PERFORMANCE COMPARISON OF CPLD AND PLD BASED TRAFFIC LIGHT CONTROL SYSTEM

A Thesis

Submitted to the Department of Computer Science and Engineering

of

BRAC University

by

ISHTAQUE ASAD

Student ID: 06310016

MD. ASADUZZAMAN

Student ID: 06310025

TANVIR SOBHAN

Student ID: 06310015

KHANDKER MOHD. SABBIR MORSHED

Student ID: 06110025

Supervised by

Dr. A.K.M. Abdul Malek Azad

In Partial Fulfillment of the

Requirements for the Degree

of

Bachelor of Science in Electronics and Communication Engineering ,Sep 2009

DECLARATION

We hereby declare that this thesis is based on the results we found by our work. Contantes of work found by other researcher are mentioned by reference. This thesis , has never been previously submitted for any degree neither in whole nor in part.

Signature of
Supervisor

Signature of
Author

ACKNOWLEDGMENTS

Special thanks to all the people who believed our ability, people who taught us how to use FPGA, to Chowdhury Mohd. Iftekharul Islam, Md. Asaduzzaman All Faruq, and to Md. Ahmed Al Amin , seniors who have tought us how to think freely and using many devices not femilier to us , Suhailey Farzana for aiding to design our Traffic light model, and to Dr. A.K.M. Abdul Malek Azad for accepting the difficult task of overseeing this work to completion and giving us the chance to learn under him.

ABSTRACT

PLD and CPLD has been extensively used for custom made circuits. That is why they are perfect for designing traffic light control systems. Our thesis represents the performance comparison of a traffic light control system designed on GAL (Generic Array Logic) using Programmable Logic Device (PLD) and on FPGA (Field Programmable Gate Array) using Complex Programmable Logic Device (CPLD). For our PLD implementation, we have considered GAL (16V8) chips, which can be reprogrammed and erased. For the CPLD implementation, we have considered FPGA (Altera's Flex 10k family's EPF10K10TC144-4) chip, which is a 144 pin SRAM. The CPLD design was developed using the CPLD programming software MAX PLUS2 v 9.23. The traffic light controller consists of traffic signals (Red, Yellow/Amber & Green). We have designed the traffic controller using both CPLD and PLD. Then we have taken the real time waveform as well as the simulated waveform for different frequencies. The Digital Storage Oscilloscope (DSO).was used to generate the real time wave from the traffic controllers. The results from the real time waveform clearly illustrates that CPLD has the better performance over the PLD technology. Further more we have designed complex circuits for automated detection of railway crossing and A Five road junction ontrolling Traffic light system.

Keywords: GAL**,** PLD, FPGA, CPLD, Leap, Altera, traffic light control system

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

9

CHAPTER I

Introduction

1.1 An overview

PLD and CPLD have been used for a wide range of applications. After the introduction of the first PLD in the early 1970s by Phillips, the field of programmable logic has expanded exponentially.[4] Due to its ease of design and maintenance, implementation of custom made chips has shifted to PLD. Also there is a fact that cost of building a mask production of a custom VLSI, especially for small quantity is great.[2]

However, PLD such as PALs and GALs are available only in small sizes, equivalent to a hundred of logic gates. For Larger logic gates, Complex Programmable Logic Device (CPLD) was needed. Now, CPLD can replace thousands, or even hundreds of thousands, of logic gates [5]. But CPLDs doesn’t have much memory. That’s why devices which require lots of flip flops are not good candidates for CPLD [6]. For this reason machines such as a finite state machine class circuits such as traffic light controller are perfect for implementation [6].

A traffic light or traffic signal is a signaling device which is placed on a road intersection or pedestrian crossing to indicate when it is safe to drive, ride or walk, using a universal color code (usually red, yellow and green) [7].With the growing number of vehicles, the traffic congestion and transportation delay on urban arterials are increasing worldwide; hence it is imperative to improve the safety and efficiency of transportation [3]. That is why improve traffic light control system is essential.

1.2 Research Objective

The objective of our work is to implement a Traffic light Control system for a four road junction with both PLD and CPLD. And thereby Compare the performance of the two device. As the CPLD is a new technology to the country. These advanced technology implementations will lead the countries systems in the near future. So for a better future,its today we have to work with.

CHAPTER II

Brief of the Technologies

2.1 An Overview of the Technologies used

PLD and CPLD have been used for a wide range of applications. After the introduction of the first PLD in the early 1970s by Phillips, the field of programmable logic has expanded exponentially.[4] Due to its ease of design and maintenance, implementation of custom made chips has shifted to PLD. Also there is a fact that cost of building a mask production of a custom VLSI, especially for small quantity is great.[2]However, PLD such as PALs and GALs are available only in small sizes, equivalent to a hundred of logic gates. For Larger logic gates, Complex Programmable Logic Device (CPLD) was needed. Now, CPLD can replace thousands, or even hundreds of thousands, of logic gates [5]. But CPLDs don't have much memory. That's why devices which require lots of flip flops are not good candidates for CPLD [6]. For this reason machines belonging to finite state class machine circuits such as traffic light controller are perfect for implementation [6].

2.2 Programmeble Logic Device (PLD)

A programmable logic device or PLD is an electronic component used to build reconfigurable digital circuits. Unlike a logic gate, which has a fixed function, a PLD has an undefined function at the time of manufacture. Before the PLD can be used in a circuit it must be programmed. PLD is classified into PAL (Programmable Array Logic) and GAL (Generic Array Logic). MMI introduced a breakthrough device in 1978, the Programmable Array Logic or PAL. Which is a fixed circuit ambedded in a chip. An innovation of the PAL was the generic array logic device, or GAL, invented by Lattice Semiconductor in 1985. Both of the technologies consist of small chips. For our thesis we used GAL as it is reprogrammable and more afficient.

2.2.1 Generic Array Logic (GAL)

The Generic Array Logic (GAL) device was an innovation of the PAL and was invented by Lattice Semiconductor. The GAL was an improvement on the PAL because one device was able to take the place of many PAL devices or could even have functionality not covered by the original range. Its primary benefit, however, was that it was eraseable and re-programmable making prototyping and design changes easier for engineers. The GAL is very useful in the prototyping stage of a design, when any bugs in the logic can be corrected by reprogramming. GALs are programmed and reprogrammed using a PAL programmer, or by using the in-circuit programming technique on supporting chips.Lattice GALs combine CMOS and electrically erasable (E^2) floating gate technology for a high-speed, low-power logic device.For our thesis we have used GAL 16v8 chips.

Fig 2.2.1: GAL 16v8

The GAL16V8, at 3.5 ns maximum propagation delay time, combines a high performance CMOS process with Electrically Erasable (E2) floating gate technology to provide the highest speed performance available in the PLD market. High speed erase times (<100ms) allow the devices to be reprogrammed quickly and efficiently.

2.3 Complex Programmable Logic Device (CPLD)

A complex programmable logic device (CPLD) is a programmable logic device with complexity between that of PALs and FPGAs, and architectural features of both. The building block of a CPLD is the macro cell, which contains logic implementing disjunctive normal form expressions and more specialized logic operations. Now a days CPLD has progressed in such a manner that is has become impossible to distinguish between CPLD and FPGA technology .There is very minor differences between these two technologies.

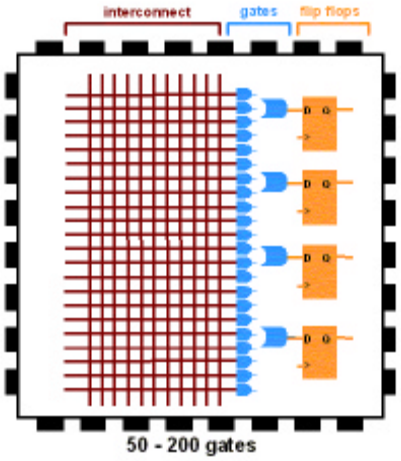


Figure 2.3 Complex Programmable Logic Device (CPLD)

Features in common wit FPGAs :

- Large number of gates available. CPLDs typically have the equivalent of thousands to tens of thousands of logic gates, allowing implementation of moderately complicated data processing devices. PALs typically have a few hundred gate equivalents at most, while FPGAs typically range from tens of thousands to several million.
- Some provisions for logic more flexible than sum of products expressions, including complicated feedback paths between macro cells, and specialized logic for implementing various commonly-used functions, such as integer arithmatic.

The most noticeable difference between a large CPLD and a small FPGA is the presence of on-chip non-volatile memory in the CPLD. This distinction is rapidly becoming less relevant, as several of the latest FPGA products also offer models with embedded configuration memory.

The characteristic of non-volatility makes the CPLD the device of choice in modern digital designs to perform ‘boot loader' functions before handing over control to other devices not having this capability. A good example is where a CPLD is used to load configuration data for an FPGA from non-volatile memory.

2.3.1 Benefits of using CPLD

CPLDs enable ease of design, lower development costs, and more product revenue for your money, and the opportunity to speed your products to market.

Ease of Design: CPLDs offer the simplest way to implement a design. Once a
design has been described, by schematic and/or HDL entry, you simply use CPLD development tools to optimize, fit, and simulate the design. The development tools create a file that is used to customize (that is, program) a standard off-the-shelf CPLD with the desired functionality. This pro- vides an instant hardware prototype and allows the debugging process to begin. If modifications are needed, you can enter design changes into the CPLD development tool, and re-implement and test the design immediately.

Lower Development Costs: CPLDs offer very low development costs. Because CPLDs are re-programmable, you can easily and very inexpensively change your designs. This allows you to optimize your designs and continue to add new features to enhance your products. CPLD development tools are relatively inexpensive (or in the case of Xilinx, free). Traditionally, designers have had to face large cost penalties such as rework, scrap, and development time. With CPLDs, you have flexible solutions, thus avoiding many traditional design pitfalls.

More Product Revenue : CPLDs offer very short development cycles, which means your products get to market quicker and begin generating revenue sooner. Because CPLDs are re-programmable, products can be easily modified using ISP over the Internet. This in turn allows you to easily introduce additional features and quickly generate new revenue. (This also results in an expanded time for revenue). Thousands of designers are already using CPLDs to get to market quicker and stay in the market longer by continuing to enhance their products even after they have been introduced into the field. CPLDs decrease TTM and extend TIM.

Reduced Board Area: CPLDs offer a high level of integration (that is, a large number of system gates per area) and are available in very small form factor packages. This provides the perfect solution for designers whose products which must fit into small enclosures or who have a limited amount of circuit board space to implement the logic design.

Cost of Ownership: Cost of Ownership can be defined as the amount it costs to maintain, fix, or warranty a product. For instance, if a design change requiring hardware rework must be made to a few prototypes, the cost might be relatively small. However, as the number of units that must be changed increases, the cost can become enormous.

2.3.2 Field Programmable Logic Array (FPGA)

A field-programmable gate array (FPGA) is a semiconductor device that can be configured by the customer or designer after manufacturing—hence the name "field-programmable". FPGAs are programmed using a logic circuit diagram or a source code in a hardware description language (HDL) to specify how the chip will work. They can be used to implement any logical function that an application-specific integrated circuit (ASIC) could perform, but the ability to update the functionality after shipping offers advantages for many applications.

There are two basic types of FPGAs: SRAM-based reprogrammable (Multi-time Programmed MTP) and (One Time Programmed) OTP. These two types of FPGAs differ in the implementation of the logic cell and the mechanism used to make connections in the device. The dominant type of FPGA is SRAM-based and can be reprogrammed as often as you choose. In fact, an SRAM FPGA is reprogrammed every time it's powered up, because the FPGA is really a fancy memory chip. That's why you need a serial PROM or system memory with every SRAM FPGA.

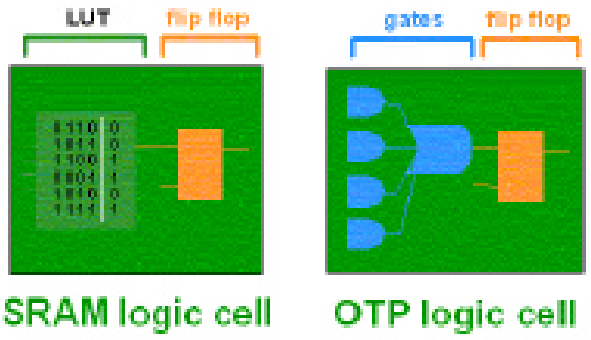


Figure 2.3.2 Types of FPGA

In the SRAM logic cell, instead of conventional gates, an LUT determines the output based on the values of the inputs. (In the "SRAM logic cell" diagram above, six different combinations of the four inputs determine the values of the output.) SRAM bits are also used to make connections. OTP FPGAs use anti-fuses (contrary to fuses, connections are made, not "blown," during programming) to make permanent connections in the chip. Thus, OTP FPGAs do not require SPROM or other means to download the program to the FPGA. However,

every time you make a design change, you must throw away the chip! The OTP logic cell is very similar to PLDs, with dedicated gates and flip-flops.

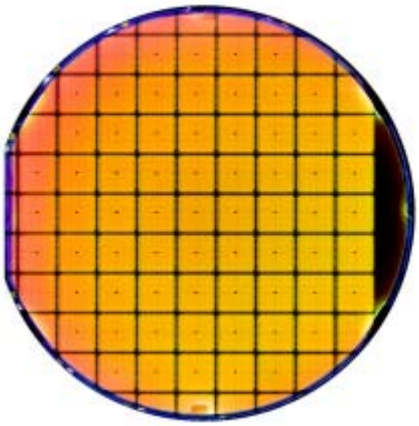


Figure 2.3.2. Silicon Wafer containing 10,000 gate FPGAs

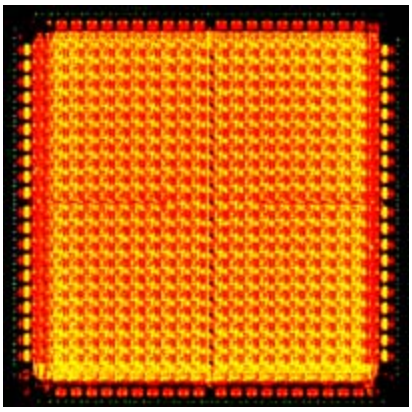


Figure 1.7 Single FPGA Die

2.3.3 The Internal Structure of FPGA

A typical FPGA is composed of three major components: logic modules, routing resources, and input/output (I/O modules) Figure 1.8 depicts the conceptual FPGA model. In an FPGA, an array of logic modules is surrounded or overlapped by general routing resources bounded by I/O modules. The logic modules contain combinational and sequential circuits that implement logic functions. The routing resources comprise pre-fabricated wire segments and programmable switches. The interconnections between the logic modules and the I/O modules are userprogrammable.

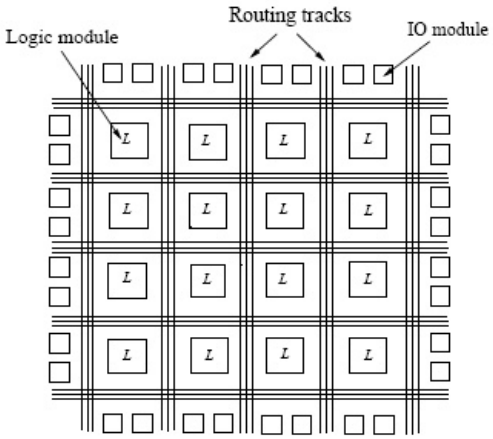


Figure 2.3.3: A typical FPGA architecture with three major components: logic modules, routing resources, and I/O modules.

2.3.4 Applications of FPGAs

FPGAs have gained rapid acceptance and growth over the past decade because they can be applied to a very wide range of applications. A list of typical applications includes: random logic, integrating multiple SPLDs, device controllers, communication encoding and filtering, small to medium sized systems with SRAM blocks, and many more. Other interesting applications of FPGAs are prototyping of designs later to be implemented in gate arrays, and also emulation of entire large hardware systems. The former of these applications might be possible using only a single large FPGA (which corresponds to a small Gate Array in terms of capacity), and the latter would entail many FPGAs connected by some sort of interconnect; for emulation of

hardware, QuickTurn [Wolff90] (and others) has developed products that comprise many FPGAs and the necessary software to partition and map circuits. Another promising area for FPGA application, which is only beginning to be developed, is the usage of FPGAs as custom computing machines. This involves using the programmable parts to “execute” software, rather than compiling the software for execution on a regular CPU. The reader is referred to the FPGA-Based Custom Computing W orkshop (FCCM) held for the last four years and published by the IEEE. When designs are mapped into CPLDs, pieces of the design often map naturally to the SPLD-like blocks. However, designs mapped into an FPGA are broken up into logic block-sized pieces and distributed through an area of the FPGA. Depending on

the FPGA’s interconnect structure, there may be various delays associated with the interconnections between these logic blocks. Thus, FPGA performance often depends more upon how CAD tools map circuits into the chip than is the case for CPLDs. We believe that over time programmable logic will become the dominant form of digital logic design and implementation. Their ease of access, principally through the low cost of the devices, makes them attractive to small firms and small parts of large companies. The fast manufacturing turn-around they provide is an essential element of success in the market. As architecture and CAD tools improve, the disadvantages of FPDs compared to Mask-Programmed Gate Arrays will lessen, and programmable devices will dominate.

CHAPTER III

Traffic Light controller implementation with CPLD/FPGA

3.1 Designing Method

Designing with CPLD/FPGA is relatively easier compared to previously used PAL , PEEL , and GAL. As the devices are really user friendly. Our Leap electronics LP – 2900 device a student kit was attached with the FPGA chip ,EPF10K10TC144-4 of FLEX 10K family , a 144 pin chip. We used Altera Maxplus II Baseline version 9.23 used for programming.

The very first step for the design was to generate the state equations. For that the states were elloborated in the state table and then Karnaugh Map was generated using the state table.Finally Output equations were implemented as circuits in the Altera Maxplus II Baseline version 9.23 programming software.

Altera Maxplus II Baseline version 9.23 is really user friendly ,and has all the functions necessary to deal with complex circuits. After implementing smaller logic circuits we mapped them to blocks of circuict which were then utilized in more complex higher level of the circuit leading to complision.

Once the circuit is made it is then placed in the chip using the Floorplan option we assigned the input and output pins. Then the circuit if mounted on the FPGA chip EPF10K10TC144-4 which is sealed with the FPGA kit. Then the chip generates all the controlling instruction and the output were connected to the Four road junction model we have created for our thesis.

3.2 State tables

The state tables shown in the figure below are generated from the state of the traffic light. The traffic light orientation for our thesis we have chosen to be a four road junction with horizontal and vertical roads crossing which other . the states for the horizontal roads are same as horizontal green to horizontal yellow to horizontal red. In the mean time the vertical roads,who have same state remain at red light . As the horizontal roads are shown red light, the vertical lights move to veritical green and then to vertical yellow. As the vaertical road light becomes red the horizontal roads are open.

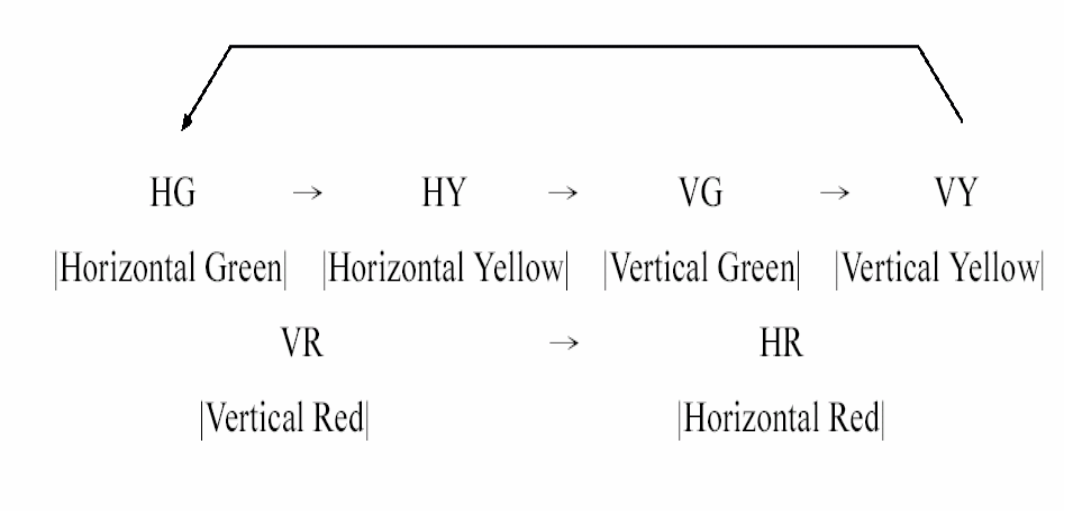


Fig 3.2 : State orientation of the Traffic light

Keeping the above state orientation in mind we generated the state table as,

Present State	Inputs x_1x_2				Outputs
	00	01	10	11	Abcd
00	00	01	00	00	1000
01	01	01	01	11	0100
10	00	10	10	10	0010
11	11	11	10	11	0001

Fig 3.2 : State Table of the Traffic Light Controller

3.3 Karnaugh Map generation from the state table

Next phase for the implementation is the generation of the Karnaugh Maps from the state table. These Karnaugh maps lead us to the state and output equations.First we observe the karnaugh maps for the state equations. These karnaugh maps start from the present state input.

Karnaugh Map of D_0, $D_0 = x_1'x_0 + Y_0x_1' + Y_0x_0$

D_0		Present State Input x_1x_0			
		00	01	11	10
Y_1Y_0 Present State Input	00	0	1	0	0
	01	1	1	1	0
	11	1	1	1	0
	10	0	1	0	0

Karnaugh Map of D_1, $D_1 = Y_1x_1' + Y_1x_0 + Y_1'Y_0x_1x_0'$

D_1		Present State Input x_1x_0			
		00	01	11	10
Y_1Y_0 Present State Input	00	0	0	0	0
	01	0	0	0	1
	11	1	1	1	0
	10	1	1	1	0

The first karnaugh map give us the equation for D_0, which is our horizontal road state. Y_1Y_0 , X_1X_0 are the Present state inputs.
Which gives us the state equations,

$D_0 = X_1'X_0 + Y_0X_1' + Y_0X_0$ And $D_1 = Y_1X_1' + Y_1X_0 + Y_1'Y_0X_1X_0'$

And the Output equations are found from karnaugh maps shown below as,

a= Y1'Y0' which is a=input1 ' * input0 '

b= Y1'Y0 which is b=input1 ' * input0

c= Y1 Y0' which is c=input1 * input0 '

d= Y1 Y0 which is d=input1 * input0

Karnaugh Map of a, $a = Y_1'Y_0'$

a		Present State Input x_1x_0			
		00	01	11	10
Y_1Y_0 Present State Input	00	1	1	1	1
	01	0	0	0	0
	11	0	0	0	0
	10	0	0	0	0

Karnaugh Map of b, $b = Y_1'Y_0$

b		Present State Input x_1x_0			
		00	01	11	10
Y_1Y_0 Present State Input	00	1	1	1	1
	01	0	0	0	0
	11	0	0	0	0
	10	0	0	0	0

Karnaugh Map of c, $c = Y_1Y_0'$

c		Present State Input x_1x_0			
		00	01	11	10
Y_1Y_0 Present State Input	00	1	1	1	1
	01	0	0	0	0
	11	0	0	0	0
	10	0	0	0	0

Karnaugh Map of d, $d = Y_1Y_0$

d		Present State Input x_1x_0			
		00	01	11	10
Y_1Y_0 Present State Input	00	1	1	1	1
	01	0	0	0	0
	11	0	0	0	0
	10	0	0	0	0

3.4 Circuit implementation in Max plus II

Max Plus II is the FPGA chip programming software we have used to program our FPGA chip .As discussed earlier from the sate equations and the Output equations we implemented the circuit.The overall circuit along with the internal blocks are shown in the figure below.

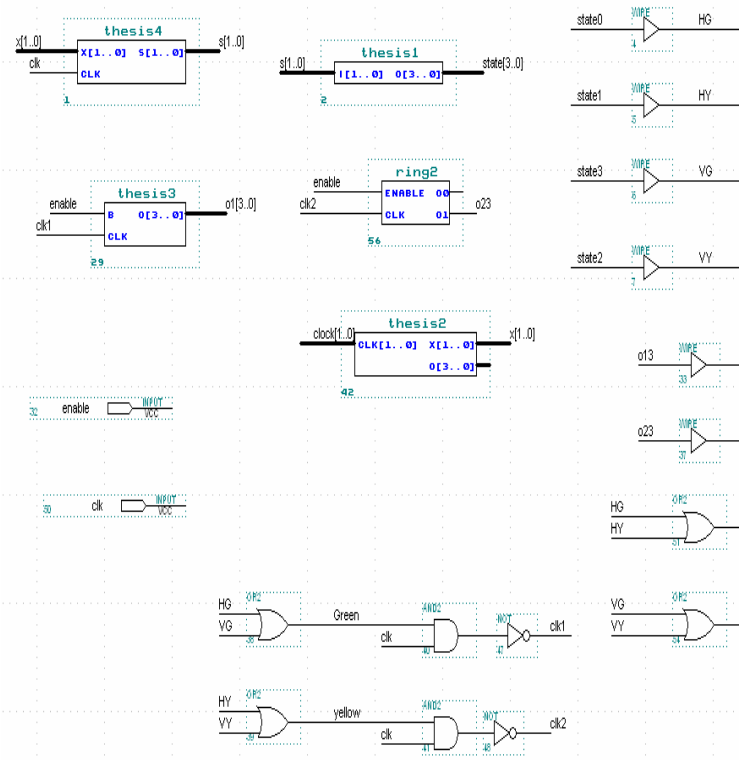


Fig 3.4 : Internal structure of the Traffic light Controller block

These internal blocks are implemented using the state equations and the output equations we generated for the karnaugh map. The total circuit is then mapped into a single block named Traffic light control system as shown below. The Signal A and Signal B are the sensor triggered signals as sensor A is attached with the Horizontal and Sensor B with the Vertical road.

The output from the Traffic Light Control system block are the states for the horizontal and vertical roads. RED A,YELLOW A,GREEN A for the Horizontal road and the RED B , YELLOW B and GREEN B for the Vertical roads.

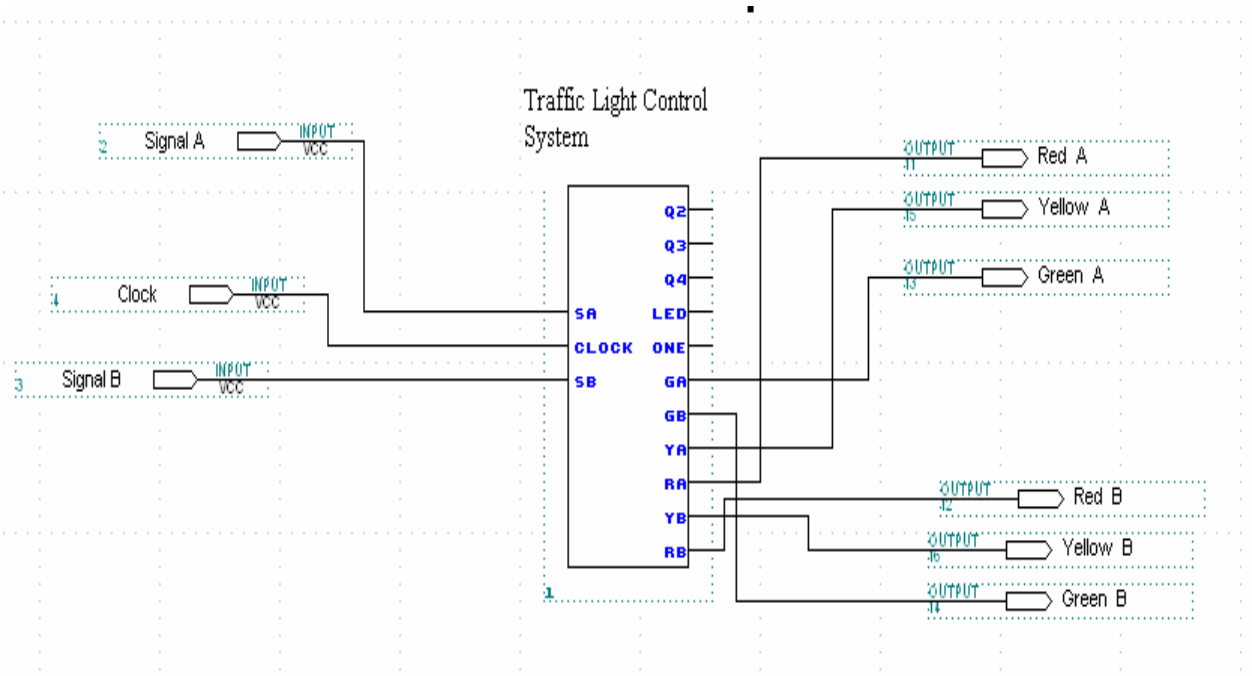


Fig 3.4 b : Traffic Light Controller CPLD based

3.5 CPLD based Output waveform with DSO

Output wave forms for the circuit have been taken Using the Digital Oscilloscope (DSO) for different time cycles / frequency between 1kHz to 2.5 MHz.But for our thesis the ration of light orientation from red to green to yellow is taken as 6 : 2 : 4 constantly.

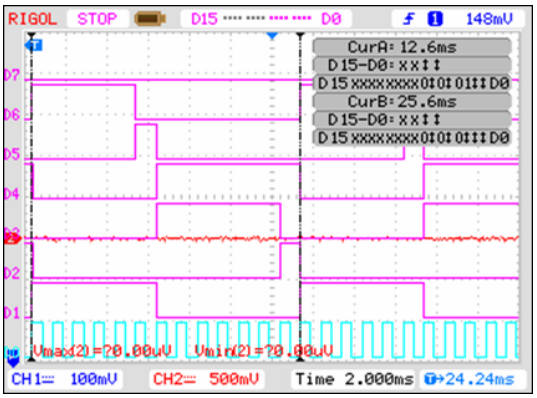


Figure 3.5 DSO output of the CPLD implementation at 1 khz

D7,D6,D5 three outputs represents horizontal Green , Yellow , Red. D4,D3,D2 outputs represents vertical Green , Yellow , Red sequentially. Bottom pulses in the bottom is the Clock pulse.

CHAPTER VI

Traffic Light controller implementation with PLD (GAL)

4.1Design Method

Programmable logic devices can be programmed and reprogrammed easily.For our thesis we used GAL 16v8 chips with MDA-PLD trainer board using software WINIDE Studio v 3.5.11.

GAL 16v8 chips are made using both CMOS and electrical erasable(E^2) technologies. As a result they have very high processing speed and low power consumption. Apart from that the erasing time is less than 100ms and the maximum delay is about 3.5nsec.

Just as the CPLD implementation or any other we had to create the state table keeping the state orientations in mind.And then to Karnaugh map from the state table.Which eventually gives us the state equations and the output equations.

For desidning purpose first we attach the Gal16v8 chip to the MDA-PLD trainer board and the software WINIDE Studio v 3.5.11 in the computer attached to the trainer board detects the chip.Once that is done there are two mathods of Gal programming equation based and code based.We used equation based system,in which the state and output equations are written in the software.And then the input symbols in the equation are assigned to the input pins and the output symbols to the output pins.Then the compiler on the software complies thye equation and the complied form of the equation are mounted on the chip.For the circuit we needed two GAL 16v8 chips.

4.2 State Table For the PLD based traffic light

The state Table for the PLd based Traffic light controller is same as the states are the same.But as the GAL 16v8 chip is unable to perform the bid equations single handed,we had to break down the state orientation mechanism to be performed into two chips and thereby the state table has changed,leading the state equations and the output equations to change.

	SaSb									
	00	01	10	11	*Ga*	*Ya*	*Ra*	*Gb*	*Yb*	*Rb*
S0	S1	S1	S1	S1	1	0	0	0	0	1 {Green A, Red B}
S1	S2	S2	S2	S2	1	0	0	0	0	1
S2	S3	S3	S3	S3	1	0	0	0	0	1
S3	S4	S4	S4	S4	1	0	0	0	0	1
S4	S5	S5	S5	S5	1	0	0	0	0	1
S5	S5	S6	S5	S6	1	0	0	0	0	1
S6	S7	S7	S7	S7	0	1	0	0	0	1 {*Ya*, *Rb*}
S7	S8	S8	S8	S8	0	0	1	1	0	0 {*Ra*, *Gb*}
S8	S9	S9	S9	S9	0	0	1	1	0	0
S9	S10	S10	S10	S10	0	0	1	1	0	0
S10	S11	S11	S11	S11	0	0	1	1	0	0
S11	S12	S11	S12	S12	0	0	1	1	0	0
S12	S0	S0	S0	S0	0	0	1	0	1	0 {*Ra*, *Yb*}

Fig 4.2 State table for PLD based Traffic-Light Controller

This state table eventually yields the state equations as,

D1=Q1Q2’+Q2Q3Q4

D2=Q1’Q2’Q3Q4+SAQ1Q3Q4+SB’Q1Q3Q4+Q1’Q2Q4’+Q1’Q2Q3’

D3=Q3Q4’+SBQ3’Q4+Q2’Q3’Q4+SA’SBQ1Q4

D4=SA’SBQ1Q3+Q2’Q4’+Q1’Q4’+SASB’Q2Q3Q4

And the output equations are ,

YA=Q2Q3Q4’ RA=Q1+Q2Q3Q4 GA=Q1’Q3’+Q1’Q2’

YB=Q1Q2 RB=Q1’Q2’+Q1’Q4’+Q1’Q3’ GB=Q1Q2’+Q2Q3Q4

Where the YA,GA,RA are the horizontal road output equations and the RB,YB,GB are the Vertical road output equations.

4.3 Circuit implementation in WINIDE Studio v 3.5.11

Applying the state and the output equations to the WINIDE Studio v 3.5.11 software and performing the methods discussed earlier yields the to GAL 16v8 chips capble of performing our desired Traffic light comtrol system With Sensor SSa for horizontal road and Sensor SSb for the vertical road.then the two GAL 16v8 chips named GAL C and GAL B are connected using wires in the trainer board as the diagram shows below.

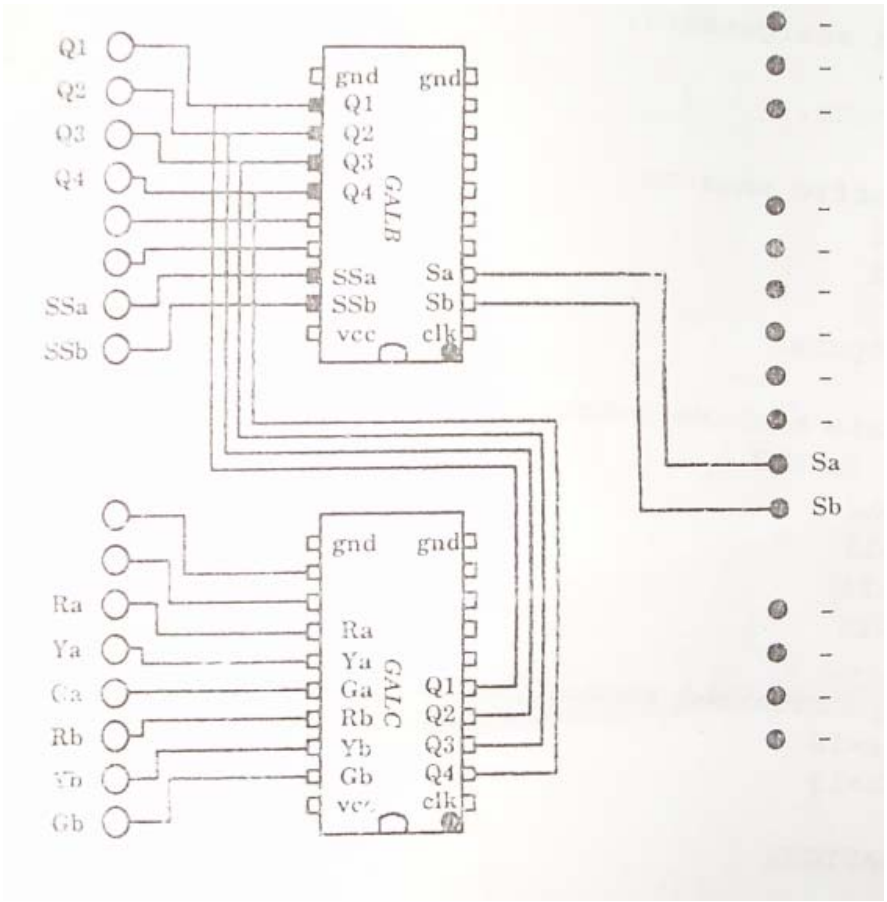


Fig 4.3 : Circuit diagram of PLD based Traffic Light Control System

The Sa and Sb combination shows the state of the orientation. Outputs Ra,Ya,Ga for the horizontal road and the outputs Rb,Yb,Gb for the vertical roads.

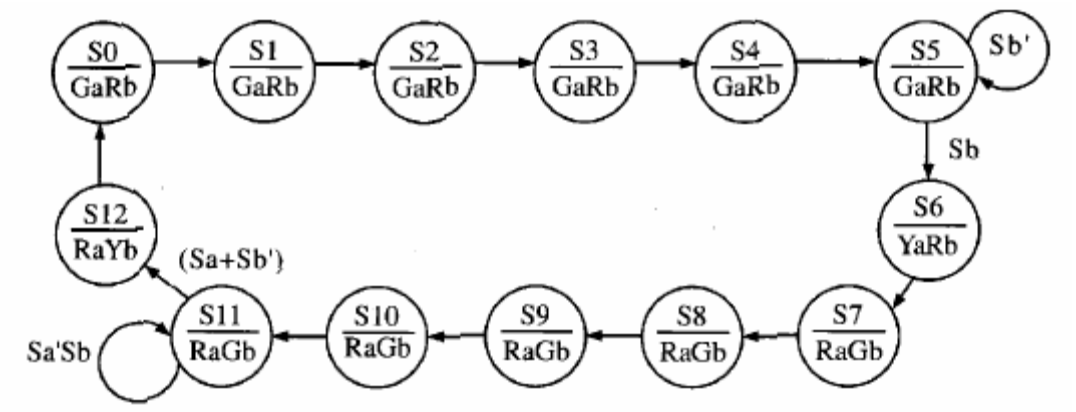


Figure 4.3(b): State Graph for Traffic Light Controller

The Figure shows the Moor state graph for the controller. For timing purposes the sequential network is driven by a clock with a 10-second period. So, a state change can occur at most once every 10 seconds. The notations are used like : GaRb in a state means that Ga=Rb=1 and all other output variables are zero. Sa'Sb on an arc means that Sa=0 and Sb=1 cause a transition along that arc. An arc without a label implies that a state transition will occur when the clock runs, which is independent of input variables. Thus the Green "A" light will stay on for 6 clock cycles (60 seconds) and then change to yellow if a car is waiting on "B" street.

4.4 PLD based Output waveform with DSO

PLD based output waveform shows more delay than the CPLD based output waveform. But the time cycle is performed according to the given ratio.

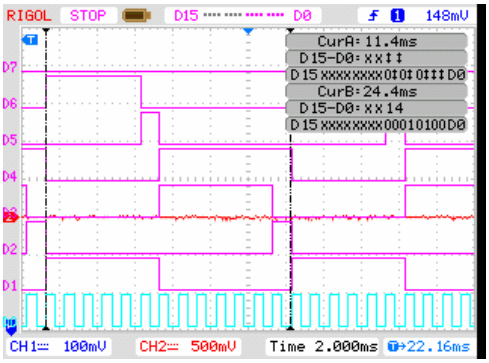


Figure 4.4 DSO output of the PLD implementation at 1 khz (Cycle time is 13 ms)

CHAPTER V

Performance Comparison Of CPLD and PLD based Circuits

5.1 An overview

The Real time ouputs of the implemented circuits have taken using the Digital storage oscilloscope (DSO) for both CPLD and PLD. Initially the cycle time for various frequencies were measured. We observed that the cycle time for both PLD and CPLD were approxamitely same various frequencies. However, when we measured the response time for various frequencies, CPLD was performing twice as better than PLD.

5.2 Comparison of DSO outputs with respect to variable frequencies

For frequencies varying from 1 Khz to 2.5 MHz we have taken multiple readings of the total orientation of the Traffic light Control system.That is for different time of cycles , Keeping the ratio of Green: yellow: Red as 6 : 2 : 4.

The DSO based outputs show the real time outputs. Which consist for some delay in response. The figures shown below are the Dso outputs for both CPLD and PLD based waveform. The following two pictures were taken directly from the DSO.

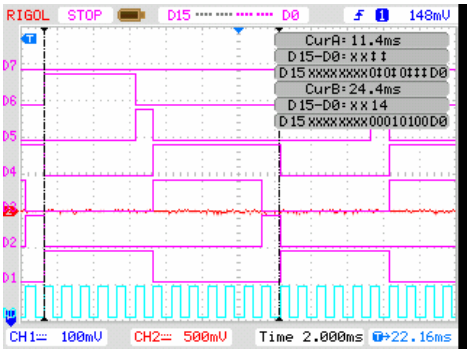


FIG 5.2a: DSO output of the PLD implementation at 1 khz (Cycle time is 13 ms)

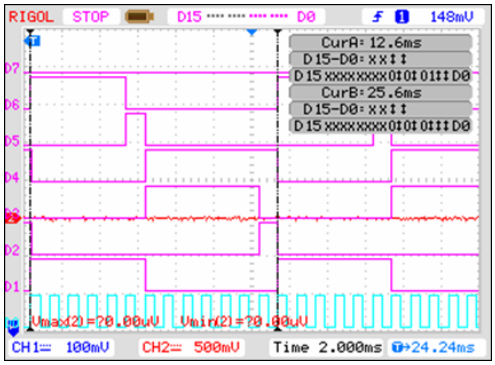
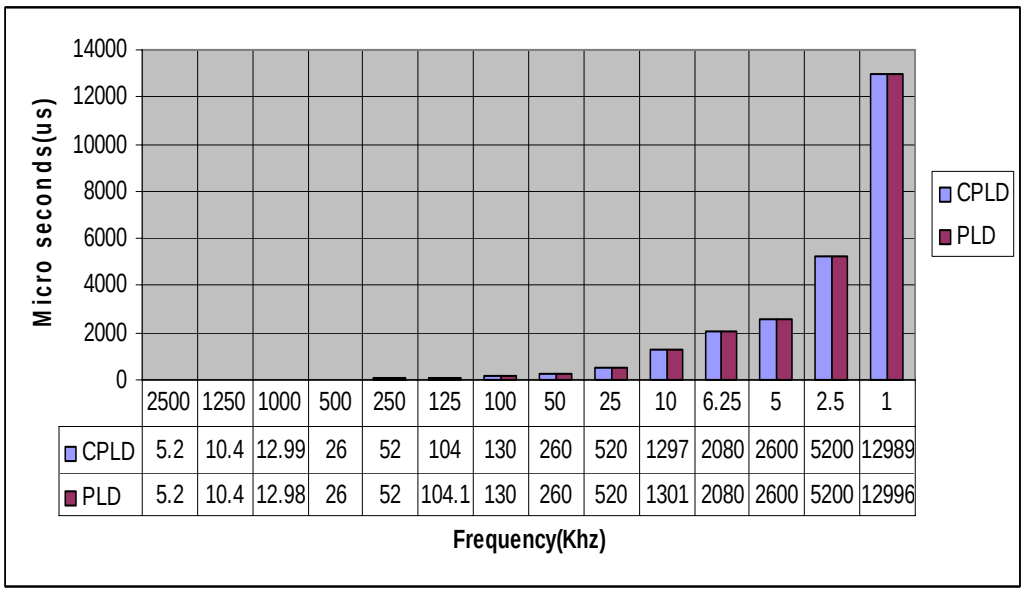


Fig 5.2b : Figure: DSO output of the CPLD implementation at 1 khz
(Cycle time is 13 ms)

Comparing the above waveforms we see for low frequencies the total time taken for the orientation of the light are same.But the PLD based circuit shows a delayed response.

Putting all the data found in the experiments conducted by us we have plotted them in to a graph.Which can guide us to the proper analogy of the changes.



	2500	1250	1000	500	250	125	100	50	25	10	6.25	5	2.5	1
CPLD	5.2	10.4	12.99	26	52	104	130	260	520	1297	2080	2600	5200	12989
PLD	5.2	10.4	12.98	26	52	104.1	130	260	520	1301	2080	2600	5200	12996

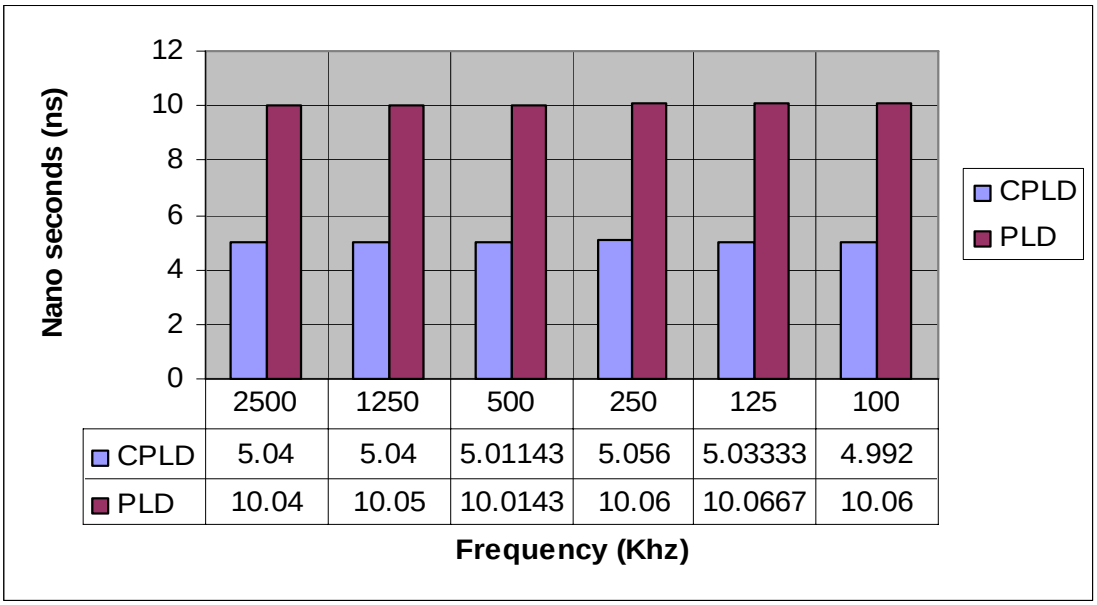
Figure 5.2 c: Cycle time vs frequency

The graph shows relatively similar patterns of response for both CPLD and PLD.Though as the Frequency increases the time cycle which is the resiprocal of the frequency decreases,there is no change in the response for the PLD and CPLD devices.The maintain the same time period given to them.

5.3 Delay response with respect to variable frequencies

For variable frequencies we have taken the output response graphs and from that the delay in the response have been measured. This is the delayed time taken to start the the operation.

The following graph states the proper output pattern found in our experiments,



	2500	1250	500	250	125	100
CPLD	5.04	5.04	5.01143	5.056	5.03333	4.992
PLD	10.04	10.05	10.0143	10.06	10.0667	10.06

Figure 5.3a : Delay response wrt clock for various frequencies

This Figure shows the delayed response found on the experiment using CPLD and PLD.The amount of delay shown by the PLD for variable frequencies remain in the same range.And for CPLD its another range but the margin is always similar. As for 500 kHz frequency the CPLD starts working after 5.01143 nanoseconds after the trigger and the PLD based device responses after 10.0143 nsec. But the difference is nearly the same through out the variable frequency range.

This figure shows the difference in response for variable frequencies,

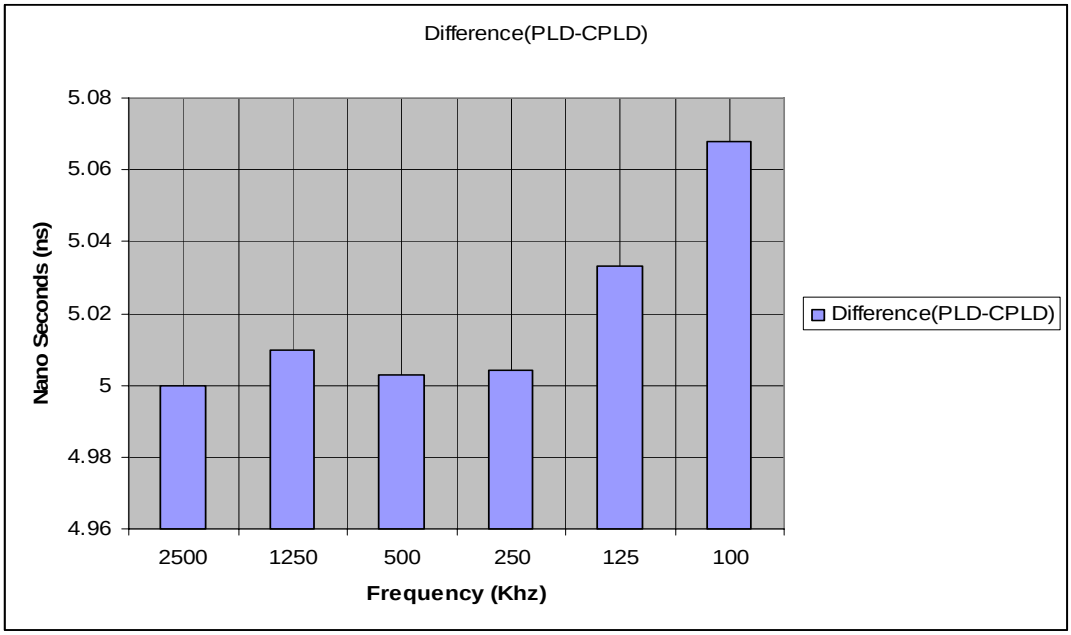


Figure 5.3 b : Difference of delay response (PLD – CPLD)

The graph was plotted from the difference of the delays in the PLD and CPLD based device outputs.The difference between the CPLD and PLd is always in the range from 5 – 5.1 nsec. From the graph we see that as the frequency decreses the margin of delay is increasing but in .01 nsec rate. So for a non time sensitive big logic device it is not that important to use CPLD, PLD can performe good enough. But for time sensitive devices like rocket launch or missile triggers, CPLD based circuits should be implemented.

CHAPTER VI

Complex Circuit Implementation

6.1 Complex circuits

Two complex circuit have been implemented for the pupose of justifying the ability of the FPGA. As the FPGA device is capable of operating huge logical circuits single handed , we decided to go beyond the objective of our thesis and establish complex circuits. These complex circuits can be really effective if implemented for the real world. The circuits we have made are firstly a five road junction where the fifth road is sensor triggered. And the second complex circuit is an automatic sensore triggered railway crossing system.

6.2 Five road junction

The first of the two complex circuits is the five road junction.As there are a lot of junctions with more roads criss crossing the rtoad map of our country.And no proper traffic light control system on them have been hounting the roads. We decided to establish a role model for such roads. Our five road junction in which the fifth additional road is chosen for having the least traffic density among all five. This fifth road has a sensor attached to the base of the traffic light controller.As described before the junction will be assumed to a four road junction and operate like the four road junction. As the traffiuc density in the fifth road is marginal so a sensore dedicated for the road will send signal to the main traffic light controller when a fair amount of car comes to the fifth road . And then only the fifth road will be shown the green light and all other four road traffic light will turn red.the cars on the fifth road will be given the previlage to pass to any road and after all the vehicle in the fifth road have passed , the roads will again perform the traffic light orientation of the four road junction.

The circuit diagram of the five road junction along with its simulated waveform given below,

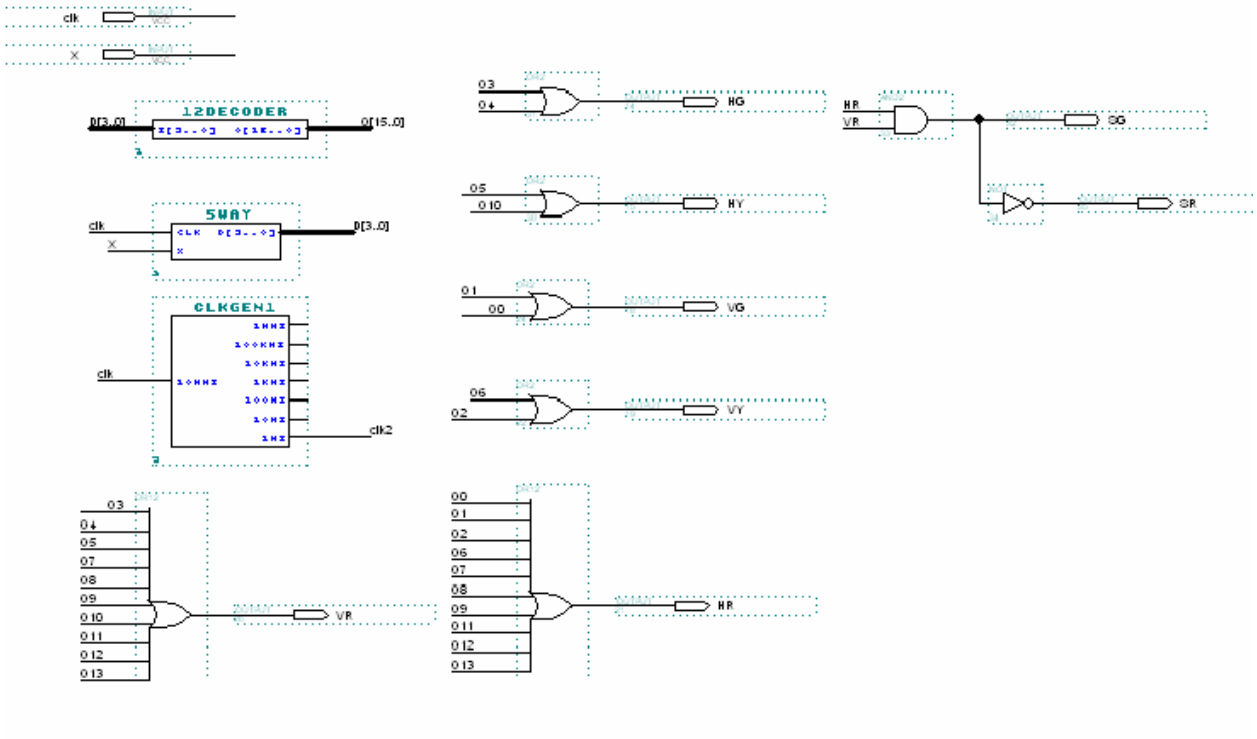


Figure 6.2 Five road junction Traffic Controller

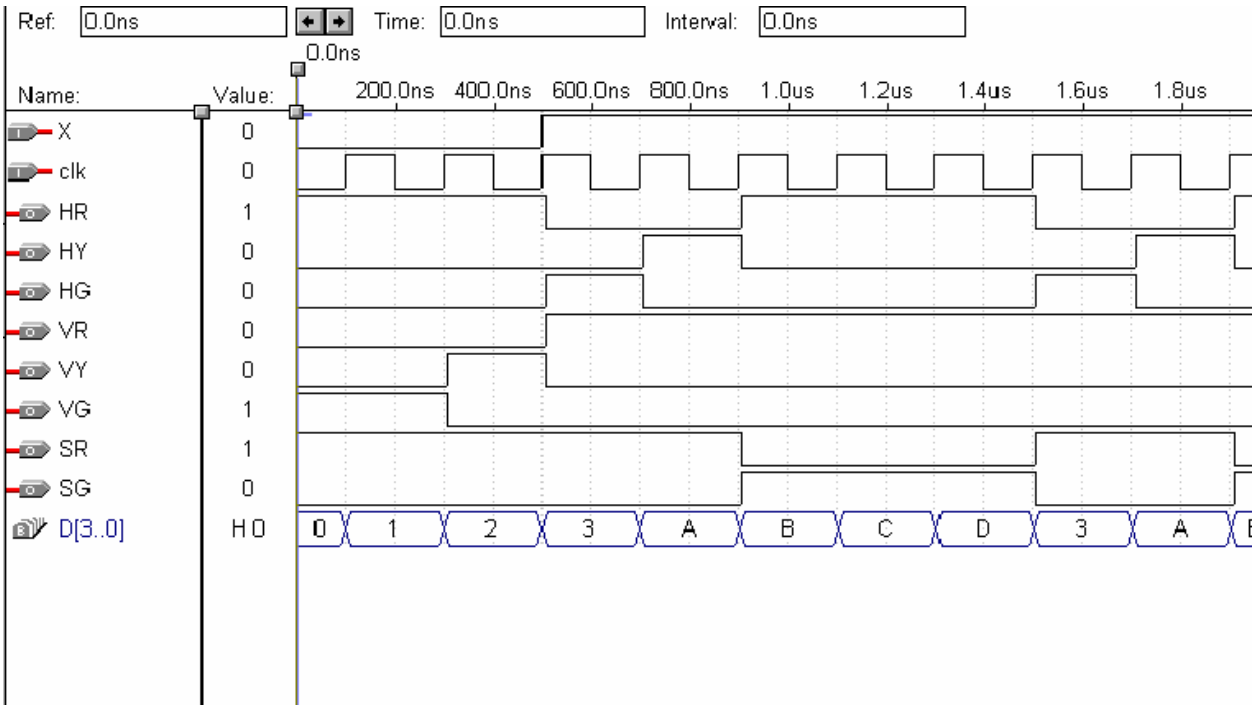


Figure 6.2b Simulated waveform of the five rioad junction using MAX PLUS II

6.3 Sensor triggered Automatic Railway Crossing

The sensore triggered railway crossing is one of the most useful system in todays world,as thousands are being killed all around the world due to railway croosing accidents. Keeping recent day accidents happening in the country regarding the railway crossing we decided to go for this complex circuit. The operation the automatic railway crossing is very important ,we have decided to go for a scenario like the Mohakhali railway crossing just with a four road junction. The main traffic controller controls the four road traffic light orientation. But when ever a train approaches, transmitted signals from the approaching train's transmitter will prompt the main traffic light controller to another state. In this state the roads parallel to the train shall remain open with green light shown to them.But the road over which the train shall cross remains closed. The term automatic is used as the triggering signal from the train shall not only prompt the alarm informing all about the approaching train but also pull down the bar on the roads giving enough time to the rest of the vehicle in the middle of the junction to cross. There is also a transmitter sensor near the cross bar, which shall send signals to the approaching train to slow down if for some reason the bar can not be placed correctly and roads are not clear of vehicles.

This circuit can be really effective in todays railway crossing infrastructure.Though we have not tested using all the sensors and the transmitter and the receivers. We have only taken a logic high for the sensore to be active and sending signals, and logic low for inactive mode.

But we are confident that this circuit shall impact the railway controlling system big time if implemented with right measures. The circuit diagram along with the simulation waveform have been shown in figure 6.3 and 6.3b.

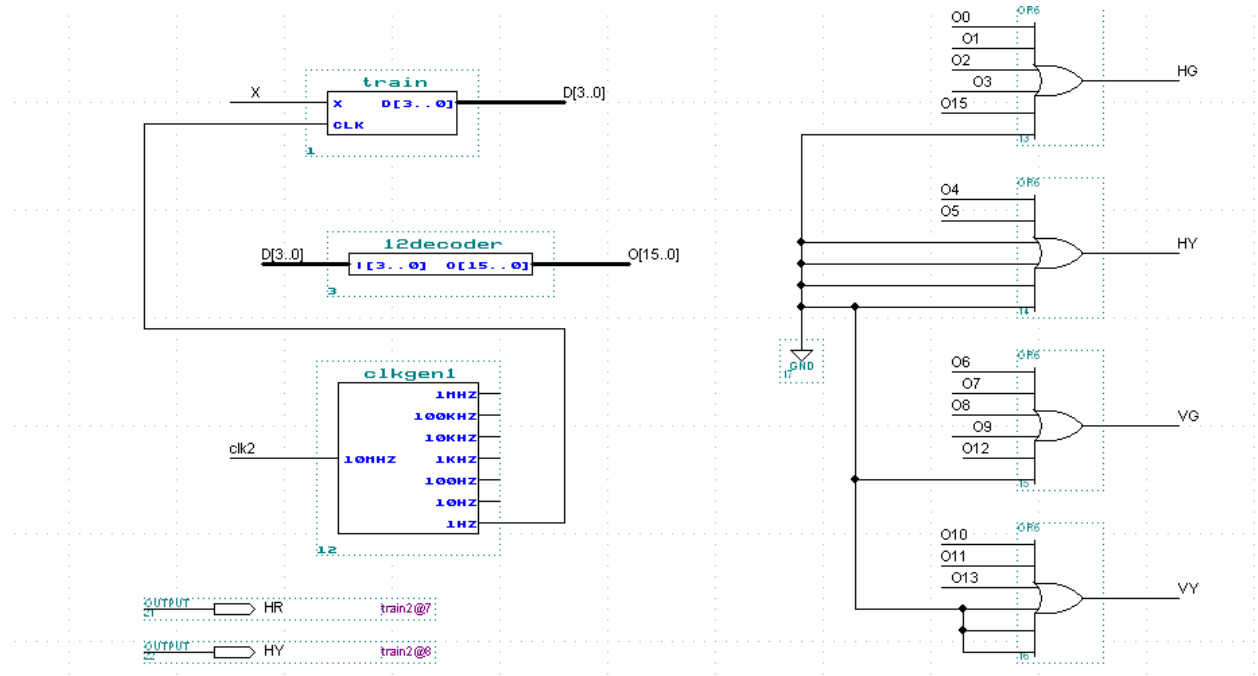


Figure 6.3 Sensor triggered Automatic Railway Crossing using CPLD

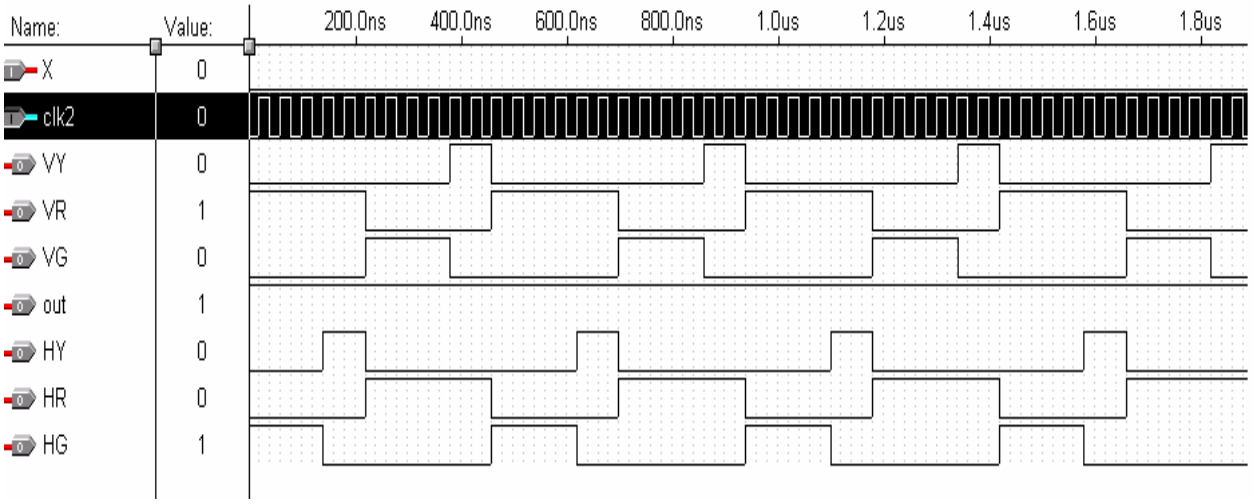


Figure : Simulated waveform of CPLD based Sensor triggered Automatic Railway Crossing circuit

CHAPTER VII

Conclusion

Based on our work on comparison of performance between CPLD and PLD technologies using traffic light control system. We have implemented the circuits using both CPLD and PLD technologies. For CPLD, we have used EPF10K10TC144-4 chips and For PLD, we have used two GAL 16V8. From our lab experiment, we observed that CPLD and PLD perform similarly at a micro second level. However, while observing the response with respect to clock, we found out that delay response of PLD is twice as much than the delay response of CPLD at a nano second level. So, we can conclude that for a traffic system which requires fast response, CPLD would be the best choice. Further More we have implemented more complex circuits and tested the capability of the CPLD (FPGA).
To sum up this new technology though expensive respect to the countries economic stands, can help modify the next generation of control systems and help modify the electronic domain of the country.

LIST OF REFERENCES

[1] K. Brunham and W. Kinsner, "RUN-TIME RECONFIGURATION: TOWARDS REDUCING THE DENSITY REQUIREMENTS OF FPGAS," *Canadian Conference on Electrical and Computer Engineering*, Publication Date: 2001, Volume: 2, on page(s): 1259-1264 vol.2

[2] WM El-Medany, MR Hussain, "FPGA Based Advanced Real Traffic Light Controller System Design." *4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 6-8 Sept. 2007 Page(s):100 – 105.

[3] Yi-Sheng Huang, Member, IEEE, Ta-Hsiang Chung, Ting-Hui Lin, "Design and Analysis Urban Traffic Lights Using Timed Colour Petri Nets," *Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC '06)*. 2006, Page(s):248 – 253.

[4] STEPHEN BROWN, JONATHAN ROSE, "FPGA and CPLD architectures: A Tutorial," *Design & Test of Computers, IEEE*, Volume 13, Issue 2, summer 1996 Page(s):42 – 57.

[5] CPLD Logic Design and Practices (*LP 2900 manual*)

[6] Wikipedia, "Programmable Logic device," *http://en.wikipedia.org/wiki/Programmable_logic_device*.

[7] Wikipedia, "Traffic light," *http://en.wikipedia.org/wiki/Traffic_light*.

[8] Charles H. Roth, Jr. "Digital Systems Design Using VHDL".