DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OBAFEMI AWOLOWO UNIVERSITY

ILE-IFE, OSUN STATE

CPE 509

HARDWARE DESIGN LABORATORY

DR. A.O. OLUWATOPE

GROUP MEMBERS

TAIWO OLUWATOMISIN O.	CSC/2014/081
TALABI OLUWASEGUN O.	CSC/2014/082
YEKO JOSEPH O.	CSC/2014/085
YUSUFF AIRAT O.	CSC/2014/086
ABON ADEPEJU T.	CSC/2015/109

Table of Contents

CHAPTER ONE

INTRODUCTION

1.1 Background Study

A logic analyzer is an electronic instrument that captures and displays multiple signals from a digital circuit or system. This is done by converting the captured data into timing diagrams, assembly language or correlating assembly with source level software. This system captures digital data in large amounts and can be triggered on a complicated sequence. It uncovers defects in hardware that are not found in simulations due to complex models or multiple clock crossing domains.

1.2 Aim

The aim of the project is to build a digital logic analyzer, which analyzes input digital signals and displays in form of a waveform.

1.3 Objectives

1. To design an annular pulse generator which will serve as the signal to be analyzed for the logic analyzer.
2. To design the logic circuit for the logic analyzer, with the output pulses from the pulse generator as input signal.
3. To simulate and implement both designs and load them unto the FPGA boards.
4. To test and evaluate the implemented system.

1.4 Methodology

1. VHDL code will be written to implement the operations of a pulse generator and the output will be in form of continuous pulses.
2. The logic for the analyzer to translate and analyze the output gotten from the pulse generator is also written, compiled and implemented.
3. The two systems will be loaded unto the FPGA Altera boards to test proper functionalities.
4. The logic analyzer will be aided by the inbuilt SignalTap II Analyzer for Altera board and trigger conditions and clock configurations are set up to initiate the analysis.
5. After all configurations and testing, the analyzer is loaded and the analysis from the pulse generator outputs will be displayed in form of waveforms.

CHAPTER TWO

METHODOLOGY

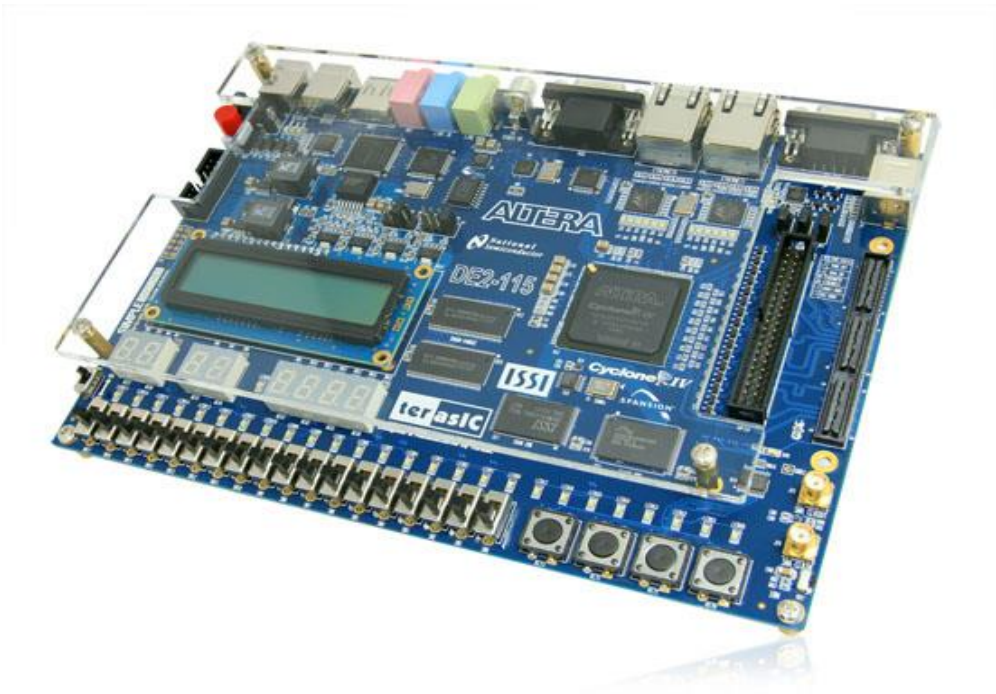
2.1 System Architecture

The digital logic analyzer was developed using readily available devices and software, that fit into the realization of its purpose, which when implemented on a large scale is fully practicable with very little margin of error or design variation. The subsection below describes the equipment used, their descriptions and how they function in the realization of this project.

2.1.1 Altera DE2

The Altera DE2 is a development and education board used to provide a backbone or "processor type support" for prototypes in several practical implementations of real-life projects in multimedia, storage and networking. It uses ultra-modern technology in hardware tools to expose designers and engineers to a vast range of topics. The Altera DE2 is a development kit having several physical components such as generic expansion headers, VGA Output, 7 Segment Display, LED, Push Button, Slider Switch and so much more.

To implement this digital logic analyzer, two Altera De2 boards were used. the FPGA on the Altera Board typically works as a processor for the system development. It executes the code written in VHDL, and generates an output based on pre-set conditions from the code.



Altera DE2 Board

2.1.2 FEMALE-TO-FEMALE WIRES

A female wires is a type of connector that consists of a jack into which a male connector can be inserted. It is usually present at the end of a cable or other hardware such that a secure physical and electrical connection between two or more devices is made possible.

A female-to-female wire is thus a connector with jack into which another female wire can be inserted.

Female to Female Jumper Wires

2.2 Mode of Operation

This system comprises two sub-systems, both based on two different functionalities of the Altera De2 board.

- The VHDL code is compiled on the Quartus II software and sent to an Altera DE 2 board. This is used to generate pulse signals to be analyzed.
- An input pin was assigned to a 50MHz inbuilt clock of the board, to serve as a driver. The design is loaded unto the board, having the output sent to four pins in the expansion header, this having been already configured in the design. The output here is the generation of four continuous pulses. This is the first sub-system.
- A different VHDL code is compiled using the same Quartus II software, and sent to another Altera DE 2 board, which serves as the second sub-system. This code utilizes the signal-tap functionality of the software for analysis of the pulses generated from the first sub-system.

- The four pins in the expansion header of the first board are connected via female-to-female wires to the expansion header of the second board, which had been pre-loaded with the analyzer logic.
- Consequent upon the connection of the two sub-systems, the whole system is triggered, based on the analyzer logic in the second sub-system. This generates continuous waveforms as output.

CHAPTER THREE

IMPLEMENTATION AND RESULTS

This chapter consists of the implementation of the design, testing and evaluation of the achieved system.

3.1 Implementation

The system is made up of two system setups: Annular pulse generator and the logic analyzer. The implementation is set up as follows:

3.1.1 Annular Pulse Generator

An annular pulse generator produces regular pulse sequences with equal or unequal time intervals. This serves as the input to the digital logic analyzer.

A block diagram was designed to simulate the operation of the pulse generator. The design consists of an input pin, four D-flipflops, AND, NOT and NAND gates, a VCC pin and four output pins. The input pin was assigned to one of the clock oscillators inbuilt into the FPGA board (the 50MHz clock) and used to drive the system. The four output pins were then used to drive the second system (the logic analyzer). The design is loaded unto the board, the outputs were sent to four configured pins on the expansion header and they had values of the generated pulse signals from the system. The VHDL code derived from the block diagram used to generate the pulses as well as the RTL views are included in the appendix.

3.1.2 Logic Analyzer

The logic analyzer was implemented using the FPGA board's SignalTap II analyzer. The outputs from the pulse generator were sent in to another FPGA board using the pins in the board's

expansion header for analysis. The VHDL code used to configure the operation of the analyzer is included in the appendix. In the SignalTap window, the signal clock is configured to use one of the inbuilt clock oscillators (50MHz clock) and one of the continuous pulse outputs from the generator sent in as input to the analyzer is used as a trigger. The trigger is configured to be an advanced triggers of two discrete switch inputs and one continuous pulse input. It's set up in such a way that low and high states from the inputs initializes the analysis and the results are displayed in a waveform. The design is compiled and loaded onto the second FPGA board and the triggers are used to initialize the analyzer.

Thus, the system operates in the following way: The pulse generator is loaded unto the first FPG board and this generates four outputs of continuous pulses. These pulses are then sent to the second FGPA board, which has been loaded with the analyzer logic, and the triggers initiate the analysis of the input pulses. The analysed pulses is shown in form of a waveform in real time.

3.2 Testing and Evaluation

This describes the testing and evaluation of the logic analyzer. Software testing was mainly used and it refers to the testing of the VHDL code by determining whether the VHDL code of both the pulse generator and the logic analyzer is consistent with the function for which it was written, whether they can be loaded on the FPGA board, whether after being loaded on the board, the desired outputs are generated.

The evaluation reviews the objectives and requirements of the system. It reports on the entire implementation and whether it works as an FPGA-based system. The results of the evaluation module as obtained from the testing are presented.

3.2.1 Testing

The VHDL codes were tested and errors in it are detected and corrected. The modified code is then re-tested, (and re-modified if necessary) until it is error-free. The development of the project was carried out in a way to ensure that the development and testing occurred in parallel and were thus closely intertwined. Software testing involves both verification and validation. Verification involves checking that the program conforms to its specification while validation involves checking that the program as implemented meets the expectations of the user. Static checking techniques include program inspections and analysis. Dynamic techniques (tests) involve exercising the system. The testing process normally has five stages. Firstly, individual units are tested in unit testing. Module testing tests modules (usually a collection of dependent units). Sub-system testing tests collections of modules and often exposes sub-system interface mismatches. System testing tests the system as a whole. Module testing was carried out on both the pulse generator design as well as the signalTap II analysis. Subsystem testing mainly involved testing subsystems, which is basically the pulse generator and the logic analyzer. System testing was then carried out for the complete system. When carrying out software testing, there are several strategies that can be adopted. The two most basic strategies are top-down and bottom-up testing. Bottom-up testing was used to test each component and module. All the components in various orders and combinations were tested.

3.2.2 Evaluation

A decision was taken early on in the project to use the Quartus II project software, since the project required the use of a FPGA. Apart from it being an accepted format, it lends itself to the integration of design and implementation. Evaluation, being of significant mention in a development process has been incorporated into this project at various stages during development.

CHAPTER FOUR

CONCLUSION AND RECOMMENDATON

4.1 Conclusion

In this project we designed a digital logic analyser that can be used to analyse both discrete and continuous pulse signals from different systems. The project made use of two FPGA boards, Altera DE1 and DE2. The system that was analysed in this project was an annular pulse generator and it was used to generate four continuous pulse signals. These signals were then sent in as input to the logic analyser system, one of the input signals was used as a trigger and analysed. The results of the signal analysis showed that different digital circuit signals can be analysed by simply sending them as input to the signalTap II analyser customized with the logic analyser code.

4.2 Recommendation

For further study and design the following recommendation will be necessary in order to come up with a better design:

- A physical device should be used to generate the signals to be analysed instead of designing a pulse generator as input.
- A complete analyser could be built from scratch instead of customising the use of the inbuilt analyser on the FPGA board. This, however, would require more resources.

APPENDIX

VHDL Code for Annular pulse generator

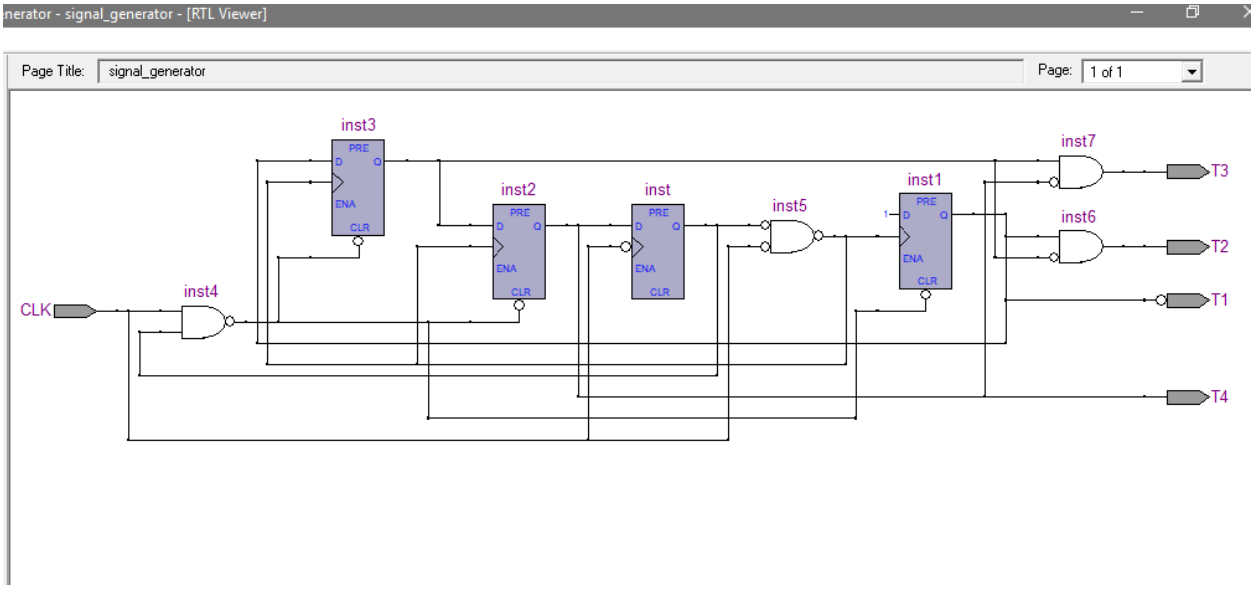
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY work;
ENTITY signal_generator IS
        PORT (CLK :  IN  STD_LOGIC;
                T1 :  OUT  STD_LOGIC;
                T2 :  OUT  STD_LOGIC;
                T3 :  OUT  STD_LOGIC;
                T4 :  OUT  STD_LOGIC);
END signal_generator;

ARCHITECTURE bdf_type OF signal_generator IS
SIGNAL      SYNTHESIZED_WIRE_13 :  STD_LOGIC;
SIGNAL      SYNTHESIZED_WIRE_14 :  STD_LOGIC;
SIGNAL      SYNTHESIZED_WIRE_15 :  STD_LOGIC;
SIGNAL      SYNTHESIZED_WIRE_16 :  STD_LOGIC;
SIGNAL      SYNTHESIZED_WIRE_3 :  STD_LOGIC;
SIGNAL      SYNTHESIZED_WIRE_17 :  STD_LOGIC;
SIGNAL      SYNTHESIZED_WIRE_18 :  STD_LOGIC;
SIGNAL      SYNTHESIZED_WIRE_19 :  STD_LOGIC;
SIGNAL      SYNTHESIZED_WIRE_20 :  STD_LOGIC;
SIGNAL      SYNTHESIZED_WIRE_9 :  STD_LOGIC;
SIGNAL      SYNTHESIZED_WIRE_11 :  STD_LOGIC;
BEGIN
SYNTHESIZED_WIRE_3 <= '1';
PROCESS(SYNTHESIZED_WIRE_13)
BEGIN
IF (RISING_EDGE(SYNTHESIZED_WIRE_13)) THEN
```

```
        SYNTHESIZED_WIRE_19 <= SYNTHESIZED_WIRE_14;
END IF;
END PROCESS;

PROCESS(SYNTHESIZED_WIRE_16,SYNTHESIZED_WIRE_15)
BEGIN
IF (SYNTHESIZED_WIRE_15 = '0') THEN
        SYNTHESIZED_WIRE_18 <= '0';
ELSIF (RISING_EDGE(SYNTHESIZED_WIRE_16)) THEN
        SYNTHESIZED_WIRE_18 <= SYNTHESIZED_WIRE_3;
END IF;
END PROCESS;

T4 <= NOT(SYNTHESIZED_WIRE_17);
T1 <= NOT(SYNTHESIZED_WIRE_18);
SYNTHESIZED_WIRE_17 <= NOT(SYNTHESIZED_WIRE_14);
SYNTHESIZED_WIRE_9 <= NOT(SYNTHESIZED_WIRE_19);
SYNTHESIZED_WIRE_13 <= NOT(CLK);

PROCESS(SYNTHESIZED_WIRE_16,SYNTHESIZED_WIRE_15)
BEGIN
IF (SYNTHESIZED_WIRE_15 = '0') THEN
        SYNTHESIZED_WIRE_14 <= '0';
ELSIF (RISING_EDGE(SYNTHESIZED_WIRE_16)) THEN
        SYNTHESIZED_WIRE_14 <= SYNTHESIZED_WIRE_20;
END IF;
END PROCESS;

PROCESS(SYNTHESIZED_WIRE_16,SYNTHESIZED_WIRE_15)
BEGIN
IF (SYNTHESIZED_WIRE_15 = '0') THEN
```

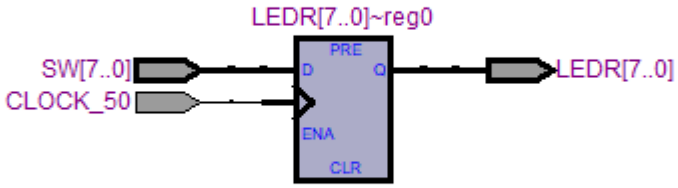
```
        SYNTHESIZED_WIRE_20 <= '0';
ELSIF (RISING_EDGE(SYNTHESIZED_WIRE_16)) THEN
        SYNTHESIZED_WIRE_20 <= SYNTHESIZED_WIRE_18;
END IF;
END PROCESS;

SYNTHESIZED_WIRE_15 <= NOT(CLK AND SYNTHESIZED_WIRE_19);
SYNTHESIZED_WIRE_16 <= NOT(SYNTHESIZED_WIRE_9 AND SYNTHESIZED_WIRE_13);
T2 <= SYNTHESIZED_WIRE_18 AND SYNTHESIZED_WIRE_11;
T3 <= SYNTHESIZED_WIRE_20 AND SYNTHESIZED_WIRE_17;
SYNTHESIZED_WIRE_11 <= NOT(SYNTHESIZED_WIRE_20);
END bdf_type;
```

VHDL code for Logic Analyzer

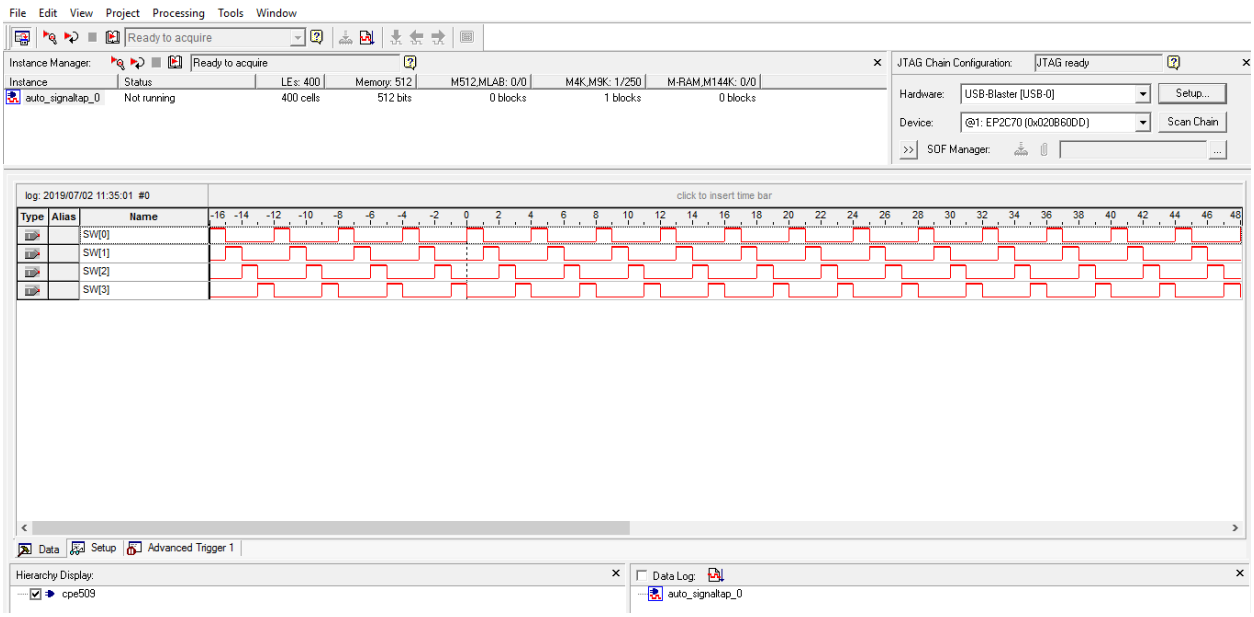
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY switching_circuit IS
        PORT (CLOCK_50 : IN STD_LOGIC;
                SW : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                LEDR: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END switching_circuit;

ARCHITECTURE Behaviour OF switching_circuit IS
BEGIN
        PROCESS (CLOCK_50)
                BEGIN
                IF(RISING_EDGE(CLOCK_50)) THEN
                        LEDR <= SW;
                END IF;
END PROCESS;
END Behaviour;
```



RTL View of Annular Pulse Generator



RTL View of Logic Switching Circuit



Signal Analysis Results