Journal of Advanced Computer Science & Technology, 3 (2) (2014) 120-128
©Science Publishing Corporation
www.sciencepubco.com/index.php/JACST
doi: 10.14419/jacst.v3i2.2818
Research Paper

Optimized design for controlling LED display matrix by an FPGA board

K. Mateur [1]*, R. Elgouri [1, 2], H. Dahou [1], L. Hlou [1]

[1] *Laboratory of Electrical Engineering & Energy Systems Faculty of Science Ibn Tofail University, Kenitra, Morocco*
[2] *National School of Applied Sciences (ENSA), Ibn Tofail University, Kenitra, Morocco*
**Corresponding author E-mail: khalidmateur@gmail.com*

Abstract

The very important objective of the digital designer when using discrete gates for implement a Boolean function is to keep the number of used gates to a minimum and win a memory space without lost the original information. In this sense, the Simplification is very important and could be achieved by a purely algebraic process, but it can be tedious when it arrived to a very large number of variables. In this paper we describe an automat solution based on finite state machine (FSM) for simplify and practically optimize the complex logical functions.
This method is programmed and tested on a display system which is based on light emitting diodes (LED) matrix and programmable platform with Field Programmable Gate Array (FPGA). The module is implemented in Spartan 3E family XC3S500E FPGA board.

Keywords: *Display Board, FPGA, FSM, LED, LED-Driver, Logic Simplification, Multiplex.*

1. Introduction

The PLD (Programmable Logic Devices) circuits are used increasingly as a base of the complex applications for science in life that surround us like security, healthcare, Telecommunications and in research field of vision and imaging. Every digital design which produced these days by using the Logic programmable circuit, like FPGA, consists mostly of high-density devices, it's not applied only to custom devices like processors, but also for the complex applications such as finite-state machine, in this sense the logical capacity is very important, it measured by number of gates in traditional gates arrays; in other words, the capacity of an PLD, FPGA for example, is measured by number of logic gates. But this number is limited and it can be obstacle to implementing complex applications that require a most logic gates, wherefore find logical simplification to minimize the number of logic gates used without losing the original information [1].
The logic simplification is often based on Boolean algebra, often logical functions are simplified by Karnaugh maps, but it is too difficult to simplified when these functions exceeds a number of variables[2]. It has a lot of applications that use a very large number of variables in its logical function, for example when we want drive a lot of LEDs for display an image or video.
The LEDs have been around us for over 30 years, this simple semiconductor junction emits continuous light when the current pass through the junction at low voltage, and it has proved useful for saving energy. It has a fairly wide angle vision, and it can be arranged in matrices to display text, numbers, and even images. Today LED screens are used in road, rail facilities signage, banks, stock exchanges, airports, advertising, etc. [3]. this device has the superlative source of the information and image display with wide viewing angle, clear and bright [4].
Like other semiconductor, the LED has a PN junction and it need to apply a directly DC voltage to emitting light photons, in principle, the change of the intensity of current pass through the junction results a variation of the brightness of LED, so each LED in the display must be controlled, a very complex controller is required for an LED display board which normally consists of a large number of LEDs [5].
The purpose of this paper is to describe the technique used for drive a large number of LEDs for display Image, by using a simplifies logical functions

The proposed system of a display image divided on four parts, as showing in figure 1:

- Transform the image into logical functions (Matlab image processing).
- Simplification (minimizes the number of logic gates and wins a memory space).
- Code VHDL (generate a code VHDL)
- Drive data by FSMD (Finite State Machine with Data path)

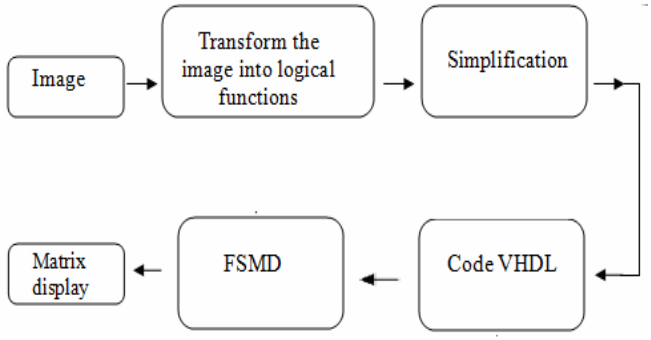

Fig. 1: Synoptic of Display System

The system starts by a simple image processing converting the image to bit map (0 and 1), changing its resolution for adapt to the display and transforms this binary image to the corresponding simplifies logical functions by using the notion of Finite Stat Machine and generate VHDL code (image circuit), The FSMD part used the VHDL code for synchronize and outputs data corresponds to each state (each line of the image) [6].

The other purpose of this paper is to present an implement of an FPGA-based for controlling a display LEDs screen with a refresh rate by using a multiplex technique.

In section 2 describes the structure and the principle technique used for display image in LED display. A description of the image processing, the simplification technique and the approach of the hardware implementation of FPGA is also given in section 3. Section 4 contains the implementation of the prototype system and the results of functional tests of the prototype in terms of image quality and overall performance details.

2. Structure and principle technique of the LED display

It is relatively simple to drive more LED individually. However, as the number of LED increases, the amount of resources needed to operate these LED is growing at an unsustainable level. As such, LEDs are often organized in matrices to make effective use of resources [8].

2.1. Structure of display element

The largest share of LED display structures are designed to minimize the complexity of printed circuit PCB and save space. The structure proposed in our paper, fig.2, facilitates their implementation and their management. In a matrix, the LEDs are arranged in rows and columns [6] [7].

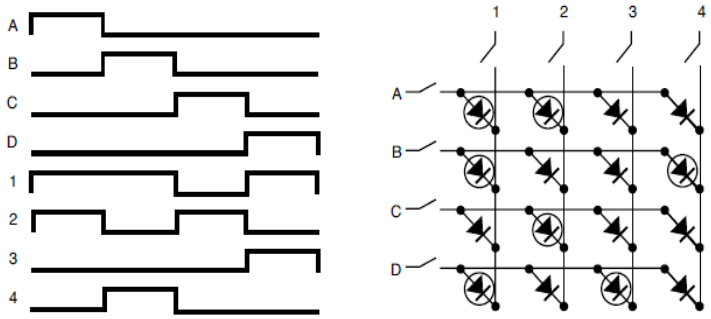

Fig. 2 : Structe and Pricipale Multiplexing of a LED Matrix

The principle is that each LED can be turned on by specifying its location in terms of rows (i) and columns (j), Pixel P(i,j) fig.2. The LED on the top left is addressed by P (A, 1), the line i=A and column j=1. This addressing method also indicates the passage of the electric current. The current flow of the A to 1 (forward bias) to turn on the pixel P(A,1), if all switches are closed to each port A to D and 1 to 4, then all LED will be light.

2.2. Structure of global screen

To facilitate data management in the case of matrix has very large number of pixels and keeping the quality of the image to display (D in figure 4), we Splits matrix into several panels (P) and modules (m) components as show in fig.3 and Fig.4; A typical configuration is a matrix 144x192 pixel divided in 3x4 panels of 48x48 pixels, which contains 2x2 modules of 24x24 LEDs.

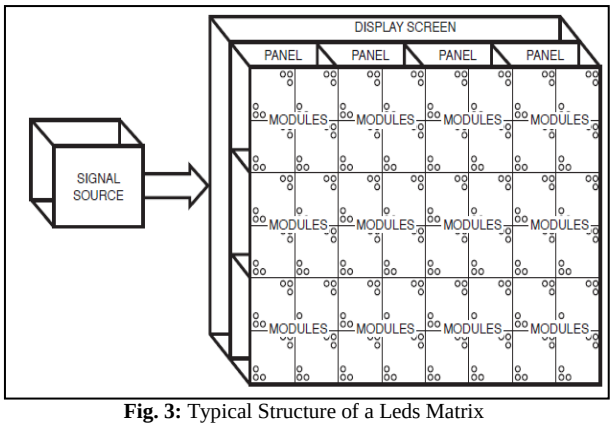

Fig. 3: Typical Structure of a Leds Matrix

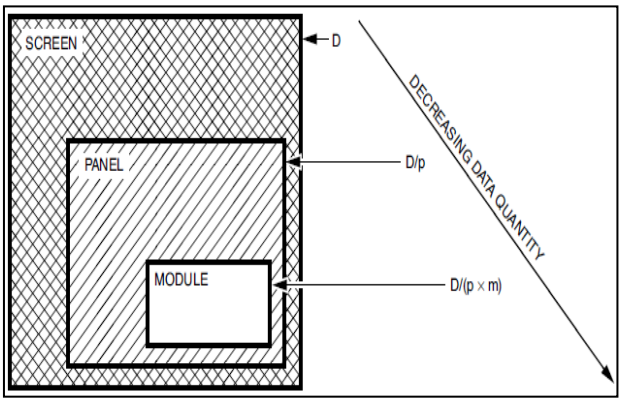

Fig. 4: Hierarchy of Display Components.

2.3. The proposed multiplexing controller

2.3.1. Principle of operation, concept of multiplexing:

Multiplexing is a technique used to localize each LED in matrix for display image. By this technique, a single row and column of the LED array activate at any time, because each anode of the LEDs in single line is related, and the same for each cathode in column, so in this system we used the configuration common anode rows and cathode columns to illustrate the concepts of multiplexing. Time division multiplexing is based on a sequence of steps as showing in figure 2, which enables a single line at a given time, the LEDs which are turned on are those of intersection between the line and columns activated [8].

2.3.2. Based structure for multiplexing a module, LED-driver.

The block who's used for multiplexing technique named LED-Driver, it has a two circuit synchronized by same Clock, fig.5, the block Select-line and the block Driver-Data, the first active line and the Driver_Data affect the data to each LED corresponds to this line.

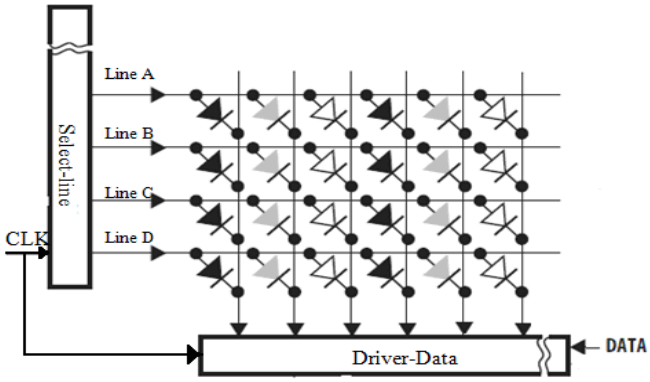

Fig. 5: Basic Construction of the LED-Driver

3. The proposed multiplexing Controller

The LED_Driver, fig.6, controlled the colour of dot matrix LEDs with similar configuration as shown in Fig.1, With respect to the technique for controlling the state of each LED within the display board described in Section 2.

3.1. Image processing by using a technique simplistic

The global system proposed in this paper, which is showing in fig.6, has three parts: the LED matrix (which already explained in section 2), Driver Data (Driver_data) and graphical user interface create by Matlab (GUI_Matlab),

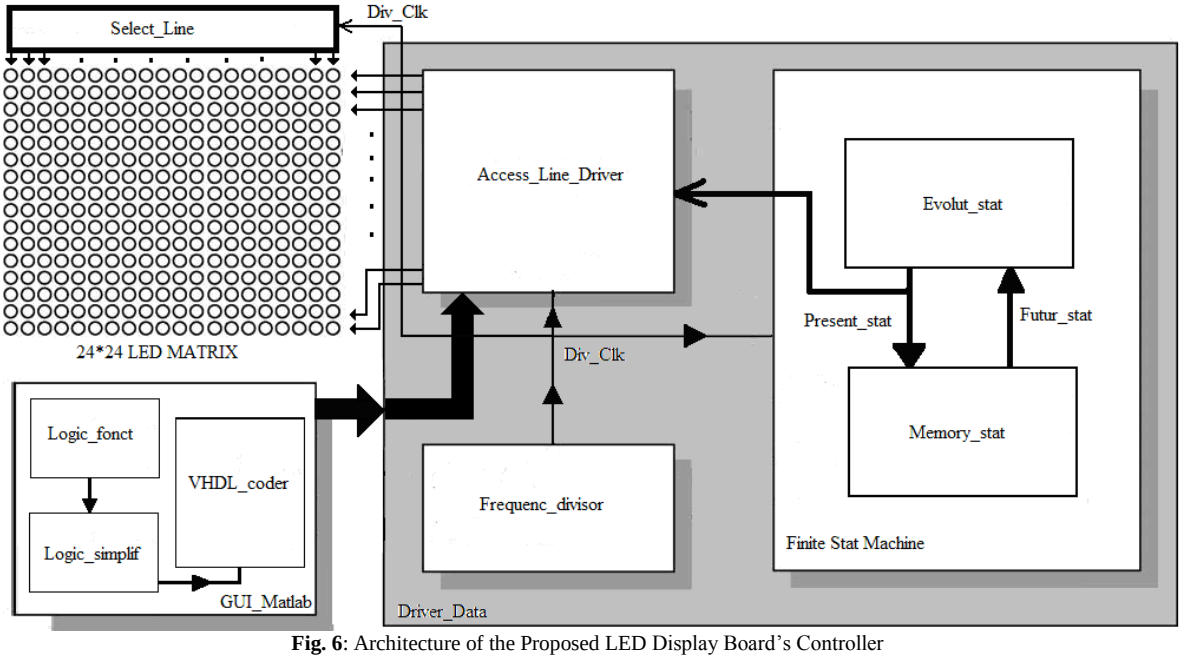

Fig. 6: Architecture of the Proposed LED Display Board's Controller

3.2. Image processing and graphical user interface

The graphical user interface, Gui_Matlab, programmed by Matlab, based on three blocks: the Logic_fonct, Logic_simplif and VHDL_coder. These blocks calculate, simplify and generate a VHDL code for the corresponding image; i-e generates command columns and matches the statements FSM.

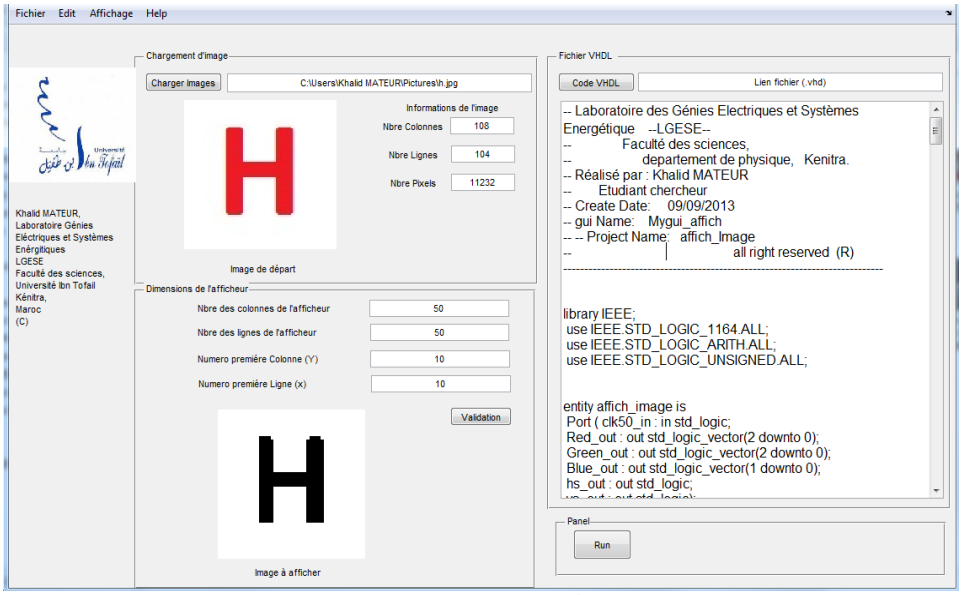

Fig. 7: Graphic User Interface

3.2.1. From image to logic fonctions (logic_fonc)

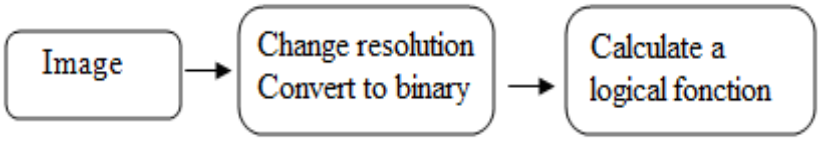

Fig.8: Diagram of Logic_Fonct

This block uses equations (1) and (2) to change the resolution of image and convert it to the bit map format, without lost the information and keep its quality,

$$f(i,j) = \sum_{j=1}^{C} \sum_{i=1}^{L} \left[\frac{(g(i,j,1)+g(i,j,2)+g(i,j,3))}{3} \right] \quad (1)$$

$g(i,j,k)$: start image

$f(i,j)$: bit map image

L, C : nembre of lines and columnes of image

$$f(i,j) = \sum_{j=1}^{Cmax} \sum_{i=1}^{Lmax} \left[g\left(\left(i * \frac{L}{Lmax} \right), \left(j * \frac{C}{Cmax} \right) \right) \right] \quad (2)$$

$Lmax, Cmax$: news dimensions of image

The result presented in this table:

Table 1: Matrix Corresponding to Image H

Logic_fonct		
Image to display	Changing resolution	Image matrix
		0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0

The logical functions are calculated by correspond to each line of the binary image, a state of the finite state machine calculates the output according to the bits of each state for example:

Table 2: The Corresponding Stat For Each Line of the Matrix

Stats	Bits Stats			Outputs
	e2	e1	e0	S0 S1 S2 S3 S4 S5 S6 S7
E0	0	0	0	
E1	0	0	1	
E2	0	1	0	
E3	0	1	1	
E4	1	0	0	
E5	1	0	1	
E6	1	1	0	
E7	1	1	1	

0 0 0 0 0 0 0 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 0

After corresponding a stat for each line of image, this component generates correspond logical functions for each outputs Si.

3.2.2. Method of the logical simplification (logic_simplif)

This block simplified the logical functions with using a Karnaugh map, the great advantage of this step is that the manual simplifications using the properties of the Boolean Algebra, which become unaffordable from a point on the number of parameters, a formalism of these properties has been developed and tested on the logical functions of parameters which are more than a hundred variable, such presented in this paper.

3.2.3. Generate a VHDL code (VHDL_coder)

This bloc generate a VHDL code of the simplify functions.

3.3. Drive data to display image (driver data)

It uses the VHDL code generated by Gui_Matlab by integrating it into a finite state machine and it has a purpose to generate outputs according to its states.
It has a three blocks: frequency divisor, finite state machine and driver data to line:

3.3.1. Frequency divisor

This block generates a refresh frequency Div_Clk, (Fig.9), it generates the rate at which the frames are refreshed. If the frequency is above a certain threshold frequency, the observer does not notice any flickering. For LED displays, a refresh rate of above 60 Hz is recommended. Persistence of vision is the human visual phenomenon that allows video images to be viewed without flicker, for find the exacted frequency with the equation (3).

$F_n = [1/(2n+1)].\ F_{Clk50MHZ}$ (3)

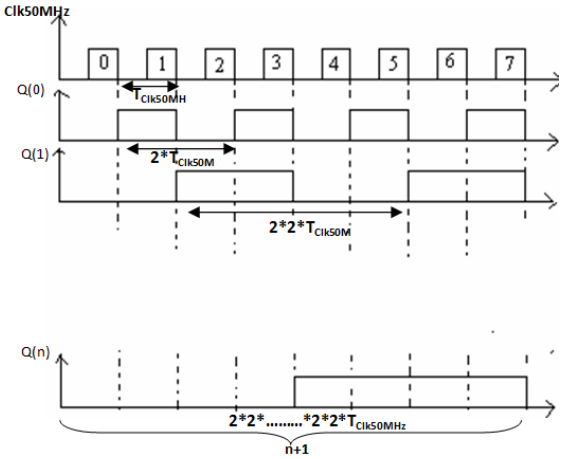

Fig. 9: Architecture Inrterne of the Frequenc_Divisor.

3.3.2. Finite stat machine

A finite state machine (FSM) also called state machine or finite state automaton is a particular type of Petri nets, a mathematical model, used to model a large number of problems. It is an abstract machine that can be in one of a finite number of states.
State Transition (Futur_stat): The machine changes to the next state when a triggering event occurs. This is called a transition. There are transition functions that map one state to the next state.
Current state (Present_stat): The machine is in only one state at a time. Current state is the state the machine is at any given time.
It Calculates and outputs the present state, that will be used in the block Acces_Line_Driver

3.3.3. Acces_line_driver

It corresponds to each output value according to the present state generated by FSM. This block is updated according to the image displayed in the GUI Matlab; Fig.8 shows the state diagram followed to calculate the output of this block.

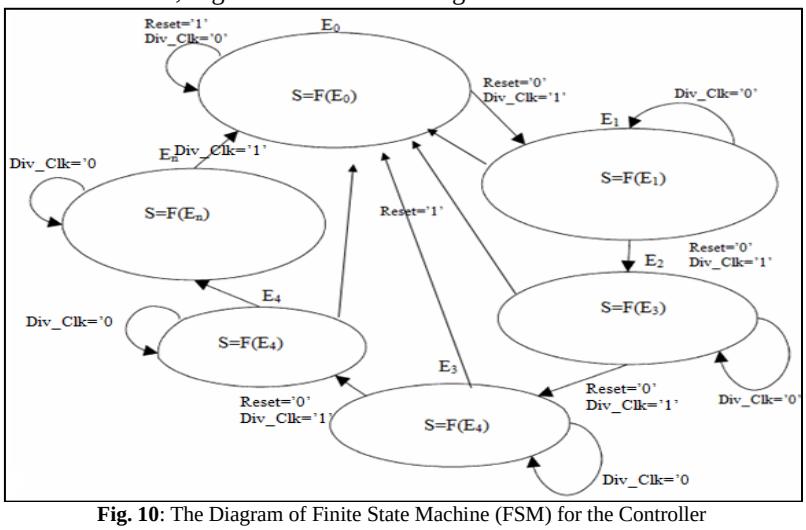

Fig. 10: The Diagram of Finite State Machine (FSM) for the Controller

4. Experimental results

This approach was implementing as part of our research to construct series of programs to simplify any logical function. We tested this application in imagery by using prototype LED display board, matrix LEDs of 10.6 cm, both in width and height, with 0.3 mm dot size and 7 mm between two LEDs were chosen for the optimized resolution. The arrangement of the LEDs in the prototype display board was 24*24 LEDs.

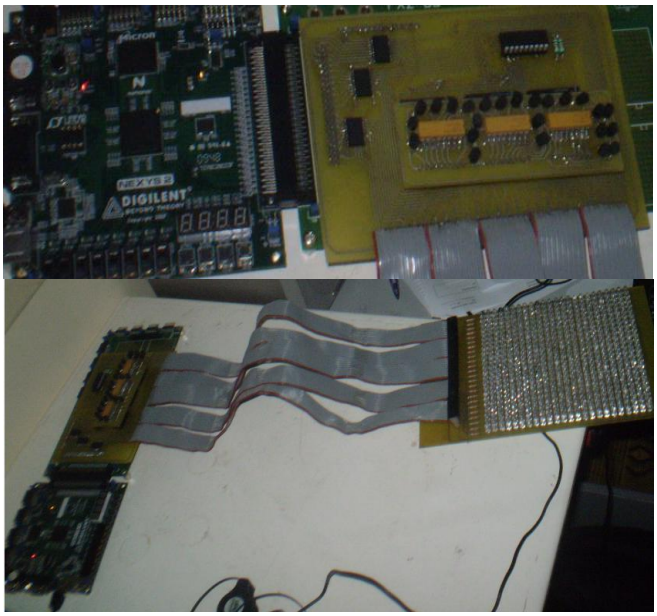

Fig. 11: Experimental Setup:the Global Disply System with FPGA Board Controller.

The LED display board was controlled by the Xilinx FPGA spartan3E XC3S500E FG320 which has a 500 103 gates [9]. As showing in figure 11, This FPGA able to generate 40 outputs port FX2. We used 24 outputs ports only to the block Driver_Data.
The block Select_line realized by card a based microcontroller, the result which we find is showing in table c and d.
The result obtained by the hardware simulation using display system as showing in figure 10 and 11, present the informations after and before the logical simplification, the number of gates before simplification is more than number of gates after simplification.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	69	9,312	1%	
Number of 4 input LUTs	81	9,312	1%	
Number of occupied Slices	55	4,656	1%	
Number of Slices containing only related logic	55	55	100%	
Number of Slices containing unrelated logic	0	55	0%	
Total Number of 4 input LUTs	107	9,312	1%	
Number used as logic	81			
Number used as a route-thru	26			
Number of bonded IOBs	5	232	2%	
IOB Latches	3			
Number of BUFGMUXs	2	24	8%	
Average Fanout of Non-Clock Nets	2.88			

Fig.12: Number of Used Gates Before Simplification

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	45	9,312	1%	
Number of 4 input LUTs	30	9,312	1%	
Number of occupied Slices	32	4,656	1%	
Number of Slices containing only related logic	32	32	100%	
Number of Slices containing unrelated logic	0	32	0%	
Total Number of 4 input LUTs	62	9,312	1%	
Number used as logic	30			
Number used as a route-thru	32			
Number of bonded IOBs	25	232	10%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	3.77			

Fig. 13: Number of Used Logic Gates after Simplification

Table 3: Example to Display Image in the Matrix

Image to display	Image displayed on the broad
	

Table 4: Example to Animation (Scroll) the Image in the Matrix

Image to display	Image displayed on the broad
	

5. Conclusion

- In this paper the processes of designing hardware, the prototype to display image in LED display board, were realize and examine. Proposed procedures for generate simplifying logical function was described in detail three

benefits for the proposed processes: Utilization less complex hardware without compromising in capability to control a group of LEDs simultaneously.

- Simplification the logical functions to win memory and logical gates in FPGA circuit Reducing number of buffer which used by the controller to 24 only for driving 576 LEDs

References

[1] B. Holdsworth and R.C. Woods, ''Karnaugh maps and function simplification, Digital Logic Design,'' (2003), PP 43-80.
[2] D. M. Harris, S. L. Harris, ''Combinational Logic Design, Digital Design and Computer Architecture,'' (2013), PP 54-106.
[3] R. Haitz, ''Trends in LED display technology, machine design,'' *Proceedings of 24th Electron Components Conference*, (1974), pp. 2–10.
[4] E. Fred Schubert, ''Light-Emitting Diodes,'' *Second Edition Cambridge University Press*, (2006).
[5] K. Yu, X. Zou, S. Long, W. Wang, ''High performance white LED driver with single-wire serial-pulse digital dimming,'' *The Journal of China Universities of Posts and Telecommunications*, Vol. 16, (February 2009), PP 95-99,110.
[6] L. Svilainis ''LED brightness control for video display application,'' *Displays*, Vol 29, (December 2008), PP 506-511.
[7] V.S. Abramov, A.E. Puisha, I.P. Polyakova, M.G. Tomilin, A.I. Chuvashov, ''LED modules for large screens,'' *Journal of Optical Technology (Opticheskii Zhurnal)*, Vol 70, (2003), PP 492– 494.
[8] W. Kurdthongmee, ''Design and implementation of an FPGA-based multiple-colour LED display board,'' *Microprocessors and Microsystems*, Vol 29, (September 2005), PP 327-336,.
[9] Xilinx Inc., Xilinx's XC3S500E FPGA Datasheet, Online Document from WWW.XILINX.COM.
[10] P. Schiefer, R. McWilliam, A. Purvis, ''Creating a Self-configuring Finite State Machine out of Memory Look-up Tables,'' *Procedia CIRP*, Vol 11, (2013), PP 363-366.