Research Article / Review Article

FUPRE JOURNAL 9(4):176-189(2025)
DOI: /10.60787/FJSIR.v9i4.176-189

FUPRE Journal

of

Scientific and Industrial Research

ISSN: 2579-1184(Print) ISSN: 2578-1129 (Online)

http://fupre.edu.ng/journal

Implementation of Digital System Using CPLD/FPGA and Analysis Using XILINX and ISIM Simulation Software

AVWORO, O. L.[1,*]

[1]*Department of Electrical Engineering, Federal University of Petroleum Resources, Effurun, Delta State, Nigeria*

ARTICLE INFO

Received: 12/09/2025
Accepted: 06/12/2025

Keywords

Control Engineering, CPLD/FGPA, Digital Electronics, Electrical Programming and Simulation.

ABSTRACT

The quick growth of digital electronics has made it necessary to create high-performance, reconfigurable systems that can handle today's computational and control demands. With an emphasis on design flexibility, parallelism, and real-time processing efficiency, this research focuses on the implementation of digital systems employing Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). From hardware description in VHDL/Verilog to simulation, synthesis, and hardware implementation on programmable logic platforms, the entire design cycle is examined. Numerous digital modules, including control circuits, arithmetic units, and communication interfaces, were created, tested, and optimized for power efficiency, speed, and logic utilization. For embedded system applications, a comparison of CPLD and FPGA implementations reveals their respective advantages in terms of scalability, cost, and timing performance. Experimental findings show that while CPLDs offer deterministic timing and low-power operation in more straightforward control-oriented designs, FPGA-based systems offer greater computational performance and reconfigurability. By demonstrating how reconfigurable logic devices can close the gap between conventional fixed hardware and adaptable software solutions for contemporary digital systems, this work advances digital design techniques. The research will show how we can implement a traffic light controller system using a CPLD system, self-designed ALU state table and how to obtain the schematic representation of how logic gates and registers interact in a logic circuit system.

1. INTRODUCTION

An arithmetic logic unit (ALU) is a digital circuit that performs integer arithmetic and logical operations. The ALU is a fundamental building block of the central processing unit of a computer, and even the simplest microprocessors contain one for purposes such as maintaining timers. The processors found inside modern CPUs and graphics processing units (GPUs) accommodate very powerful and very complex ALUs; a single component may contain a number of ALUs (Compton and Hauck 2002), Armstrong and Gray (2008), Ashenden (2008).

ALU (Arithmetic and Logic unit) is a digital circuit that performs logical and arithmetic operations. ALU is found inside every microprocessor or microcontroller in order to perform logical and arithmetic operations. Logical operation involves all logical functions like adding two or more inputs, OR, NOT, NOR etc. Arithmetic operation involves function like addition, subtraction etc.

*Corresponding author, e-mail: leslieavworo@gmail.com
DIO:/10.60787/FJSIR.v9i4.25-46
©Scientific Information, Documentation and Publishing Office at FUPRE Journal

In this research, an 8-bit ALU is being designed. An 8 bits ALU simply means it will process a 8 bit data and provide output of also 8 bits. So, in the entity section A and B are two inputs of 8 bits and R is the output of also 8 bits. OP is a selection line of 4 bits which is required to perform various function on the inputs A and B (Inputs). So for different value of OP different operations are performed on the two inputs for example when Op="0001" A and B are passed through an OR gate and an output is generated on R (Output). In the architecture case statement is used in which according to changing value of OP different operations are performed on inputs A and B. This is a program of basic ALU was further modified to add additional functions like shift left, shift right and decoder functions where also added to fulfill the requirement of the project (Kuon and Rose 2007 and Perry 2014**).**

2. LITERATURE REVIEW

2.1CPLD-Architecture

CPLDs have a unique position in the spectrum of programmable logic. In contrast to the fine-grained LUT-and-routing fabric of FPGAs, these are macro cell-based devices with a comparatively centralized programmable interconnect and combinational logic implemented using fixed-function macro cells (product-term arrays). CPLDs are well suited for glue logic, startup and reset control, state machines, address decoding, and basic protocol bridging in embedded systems because of their structural differences, which also give them deterministic timing behavior, predictable power/boot characteristics, and instantaneous (or very fast) configuration behavior. On the other hand, CPLDs are rarely the first option for high data path or large-memory applications because they usually offer lesser logic density and fewer embedded resources (block RAMs, DSP slices) than FPGAs. When advocating CPLDs for control-oriented jobs and FPGAs for compute- or memory-intensive designs, reviews and technical guides highlight these trade-offs (Kuon *et al.* 2008).

2.2 Xilinx-CPLD-software

The XC9500 series and CoolRunner and CoolRunner-II devices are examples of Xilinx's CPLD families. Due to their affordability, extensive tool support, and comprehensible documentation (device datasheets, application notes, and quick-start guides), these families were incorporated into Xilinx's ISE/WebPACK pipeline and were extensively utilized in both industry and academics (Patel and Thakkar 2019). For instance, the CoolRunner-II family, designed for low-power, multi-voltage embedded applications, is specifically documented for support in ISE's WebPACK; datasheets and quick-start guides provide information on macrocell counts, I/O behavior, and in-system programming features that directly influence implementation choices (timing constraints, reset strategies, and JTAG programming). Researchers must match the macrocell count, I/O voltage domains, and configurable features to the functional and non-functional needs of the design when choosing a CPLD for a project.

3. METHODOLOGY

3.1 Design procedure

a. Analyze the system and provide discussions on the requirements and appropriate ways to design it.
b. Develop a VHDL code to design the system as per the requirement and your design concept (Traffic light controller).
c. Functional verification of the VHDL code using Xilinx software tool with emphasis on the RTL **(**Register transfer level) view.

d. Implementing this using the CPLD/FPGA kit.

3.2 Analyze the system and provide discussions on the requirements and appropriate ways to design it

The first step into designing an ALU is to design a table of operation or a function table which shows the simple operations mechanism of the ALU. Table 1 below shows the functions and codes to establish certain functions in the ALU to be designed.

Table 1: Operation Table

ALU OPERATION	DESCRIPTION	OPERATION CODE
ADDITION	A+B	0000
SUBTRACTION	A-B	0001
AND	A AND B	0010
NOR	A NOR B	0011
OR	A OR B	0100
NAND	A NAND B	0101
XNOR	A XNOR B	0110
NOT	A NOT	0111
SHIFT LEFT		1000
SHIFT RIGHT		1001
DECODER		1010

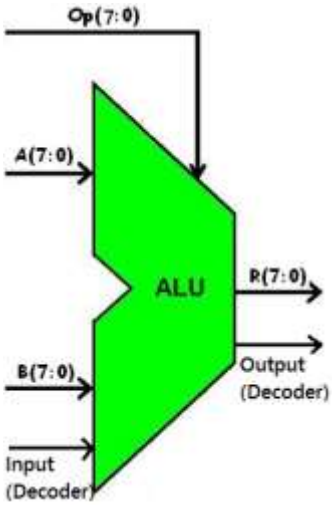


Figure 1: ALU Configuration (Nagarjuna et al. 2018)

3.3 Develop the VHDL codes to design the system as per the requirement and your design concept.

The VHDL code was developed using the XILINX ISE software. Firstly. VHDL (VHSIC HARDWARE DESCRIPTION LANGUAGE) it's simply a hardware description language used in the design of automated electronics. In this case its being used for general purpose programming. VHDL programming is very similar to the very popular C-Programming but much simpler. The first event of a VHDL program is the declaration of entity followed by the architecture of the system and at the end, also processes are properly ended. To showcase the functionality of our designed ALU, we would be implementing it into a traffic light controller system to show how our simple arithmetic table can be applied

in and electronic circuit (Payal 2020), Raghuram and Bhupathi (2017), and Gopal *et al.* (2014).

In the traffic light controller VHDL code designed for this project, clock and reset where declared as single bit inputs while LED was declared as a 12bit vector output during the entity declaration phase also the signal count and the different stages of the controller was implemented into signal in the beginning of the architecture. The code was designed in such a way that when the reset is high, the system starts counting in stage 0 and after the first 3 seconds it moves over to stage 1 and after 5 seconds it drifts to stage 3. The process continues until the last stage is reached and the system loops back to the start so that the whole process begins again (Nandkishor et al. 2012). The different stages are associated with the logic designed in task A. After the codes was completed, it was checked for syntax errors to see if it was correctly written. Screenshots of the designed code is shown in the Figures 3 and 4 below.

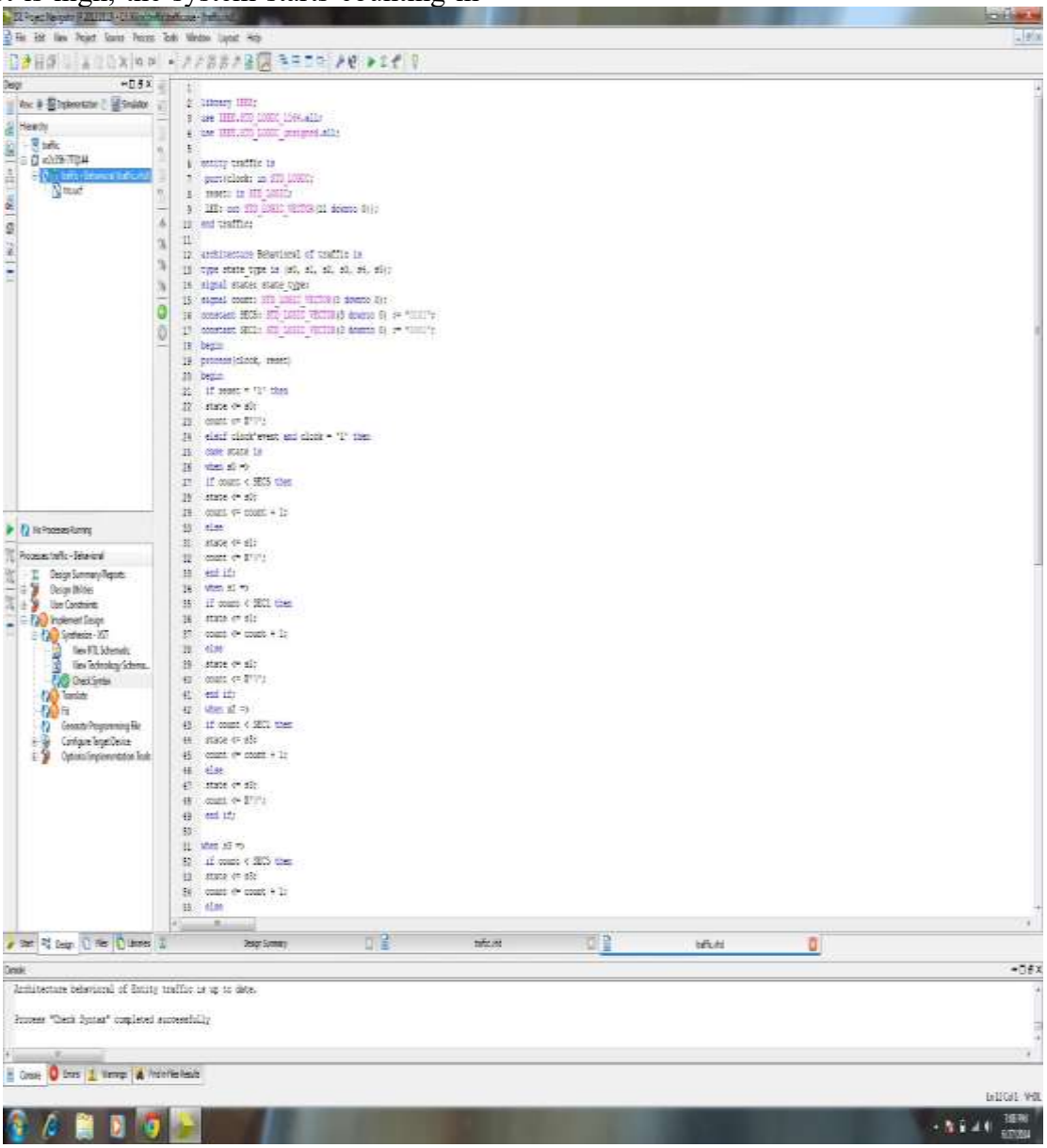


Figure 2: VHDL code for traffic light controller Part 1 (Brown,S. and Vranesic, Z., 2019)

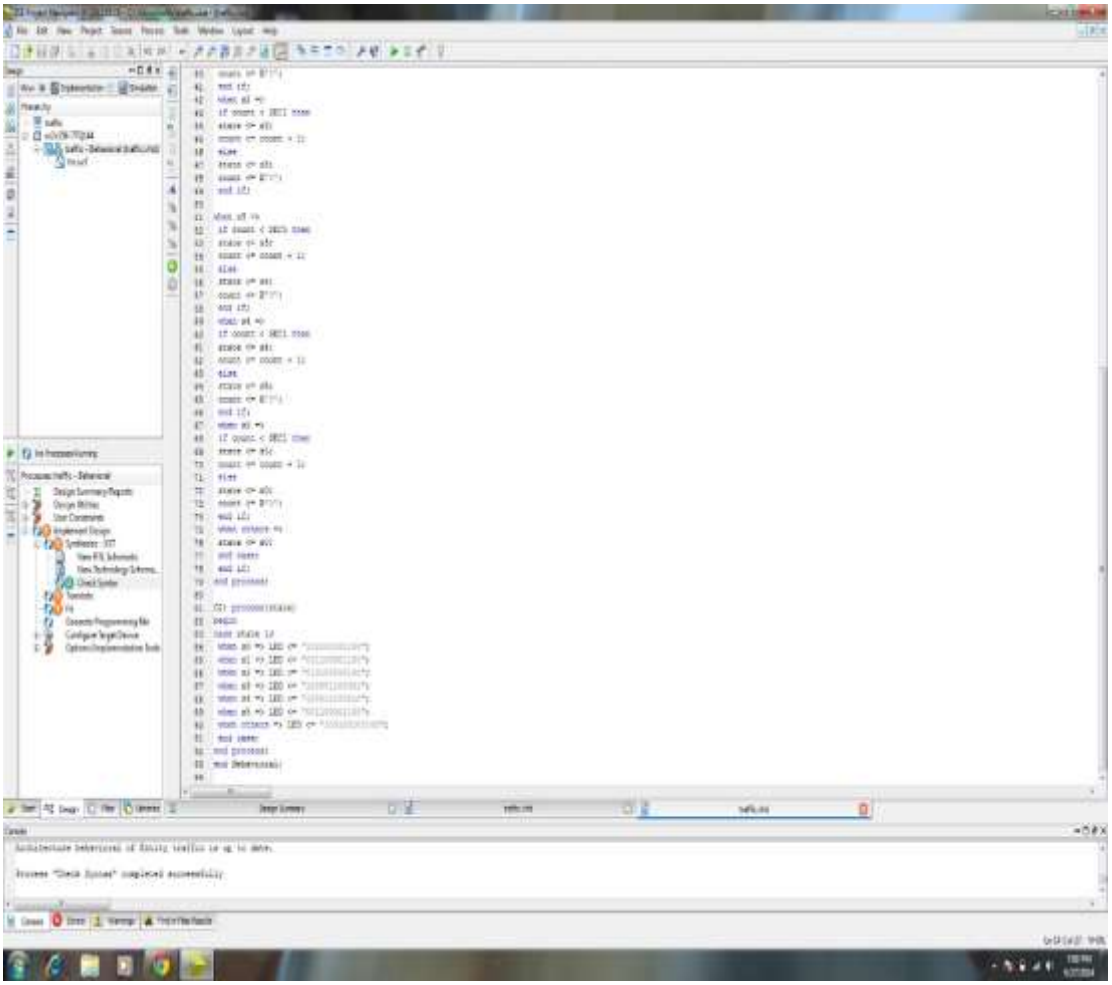


Figure 3: VHDL cod for traffic light controller Part 2 1 (Brown and Vranesic 2019)

3.4 Functional verification of the VHDL code using Xilinx software tool with emphasis on the RTL (Register transfer level) view

For the simulation of the written VHDL code to get the proper logic, a test-bench source file was opened. This helps to activate the pulse of clock generation in the code. The XILINX platform was switched to simulation mode so as to simulate the newly formulated test-bench source file to ensure that the developed code is functioning properly. From the simulation, the output representation was a replica of the logic designed in task A which proves to show that the code was appropriately designed. The figures 4 to figure 6 below show the test-bench code and the ISIM simulation of the VHDL code in XILINX.

RTL (Register transfer level) view of the designed system

The RTL viewer simply allows for the designer to view the schematic representation of the internal structure of its design. The two stages of the RTL view representation is show in the figure below. Firstly, RTL viewer option is double clicked in the XILINX platform after that, every single detail that appears on the next page is added to the RTL view list and then the create schematic option is clicked. This provides the schematic representation of the entire VHDL code designed for the traffic light system. Below are the figures for the RTL view.

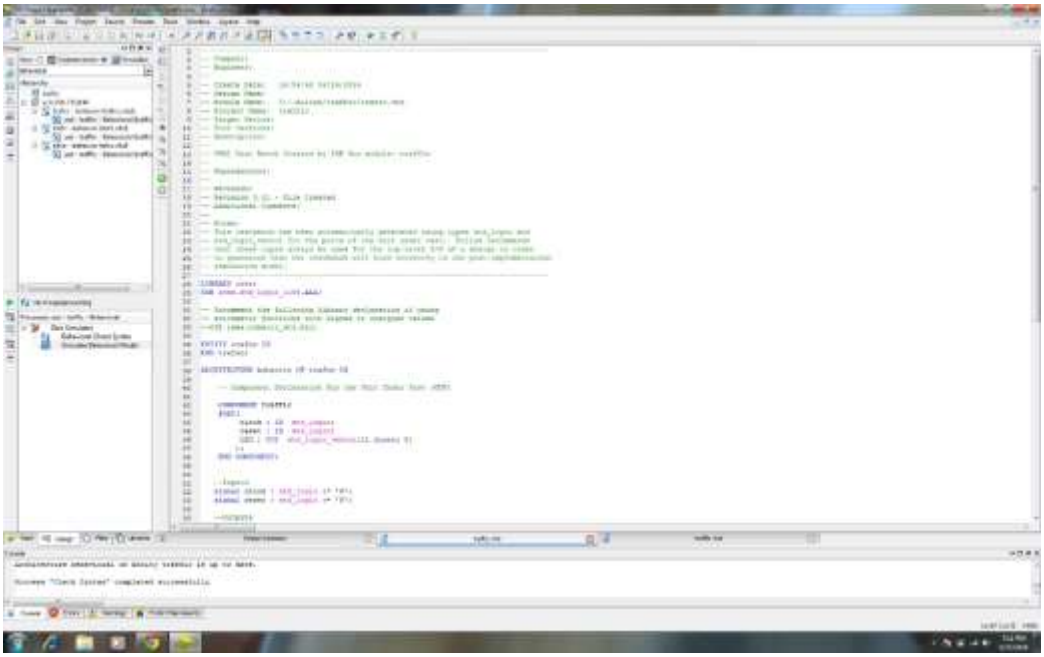


Figure 4: Test Bench code Part 1 1 (Brown and Vranesic 2019)

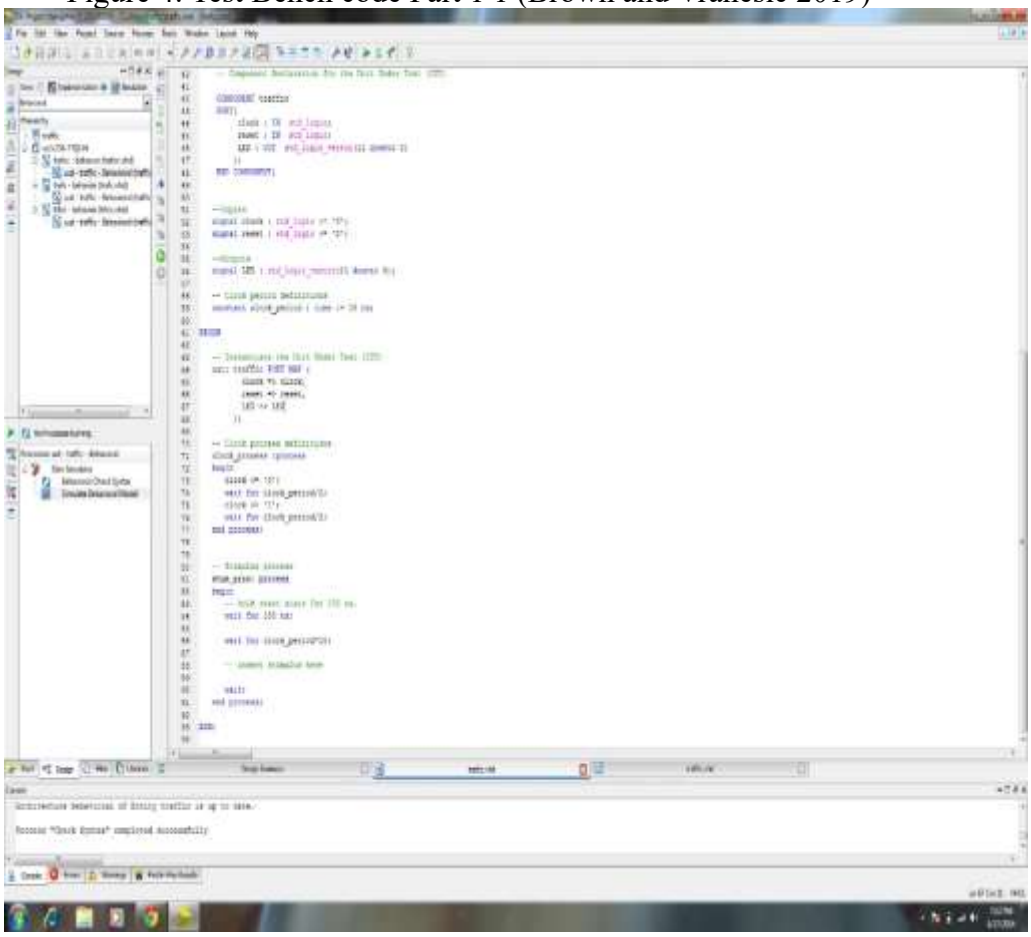


Figure 5: Test Bench code Part 2 1 (Brown and Vranesic 2019)

*Corresponding author, e-mail: leslieavworo@gmail.com
DIO:/10.60787/FJSIR.v9i4.25-46
©Scientific Information, Documentation and Publishing Office at FUPRE Journal

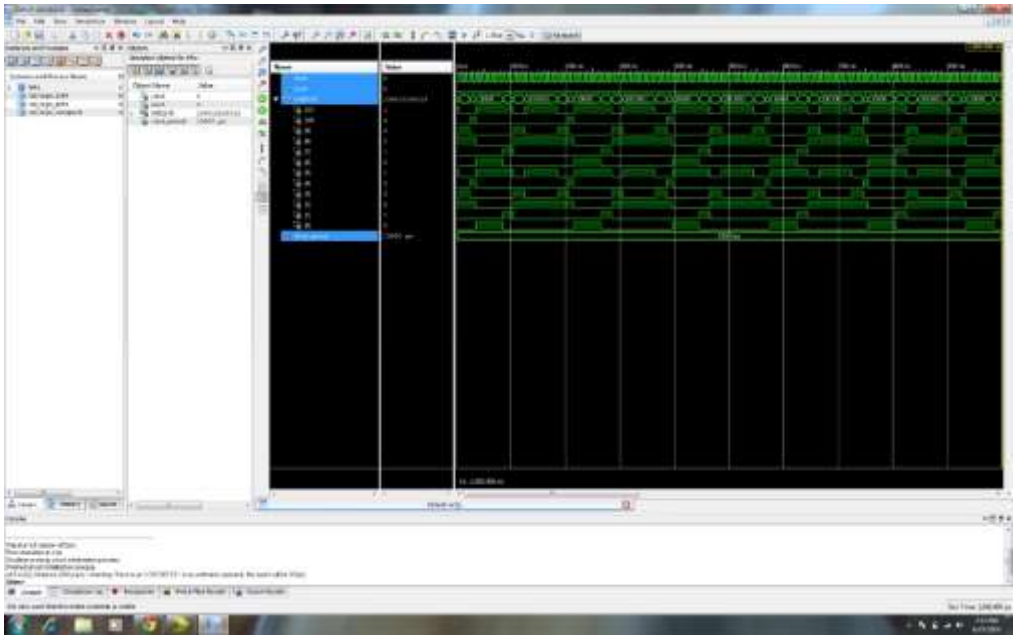


Figure 6: Test Bench Simulation of Traffic light system 1 (Brown and Vranesic 2019)

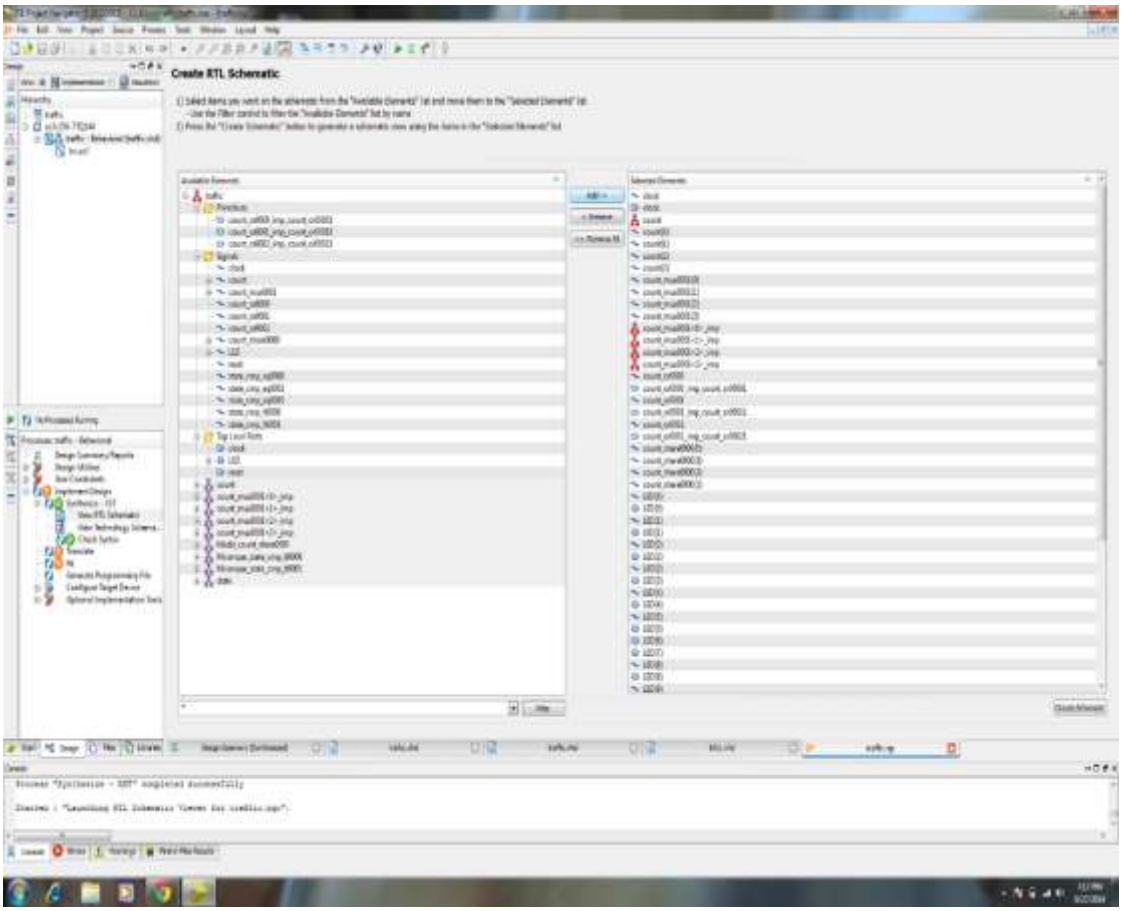


Figure 7 RTL (Register transfer level) Schematic step 1 1 (Brown and Vranesic 2019)

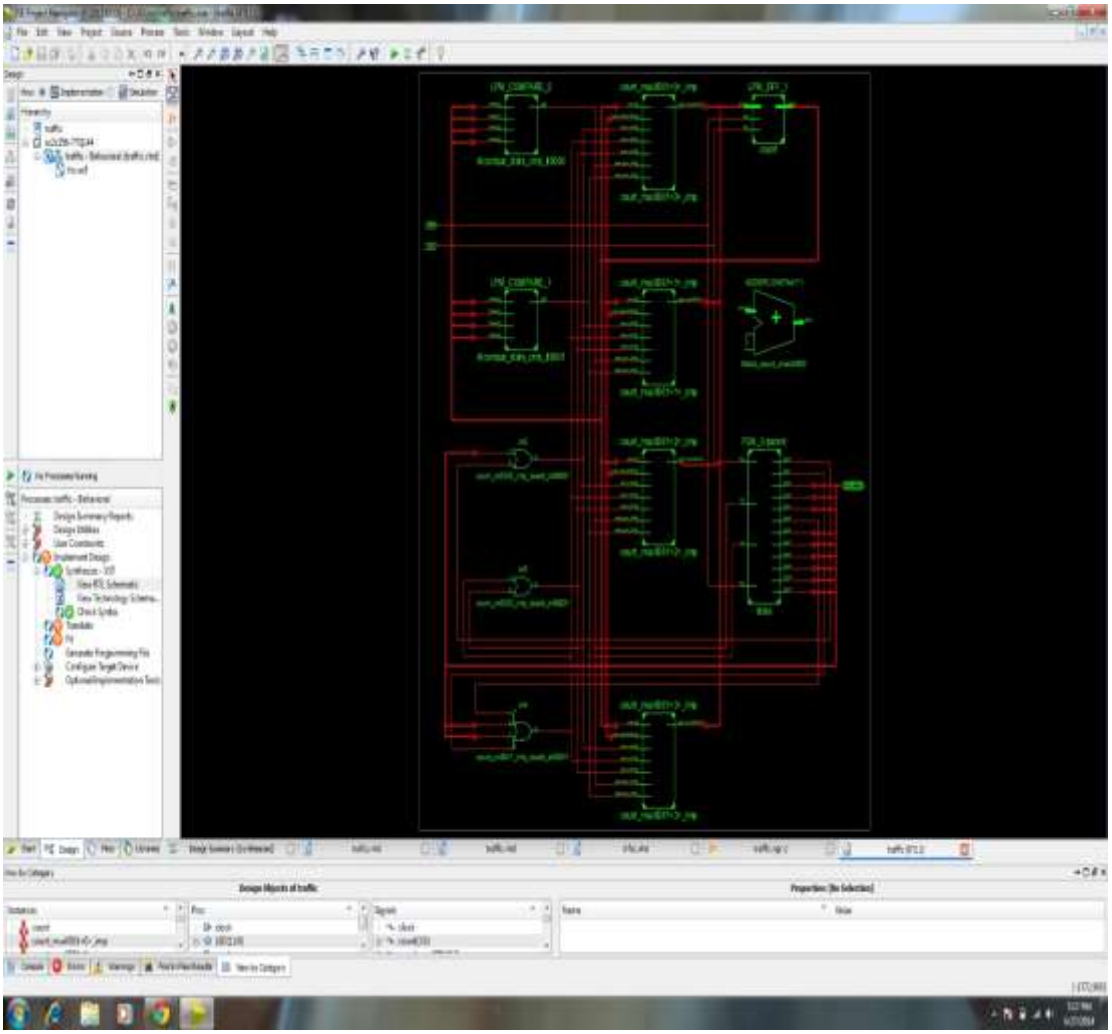


Figure 8 RTL (Register transfer level) SCHEMATICS 1 (Brown and Vranesic 2019)

3.5 Implementation of the design using the CPLD kit

The Cool Runner II XC2C256 CPLD kit was used for the design implementation, this process of physically placing the design into the target device.

The implementation of the design on the CPLD kit is another complicated process involved in the FPGA/CPLD implementation. It involves a lot of steps that are shown in the figures below. Firstly, after the VHDL program has been completed, the configuration is the next step which is used to assign functions to particular pins in the CPLD kit corresponding to the design layout of the VHDL program (Jamie, P. 2020). The next phase is the time constrain configuration which is set to global clock is used to configure the system delay and also to set the CPLD internal clock to function as the clock pulse for the system. Lastly, configure impact mode is used to burn the designed program into the CPLD kit and during this process, the kit is already connected to a PC and power supply. After the kit has been completely programmed, it is then verified and ready for connection (Gopinath and Arikesh 2013).

The traffic light LEDs are connected in a breadboard which are connected to their assigned pins in the CPLD board. When the kit is turned on and the reset is high, the traffic light controller system starts functioning as designed in the operational logic table meaning the project is success.

The figures 9 to figure 17 show the various steps involved in the hardware implementation and the final hardware design is shown below.

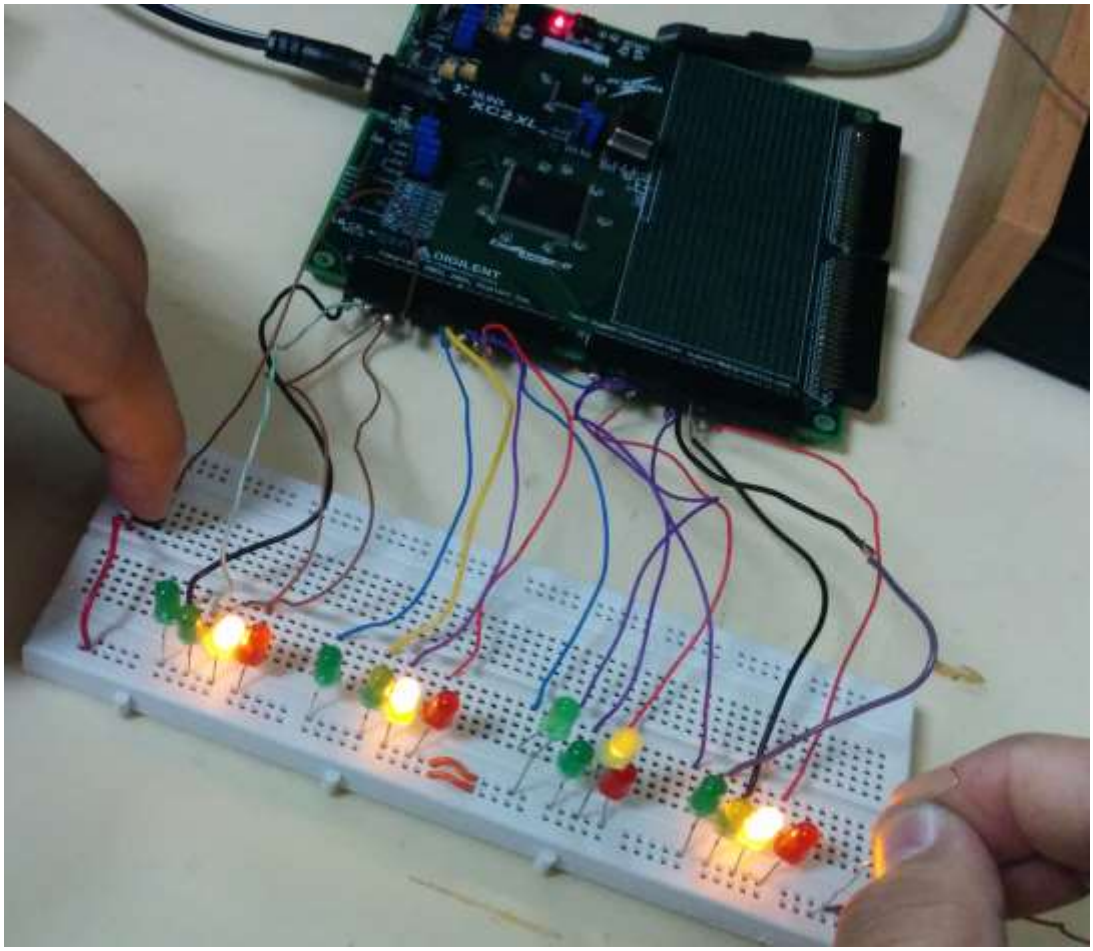


Figure 9 Hardware Design

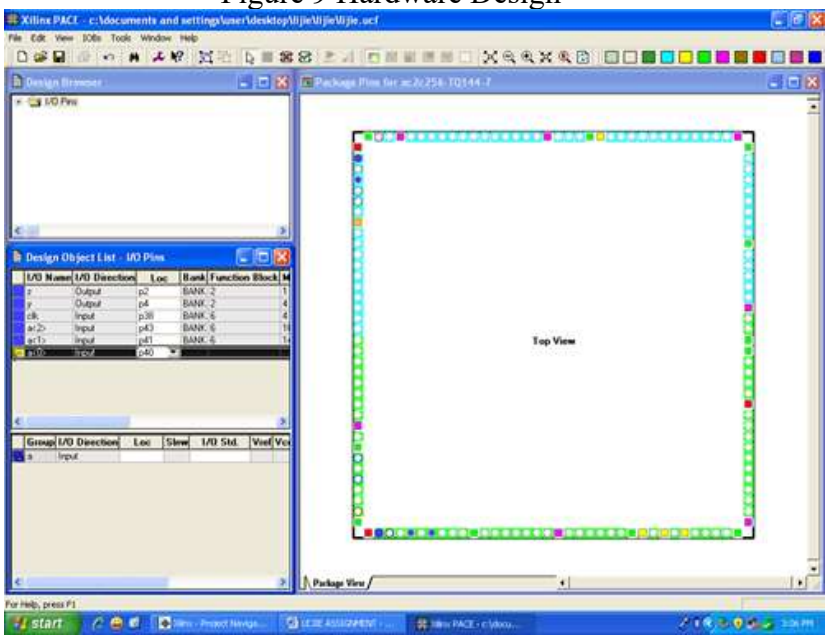


Figure 10 Pin configuration (Xilinx Pace) (Chu, P., 2017)

*Corresponding author, e-mail: leslieavworo@gmail.com
DIO:/10.60787/FJSIR.v9i4.25-46
©Scientific Information, Documentation and Publishing Office at FUPRE Journal

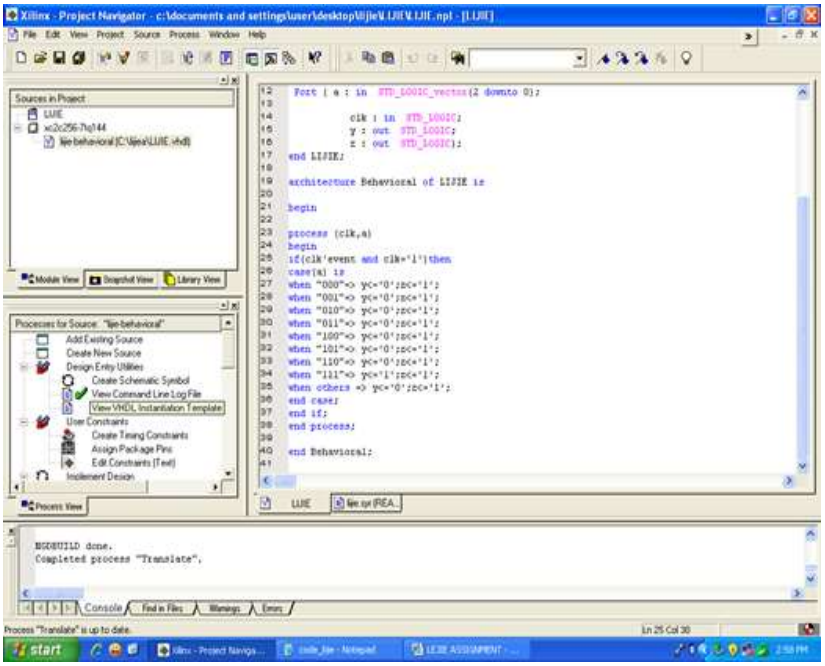


Figure 11 VHDL module after pin configuration (Chu, P., 2017)

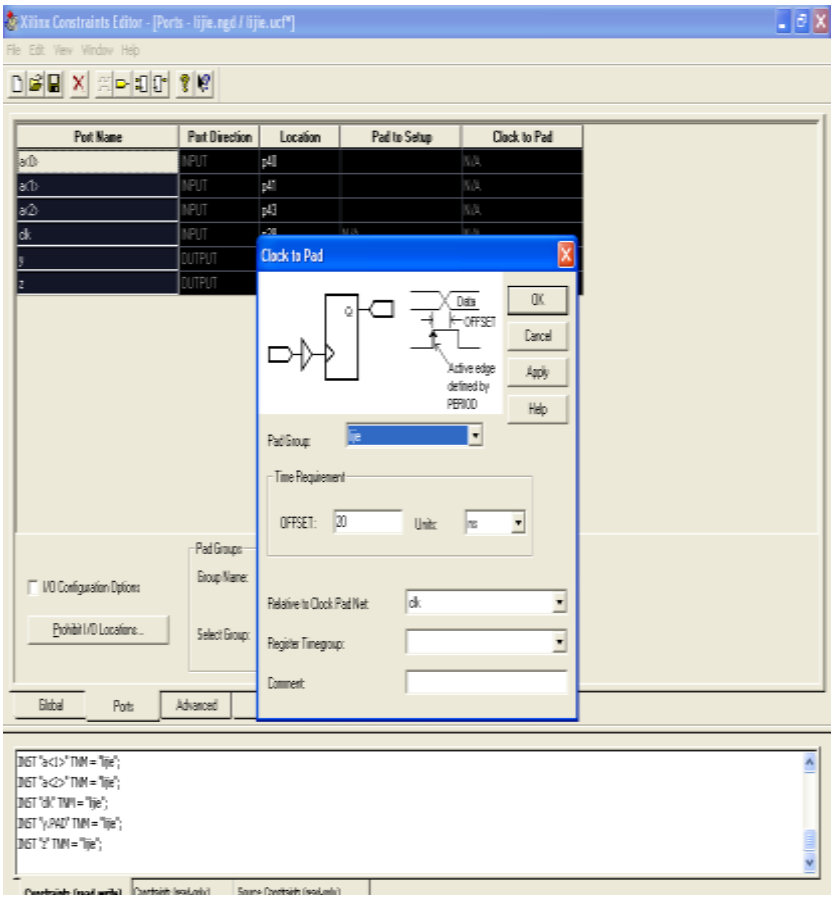


Figure 12 Timing constrain configuration (Chu, P., 2017)

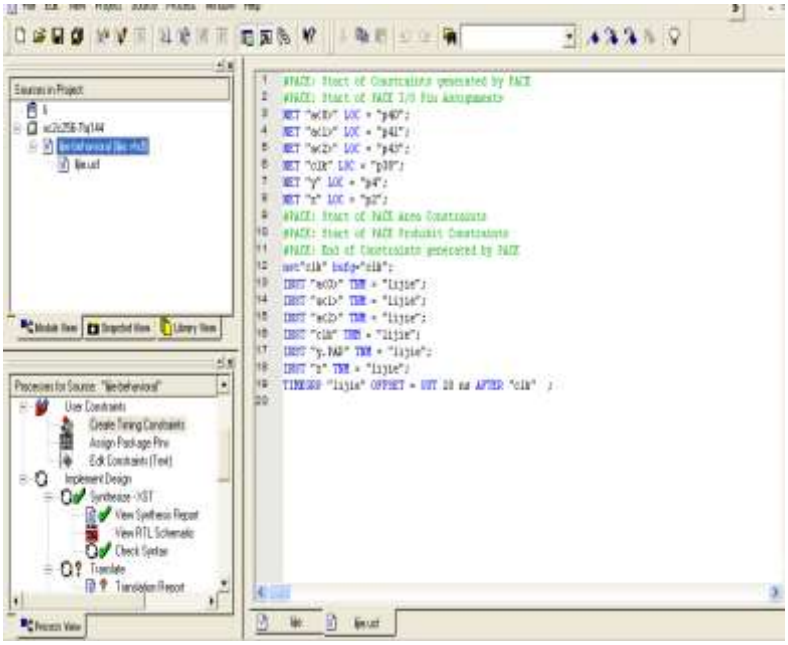


Figure 13 Module after Timing constrain (Chu, P., 2017)

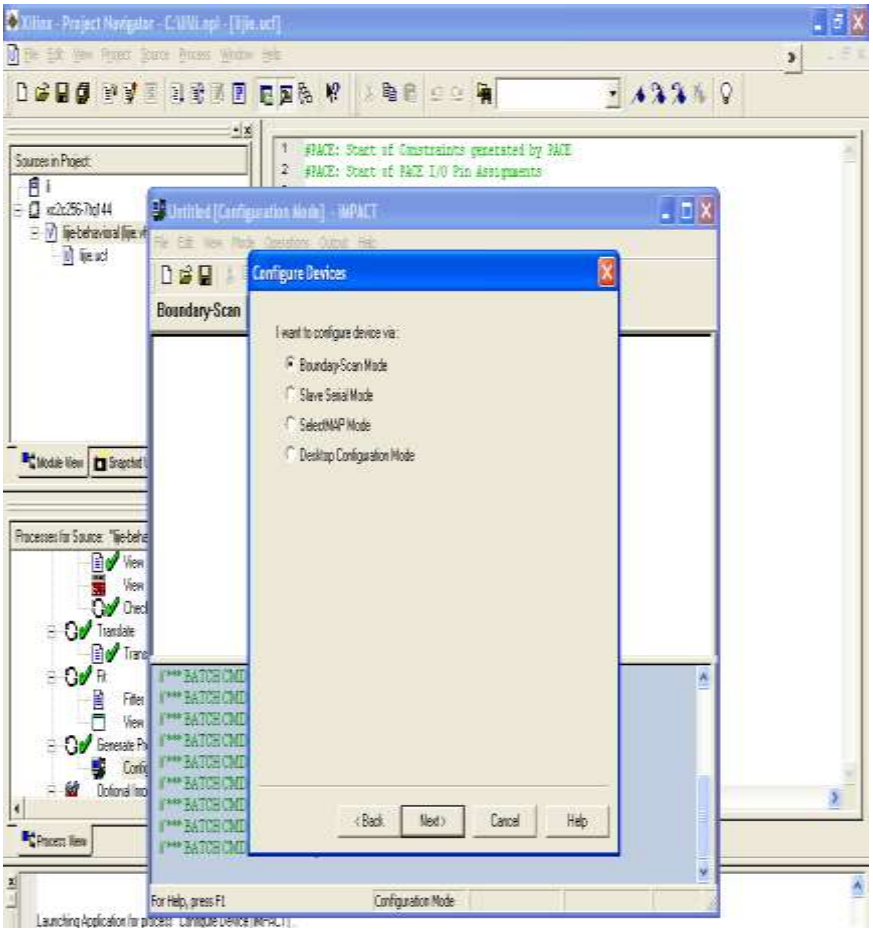


Figure 14 Configuration Impact mode (Chu, P., 2017)

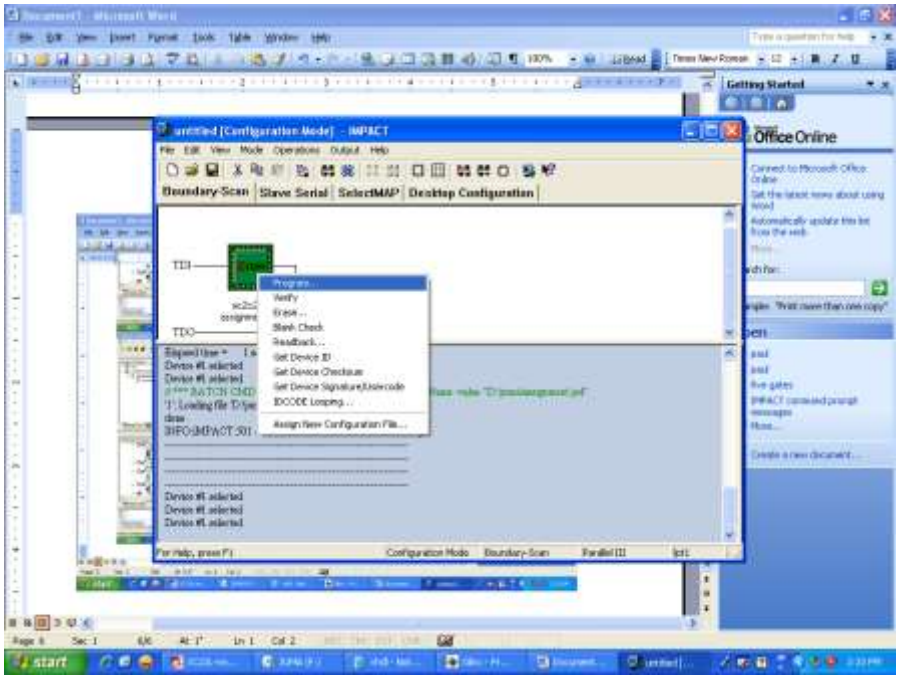


Figure 15 Configuration impact mode (Routine) (Chu, P., 2017)

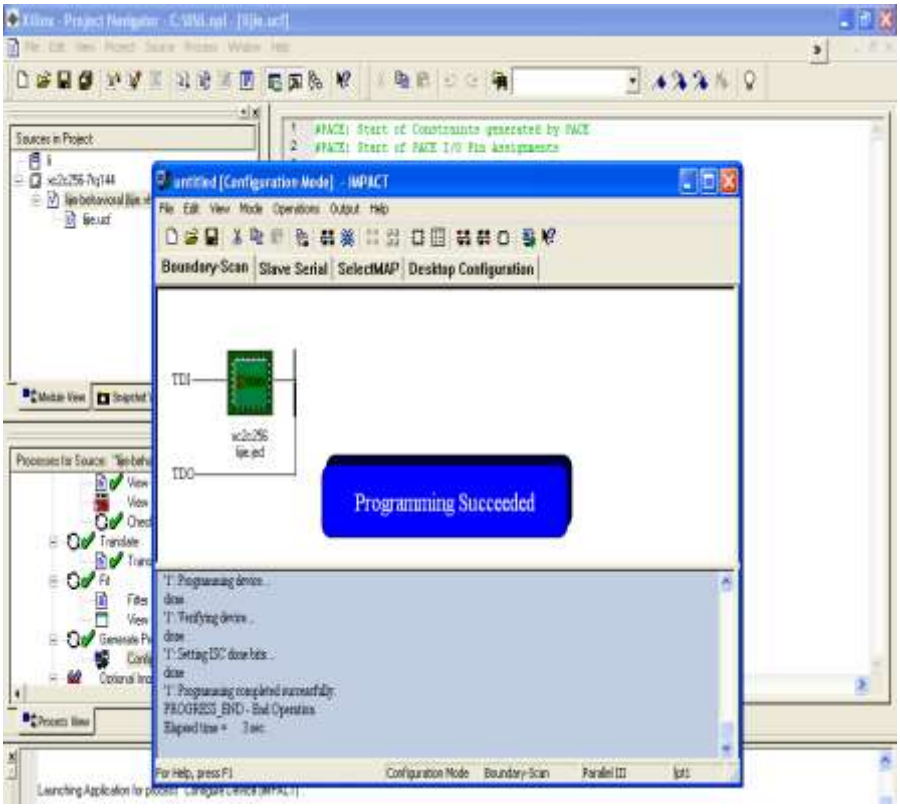


Figure 16 Configuration Succeeded (Chu, P., 2017)

The screen below pops out when the program which has been successfully burnt into the CPLD kit was verified

It displayed “verify succeeded” this implies that all the procedures were done and the burnt in program is fully functional.

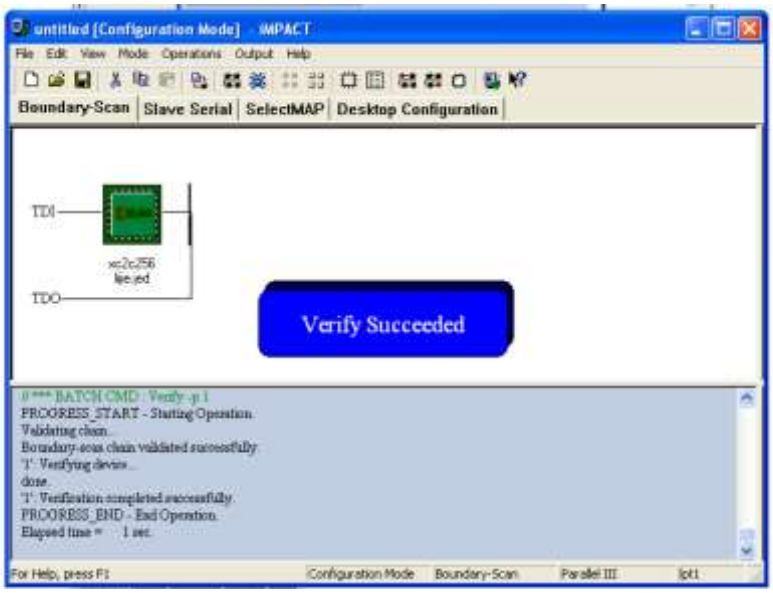


Figure 17 Verification Process succeeded (Chu, P., 2017)

4. RESULT DISCUSSION

4.1 Result

After drawing the state table and the state diagram using the mealy machine design for the logic circuit there was no need to minimize the state diagram. The state diagram information was used to write a VHDL program for the logic circuit program (Traffic light controller) and implemented using the Xilinx IDE tool.

The designed program was benched marked and tested for correctness and copied into the CPLD hardware kit for functionality testing using the XILINX software. During testing, it was noticed that the traffic controller system was functioning according to design and performing the basic arithmetic functions it was programmed to do.

The project also made it possible to view Register transfer level view (RTL view) that show the digital circuit internal structure and graphical schematic representation that show how data is transferred between logic gates and registers.

5. CONCLUSION

This system shows how we can design and implement an ALU and programmed it into a CPLD hardware kit for the designing of basic electronic devices. The software analysis tools was capable of showing the interactions of registers and logic gates in our designed system. The counter was design and implemented using Xilinx software tool to simulate the ISE simulation. A vast amount of knowledge and the use of VHDL/Xilinx software was gain during this research and also knowledge of designing sequential circuits and implementation.

REFERENCES

Armstrong, J. R. and Gray, F., 2008. *VHDL Design Representation and Synthesis*. 2nd ed. Upper Saddle River, NJ: Prentice Hall.

Ashenden, P. J., (2008). *The Designer's Guide to VHDL*. 3rd ed. Burlington, MA: Morgan Kaufmann.

Brown, S. and Vranesic, Z., 2019. *Fundamentals of Digital Logic with VHDL Design*. 3rd ed. New York: McGraw-Hill.

Chu, P., 2017. *FPGA Prototyping by VHDL Examples: Xilinx Zynq-7000 Edition*. Hoboken, NJ: Wiley.

Compton, K. and Hauck, **S.** (2002). Reconfigurable computing: a survey of systems and software, ACM Computing Surveys**,** 34(2): 171–210.

Gopal, P. G., Shruti, V. G., Irshad, A. Z. and Sagar, P. R. (2014). Review of speed control and automatic braking system", *International Journal of Engineering Research and Technology*, 3(2): 474-476.

Gopinath, G. and Arikesh, A. (2013). "Development of Three Step Charging H-Bridge based Sine wave Inverter", *International Journal of Advanced Trends in Computer Science and Engineering*, 2(2):149-153.

Jamie, P. D. (2020). How electronic throttle control systems works. [Online]. Available: https://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/electronic-throttle-control-systems.htm. [Accessed: 11 – August – 2020].

Kuon, I. and Rose, J. (2007). Measuring the gap between FPGAs and ASICs', IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 26(2): 203–215.

Kuon, I., Tessier, R. and Rose, J. (2008). FPGA architecture: Survey and challenges', Foundations and Trends in Electronic Design Automation, 2(2):135–253.

Nagarjuna, G., Padmini, M., and Kumar, R. (2018). Low power delay product 8-bit ALU design using decoder and data selector. International Journal of Pure and Applied Mathematics, 118(16): 1195–1208.

Nandkishor, P. J. and Ajay P. T. (2012). "Speed control of DC motor using analogy PWM Technique". *International Journal of Engineering Research and Technology*, 1(9):1-5.

Patel, M. and Thakkar, H. (2019). Design and implementation of FPGA-based systems: A review', International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 8(6): 2051–2060.

Payal, D. (2020). Lack of sign boards, faulty design of speed breakers causes accidents too. [Online]. Available: https://timesofindia.indiatimes.com/city/ludhiana/lack-of-sign-boards-faulty-design-of-speed-breakers-cause-accidents-too/articleshow/76737880.cms. [Accessed: 4-August-2020].

Perry, D.L., (2014). *VHDL Programming by Example*. 5th ed. New York: McGraw-Hill Education.

Raghuram, S. and Bhupathi, E., (2017). *VHDL Made Easy!* New Delhi: Laxmi Publications.