MEKATRONIKA
e-ISSN: 2637-0883
VOL. 3, ISSUE 1, 52 – 60
DOI: https://doi.org/10.15282/mekatronika.v3i1.7152

ORIGINAL ARTICLE

Implementation Study Of Field Programmable Gate Array (FPGA) And Complex Programmable Logic Device (CPLD) In Collision Avoidance System Using Vhsic Hardware Description Language (VHDL)

Mohammad Naqiuddin Fahmi Fathli[1] and Zulkifli Md Yusof[1,*]

[1]Faculty of Manufacturing Engineering, Universiti Malaysia Pahang, 26600 Pahang, Malaysia.

ABSTRACT – A collision avoidance system, also known as a pre-crash system, forward collision warning system, or collision mitigation system, is a sophisticated driver-assistance system that aims to avoid or mitigate the severity of a collision. For this research, collision avoidance system will be fabricating to show that this system can detect avoidance range before apply the braking action to prevent collision. The ultrasonic sensor will be used in this system to detect the avoidance range. In this collision avoidance system, there will be uses of Field Programmable Gate Array (FPGA) and Complex Programmable Logic Device (CPLD). This research will observe how to implement FPGA and CPLD in the collision avoidance system using VHSIC Hardware Description Language (VHDL). The VHDL will be done in Quartus II 15.0 Software. In this research, Terasic DE-10 Standard board has been used. It contains FPGA microcontroller model Cyclone V SoC 5CSXFC6D6F31C6N. Max II board is used because it contains CPLD microcontroller model EPM240T100C5.

ARTICLE HISTORY
Received: 12th April 2021
Revised: 2nd May 2021
Accepted: 19th June 2021

KEYWORDS
FPGA
CPLD
VHDL
Collision Avoidance System
Microcontroller

INTRODUCTION

There are two well-known digital logic chip type in this world, that is FPGAs and CPLD. In terms of internal architecture, these two chips are clearly distinct. FPGA stands for Field Programmable Gate Array, is a type of programmable chip. This chip is programmed to perform any digital operation [1]. The architecture of the FPGA enables the chip to have a large logic capability [2]. Because of its architecture, it is used in designs that involve a high gate count and have unpredictable delays. The FPGA is referred to as ‘fine-grain' because it comprises a large number of small logic blocks, which may number in the thousands. Flip-flops, combination logic, and memory are all included [3]. It's made for more advanced applications.

On the other hand, CPLD stand for Complex Programmable Logic Device. This chip is designed by using EEPROM (electrically erasable programmable read-only memory) [4]. It works best in designs with a low gate count. The delays are much more stable and it is non-volatile because it is a less complex architecture. For simple logic applications, CPLDs are often used [5]. It is made up of only a few logic blocks and can achieve a maximum of 100 [6]. CPLDs, on the other hand, are referred to as ‘coarse-grain' instruments. Because of its simpler, "coarse grain" design, CPLDs are inexpensive and have a much faster input to output time.

In this project, collision avoidance systems will be fabricating using ultrasonic sensor to detect the avoidance range before applied the braking action. In this systems, there will be two kind of chips that will be used. That is FPGA and CPLD. This research will study the implementation of these two logic chips. Since there are two types of Hardware Description Language (HDL), that is VHSIC Hardware Description Language (VHDL) and Verilog Hardware Description Language (Verilog HDL), VHDL will be used for this project.

AIMS & OBJECTIVES

The aims are to design the collision avoidance system that can detect avoidance range before apply the braking action using FPGA and CPLD. The objective of this research are:

I. To develop a collision avoidance system using FPGA and CPLD.
II. To compare FPGA and CPLD.

*CORRESPONDING AUTHOR |Zulkifli Md Yusof | ✉ zmdyusof@ump.edu.my

RELATED WORK

Collision avoidance systems are implemented in a wide range of applications and in a variety of conditions. Radar, lidar, and vision sensors are common sensors used to identify obstructions. Based on research by J. Jansson, he stated state that sensors in the collision avoidance system monitor the surroundings directly in front of the host vehicle. Based on data from forward-looking sensors, the decision to deploy countermeasures to avoid impending frontal collisions can be taken. Warning signs (audible, visual, or haptic/tactile), braking, and steering are common interventions [7]. An example of braking system is shown in Figure 1.

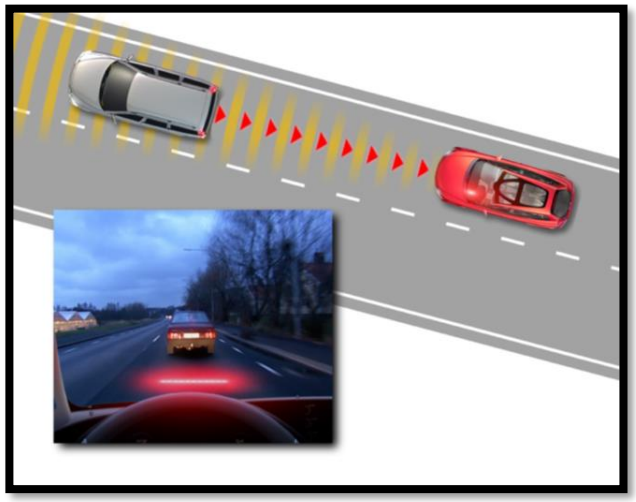

Figure 1: Example of a Collision Avoidance System

This research is similar to Design of Fabrication of Automatic Braking System by Ashish Deshmukh, Suraj Singh, Shreesh Shukla and Vivek Bachhav. Their research is about technology that allows vehicle to detect an impending forward collision with another vehicle or an obstacle and automatically apply the brakes. They used ultrasonic sensor and an 8051 microcontroller and the system flow is illustratted in Figure 2. The only different between this research and their research is, this research will use FPGA and CPLD [8].

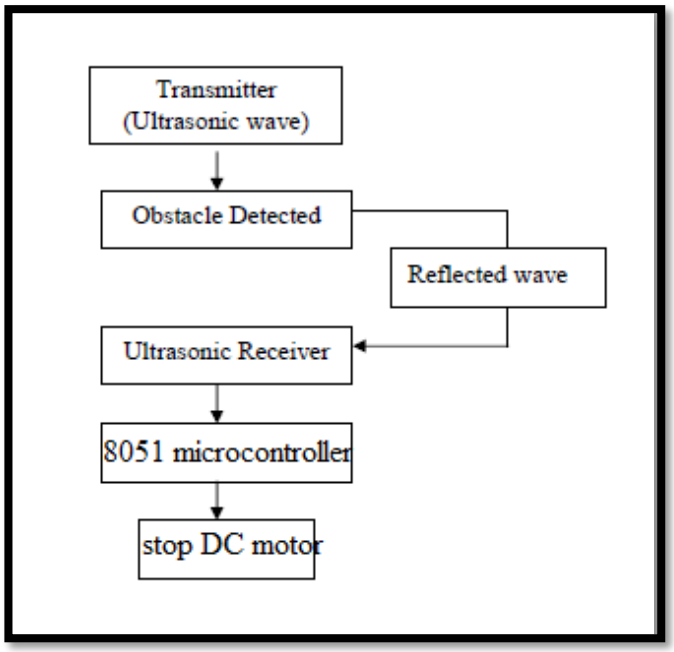

Figure 2: Flow algorithm of the Auto-Braking System

Based on their research, there are few advantages of their system, that is their system is more accurate and quick to respond. The use of mechatronics system makes safe and reliable. Their system can be used to detect obstacles as well as calculate distances, which is useful when parking and driving [8].

METHODOLOGY

Flow Chart of FPGA and CPLD Implementation

Figures 3 and 4 illustrates the Flow Chart of FPGA and CPLD Implementation and Block Diagram for Collision Avoidance System used for this research.

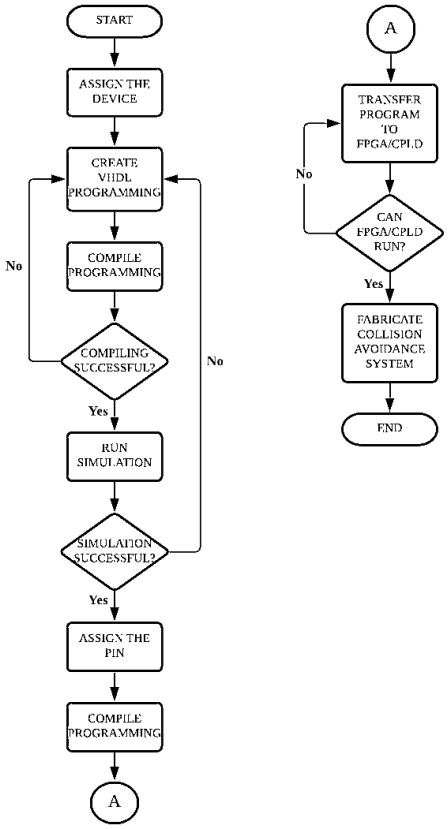

Figure 3: Flow Chart of FPGA and CPLD Implementation

Block Diagram for Collision Avoidance System

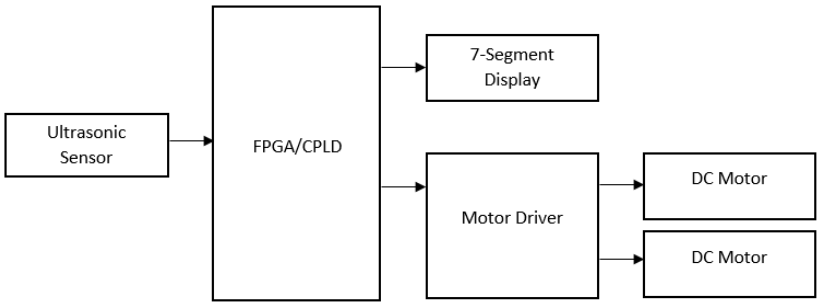

Figure 4: Block Diagram for Collision Avoidance System

Quartus II 15.0 Software

Quartus II 15.0 Edition software has been used to write VHDL programming. All the simulation has been done in this software. All the program will be uploaded into FPGA and CPLD using this sofware. Quartus II 15.0 Edition software need to be used in this research because this software can run the FPGA model Cyclone V SoC 5CSXFC6D6F31C6N and CPLD model EPM240T100C5 which will be use in this research.

VHDL Programming

```
entity DistanceSafety is
   Port ( pulse_pin : in STD_LOGIC;
          Trigger_pin : out STD_LOGIC;
          clk : in STD_LOGIC;
          topsegA : out STD_LOGIC;
          topsegB : out STD_LOGIC;
          topsegC : out STD_LOGIC;
          topsegD : out STD_LOGIC;
          topsegE : out STD_LOGIC;
          topsegF : out STD_LOGIC;
          topsegG : out STD_LOGIC;
          topsegA1 : out STD_LOGIC;
          topsegB1 : out STD_LOGIC;
          topsegC1 : out STD_LOGIC;
          topsegD1 : out STD_LOGIC;
          topsegE1 : out STD_LOGIC;
          topsegF1 : out STD_LOGIC;
          topsegG1 : out STD_LOGIC;
          sw: in  STD_LOGIC;
          pwm1, pwm2, pwm3, pwm4: out  STD_LOGIC
          );
```

Figure 5: Entity 'DistanceSafety'

Figure 5 above shows the entity of the program. It showed the input and the output of the system. The clock signal which is clk, is a signal that controls the speed of a flip flop (or a set of flip flops) in a digital circuit. The clock signal will activate all the flip flops and RAM blocks according to the clock frequency. For the pulse_pin and Trigger_pin, they were act as a port for ultrasonic sensor. Trigger_pin act as transmitter while pulse_pin act as receiver.

As for output port topsegA, topsegB, topsegC, topsegD, topsegE, topsegF, topsegG, topsegA1, topsegB1, topsegC1, topsegD1, topsegE1, topsegF1, and topsegG1 they were act as port for 7-segment display. In this research, two 7-segment displays are used to display the range of the collision avoidance system from the obstacle. The range that was display on 7-segment displays will be in cm.

pwm1, pwm2, pwm3 and pwm4 are acted as Pulse-Width Modulation (PWM). PWM are used to control the speed of DC motor. This system used two DC motor. Port pwm1 and pwm2 are connected to DC motor 1 while pwm3 and pwm4 are connected to DC motor 2. The speed of the motors is set according to certain distance.

The System Implementation for FPGA

Figures 6 and 7 shows the pin planner and program upload of the FPGA.

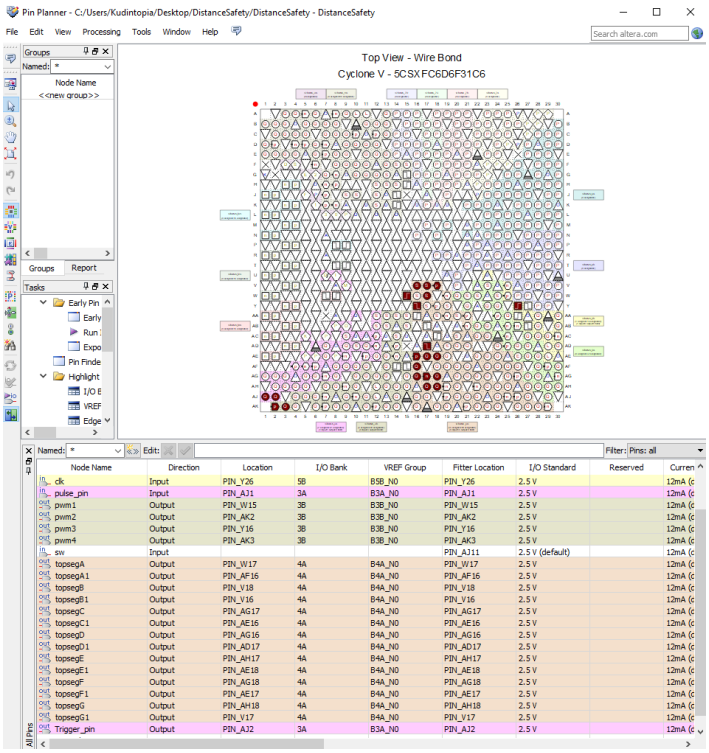

Figure 6: Pin Planner for FPGA

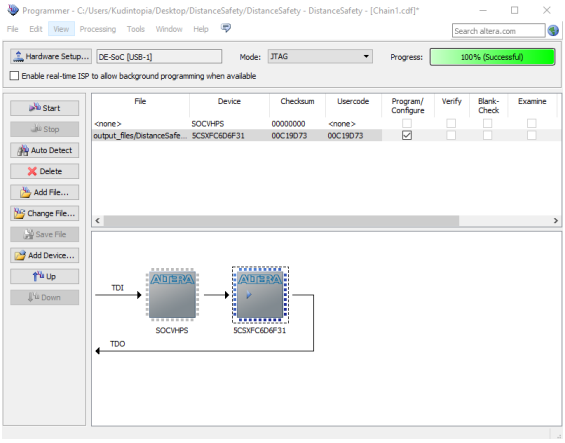

Figure 7: Programmer Upload for FPGA

The System Implementation for CPLD

Figures 8 and 9 shows the pin planner and program upload of the CPLD.

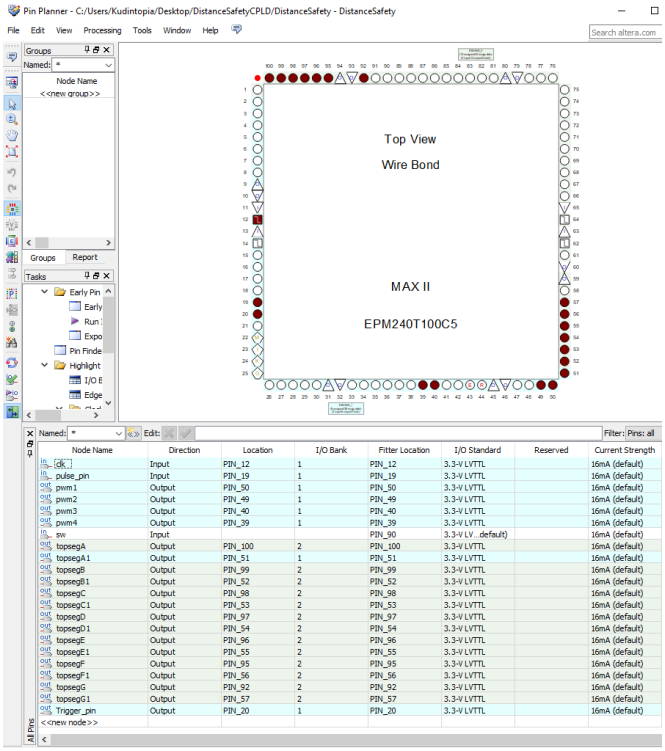

Figure 8: Pin Planner for CPLD

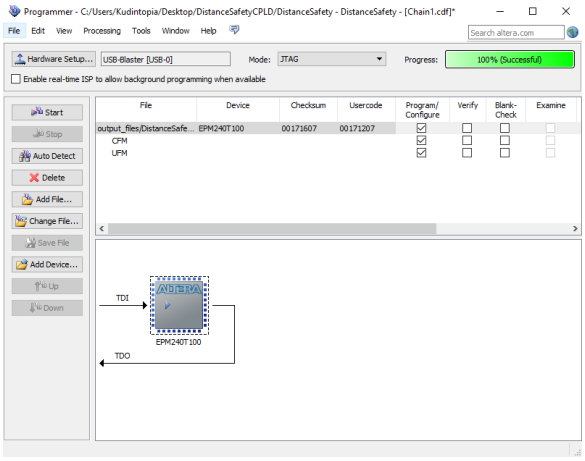

Figure 9: Programmer Upload for CPLD

SignalTap II Logic Analyzer Editor

Figure 10 illustrate the SignalTap II Logic Analyzer Editor for FPGA.

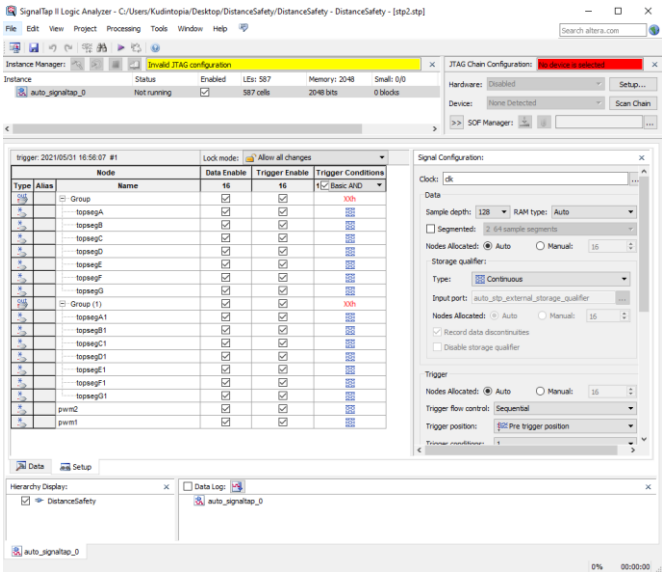

Figure 10: SignalTap II Logic Analyzer Editor for FPGA

The hardwarde used are the Terasic DE-10 Standard board, Max II board, L298N, HC-SR04, Common Anode 7-Segment Display and DC Motor. The integration of hardware is shown in Figures 11 and 12 below.

Hardware Implementation

Figure 11: Hardware Implementation for FPGA

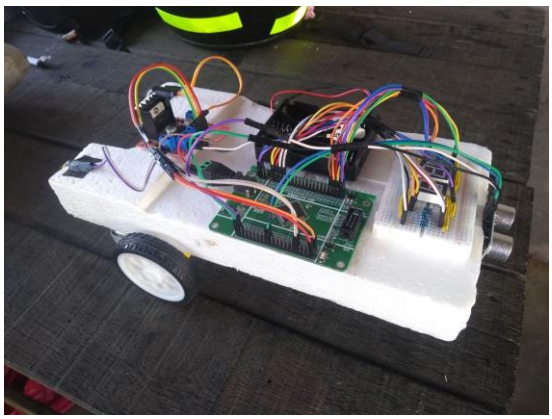

Figure 12: Hardware Implementation for CPLD

RESULTS AND DISCUSSION

Flow Elapsed Time

Flow Elapsed Time

	Module Name	Elapsed Time	Average Processors Used	Peak Virtual Memory	Total CPU Time (on all processors)
1	Analysis & Synthesis	00:00:34	1.0	4944 MB	00:00:59
2	Fitter	00:01:03	1.2	6252 MB	00:01:37
3	Assembler	00:00:11	1.0	4890 MB	00:00:10
4	TimeQuest Timing Analyzer	00:00:16	1.2	5244 MB	00:00:14
5	EDA Netlist Writer	00:00:03	1.0	4819 MB	00:00:03
6	Total	00:02:07	--	--	00:03:03

Figure 13: Flow Elapsed Time for FPGA

Flow Elapsed Time

	Module Name	Elapsed Time	Average Processors Used	Peak Virtual Memory	Total CPU Time (on all processors)
1	Analysis & Synthesis	00:00:15	1.0	4834 MB	00:00:31
2	Fitter	00:00:03	1.0	5181 MB	00:00:03
3	Assembler	00:00:01	1.0	4730 MB	00:00:01
4	TimeQuest Timing Analyzer	00:00:03	1.0	4765 MB	00:00:02
5	EDA Netlist Writer	00:00:01	1.0	4694 MB	00:00:01
6	Total	00:00:23	--	--	00:00:38

Figure 14: Flow Elapsed Time for CPLD

Based on both result in Figure 13 and 14 shows that Flow Elapsed Time result for FPGA takes longer than CPLD. For FPGA, the longest time taken is 1.03 minutes which is used for Fitter process while the total time taken for Flow Elapsed Time is 2.07 minutes. Fitter process is used for aligns the project's logic and temporal requirements with the target device's available resources. It chooses the ideal logic cell placement for routing and timing for each logic function, as well as appropriate interconnection pathways and pin assignments.

For CPLD, the total time taken for Flow Elapsed Time is 0.23 minutes. The longest process is Analysis and Synthesis, which is 0.15 minutes. Analysis and Synthesis is a procedure that checks for logical completeness and consistency in a project, as well as boundary connectivity and syntax problems. It also synthesises and maps the logic in the design entity or project's files to technology. Flip flops, locks, and state machines are all inferred from VHDL. It generates state assignments for state machines and makes resource-saving decisions.

The Assembler process is a process to generates programming files. It converts Fitter's device, logic cell and pin assignments into a programming image such as Programmer Object Files (.pof) for CPLD or SRAM Object Files (.sof) for FPGA. For FPGA, it takes 0.11 minutes for the Assembler process while for CPLD, it only takes 0.01 minutes.

Resource Usage Summary

	Analysis & Synthesis Resource Usage Summary	
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	383
2		
3	Combinational ALUT usage for logic	480
1	-- 7 input functions	1
2	-- 6 input functions	99
3	-- 5 input functions	86
4	-- 4 input functions	61
5	-- <=3 input functions	233
4		
5	Dedicated logic registers	627
6		
7	I/O pins	22
8	Total MLAB memory bits	0
9	Total block memory bits	2048
10		
11	Total DSP Blocks	0
12		
13	Maximum fan-out node	altera_internal_jtag~TDO
14	Maximum fan-out	367
15	Total fan-out	4263
16	Average fan-out	3.63

Figure 15: Resource Usage Summary for FPGA

Figure 15 displays the Resource Usage Summary for FPGA. LUT is a Lookup Table while ALUT is an Adaptive Lookup Table. Cyclone V contains two combinational logic LUTs and two registers, which are linked together to form a Combinational ALUT/register. In this FPGA, the number of Combinational ALUT use are 480. Combinational ALUT usually have up to 7 input function. The total memory block that has been used in this FPGA is 2,048 from 5,662,720, which is less than 1%. The total I/O pins for this FPGA is 499, while the number of I/O pin that has been used is 22. Only 4% from the total I/O pin that has been used.

	Analysis & Synthesis Resource Usage Summary	
	Resource	Usage
1	Total logic elements	218
1	-- Combinational with no register	144
2	-- Register only	11
3	-- Combinational with a register	63
2		
3	Logic element usage by number of LUT inputs	
1	-- 4 input functions	104
2	-- 3 input functions	32
3	-- 2 input functions	68
4	-- 1 input functions	3
5	-- 0 input functions	0
4		
5	Logic elements by mode	
1	-- normal mode	139
2	-- arithmetic mode	79
3	-- qfbk mode	0
4	-- register cascade mode	0
5	-- synchronous clear/load mode	0
6	-- asynchronous clear/load mode	46
6		
7	Total registers	74
8	Total logic cells in carry chains	83
9	I/O pins	22
10	Maximum fan-out node	clk
11	Maximum fan-out	65
12	Total fan-out	821
13	Average fan-out	3.42

Figure 16: Resource Usage Summary for CPLD

Figure 16 illustrates the Resource Usage Summary for CPLD. For CPLD, the number of LUT input used is 207. The maximum number of input for input function that has been used in this CPLD is 4 input function. The I/O pins that used for CPLD is same as FPGA that is 22 pins. But for CPLD, the total number of I/O pins is 80 and that make the percentage of I/O pin that used is 28%.

SignalTap II Logic Analyzer Editor

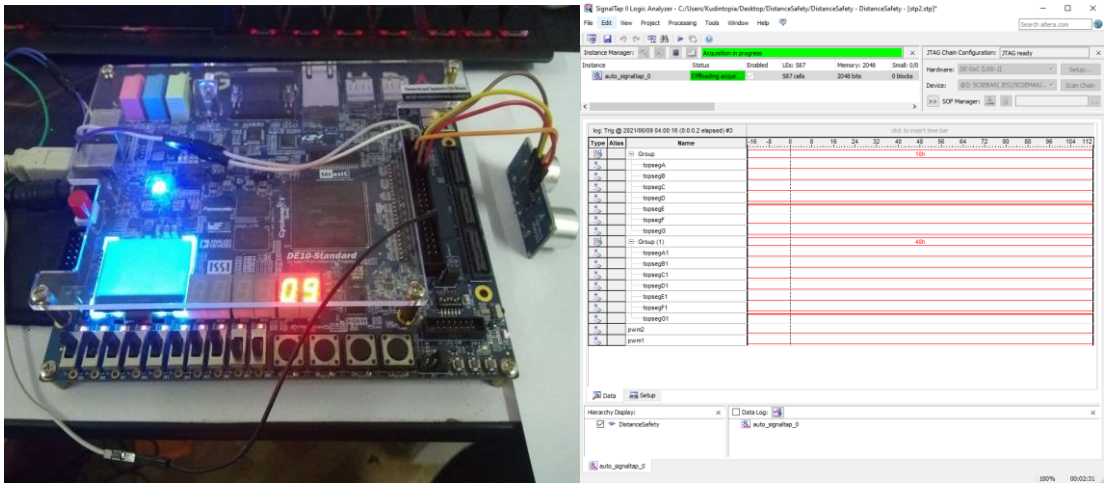

Figure 17: SignalTap II Logic Analyzer Editor for FPGA

The SignalTap II Logic Analyzer showed in Figure 15 shows the value based on reading detect by the ultrasonic sensor. There are two common anode 7-segment displays used in this project. First 7-segment display showed the number 0 because the signal on topsegG1 is "1" while the other signal is "0". For the second 7-segment display, it showed the number 9 because the signal on topsegE is "1" while the other is "0". This showed that the result on the SignalTap II Logic Analyzer is same as the result on the hardware. The pwm1 and pwm2 signal is "0" because the motor is not moving since it is in stop range.

For the CPLD, there are no result for SignalTap II Logic Analyzer since the SignalTap II Logic Analyzer does not support the Max II board.

CONCLUSION

In conclusion, there are several different between FPGA and CPLD in implementation of Collision Avoidance System. Even though, both of the system using same VHDL programming and same components. Both of the system use L298N motor driver, Common Anode 7-segment display, HC-SR04 ultrasonic sensor and DC motor.

The hardware result showed that the Collision Avoidance System in both FPGA and CPLD function according to the program. As the distance between the system and obstacle become closer, the speed of the motor become lower and the motor will stop before the system make contact with the obstacle. This showed that the PWM method is working.

Based on the research, it showed that FPGA contains logic blocks up to 100,00 while CPLD only contain around hundred logic blocks. For the hardware, FPGA showed a huge advantage over CPLD. One of it is the number of I/O pin. For FPGA, it contains almost 499 I/O pins while CPLD only contain 80 I/O pins. FPGA also have many built-in components such as 7-segment display, LCD and many more. Because of the high number of logic block and high number of I/O pins, FPGA can perform more complex computation and application than CPLD.

Some of the result showed that CPLD also has an advantage over FPGA. During compilation, CPLD only take less time than FPGA. The huge advantage of CPLD is, it has non-volatile storage while FPGA does not have it. It means that CPLD will not lose its configuration when the power is removed. FPGA need to load configuration data every time the FPGA is power ON. CPLD usually comes with small size compare to FPGA. Because of the small size and non-volatile storage, CPLD become more portable than FPGA.

REFERENCES

[1] T. Soniya, I. Ragasudha, and P. N. Valli, "Routing Architecture and Applications of FPGA : A survey Routing Architecture and Applications of FPGA : A survey," 2021, doi: 10.1088/1742-6596/1717/1/012025.

[2] G. Krishna and S. Roy, "Fundamentals of FPGA Architecture," *Fundam. FPGA Archit.*, no. November, pp. 12–30, 2017, [Online]. Available: www.tscipub.com.

[3] E. Monmasson and M. N. Cirstea, "FPGA design methodology for industrial control systems - A review," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1824–1842, 2007, doi: 10.1109/TIE.2007.898281.

[4] V. Shah, B. Patel, and R. Kshirsagar, "Cpld based stepper motor control application."

[5] P. Malav, B. Patil, and R. Henry, "Compact CPLD Board Designing and Implemented for Digital Clock," *Int. J. Comput. Appl.*, vol. 3, no. 11, pp. 7–10, 2010, doi: 10.5120/783-1108.

[6] D. Kania, "Logic Decomposition for CPLD Synthesis," *IFAC Proc. Vol.*, vol. 33, no. 1, pp. 49–52, 2000, doi: 10.1016/s1474-6670(17)35585-4.

[7] J. Jansson, *Collision Avoidance Theory with Application to Automotive Collision Mitigation*, vol. Ph.D., no. 950. 2005.

[8] A. Deshmukh, "Design of Fabrication of Automatic Braking System," pp. 433–438.