment, seg_decode, One_sec, Alarm_time and Clock_Time, and then connect their ports.
The Seven_Segment represents a seven-segment display and handles the conversion of the input number (number) into the corresponding segment outputs (seg) and anode outputs (an).
The seg_decode entity is a decoder that takes a 4-bit input (hex_in) and generates the corresponding 8-bit segment output (seg_out) for a specific digit or character.

The One_sec entity represents a one-second counter. It takes a clock input (clk_i), a reset input (rstn_i), and generates a sec_plus signal when one second has elapsed.
The Alarm_Time entity is responsible for handling the alarm time. It takes clock input (clk_i), reset input (rstn_i), control signals for increasing/decreasing the alarm time (alram_down, alram_up), and the current mode (hrs_min). It increments the alarm hours (hrs_out) or minutes (min_out) based on the control signals and mode when the sec_plus signal is asserted.
There is also a similar component Clock_Time, which handles the clock time.
The code includes other signals and intermediate connections necessary for the functionality of the design.
The results of our FPGA implementation are shown below. In Fig. 4 the alarm set time is 2 minutes, and the running clock time is 0 minutes, and the LEDs are off. And in fig. 5 it can be seen when running clock (on right) time is equal to the alarm set time (on left) then the LEDs go on to simulate the functionality of developed alarm clock design.

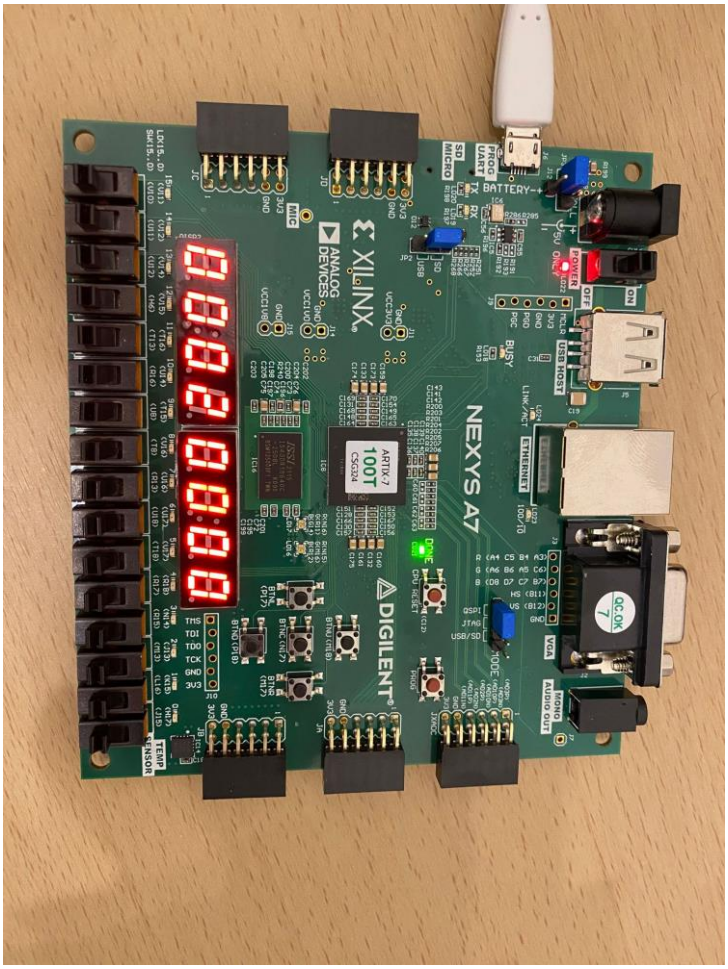


Fig. 4 Alarm off (Alarm set time 2 mins and current time 0 mins)

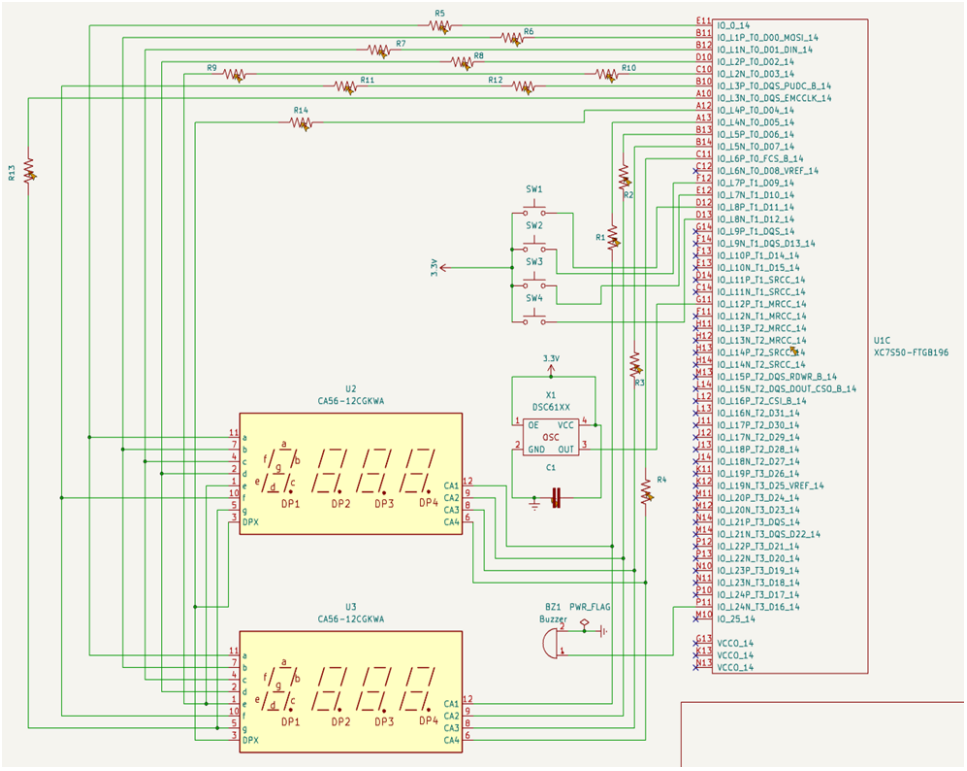
Fig. 5 Alarm on (Alarm set time 2 mins, current time 2 mins

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).