Implementation of an Alarm Clock in Verilog applicable in FPGA

Aamod B K
International Institute of Technology, Bangalore
Aamod.BK@iiitb.ac.in

***Abstract*—This document describes the design and development of a fully implementable Verilog code of a working digital alarm clock synthesizable for FPGA. Basic Verilog HDL syntax is used for the implementation of this digital design and a testbench file is also created for simulation using the Icarus Verilog compiler and the GTKWave toolkit.**

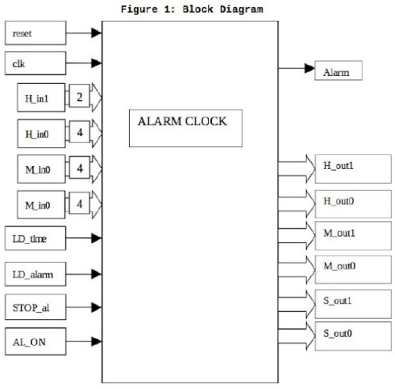

Fig. 1. Block diagram of the design

I. Introduction

All credits for the below presented digital design go to the website fpga4student.com. [1]
This design aims to apply the Verilog language into the development of a simple mechanism of a digital alarm clock. This implementation of a digital clock includes a multiple features enabling resetting the clock, setting up an alarm for a specific time and also stopping the alarm when it goes off. Basic verilog language syntax and functions are used to realize the required features. The created digital clock is capable of displaying hours, minutes and seconds upto two significant digits. Alarm can also be set upto the accuracy of minutes. The figure 1 depicts the block diagram representing the design.

II. Design Implementation

The proposed design is implemented in a module named *aclock*. The clock provides options to do the following — reset – using *reset* input, set time – using *LD_time* input, set an alarm – using *LD_alarm* input, signal to call alarm – using *AL_ON* input, stop the alarm – using *STOP_al* input.
The code also consists of the following output registers — *Alarm* – Goes to high if alarm time equals current time, *H_out1* – Most significant digit of hours, *H_out0* – Least significant digit of hours, *M_out1* – Most significant digit of minutes, *M_out0* – Least significant digit of minutes, *S_out1* – Most significant digit of seconds, *S_out0* – Least significant digit of seconds.

A number of count variables are used to keep track of the hours, minutes and seconds from reset/set, whose values are continuously read by the output variables in an *always* loop to keep updating the time. A clock variable of frequency 10 Hz is created in the testbench to keep track of time. This clock is used to update the 1-second clock so that the temporary variables can keep up with the change in time. The alarm functionality is satisfied by continually checking for the *AL_ON* input and if it is high, the *Alarm* register is made high. Also, if the system then recieves the *STOP_al* signal, the alarm is turned off. This code is run and tested using the Icarus Verilog compiler and the plots w.r.t time are obtained using GTKWave [2].

III. Testing

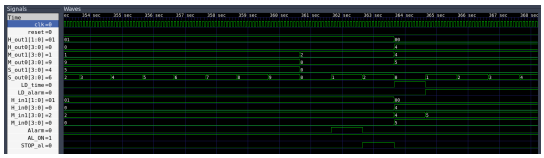

Fig. 2. GTKWave output for the testbench

In the testbench file, an Under Unit Test (UUT) is instantiated adn variables for input and output are provided to the test module. The clock variable is set to a frequency of $10\,Hz$. The testing proceeds as follows – Initially, the time is set to $10:14$. After 1 second, an alarm is set at the time $10:20$. The alarm goes off at the set time of $10:20$ as seen from the figure 2. The alarm is then turned off after 1 second and the clock-time is reset to $04:45$. Next, another alarm is set to $04:55$, which functions as expected.

Conclusion

The results obtained in the GTKWave plots lead to the conclusion that the design for the alarm clock and the corresponding Verilog code is accurate and implementable.

References

[1] Verilog code from – www.fpga4student.com
[2] GTKWave and IcarusVerilog guide from – iverilog.fandom.com