DESIGN AND VERIFICATION OF FSM BASED TIMING GENERATOR CIRCUIT USING HDL

K.R.N. karthik[1], P. Kaveri[2], Fazal Noorbasha[3]
[1,2]VLSI Systems Research Group, Department of Electronics and Communication Engineering, K.L University, Guntur, AP – India
[3]Assoc. Professor VLSI Systems Research Group, Department of Electronics and Communication Engineering, K.L University, Guntur, AP – India

ABSTRACT

As the technology is going on rapidly the major parameters affecting the chip design is area, density and clock speeds. Every digital circuitry [1]. is based on the clock and design of clock for each and every individual module is a bit complex process and so this paper deals with the real time clock in which this clock module can be integrated to various circuits as per their requirement and alarm functionality is also implemented such that this can be helpful in waking up the devices at particular time from the sleep mode to save the power consumption this paper is completely based on the FSM plot[3] also consisting of the counters and the registers this is implemented on the Spartan 3 FPGA board and the complete code coverage report have been generated design have been done using verilog HDL in Xilinx and the verification part have been done in mentor graphics tool

KEYWORDS: *Digital Circuits, FSM, FPGA, sequential circuits.*

I. INTRODUCTION

A digital clock [2] is a type of clock that displays the time digitally, i.e. in numerals or other symbols, as opposed to an analog clock, where the time is indicated by the positions of rotating hands. Digital clocks are often associated with electronic drives, but the "digital" description refers only to the display, not to the drive. the construction of the real time clock consists of various modules such as the registers and counters and the frequency of the clock can be changed by changing the counting sequence of the counter design is done using verilog and when coming to the need of verification design. This split has led to the development of narrowly focused language for verification and to the bifurcation of engineers into two largely independent disciplines. This specialization has created substantial bottlenecks in terms of communication between the two groups. System Verilog addresses this issue with its capabilities for both camps. Neither team has to give up any capabilities it needs to be successful, but the unification of both syntax and semantics of design and verification tools improves communication. For example, while a design engineer may not be able to write an object-oriented test bench environment, it is fairly straightforward to read such a test and understand what is happening, enabling both the design and verification engineers to work together to identify and fix problems. Likewise, a designer understands the inner workings of his or her block, and is the best person to write assertions about it, but a verification engineer may have a broader view needed to create assertions between blocks. Another advantage of including the design, test bench, and assertion constructs in a single language is that the test bench has easy access to all parts of the environment without requiring specialized APIs. The value of an HVL is its ability to create high-level, flexible tests, not its loop constructs or declaration style [1]. System Verilog is based on the Verilog constructs that engineers have used for decades.

II. ARCHITECTURE

The architecture consists of the block diagram based on the FSM [3] and it consists of six modules and alarm controller plays a major role and here load signals for controlling the counters are generated by the alarm controller, key register is used to enter the new key values time generator generates the one minute and one second pulse alarm register is used to store the particular time value and the LCD driver is designed keeping the FPGA display format in to consideration

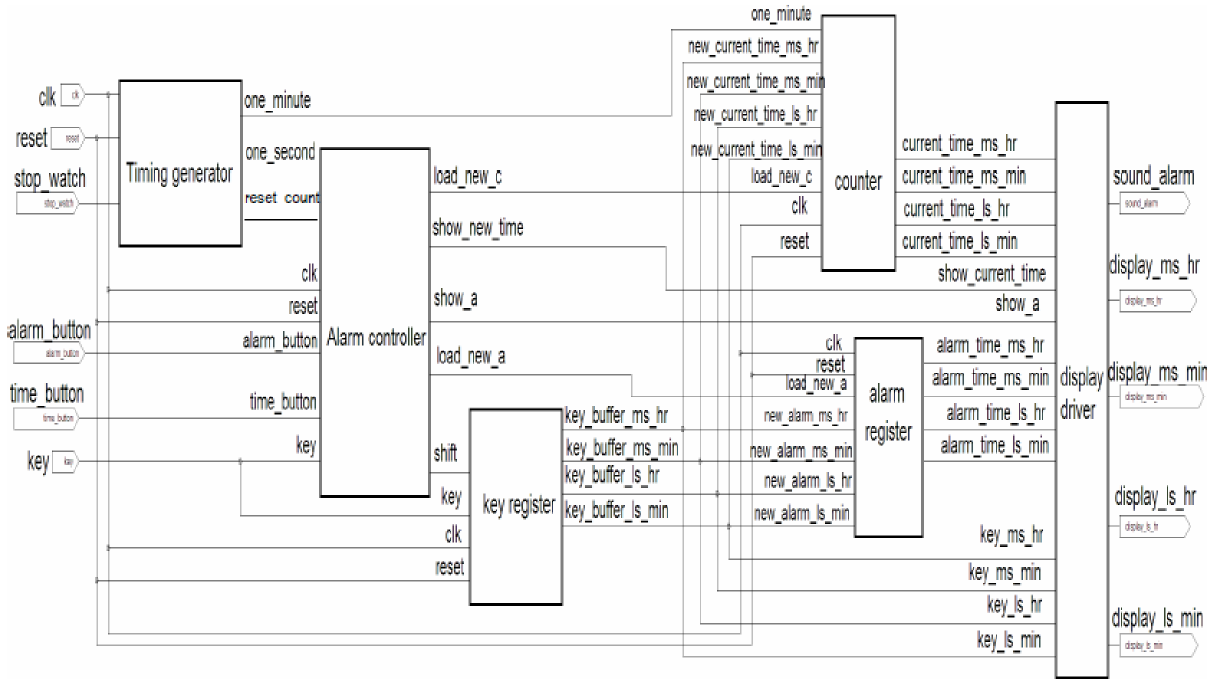


Fig 1: Block diagram of the timing circuit

2.1. Timing generator:

Timing generator takes the clock and reset and stopwatch as the inputs and generates the one minute and one second pulse which can be further processed in to various blocks such as alarm controller and the counter and the obtained waveform is mentioned below.

2.2. Alarm controller:

The control signals of the alarm and time button is given to the alarm controller and here depending on the control signal only the required functionality will be displayed and alarm controller plays a major role and it also takes key value as the input, the obtained waveform is mentioned below.

2.3. Key register:

Key register is used to enter the time and alarm values manually and the entry of the key values depends on the finite state machine and the obtained wave form is mentioned below.

2.4. Counter

The counter calculates the current rime and here after reset of the counter depending on the current input values the counter will run. To change the clock frequency the counter steps must be increased binary counter can be constructed from J-K flip-flops by taking the output of one cell to the clock input of the next. The J and K inputs of each flip-flop are set to 1 to produce a toggle at each cycle of the clock input. For each two toggles of the first cell, a toggle is produced in the second cell, and so on down to the fourth cell. This produces a binary number equal to the number of cycles of the input clock signal. This device is sometimes called a "ripple through" counter. The same device is useful as a frequency divider.

2.5. Alarm register

The Alarm register is the functionality similar to the normal register and here the alarm register saves the current alarm values and it compares with the current time values and when ever bits of the current values and the alarm value matches the alarm signal will remain high.

III. FSM

A finite-state machine [3] or finite-state automaton or simply a state machine, is a mathematical model of computation used to design both computer programs and sequential logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition; this is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition.

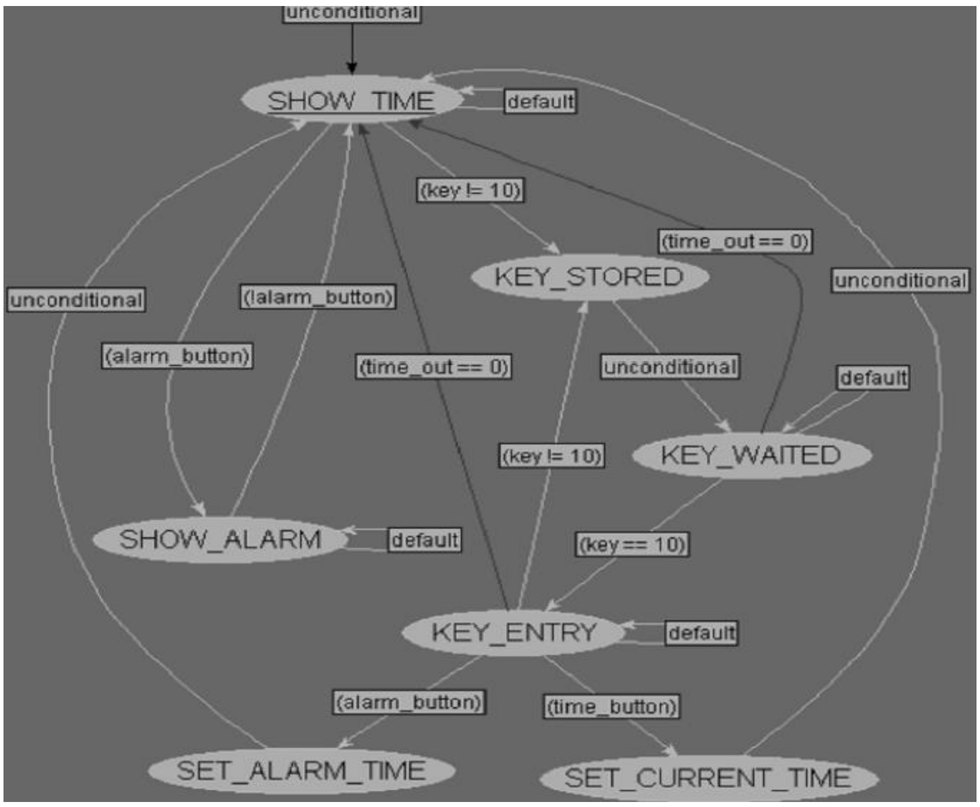


Fig 2. FSM model

In the above FSM whenever the delay or unwanted signal happens it will be reset back to the initial state and depending on the delay transition will be kept on hold and latter on depending on the condition the stage will be changed

IV. CODE COVERAGE

Coverage is used. It will measure the efficiency of your verification implementation. By default, every tool disables the code coverage. If user enables then only code coverage is done. By enabling the code coverage there is overhead on the simulation and the simulation takes more time. So it is recommended not to enable the code coverage always. Enabling the code coverage during the regression saves user time a lot.

V. RESULTS

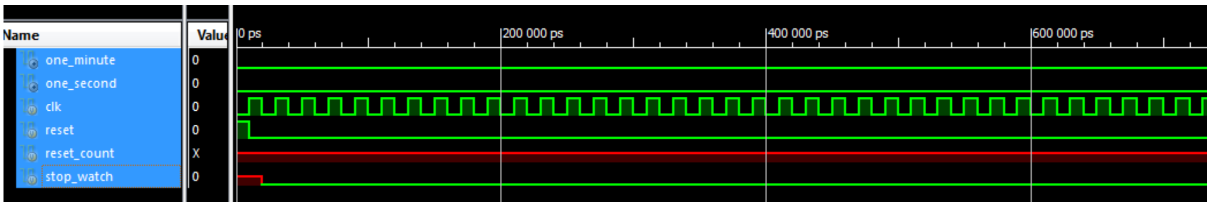


Fig 3: waveform of timing generator

In the above waveform minute pulse will be obtained depending on the seconds pulse reset_count signal will be remained unknown because of that particular signal will be obtained from the another module.

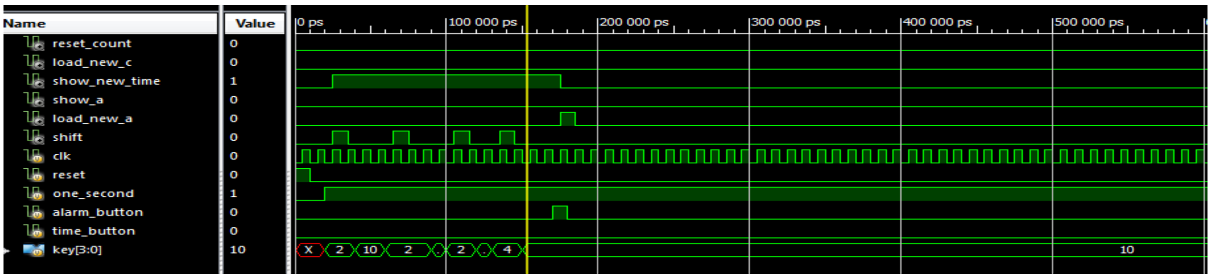


Fig 4: waveform of the alarm controller

In the above waveform key values are entered and which depending on the control signal these key values will be set as the current or the alarm timing values

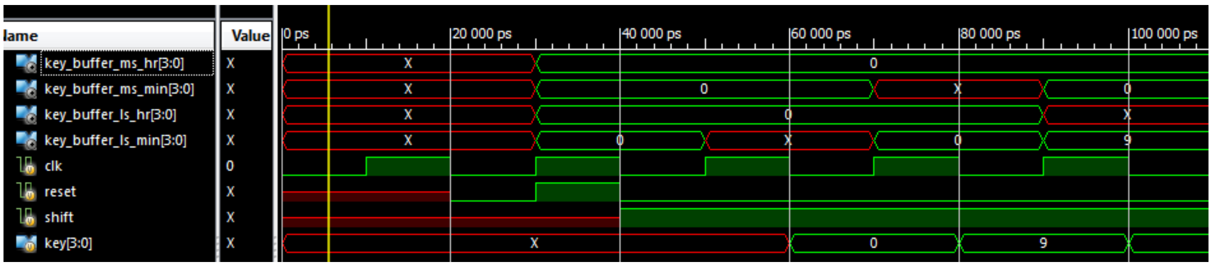


Fig 5: waveform of key register

In the above waveform initially reset is zero so unknown values are loaded and after the reset signal the key values are loaded in to the particular register

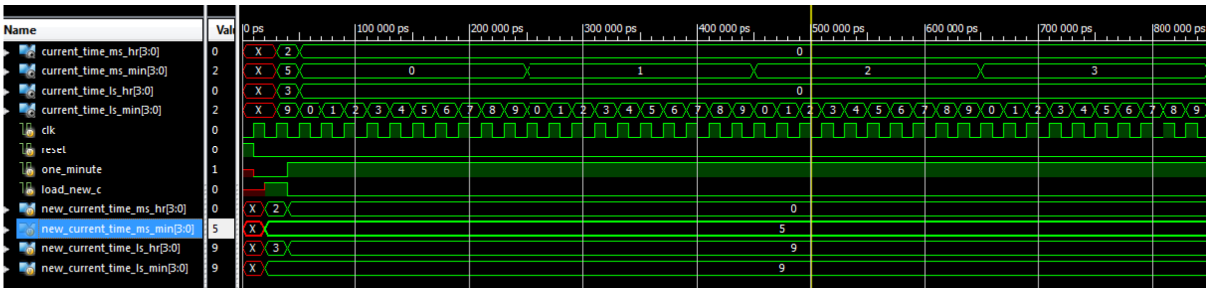


Fig 6: waveform of the counter

In the above waveform the counter will start running once the reset is kept zero and depending on the counting values the seconds and the minute pulse will be changed

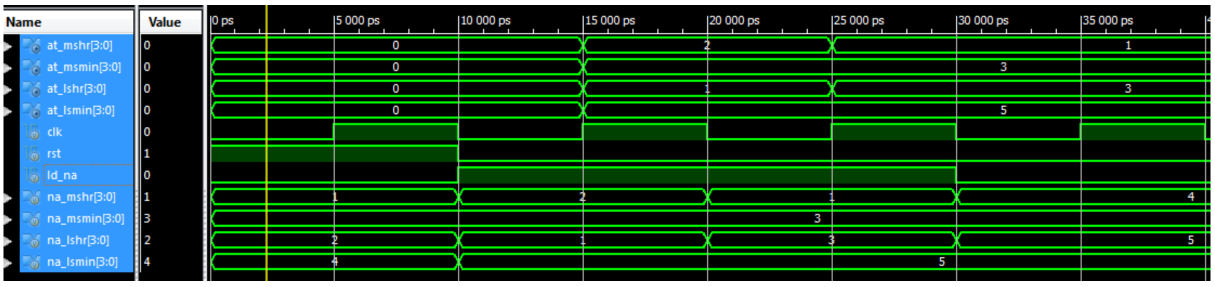


Fig 7: waveform of the alarm register

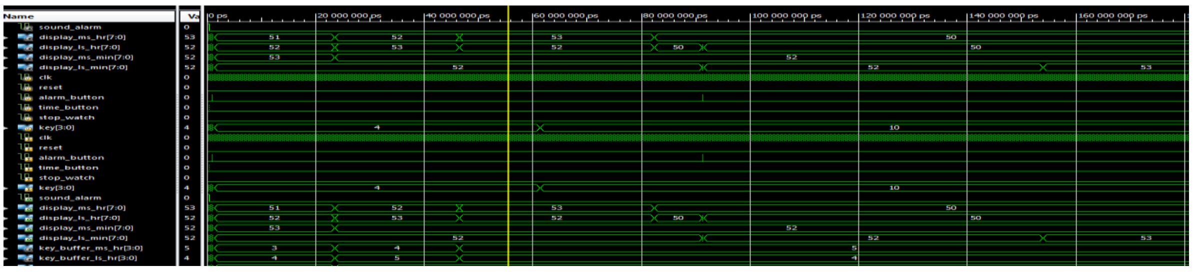
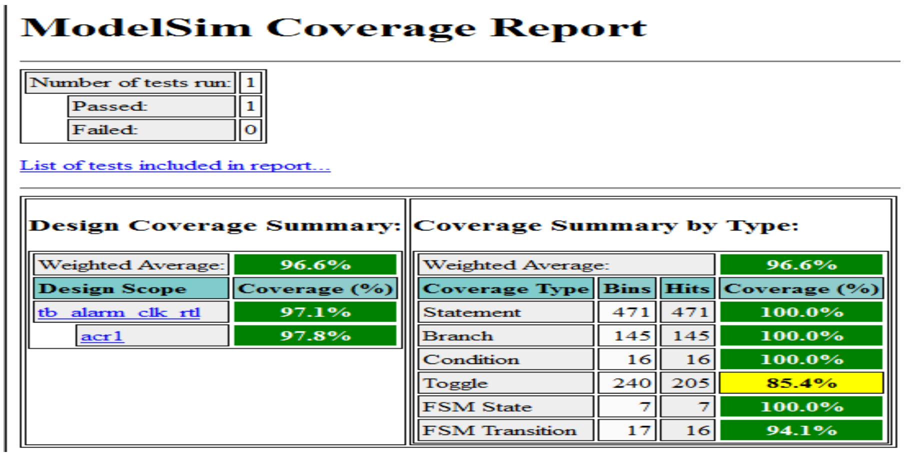


Fig 8: Final output waveform of the topmodule



ModelSim Coverage Report

Number of tests run:	1
Passed:	1
Failed:	0

List of tests included in report...

Design Coverage Summary:

Weighted Average:	96.6%
Design Scope	**Coverage (%)**
tb_alarm_clk_rtl	97.1%
acr1	97.8%

Coverage Summary by Type:

Weighted Average:			96.6%
Coverage Type	**Bins**	**Hits**	**Coverage (%)**
Statement	471	471	100.0%
Branch	145	145	100.0%
Condition	16	16	100.0%
Toggle	240	205	85.4%
FSM State	7	7	100.0%
FSM Transition	17	16	94.1%

Fig 9: Coverage report

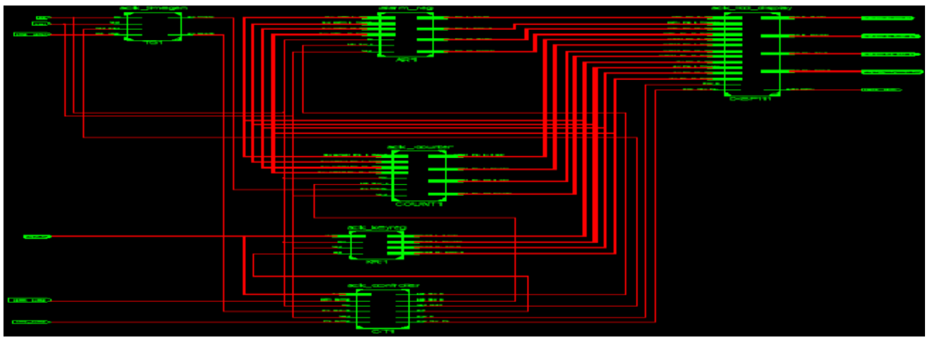


Fig 10: schematic

5.1 System verilog verification environment:

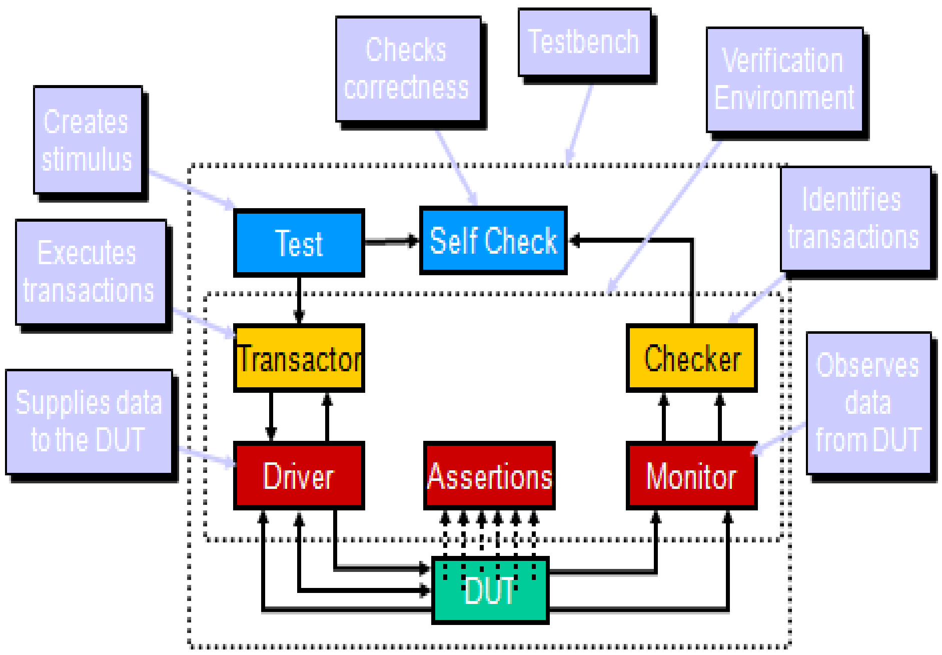
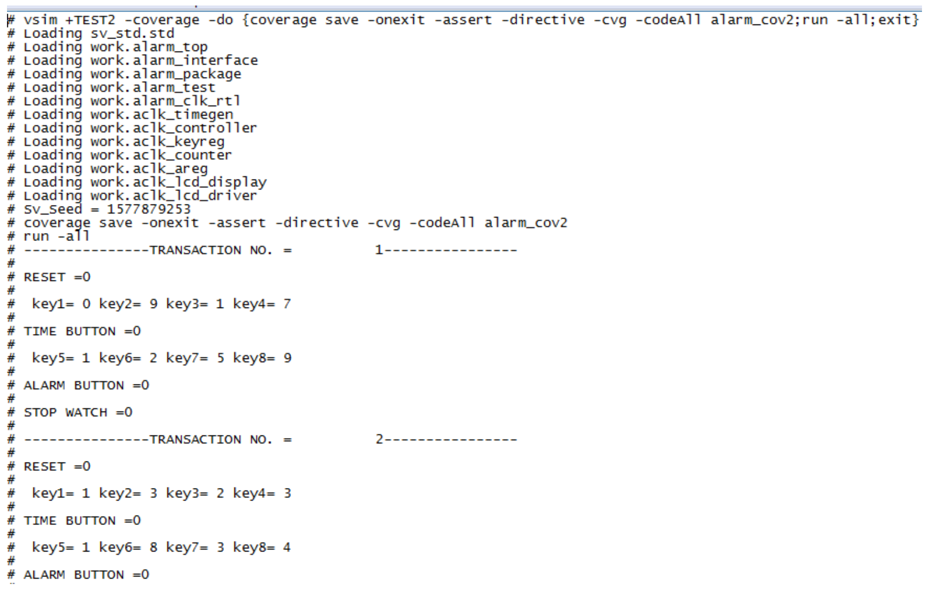


Fig 11: verification environment

In the above figure the transactor will generate the stimulus transaction that which the driver will send the stimulus to the design under test and latter on the corner cases will be verified in the monitor and the checker will verify any loss of data. The total process will be done in the dynamic environment

5.2 system verilog verification report:



```
# vsim +TEST2 -coverage -do {coverage save -onexit -assert -directive -cvg -codeAll alarm_cov2;run -all;exit}
# Loading sv_std.std
# Loading work.alarm_top
# Loading work.alarm_interface
# Loading work.alarm_package
# Loading work.alarm_test
# Loading work.alarm_clk_rtl
# Loading work.aclk_timegen
# Loading work.aclk_controller
# Loading work.aclk_keyreg
# Loading work.aclk_counter
# Loading work.aclk_areg
# Loading work.aclk_lcd_display
# Loading work.aclk_lcd_driver
# Sv_Seed = 1577879253
# coverage save -onexit -assert -directive -cvg -codeAll alarm_cov2
# run -all
# ---------------TRANSACTION NO. =          1----------------
#
# RESET =0
#
#  key1= 0 key2= 9 key3= 1 key4= 7
#
# TIME BUTTON =0
#
#  key5= 1 key6= 2 key7= 5 key8= 9
#
# ALARM BUTTON =0
#
# STOP WATCH =0
#
# ---------------TRANSACTION NO. =          2----------------
#
# RESET =0
#
#  key1= 1 key2= 3 key3= 2 key4= 3
#
# TIME BUTTON =0
#
#  key5= 1 key6= 8 key7= 3 key8= 4
#
# ALARM BUTTON =0
```

VI. CONCLUSIONS

This type of design is very flexible and also easy to interface with the system which requires the clock signal and the alarm functionality which will be used to wake up the devices which are in the sleep mode this can be further developed as the clock with variable frequency of technology independent module so that various projects can use this single module and by this type of design the total design time of a digital circuitry will be reduced

REFERENCES

[1]. J.R. Armstrong and G.W. Woodruff, "Simulation Techniques for Microprocessors," Proc. Design Automation Conf., June 1977, pp. 225-229.
[2] J.R.Armstrong, J. R., "Chip-Level Modelling of LSI Devices, '' IEEE Trans. CAD of Integrated Circuits and Systems, Oct. 1984, pp. 288-297.
[3]M. J. Averill et al. , "SMAS: A program for concurrent state reduction and state assignment of finite state machines," in Proc. ISCAS , 1991, pp. 1781–1784.
[4] L. Benign and G. De Michel, "State assignment for low power dis- sipation," IEEE J. Solid-State Circuits , vol. 30, no. 3, pp. 258–268, Mar. 1995.

AUTHORS

K.R.N.Karthik completed Bachelors degree in E.C.E from MallaReddy engineering college affiliated to JNTU HYD and pursuing Masters from KL University, Vaddeswaram, and Guntur district

P.Kaveri completed Bachelors degree in E.C.E from Sri Vishnu engineering college for women affiliated to JNTU and pursuing Masters from KL University, Vaddeswaram, and Guntur district

Fazal Noorbasha was born on 29th April 1982. He received his, B.Sc. Degree in Electronics Sciences from BCAS College, Bapatla, Affiliated to the Acharya Nagarjuna University, Guntur, Andhra Pradesh, India, in 2003, M.Sc. Degree in Electronics Sciences from the Dr. HariSingh Gour University, Sagar, Madhya Pradesh, India, in 2006, M.Tech. Degree in VLSI Technology, from the North Maharashtra University, Jalgaon, Maharashtra, INDIA in 2008, and Ph.D. Degree in VLSI Specialization from Department Of Physics and Electronics, Dr. HariSingh Gour Central University, Sagar, Madhya Pradesh, India, in 2011. Presently he is working as a Associate Professor and Head of VLSI Systems Research Group at KL University.