MIDDLE EAST TECHNICAL UNIVERSITY

ELECTRICAL/ELECTRONICS ENGINEERING DEPARMENT

EE400 SUMMER PRACTICE REPORT

Student Name: Burak Topçu

Student ID: 2167385

SP Company: ROKETSAN A.S

Department: Guidance Sysytems Design Department & Tactical Missile Systems

SP Date: 11.07.2019-08.08.2019

Advisor: Suhip Tuncer Soyer

Advisor mail address: tuncer.soyer@roketsan.com.tr

Advisor phone: +90 312 860 55 00 / Ext.: 6783

1. TABLE OF CONTENTS

2. GENERAL INFORMATION ABOUT ROKETSAN

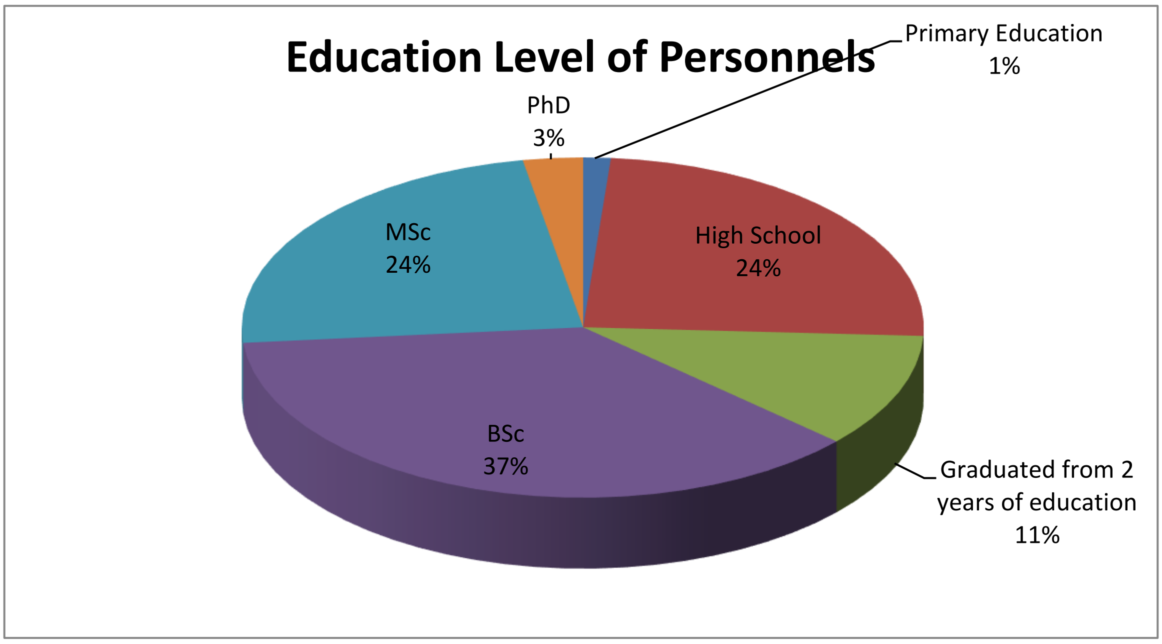
Roketsan A.Ş. was established in 1988 with the aim of leading institution in the design, development and production of rockets and missiles in our country with the decision of the Defense Industry Executive Committee. In accordance with this purpose; Roketsan has started its activities with the second largest industrial work share within the scope of the European Joint Production Program and has been continuing its activities successfully by participating in various international programs and adding original design products to the Turkish Armed Forces inventory (Roketsan) [1].

Roketsan that was founded in Ankara / Elmadağ, and now has qualified staff of 3000 people who take part in advanced technological applications for several purposes, can adapt itself to the standards of aviation / aerospace industry, and aim to present its capabilities and creativity to the service of the national defense force of country, has participated in NATO programs in its field of technology and its products, and become an institution that offers service to some countries as well.

Since its establishment, Roketsan has been in line with its mission; including technological infrastructure projects, product development and production programs, as well as launch platforms and command units; has gained the ability to design, manufacture, train user personnel and meet logistic support needs of weapon systems. The system engineering approach has played a key role in achieving these capabilities by taking into account all the elements that will include all subparts of the rocket and missile systems. For this purpose, internal ballistics, fuel technology, structural-thermal-dynamic analysis, material technology, aerodynamics, structural technology, flight mechanics, test technology, guidance engineering, sensor technology, control engineering, control-drive technology, weapon system engineering, warhead technology, software engineering which are necessary for the design of a rocket or missile system, all disciplines are all integrated with the understanding of system engineering approach at Roketsan. In addition, Roketsan has many production capabilities such as the production and insulation of rocket motor housings, the production of all kinds of mechanical and plastic parts for these rockets and missile systems, the preparation and processing of composite materials, which is one of the most critical issues in terms of durable and lightweight for metallic pieces .Roketsan also develops projects for different purposes. In addition to the demands of the National Defense Department, some other projects are also carried out in order to get more knowledge about space or sell some projects or its sub-parts to the other countries (Roketsan) [1].

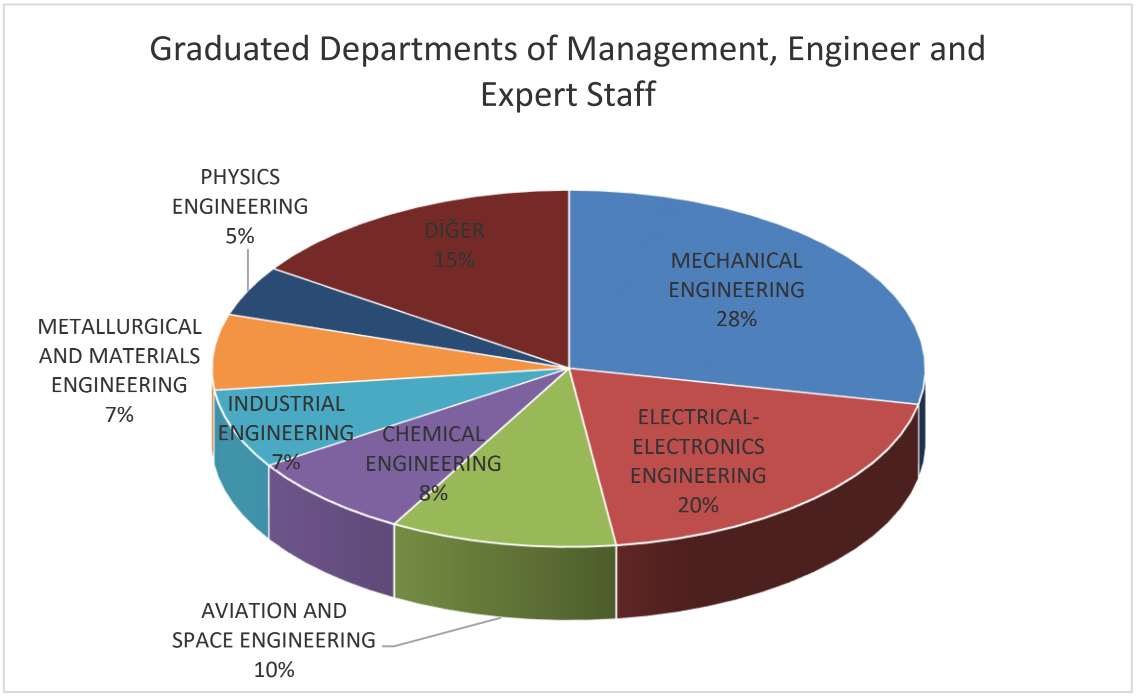
In the followings, I will present some graphics about Roketsan staff. From the graphs in figure 1, you see the percentage distribution of the expertise and bureaucratic status of the staff such as engineers, technicians and managers working at Roketsan. High school and associate degree education generally works as a technician in the company. In addition, a small part of high school and a group of elementary schools offer internal services to Roketsan employees. Other employees who are undergraduate and graduate education work as engineers in Roketsan. Some of the MSc's and the PhD's diplomaed are senior engineers and directors of the subunits such as Tactical Missile Systems.

Figure 1: Distribution of education status of Roketsan personnel on pie chart



In the pie chart in figure 2, information about the graduation departments and areas of expertise of manager, expert and engineer staff belonging to Roketsan is shared.

Figure 2: Pie chart showing the academic graduation departments of Roketsan staff



In addition, the departmental structure in Roketsan has a bureaucratic mechanism, which can be seen in the diagrams 1 and 2 below. I'm sharing the diagram with you by dividing the diagram into two because of the excess branching in the diagram. You can reach the original diagram from the website of Roketsan for detailed investigations.

Diagram 1: Hierarchical structure 1 in Roketsan

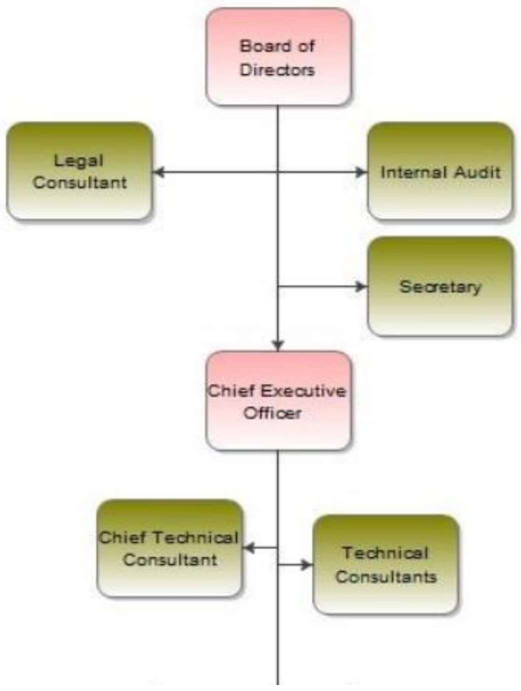
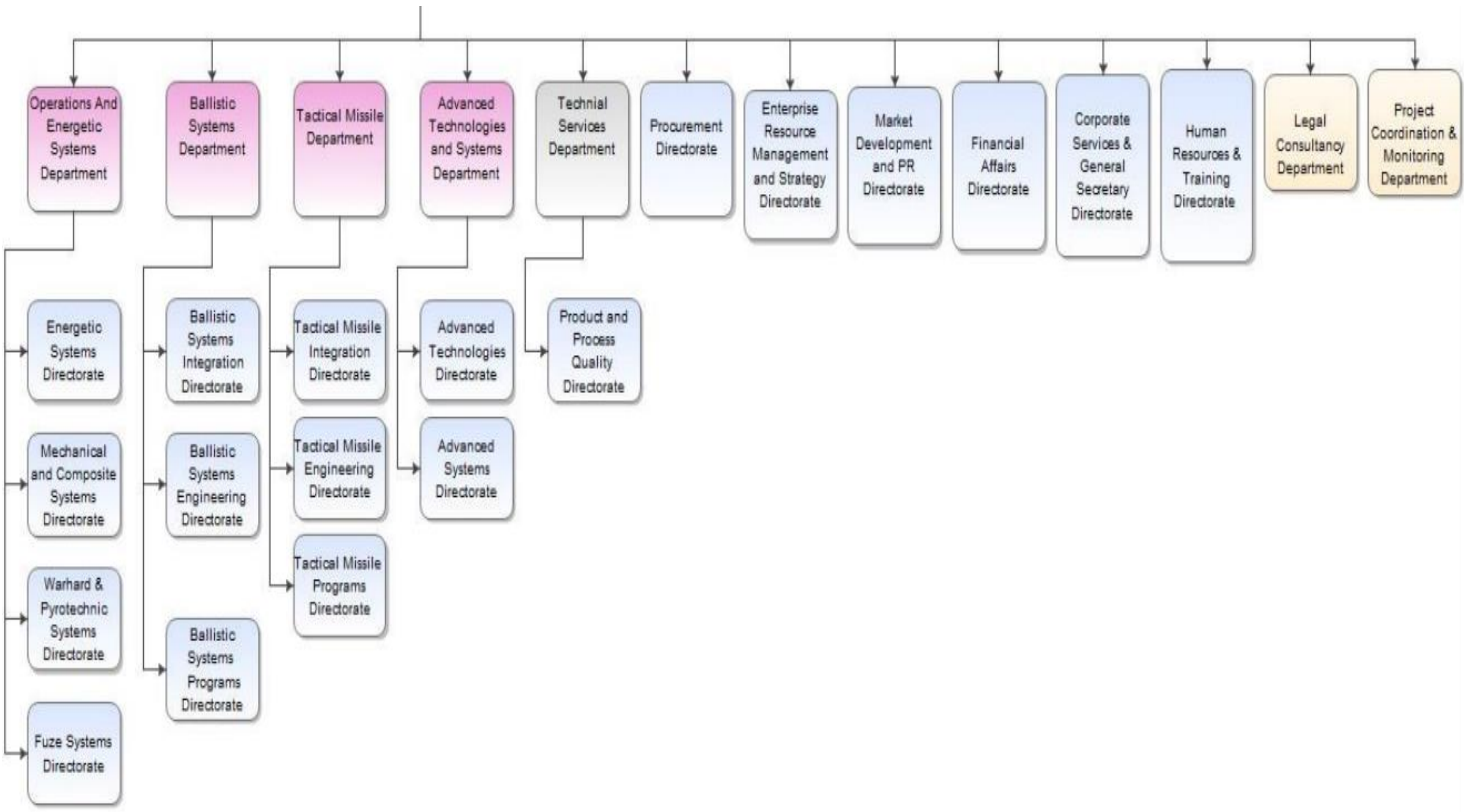


Diagram 2: Hierarchical structure 2 in Roketsan


Operations And Energetic Systems Department
Ballistic Systems Department
Tactical Missile Department
Advanced Technologies and Systems Department
Technial Services Department
Procurement Directorate
Enterprise Resource Management and Strategy Directorate
Market Development and PR Directorate
Financial Affairs Directorate
Corporate Services & General Secretary Directorate
Human Resources & Training Directorate
Legal Consultancy Department
Project Coordination & Monitoring Department
Energetic Systems Directorate
Mechanical and Composite Systems Directorate
Warhard & Pyrotechnic Systems Directorate
Fuze Systems Directorate
Ballistic Systems Integration Directorate
Ballistic Systems Engineering Directorate
Ballistic Systems Programs Directorate
Tactical Missile Integration Directorate
Tactical Missile Engineering Directorate
Tactical Missile Programs Directorate
Advanced Technologies Directorate
Advanced Systems Directorate
Product and Process Quality Directorate

3. ROKETSAN PRODUCTION ACTIVITIES

3.1. Mechanical Component Production Activities

Turning: Internal or external turning, threading and drilling operations of metal and composite parts are performed on lathes. There are 2, 3, 4 and 5 axis lathes in various brands belonging to the mechanical production unit. In CNC lathes, the machining program is written as code embedded to the machine, and in universal lathes the technician controls the machine manually.

Milling: Metal and composite parts drilling, tapping, surface machining, angle drilling operations are performed on milling machines. There are 3,4 and 5 axis milling machines of various brands belonging to mechanical production unit.

Plastering: This method is known as cold forming method. It is the process of lengthening and reducing the thickness of the hollow metal materials in the form of pipes. In this way, engine pipes of rockets are produced. It is made on CNC plastering machine.

Heat Treatment (Processing): In the mechanical parts production unit, hardening and tempering heat treatments are performed. In vacuum temper heat treatment furnace, hardening heat processes can be done under vacuum.

Welding: MIG, MAG and TIG welding and engine pipe production are carried out in mechanical production workshops. Longitudinal welding, circumferential welding and spot welding are used in engine pipe production.

3.2. Composite Parts Production Activities

A composite material is a material formed by physically and chemically combining two or more materials with different properties with the help of some machines. This material exhibits different properties than the constituent components.

Composite materials used in Roketsan have high abrasion resistance, low expansion, high strength, high melting temperature and low specific gravity. Thanks to these features, it protects the some pieces of rocket or missile system such as electronic cards from high temperature and abrasive environment caused by firing of rockets and missiles by using them for insulation in engine body, nozzle, and igniter parts. The main component is polymer-based, while the additives are glass, aramid and carbon.

In composite component production workplaces;

- Production of composite intermediates (prepreg, premix),
- Composite material forming (ribbon winding, filament winding, press, hand depositing, vacuum-bagging),
- Composite material maturation (press and autoclave),
- Composite material processing (universal lathe),
- Composite material assembly,
- Surface treatment (coating, sandblasting, and painting) operations are performed.

4. ROKETSAN DESIGN ACTIVITIES

Roketsan defines the task as a first step in the design process. Then, a configuration corresponding to this defined task is determined. This configuration design may be aerodynamic, propulsion or mechanical topics. After the design is carried out, it is subjected to simulations in order to observe the results. If the simulation result is meaningful for purpose of project, the design step starts to be physically occured. Some of the design activities I have collected from the notes I received at Roketsan sub-units and the information shared by the company during the campus trips are as follows.

I. Avionic System Design Activities:

- Guidance kit design,
- Test head kit design,
- RF System design

II. Hardware and Sensor Mechanics Design Activities:

- Test head kit design,
- Design of avionics electronic equipment,
- On-board (vetronic) electronic equipment design,
- Inertial sensor design

III. Missile System Design Activities:

- System architectural design,
- System requirements analysis,
- Technical coordination,
- Flight Performance and Qualifications Analysis

IV. Power Transmission and Motion Systems Design Activities:

- Electromechanical System Modeling, Control, Analysis and Testing,
- Hydraulic System Modeling, Control, Analysis and Testing,
- Pneumatic System Modeling, Control, Analysis and Testing,
- High speed brush / brushless DC motor development,
- Magnetic System Modeling, Servovalf and Solenoid Valve Design,
- Piezoelectric Design,
- Launching Tools Precision Orientation Mechanisms and Stabilization Systems Modeling, Control, Analysis and Testing Ability,

V. Guidance, Control and Navigation Algorithms Design Activities:

- Modeling of inertial sensors and systems,
- Design of navigation algorithms,

- Initial Alignment algorithms design,
- Support systems integration algorithms design,
- Guidance and flight control algorithms design

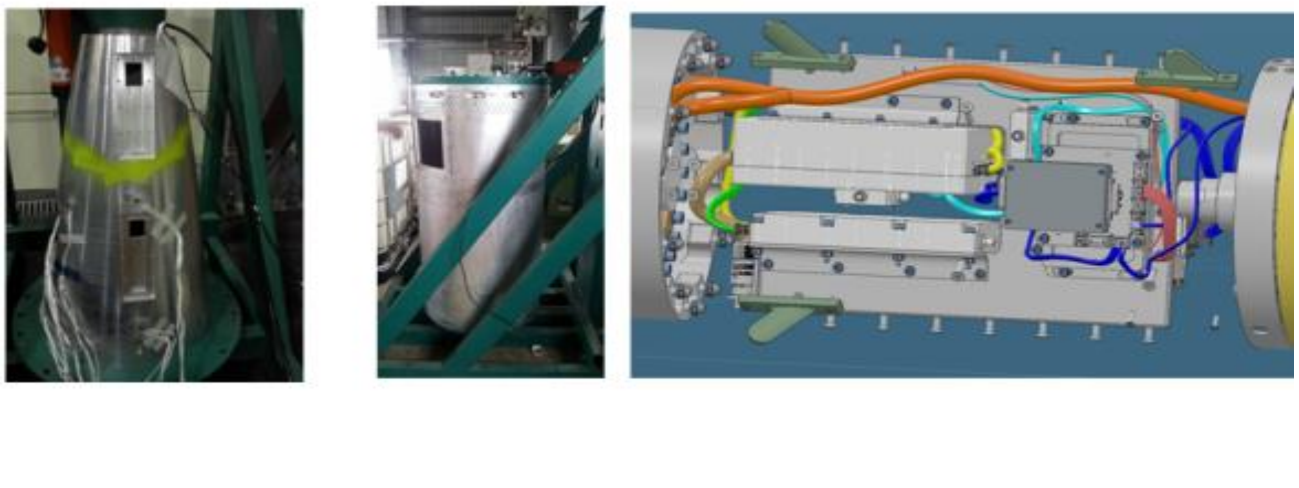
VI. Modeling and Simulation Activities:

- Flight Simulation Development,
- Flight Control Software Development,
- Development of Guidance Algorithms,
- Development of Control Algorithms for missiles or rockets,
- Ballistic Software Library Development,
- Area Division, Target Allocation, etc. Developing Algorithms, (my subject)
- Developing Deep Learning Algorithms for System Identification / Guidance and Control. (my subject)

VII. System Mechanical Design Activities:

- Missile or Rocket Structural / Dynamic Design,
- Guided Artillery Ammunition Design,
- Guidance Kit Design,
- Trial Title Design,
- Weapon System Superstructure Design,
- Launch System Design
- Thermal Insulation Systems Design,
- Useful Load Design,
- Autonomous System Design

Figure 3: Interface output for physical design of the trial title and positioning of the guidance kit in the title

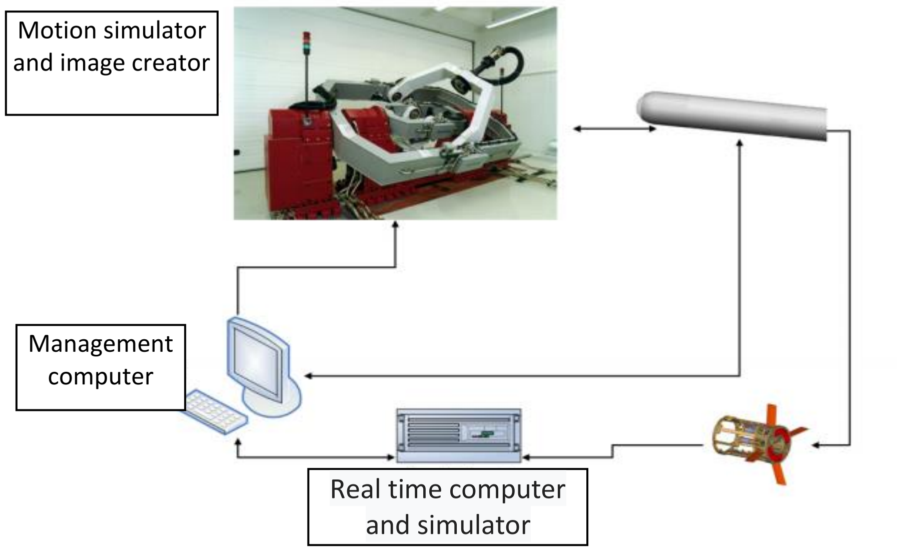


VIII. System Testing and Evaluation Activities:

- Requirements management,
- Planning of system tests,
- Determination of test prototype configuration,

- Determination of test infrastructure requirements,
- Performing system tests,
- Shooting test,
- Environmental condition tests,
- Evaluation of test results,
- Design and development of test equipment in terms of hardware and software,

Figure 4: Test, evaluation and analysis track


Motion simulator and image creator
Management computer
Real time computer and simulator

IX. Roketsan Navigation and Inertial Measurement System Devices

Navigation is the phenomenon of determining (not the shortest) the most convenient way to go from one point to another (route planning) and performing the journey on the planned route. The term navigation is an important part of the navigation business.

5. ROKETSAN INTEGRATION ACTIVITIES

I. Guidance Integration and Test Activities

- **Thermal Battery:** Provides the power the missile needs.
- **Electronic Cards:** Receive and send commands required for peripherals during flight. Acts in accordance with the algorithm.
- **Carrier Body:** Mechanical installation of thermal batteries, PCU, electronic cards and cabling is done on the carrier body.
- **Guiding Body:** It constitutes the outer body of the guiding assembly.
- **Avionics:** They are electronic systems in the missile. The complete combination of the Guidance, Seeker and Control Drive System sub-assemblies is called the Avionic Complex

They did not present us an orientation for some of the below activities within the institution. As well, I could not find anything that could indicate content since they also did not inform us about the studies. I just wanted to state that these activities were held as a result of the information that the institution shared with us.

II. **Seeker Integration and Test Activities**

III. **Mechatronic Integration and Test Activities**

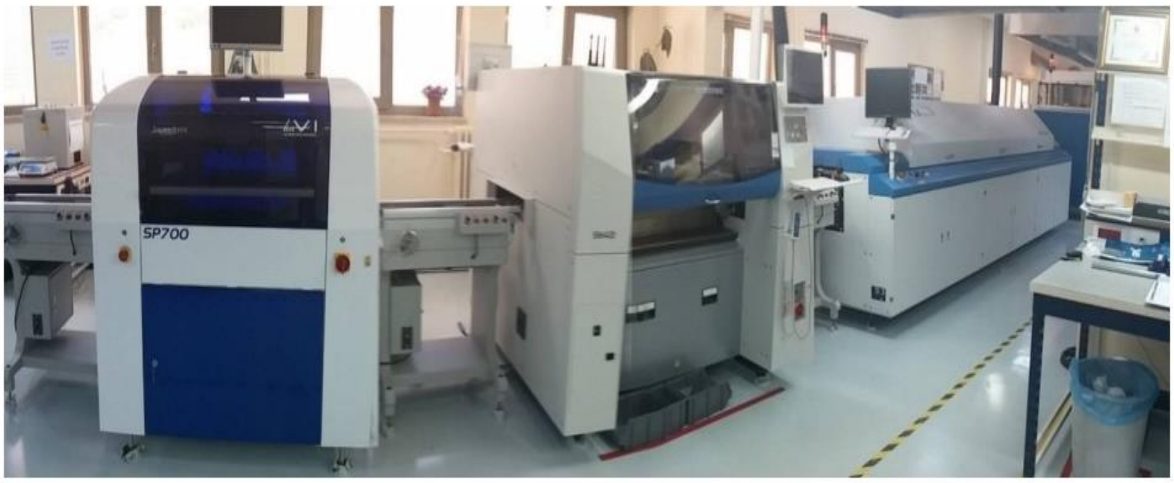
IV. **Weapon Systems Integration Activities**

V. **Control Drive System Production Activities**

VI. **Electronic Card Assembly Line Activities**

This unit is located on Roketsan's Lalahan campus. Here, the cards designed in other units are produced on the PCB card typesetting line as in the photograph shared with you in figure 5 and delivered to the desired units and added to the projects.

Figure 5: Electronic card typesetting line (PCB production unit)



6. ROKETSAN ANALYSIS ACTIVITIES

6.1. Structural Analysis and Test Activities

Rockets or missiles and their launch systems are exposed to a wide range of structural loads, from mechanical loads in assembly / storage / transportation conditions to provide aerodynamic durable during flight. The systems are required to maintain their structural integrity under these loads that they encounter from their production to the end of their service life.

If the structural requirements are not controlled by structural analysis; for example, due to the load to which a missile designed with a low wall thickness is subjected during its maneuver, the body may be buckled. On the contrary, if the wall thickness is designed to be extremely safe, it may not meet the performance requirements (weight, range, flight time, etc.) and may be costly. Therefore, there are lots of analysis activities in order to present the product that meets the design requirements in a best way, in the shortest time and with the lowest cost.

6.2. Thermal Analysis and Test Activities

All subsystems that make up rocket and missile systems are functional between the ceratin limited temperatures. These systems are exposed to different thermal loads (heat flux, radiation, etc.) from the time they wait in the warehouse until they hit the target. Thus, they are exposed to different temperature distributions throughout their entire life cycle. Temperatures under thermal loads can lead to various serious fault mechanisms on the system, such as the loss of functionality of electronic cards. Therefore, it is necessary to satisfy the performance requirements of the system with thermal analyze performed at the design stage. Furthermore, the temperatures obtained as a result of thermal analyzes constitute an important input for the structural and dynamic characteristics of the system.

6.3. Dynamic Analysis and Test Activities

The dynamic properties of rockets and missile systems (natural frequencies, mode shapes, etc.) and their resistance to dynamic loads (vibration, shock, acoustics, etc.) are important during the design phase. Designing the structure without considering the dynamic properties can cause the structure to become unstable, unable to function, resonate, or even break. Structures can break due to fatigue at low vibration loads compared to static loads. In addition, the response and resistance of the structures to shock loads (duration of action: 0.1-0.2 seconds) are very different from static loads.

7. PROJECTS PRODUCED BY ROKETSAN

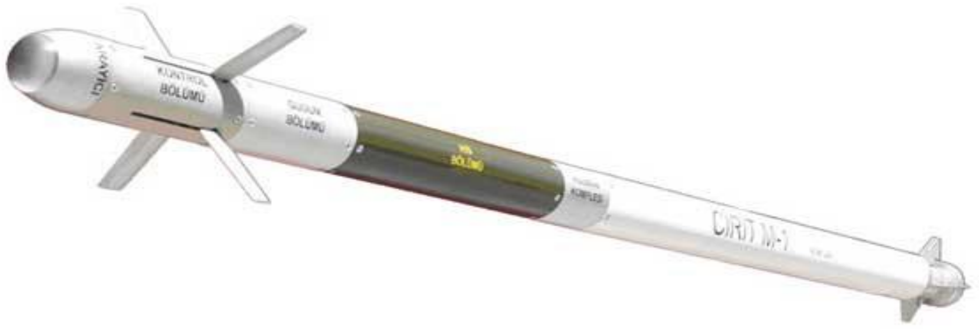
I. Cirit- Laser Powered Missile

The Laser Guided Missile is a high precision and cost effective solution used against light armored or unarmored, fixed and moving targets from attack helicopters. Cirit, a new generation missile, is designed to fill the tactical gap between unguided 2.75” rockets and guided anti-tank missiles.

Properties:

- Weight: 15 kg (Except Insulated Tube),
- Range: 1,5 - 8 km

Figure 6: Laser-guided Cirit Missile developed by Roketsan (Roketsan) [2]



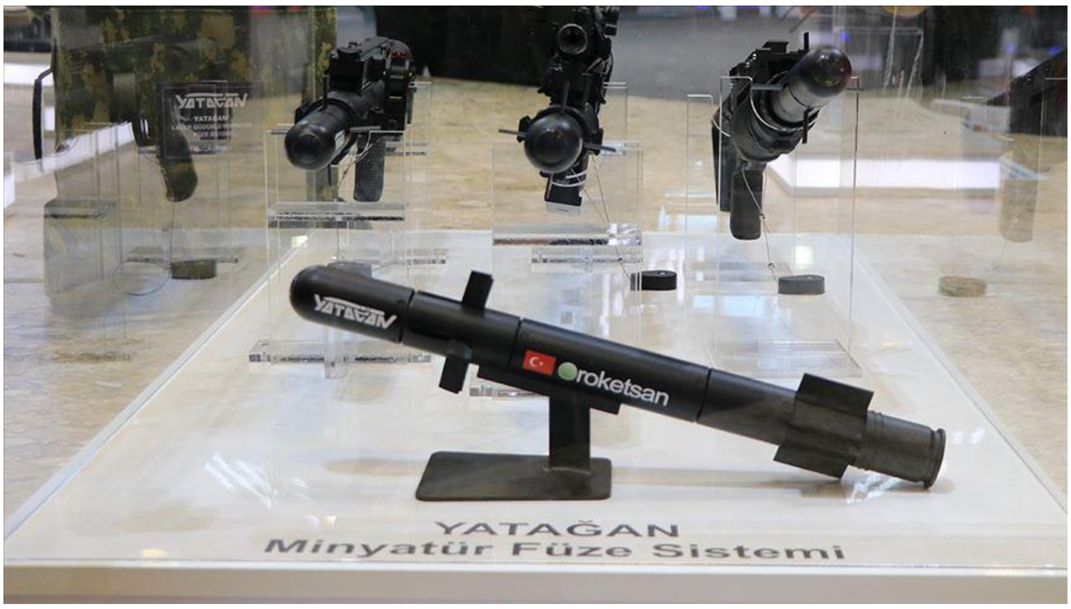
II. Yatağan - Laser Guided Miniature Missile System

Laser Guided Miniature Missile System Yatağan that can be fired the new generation 40mm bomb launcher has a more effective range compared to the its own competitors though miniature missile systems. Yatağan, which can be controlled by single personnel with bomb throwers, is designed to be integrated to mini unmanned aerial vehicles, unmanned land and sea vehicles with its light weight and small size.

Properties:

- Weight: 1 kg
- Range: 1 km

Figure 7: Laser-guided Yatağan missile developed by Roketsan (MSI, 2019) [3]



III. Bora Missile

The BORA Missile creates effective firepower on strategic targets within the battlefield. The missile is fired via 8x8 Tactical Wheeled Vehicle, and one of Roketsan's multi-purpose MLP system is

Bora weapon system. It can also be launched from other Tactical Wheel Platforms with appropriate interface for integration according to user needs.

Properties:

- Diameter: 610 mm
- Weight: 2.500 kg
- Range: 80-280 km

Figure 8: Bora missile developed and produced by Roketsan

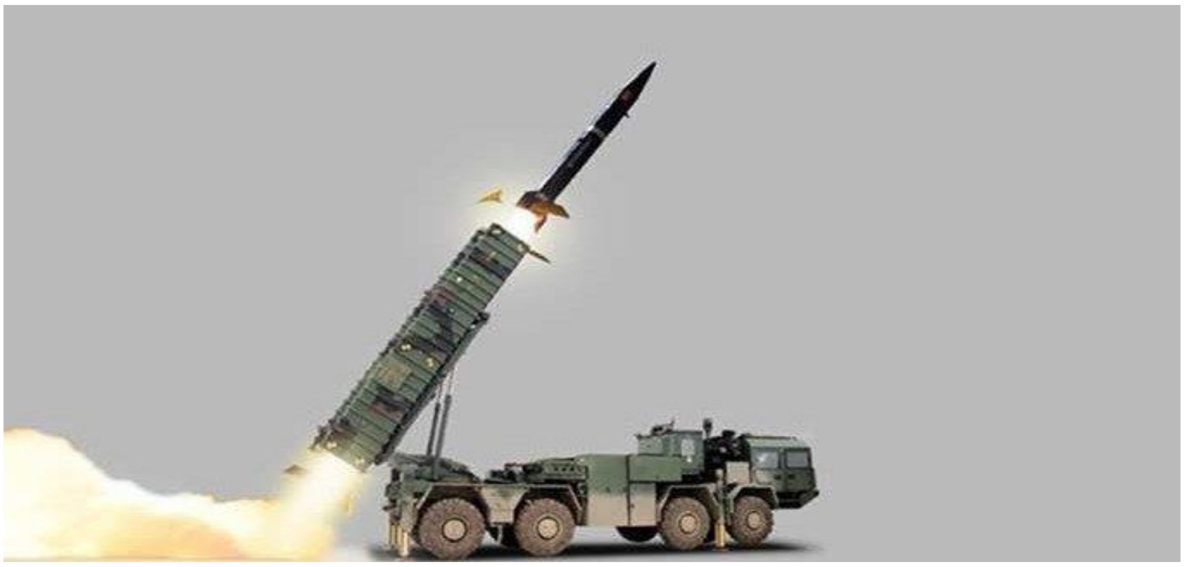
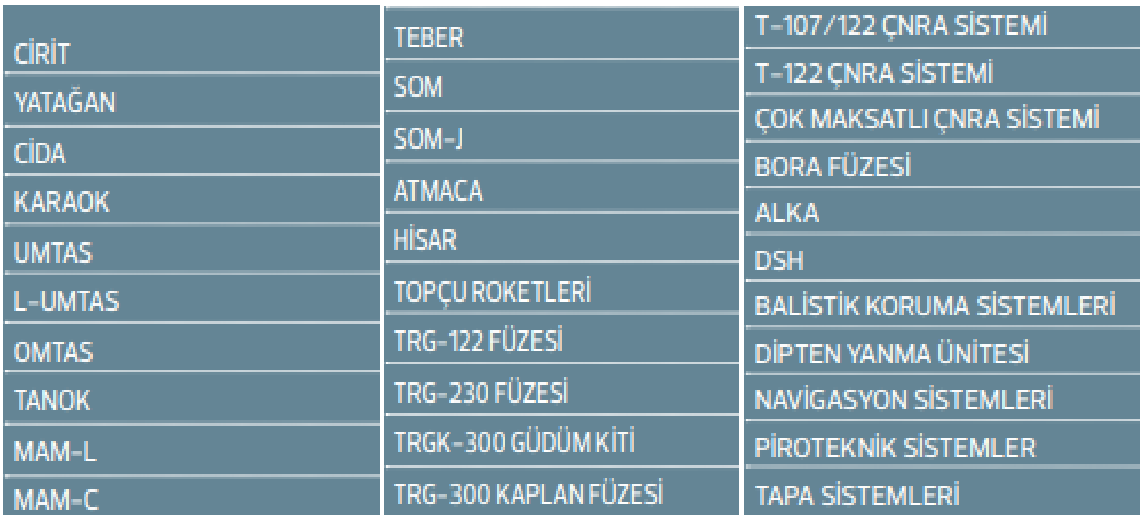


Figure 9: Other missiles, rockets and sub-parts for them produced by Roketsan



CİRİT	TEBER	T-107/122 ÇNRA SİSTEMİ
YATAĞAN	SOM	T-122 ÇNRA SİSTEMİ
CİDA	SOM-J	ÇOK MAKSATLI ÇNRA SİSTEMİ
KARAOK	ATMACA	BORA FÜZESİ
UMTAS	HİSAR	ALKA
L-UMTAS	TOPÇU ROKETLERİ	DSH
OMTAS	TRG-122 FÜZESİ	BALİSTİK KORUMA SİSTEMLERİ
TANOK	TRG-230 FÜZESİ	DİPTEN YANMA ÜNİTESİ
MAM-L	TRGK-300 GÜDÜM KİTİ	NAVİGASYON SİSTEMLERİ
MAM-C	TRG-300 KAPLAN FÜZESİ	PİROTEKNİK SİSTEMLER
		TAPA SİSTEMLERİ

8. INTRODUCTION

The aim of the summer internship was to apply and examine more closely what we learned in the lessons and what we used in the labs. Observing them in technological fields that are used in the world, and following production processes are another benefits for the students that are candidate of being an engineer. In addition, I wanted to take part in specified fields that related with our optional fields in the internship program in order to have an idea about the option I will choose for last two semesters and continued academic years. Since Roketsan has big demand in the world in addition to produced advanced technological rocket projects, and it is one of the most popular defense industry companies, I applied to there. They accepted to me for my internship programme. My internship here was a research and development internship that is abbreviated as R&D.

The reason why I prefer Roketsan is that it has proven itself internationally in the fields of missiles and rockets. Actually, I wanted to do an internship in Roketsan because it offered me the opportunity to see the corporate functioning, although the engineering profession was different than the officialism. My campus was Roketsan's later-opened campus in Lalahan. I was very pleased with my internship unit that is tactical missile systems because there were so many engineers from my university and my department here. They helped me to understand the works that is carried out and processed by this unit and find a suitable and understandable subject for my internship duration.

In Roketsan's tactical and missile systems unit, I was interested in image processing codes embedded in the Xilinx FPGA cards placed in the missiles to guide the missiles. In other words, I've worked on processing image signals coming from the camera in real-time stream. In order to do this, I applied some algorithms for edge detection, brief descriptior and corner matchings. However, I stored the incoming data in a few BRAMs and then applied image proccessing algorithms because the incoming data is not appropriate for processing. Before doing these, I read some articles first. The engineers at the company then asked me to write the code on Vivado-HLS, namely High Level Synthesis, rather than writing it on Verilog HDL. It was actually much easier to create the code on HLS, but since I could not understand the working algorithm of the functions, I continued codding for project on Verilog HDL. To try the algorithm, I wrote an algorithm on MATLAB and tried the FAST and BRIEF algorithms. I will share the results with you in the main body section.

Internship studies that I will elaborate in the main body section:

- Testing the FAST algorithm on Matlab
- Block RAM and Distrubuted RAM
- Edge Detector (FAST Algorithm)
- Feature Descriptor and Corner Matching (BRIEF algorithm)
- Use of Vivado HLS that can perform Verilog RTL implementation

9. WORK CONDUCTED AT THE ROKETSAN

My internship in Roketsan, which will continue for 20 working days, started with 3-days training and orientation seminars. In these seminars, we received lots of instructions about the confidentiality and working rules in Roketsan. After completing these seminars, interns orientated to internship units. My internship unit was laser guided systems in tactical missile systems.

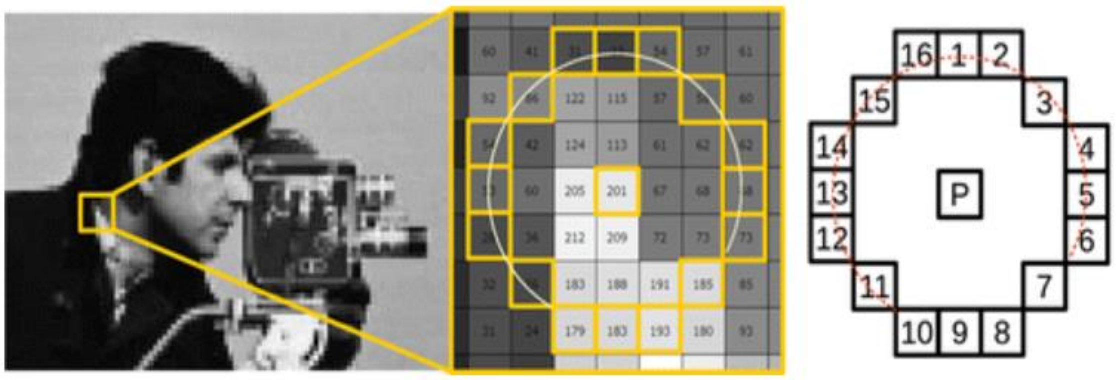
When I came to my internship unit, the engineers working on the laser systems in the unit and I found a suitable and understandable. The topic is on the field of embedded software, and they would use in their own projects. That's why I felt important to myself. What they gave me was exactly about image proccessing on stream video signals. Within the this scope, I would make them capable of processing the data that came to me on 1920x1080 pixels, applying some image processing algorithms and converting the signal back to the format that it reached me.

9.2. FAST and BRIEF Algorithm for on-board Corner Detection and Matching

I. FAST Algorithm:

Since the project that was given to me during the internship was something I did not know, I first found some articles in the literature in additon to the other articles that was given to me by the my advisor engineer. The Features from Accelerated Segment Test algorithm, abbreviated as FAST, limits the pixel area as 7x7 from HD screen that is 1920x1080 pixels, making the intensity comparison between the pixels within that area.

Figure 10: Pixels positioned by the FAST algorithm during running process (size 7x7) (Michał Fularz, 2015) [4]



In figure 10, you see a 7x7 pixel area from the random sub pixel region of 1920x1080 pixels that was taken from the photo on the left. 16 pixels marked around P contained data about the image intensity of that pixel are compared the center indicated by P in 7x7 pixel region with respect to the intensity values. These comparisons are made according to the information shown in figure 11.

Figure 11: Comparison algorithm of pixels containing 12 bit image signal data in FAST algorithm [5] (Jingjin Huang, 2018)

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \leq I_p - t \quad \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t \quad \text{(similar)} \\ b, & I_p + t \leq I_{p\to x} \quad \text{(brighter)} \end{cases}$$

I represent the 12-bit intensity stored in the each pixel. Threshold is indicated by the letter *t*. I_{p-x} represent the pixel density in the center, while I_p represents the other 16 pixels around the center. We identify the result of the 16 comparison between 16 surrounding pixels in the 7x7 pixel region and the center pixel and named the results as dark, bright or same. If the count of dark or bright result is more than 9 or equal to 9, we detect that pixel as an edge. Starting from the top left to the bottom right of a 1920x1080 pixel frame, we apply this algorithm to the all pixels by dividing them to sub-parts (7x7) to detect all edges in the frame.

In MATLAB, I wrote a code to try and understand the FAST algorithm. The results of an image read via MATLAB are shared with you in figure 12 and figure 13 below. The code can be found in the appendix. The image in figure 12 was printed out after processing on the matlab and other image that is in figure 13 was obtained. Since we do not limit the number of detection points here, too many detected points have occurred. However, in order to make advanced detection, Score algorithm can be applied in addition to this algorithm and this application can be used to compare the intensity values of the points that are detected as bright and dark within themselves. When FAST detection is performed, points designated as dark or bright with a large margin are retained, whereas points identified as dark or bright with a low margin are removing from the detected points (Michał Fularz, 2015) [4]. In this way, the number of detected points can also be limited.

Figure 12: An image taken for testing in FAST algorithm on Matlab

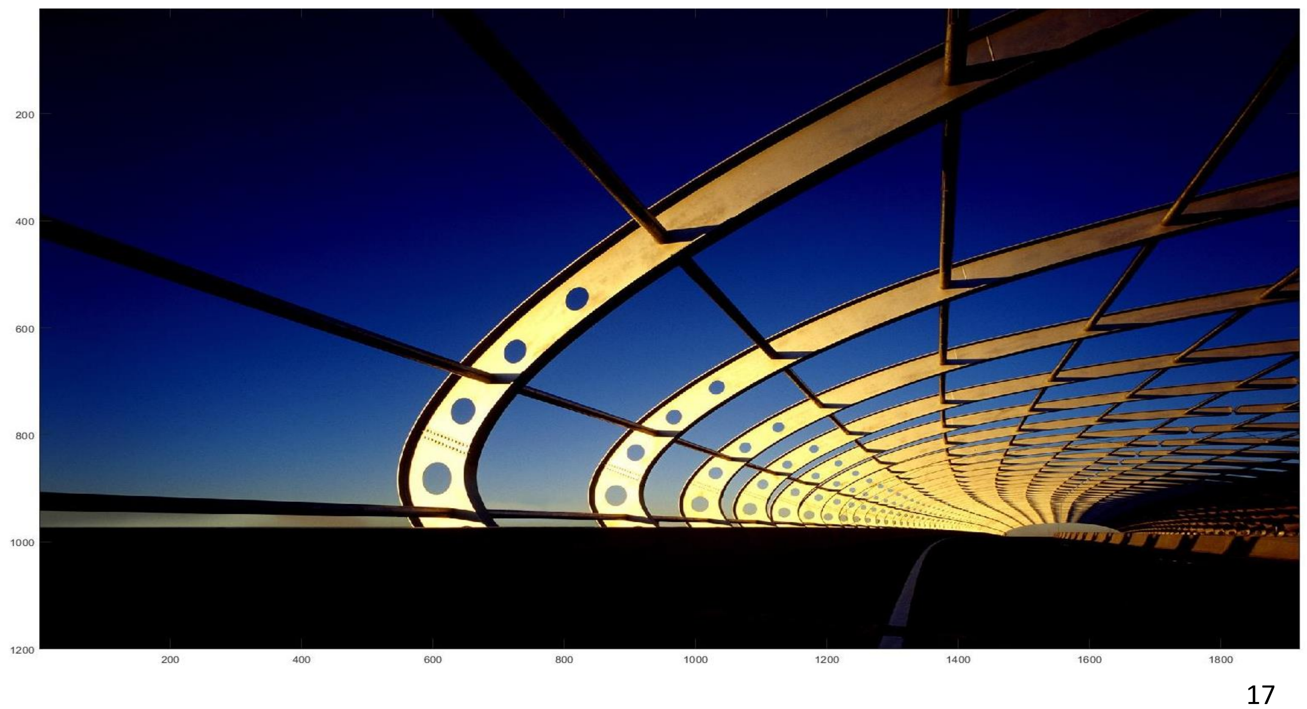
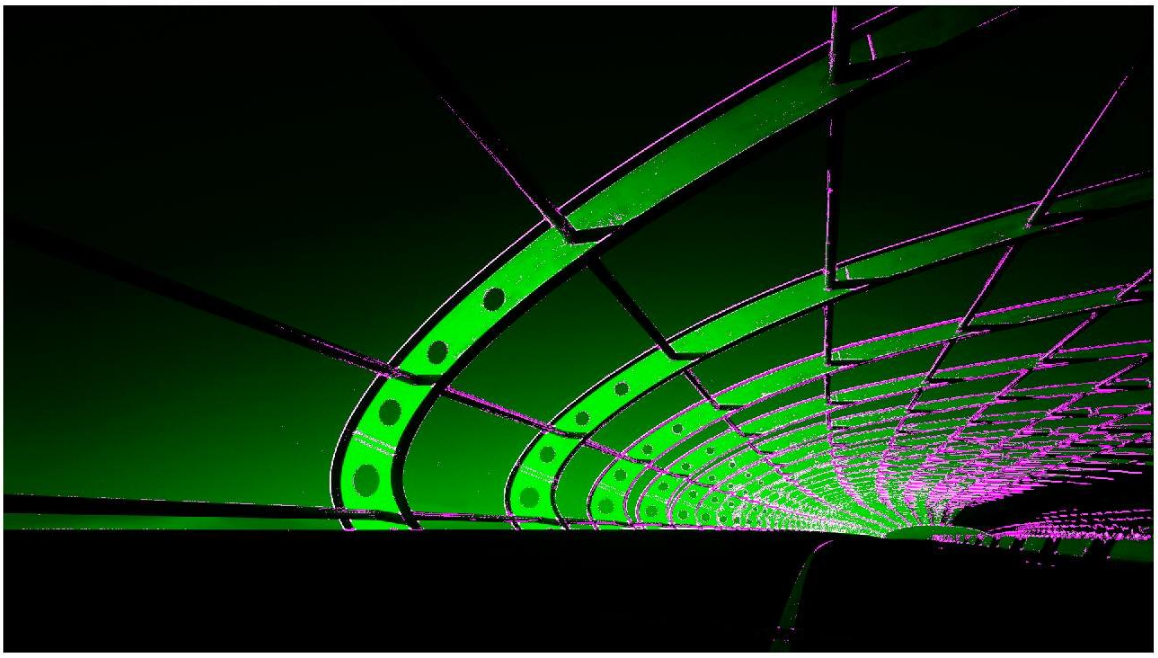


Figure 13: The resultant image of the manipulated pixel values of the detected points as a result of FAST algorithm



II. BRIEF Algorithm:

In the BRIEF algorithm, since the incoming data is very sensitive to noise, the data is passed through an averaging filter for smoothing process. Then, in a pixel slice of 33x33, 512 points are selected according to the Gaussian distribution method, and these points are matched as 256 point pairs. If the intensity difference in each pair is greater than a value called hamming distance, the points of this pair are identified as different points. If the difference in intensity for each pair points is smaller than hamming distance, these points are called similar points. The difference and similarity results are kept in a matrix of 256 bits, as 1 and 0. As shown in figure 14, schematics are representing intensity comparison for random pixel pair. In figure 15, described comparisons are shown in 33x33 pixel region, and this comparison process continues for each frame from starting the top pixel at the left side to the bottom pixel on the right side.

Figure 14: Hamming distance comparison with respect to corresponding intensity values between pixels specified in BRIEF algorithm (Michał Fularz, 2015) [4]

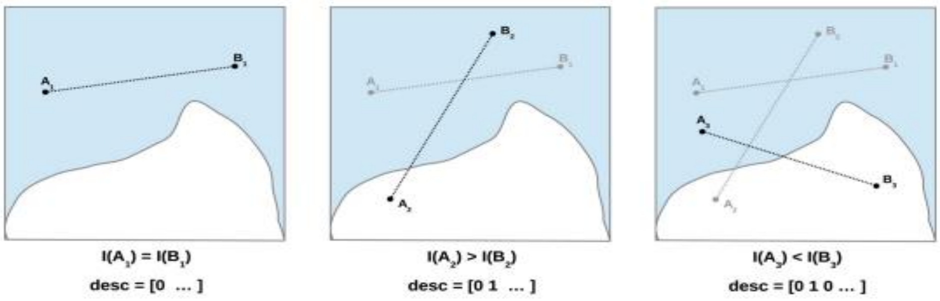
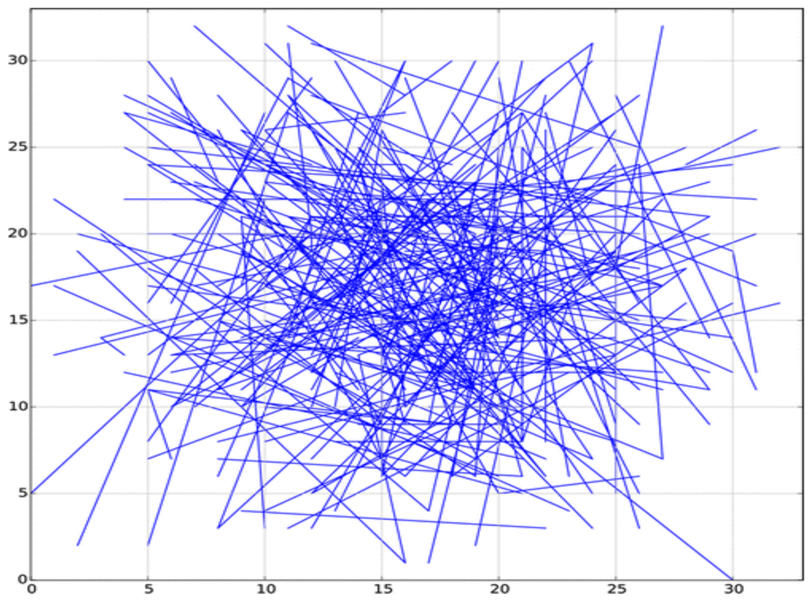


Figure 15: Selecting 512 point with respect to gaussian distribution method and grouping them as pairs on MATLAB (Michał Fularz, 2015) [4]



Each of these matrices is identified with respect to the coming video frame and the next frame is compared with previous frames with respect to the ones and zeros corresponding to the same region. XOR function is applied in matrice comparisons. If the values in the matrices match each other, it is called matched, if not, it is called unmatched. In this way, the edges and corners for a continued image signals are detected, and manipulated. An example of these comparisons with respect to different matrices that belongs same region for different frames is shown in Figure 16.

Figure 16: Comparison table with respect to XOR function, and matched or unmatched vector in BRIEF descriptor algorithm

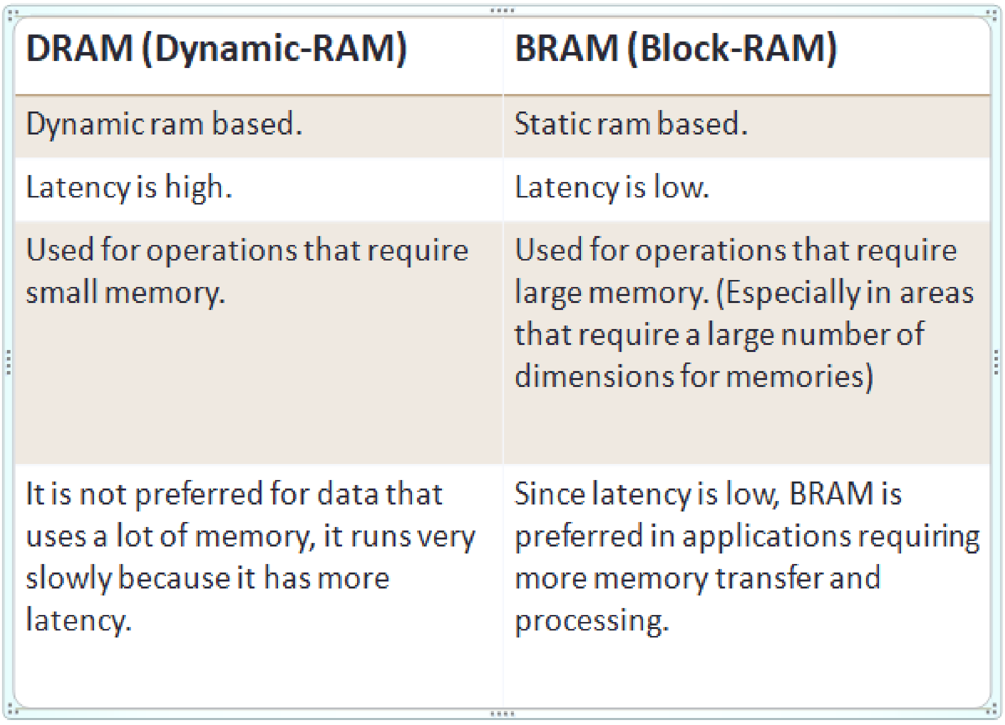
ID	Second Image	ID	First Image	XOR	Hamming Distance	Result
1	111111	1	110011	001100	2	Matched
		2	110000	001111	4	Unmatched
		3	010101	101010	3	Unmatched
2	000001	1	110011	110010	3	Unmatched
		2	110000	110001	3	Unmatched
		3	010101	010100	2	Matched
3	110100	1	110011	000111	3	Unmatched
		2	110000	000100	1	Matched
		3	010101	100001	2	Unmatched

9.2. Block RAM and Distributed RAM

In order to apply Fast algorithm, the data starting from the top line and left side of the video stream must be manipulated to get 7 pixels for each clock cycle, and these pixels are the first seven pixels of first seven lines respectively. With the help of my advisor, we first thought about how we could construct such a structure that can both store the video signals and not waste any RAM.

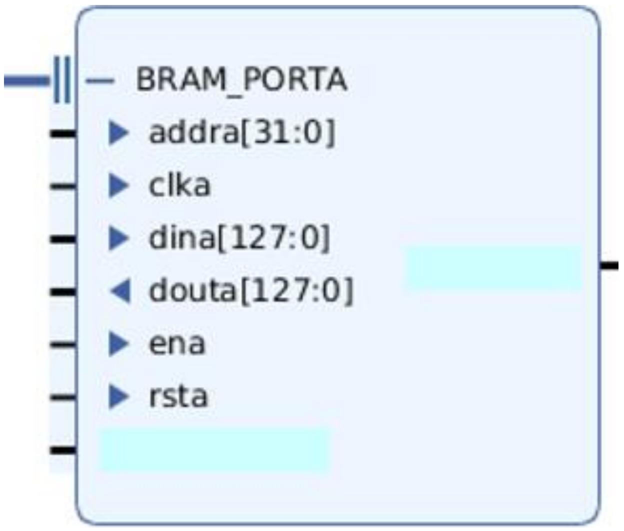
Avoiding from unnecessary RAM usage was one of the most important issues for the project, while a system that was able to work effectively was also required. I thought of creating a register for each pixel on the screen and manipulating those pixels one by one, but my advisor engineer told me that this module would not meet the requirements mentioned in the previous sentence, and led me to get to know the concepts of BRAM and DRAM. I learned these concepts as a result of some researches and prepared a table in the form of BRAM vs DRAM in my presentation to the Roketsan. It was shared with you in table 1.

Table 1: BRAM vs DRAM (Xilinx forum, 2017) [7]



DRAM (Dynamic-RAM)	BRAM (Block-RAM)
Dynamic ram based.	Static ram based.
Latency is high.	Latency is low.
Used for operations that require small memory.	Used for operations that require large memory. (Especially in areas that require a large number of dimensions for memories)
It is not preferred for data that uses a lot of memory, it runs very slowly because it has more latency.	Since latency is low, BRAM is preferred in applications requiring more memory transfer and processing.

Figure 17: Input and output ports of BRAM module in Vivado

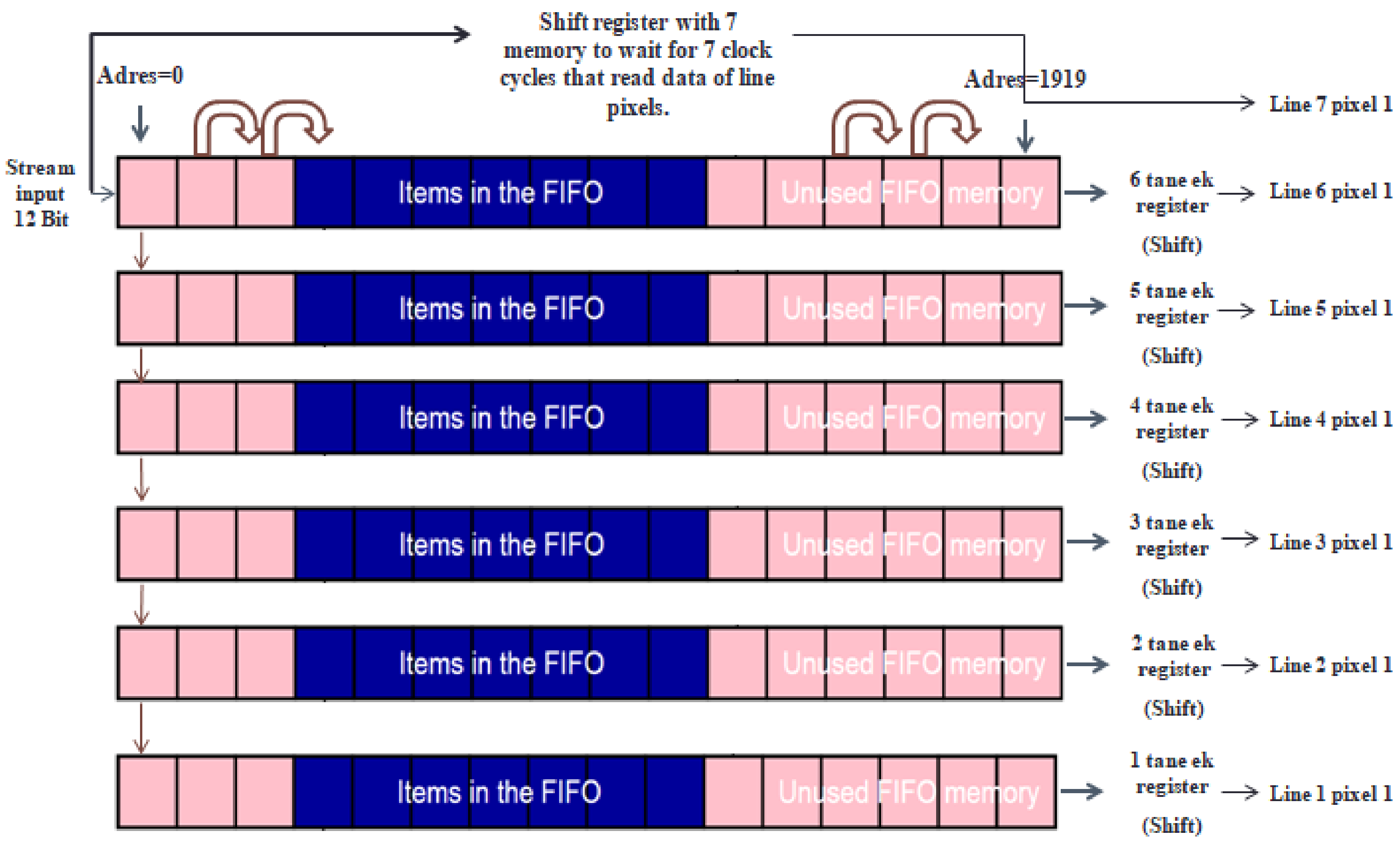


The data to be processed in the FAST algorithm must be taken from the first 7 lines as first 7 pixels at the same time and the data taking should be shifted horizontally to the right in each clock cycle. For this purpose, 7 line buffers that include 1920 memory registers have to be used to store data on the card, and use them as needed. As a result of the comparison of the two RAMs in table 1, BRAM, which is more compact than DRAM, is used in areas requiring high memory. Figure 17 that is on the left side shows the BRAM module and its ports on Vivado. The ports on the BRAM module are as follows:

- Enable port,
- Address port [1920 memory for our project],
- Clock port,
- Input and output ports,
- Reset port.

The BRAM module can be one-sided or double-sided. We preferred using one-sided BRAM modules. Different clocks can be used in a double sided BRAM module.

Diagram 3: Submodule consist of BRAMs and several registers designed to use stream data in FAST algorithm


Shift register with 7 memory to wait for 7 clock cycles that read data of line pixels.
Adres=0
Adres=1919
Stream input 12 Bit
Items in the FIFO
Unused FIFO memory
6 tane ck register (Shift)
5 tane ck register (Shift)
4 tane ck register (Shift)
3 tane ck register (Shift)
2 tane ck register (Shift)
1 tane ck register (Shift)
Line 7 pixel 1
Line 6 pixel 1
Line 5 pixel 1
Line 4 pixel 1
Line 3 pixel 1
Line 2 pixel 1
Line 1 pixel 1

The block diagram shared in diagram 3 is the hardware design that makes it possible to use the video signal that comes as a stream of the video signals pixel by pixel starting from left side of first line to right side of last line. Each BRAM module contains 1920 memory cells and delays circuit 1 clock cycle after processing 1920 cells with coming data. In addition, BRAM module embedded Xilinx via Vivado run with the first read then write command.

Initially, all BRAMs are initialized to 0 when the circuit is not running. The output of each BRAM module is connected to the input of the next BRAM module. As shown in the diagram 3, the input of the first BRAM module is connected to the stream video signals that are transferred via AXI-stream module. Once the circuit has started, during the first 1920 clock cycle, 0 will be read at all outputs of each one of the BRAMs because the BRAM module will first read the data it stores inside and then write the new incoming data to that address. After the clock cycle rises above 1920, the address port of the BRAM will return to 0 and resume reading and writing data from there.

After the address returns to 0, the data of the first line of stream video signals will be filled to the first BRAM. Therefore, while the read operator is being carried out, the information of the first line is readable from the first BRAM. When transferring this data to the second BRAM module, the data on the second line of the video signals is now inserted into the first BRAM module. From the second BRAM, the observed data is still zero because, as mentioned earlier, the BRAM modules work on the principle of first read then writes. After reading the zeros stored in the BRAM module in the second line, it starts to store the data of the video signals from the first BRAM module, and these are the data first line of video signals. In this way, the first data is passed to the next BRAM module during each 1920 clock cycle. After the sixth 1920 clock cycle, we start reading data of the first 6 lines from each BRAM module. In other words, in the meantime;

- 6th BRAM module displays 1st line data,
- 5th BRAM module displays 2nd line data,
- 4th BRAM module displays 3rd line data
- 3rd BRAM module displays 4th line data,
- 2nd BRAM module displays 5th line data
- The 1st BRAM module displays the data of the 6th line as output.

As the circuit continues and the clock signal runs, each BRAM module will store the next line data and the flow will not break. In order to prevent delays of 1 clock cycle, amount of 6, 5, 4, 3, 2 and 1 registers are respectively placed after BRAM modules.

Since the 7th BRAM module uses a bit more memory on the FPGA, I have put 7 registers on top of the 1st row of BRAM modules to receive the data in the 7 line instead, and the stream of incoming data flows to it. While the circuit is running, the information of the sixth line of the video signals is read from the 1st BRAM module and the 7th line data is written on that addresses on the 1st BRAM. In addition, the information of the 7th line of image signals is shifted by 7 registers created using the distributed RAM capacity of FPGA as shown in the diagram 3, above the BRAM modules. The reason for using 7 registers in order to receive the latest data as desired is because there will be 1 clock cycle delay in each BRAM module.

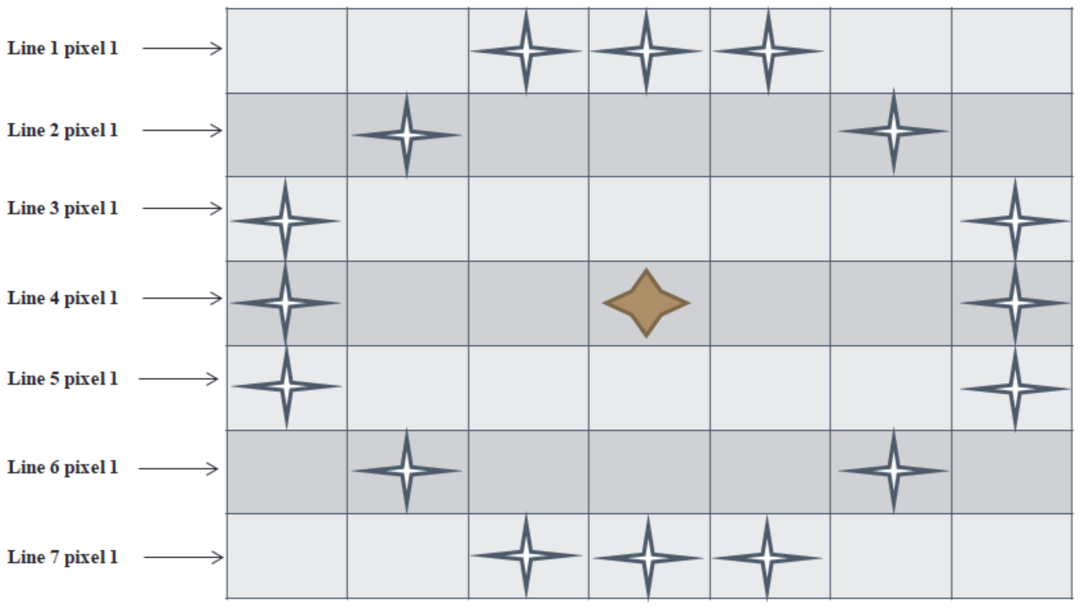
As a result, as soon as the data begins to be read after 1920x6 clock cycle, the information for the first 7 lines can be read by starting from bottom to top respectively, as indicated by the arrows on the right side in diagram 3. As the stream data of the video signals flows, the data indicated at the ends of the arrows will shift horizontally 1 by 1 pixel in each clock cycle, and begin processing in the edge detection module.

9.3. Edge Detector Submodule by Using FAST Algorithm

The FAST algorithm is an easily understandable and functional algorithm. Before I started writing the Verilog HDL code, I was able to get very meaningful results, as can be seen from my work on MATLAB by comparing images in figure 12 and figure 13. Before writing the Verilog code, I first designed the module and the algorithm that would process it. In addition to the ports created for the 7 data from the previous 6 BRAM compact modules, I also created a clk, an enable and a reset port. As the output, I gave out the address of the detected point. In addition, I have manipulated the intensity values of the points that are read and the detected points in the previous module, and wrote them into a text file and compared the results with the results obtained in the MATLAB.

In the meantime, since I wrote the algorithm on MATLAB before, I could easily write it on the Verilog HDL. After writing the Verilog code, I obtained a text file of an image from matlab to try the module since the files in the form of text files can be retrieved and read through some functions via testbench in Verilogh HDL on Vivado. I read the image pixel values of an image as 8 bits on MATLAB and converted these values to 12 bit data. I converted this data into a matrix of 1920x1080 from 2240x1400 pixels. After obtaining a matrix of the desired size, I used the rgb2gray function to obtain the black-and-white version of the colors and then export the data to a text file extension.

Diagram 4: Pixel region to which FAST algorithm is applied



In diagram 4 above, you can see the block diagram schematic of the edge detector module that I create for both my presentation in Roketsan and internship report for university. 16 comparisons are carried out between the register marked at the center and the 16 different registers around it in a 7x7 pixel region. As a result of these comparisons, according to the bright and dark counts of the center, the address can be kept in the some register on the code to manipulate later, or manipulations can be done to this center in the code automatically by putting manipulative algorithms into the module. As a result of these comparisons, if the bright or dark counts of the center pixel data are 9 or more, this address is kept in some memory in verilog HDL and manipulations can be made to this pixel as the desired ways.

I learned that $readmemb (for binary data) and $readmemh (for hexidecimal data) functions are used to read text file data in testbench. We can open a text file with $fopen (“file.path”, ”r”,”w”) and then extract data from that file using the $fgetc or $fscanf functions. For $fopen function, while “r” is used for reading the data from given directory, “w” is used for writing data to be written to a directory.

The test results of the edge detection module created in the Vivado program are shared in figure 18 and figure 19. The original photo is shown first, in figure 18. The second one was obtained by reading the text file containing pixel information of the first photograph on the MATLAB after processing and manipulating detected points by changing intensity values and writing them back to a new textfile via Vivado testbench. Writing both the values of both detected and undetected for corresponding pixels on the textfile is done with $fopen (“file.path”,”w”) function.

Figure 18: Main image that is tested for FAST algorithm in the Verilog testbench

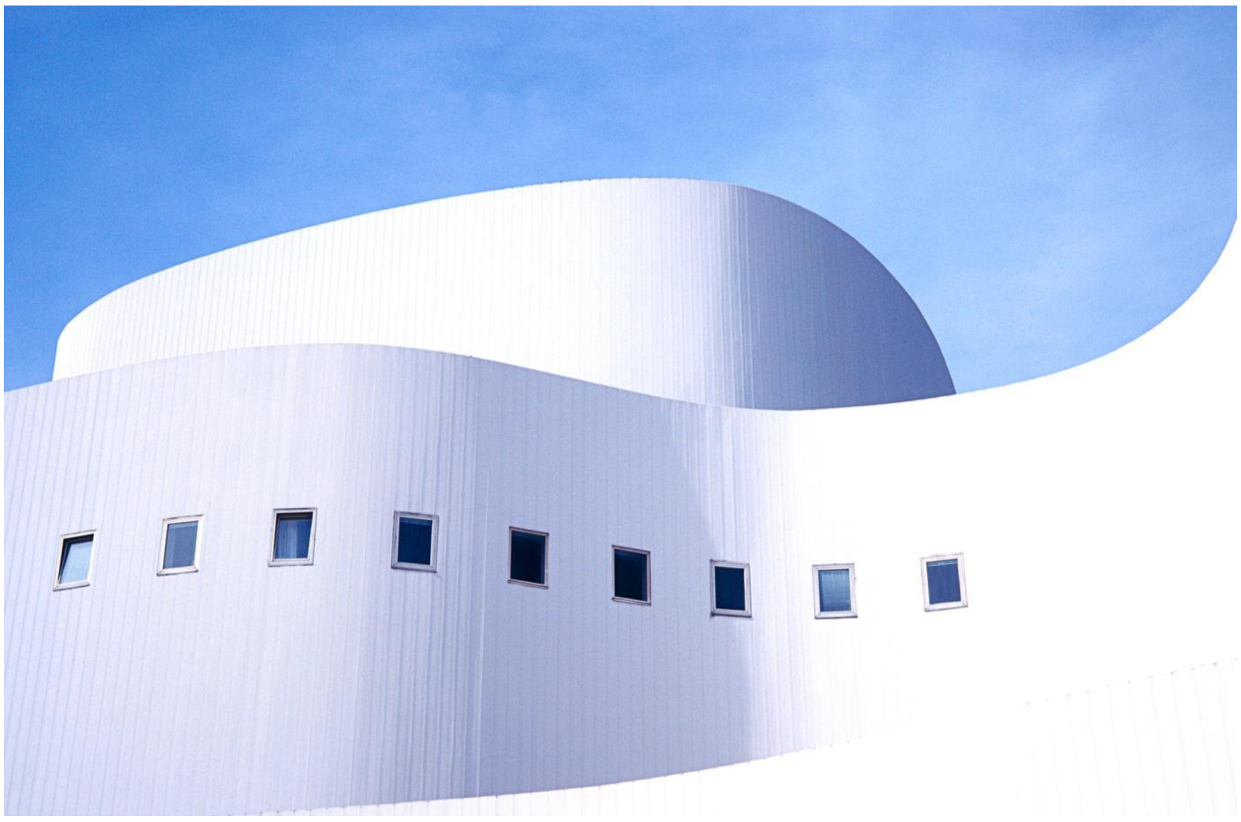
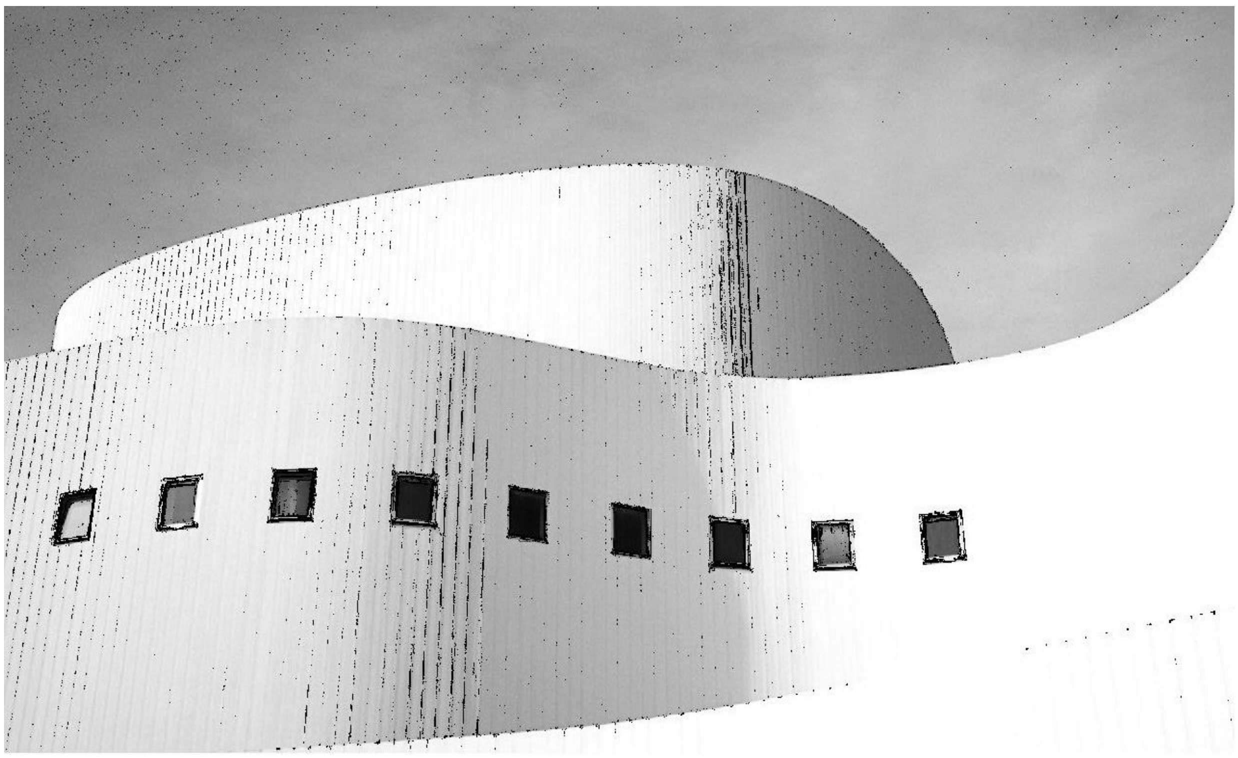


Figure 19: Resultant image after processing in FAST algoritm on verilog testbench



9.4. Vivado HLS (High Level Synthesis) Programming Language

Before starting the project, the engineers at the unit led me to learn Vivado HLS, a programming language produced by Vivado. It runs by using syntax of both C and C++ rules. Since we learned C language before, it was not difficult for me to learn C ++ (Xilinx tutorial, 2018)[6]. The reason they led me to this programming language was that it was a high-level synthesis language compared to the Verilog HDL, and that it could generate Verilog RTL implementation in itself.

After I learned Vivado HLS language that could RTL imlemepentation, I wrote codes that can simple mathematical operations such as addition or multiplication and some simple loops, and together with the engineers in the unit, we examined their results. But we could not relate the simulation results to the work of the hardware language. In other words, we were unable to found an analogy the function between a clock cycle and the for loop function. In addition, since RTL implementations produced by the HLS program are too much complicated to understand and process it through Vivado Verilog HDL.

Unfortunately, my internship duration was not enough to write and test the BRIEF algorithm, so I only learned theoretically about corner matching. In addition, I learned the existence of some AXI systems (AXI stream, AXI 4lite, so on) that enables collabratedly processing between the modules that is created for the project and have some missions such as image process and Zynq processor. I can also tell that I have a bit knowladge about the Xilinx, its productions and usages.

10. CONCLUSION

The opportunity of adding many innovations and teachings is presented to me during my 20-work day summer internship. Some of these innovations would contribute to my academic and engineering life, while others were gains in bureaucratic functioning and confidentiality in institutions such as Roketsan. Since Roketsan, which produces rockets and missiles at an international level, has an importance in terms of defense systems for Turkey Republic government, privacy is of paramount importance in many of the projects and designs. In order to ensure this confidentiality, some critical precautions have been taken such that information sharing about Roketsan is prohibited outside of the campus or confidentiality agreement for interns. I had the chance to create a network by meeting with lots of Roketsan staff that are different bureaucratic levels, and observe the Roketsan's operation in different fields.

If I can talk about the benefits it offers to me technically, I have tried to process the image features for a stream video signal by using FPGA and writing codes on Verilog HDL which I got the opportunity to get acquainted with this year's digital laboratory project. As I mentioned in detail above in body part, I tried to use the FAST algorithm for edge detection, then the BRIEF descriptor and Corner Matching methods.

There are enough opportunities within the institution to advance the projects in terms of detail R&D works. In the integration unit, for example, there were physical systems for testing rocket headers or guidance systems, as well as integrated computer interfaces. However, since the institution is an important institution for the country in terms of confidentialy, I couldn't see as much automation and integration as I expected. In other words, technicians and engineers were working in many fields of projects and productions in general instead of automated systems. If we talk about the research and development approaches of the engineers in the sub-unit where I do my internship, I can say that a detailed literature search is conducted for the missile products that are planned to be developed in various projects. I have read 3 different articles for the edge detection, brief descriptor and corner matching code design. After I read them, I acquired knowledge about the advantages and disadvantages, and I have also learned why we choose the FAST algorithm instead of using other algorithms such as Canny or Sobel edge detection methods. I think that such detailed research is carried out due to the fact that the production of the institution is so important that it will determine the fate and futures of states. In addition to such extensive literature research, I can say that they were really meticulous in the tests. In the units where quality control and testing have been conducted, I have personally witnessed that they keep the margin of errors seriously narrow in order to ensure that the products for both rockets an missiles results in accordance with their objectives in their usage areas.

My internship at Roketsan was my first summer internship. So I cannot make a technical comparison at this point. However, in my second internship, I do not think that I will be under such a formal and confidential framework since I will be an intern within Dalgakıran which is one of the industrial company that offer lots of compressor machines to all over the world. I can recommend Roketsan Institution to the students who will do internship in order to get to know defense industry. There are human resources that support students and help them to able to acquire the knowladge about the works and projects. Furthermore, technically, as I mentioned above, I can recommend this institution to other students from any techincal deparment at our university since the institution serve in many fields. The institution offers the opportunity to work in two different campuses, one in Lalahan and the other in Elmadağ. It may be far from the city, but service facilities prevent it from turning into a problem.

As a result, I have completed my EE400 internship at Roketsan. I have completed this summer internship program with lots of benefits in terms of both the understanding institutional mechanism and the technical advances. Although I cannot share all of my internship activities due to the confidentiality contrat, some studies have been shared in the appendix.

11. REFERENCES

[5]Jingjin Huang, G. Z. (2018, March 28). A new FPGA architecture of FAST and BRIEF algorithm for on-board corner detection and matching. *MDPI* , pp. 1014- 0/17.

[4] Michał Fularz, M. K. (2015, October 14). A High-Performance FPGA-Based Image Feature Detector and Matcher Based on the FAST and BRIEF Algorithms. *International Journal of Advanced Robotic Systems* .

[3] MSI. (2019, May 22). *Roketsan, YATAĞAN® ile Tek Piyadeye ve Mini İnsansız Araçlara, Hassas Güdümlü Füze Kabiliyeti Kazandırıyor*. Retrieved august 18, 2019, from MSI: http://www.milscint.com/tr/roketsan-yatagan-ile-tek-piyadeye-ve-mini-insansiz-araclara-hassas-gudumlu-fuze-kabiliyeti-kazandiriyor/

[2] Roketsan. (n.d.). *Roketsan > Ürünler & Hizmetler > Hassas Güdümlü Füzeler > CİRİT 2.75" Lazer Güdümlü Füze*. Retrieved august 18, 2019, from Roketsan.com.tr: http://www.roketsan.com.tr/urunler-hizmetler/hassas-gudumlu-fuzeler/cirit-275-lazer-gudumlu-fuze/

[1] Roketsan. (n.d.). *Roketsan, Kurumsal, Hakımızda*. Retrieved August 18, 2019, from ROKETSAN A.Ş.: http://www.roketsan.com.tr/kurumsal/hakkimizda/

[7] Xilinx forum. (2017, 11 17). *Difference between BRAM, DRAM and DMA*. Retrieved august 18, 2019, from Forums Xilinx: https://forums.xilinx.com/t5/General-Technical-Discussion/Difference-between-BRAM-DRAM-and-DMA/td-p/809324

[6] Xilinx tutorial. (2018, June 6). *High-Level Synthesis*. Retrieved August 18, 2019, from Xilinx Documentation: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug871-vivado-high-level-synthesis-tutorial.pdf

12. APPENDICES

I. FAST ALGORITHM ON MATLAB

```
red=255;
brighter=0;
darker=0;
same=0;
X=size(1080,1920);
K=imread('image.jpg');
%Z=im2uint16(K);

L=rgb2gray(K);

J=imresize(L,[1080,1920]);

for v=4:1076
for h=4:1916
        brighter=0;
        darker=0;
        same=0;
if J(v,h)>J(v-3,h)+10
                brighter=brighter+1;
elseif J(v,h)<J(v-3,h)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v-3,h+1)+10
                brighter=brighter+1;
elseif J(v,h)<J(v-3,h+1)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v-2,h+2)+10
                brighter=brighter+1;
elseif J(v,h)<J(v-2,h+2)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v-1,h+3)+10
                brighter=brighter+1;
elseif J(v,h)<J(v-1,h+3)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v,h+3)+10
                brighter=brighter+1;
elseif J(v,h)<J(v,h+3)-10
                darker=darker+1;
else
            same=same+1;
```

```
end

if J(v,h)>J(v+1,h+3)+10
                brighter=brighter+1;
elseif J(v,h)<J(v+1,h+3)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v+2,h+2)+10
                brighter=brighter+1;
elseif J(v,h)<J(v+2,h+2)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v+3,h+1)+10
                brighter=brighter+1;
elseif  J(v,h)<J(v+3,h+1)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v+3,h)+10
                brighter=brighter+1;
elseif J(v,h)<J(v+3,h)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v+3,h-1)+10
                brighter=brighter+1;
elseif J(v,h)<J(v+3,h-1)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v+2,h-2)+10
                brighter=brighter+1;
elseif J(v,h)<J(v+2,h-2)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v+1,h-3)+10
                brighter=brighter+1;
elseif J(v,h)<J(v+1,h-3)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v,h-3)+10
                brighter=brighter+1;
```

```
elseif J(v,h)<J(v,h-3)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v-1,h-3)+10
                brighter=brighter+1;
elseif J(v,h)<J(v-1,h-3)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v-2,h-2)+10
                brighter=brighter+1;
elseif J(v,h)<J(v-2,h-2)-10
                darker=darker+1;
else
            same=same+1;
end

if J(v,h)>J(v-3,h-1)+10
                brighter=brighter+1;
elseif J(v,h)<J(v-3,h-1)-10
                darker=darker+1;
else
            same=same+1;
end


if (brighter>=9)||(darker>=9)
            J(v,h)=0;
            X(v,h)=255;
            brighter=0;
            darker=0;
            same=0;

else
            brighter=0;
            darker=0;
            same=0;
            X(v,h)=0;
end
end
end
%C=imfuse(J,X);
%imshow(C);
%hold on
imshow(J);
```

II. VERILOG HDL BRAM MODULE ON VIVADO

```
module data_shifter(
        input clk,
        input enable,
        input rstb,
        input wire [11:0] inp_pixel,
            output reg [11:0] last_data_7,
            output reg [11:0] line_1_6_out,
            output reg [11:0] line_2_5_out,
            output reg [11:0] line_3_4_out,
            output reg [11:0] line_4_3_out,
            output reg [11:0] line_5_2_out,
            output reg [11:0] line_6_1_out



);
    wire [11:0] line_1_wire;
    wire [11:0] line_2_wire;
    wire [11:0] line_3_wire;
    wire [11:0] line_4_wire;
    wire [11:0] line_5_wire;
    wire [11:0] line_6_wire;

    reg [11:0] line_1_1_out;
    reg [11:0] line_1_2_out;
    reg [11:0] line_1_3_out;
    reg [11:0] line_1_4_out;
    reg [11:0] line_1_5_out;
    reg [11:0] line_2_1_out;
    reg [11:0] line_2_2_out;
    reg [11:0] line_2_3_out;
    reg [11:0] line_2_4_out;
    reg [11:0] line_3_1_out;
    reg [11:0] line_3_2_out;
    reg [11:0] line_3_3_out;
    reg [11:0] line_4_1_out;
    reg [11:0] line_4_2_out;
    reg [11:0] line_5_1_out;
    reg [11:0] last_data_1;
    reg [11:0] last_data_2;
    reg [11:0] last_data_3;
    reg [11:0] last_data_4;
    reg [11:0] last_data_5;
    reg [11:0] last_data_6;
    reg [10:0] address_counter;

     initial
        begin
           address_counter=0;        line_3_3_out=0;
           line_1_1_out=0;          line_3_4_out=0;
           line_1_2_out=0;          line_4_1_out=0;
           line_1_3_out=0;          line_4_2_out=0;
           line_1_4_out=0;          line_4_3_out=0;
           line_1_5_out=0;          line_5_1_out=0;
           line_1_6_out=0;          line_5_2_out=0;
           line_2_1_out=0;          line_6_1_out=0;
           line_2_2_out=0;          last_data_1=0;
           line_2_3_out=0;          last_data_2=0;
           line_2_4_out=0;          last_data_3=0;
           line_2_5_out=0;          last_data_4=0;
           line_3_1_out=0;          last_data_5=0;
           line_3_2_out=0;          last_data_6=0;
```

```
                                last_data_7=0;

            end

        always @ (posedge clk )
          begin
            if(!rstb)  //&&!(address_counter==0)
              begin
                line_1_1_out<=line_1_wire;                  line_2_2_out<=line_2_1_out;
                line_2_1_out<=line_2_wire;                  line_2_3_out<=line_2_2_out;
                line_3_1_out<=line_3_wire;                  line_2_4_out<=line_2_3_out;
                line_4_1_out<=line_4_wire;                  line_2_5_out<=line_2_4_out;
                line_5_1_out<=line_5_wire;
                line_6_1_out<=line_6_wire;                  line_3_2_out<=line_3_1_out;
                                            line_3_3_out<=line_3_2_out;
                line_1_2_out<=line_1_1_out;                  line_3_4_out<=line_3_3_out;
                line_1_3_out<=line_1_2_out;
                line_1_4_out<=line_1_3_out;                  line_4_2_out<=line_4_1_out;
                line_1_5_out<=line_1_4_out;                  line_4_3_out<=line_4_2_out;
                line_1_6_out<=line_1_5_out;                  line_5_2_out<=line_5_1_out;

                last_data_1<=inp_pixel;
                last_data_2<=last_data_1;
                last_data_3<=last_data_2;
                last_data_4<=last_data_3;
                last_data_5<=last_data_4;
                last_data_6<=last_data_5;
                last_data_7<=last_data_6;

              end

          end


        always @ (negedge clk)
          begin
            if((address_counter<1919)&&(!rstb))
              begin
              address_counter<=address_counter+1;
              end
else
             begin
              address_counter<=0;
//                  if(address_counter==1919)
//                    begin
//                    count_of_1920<=count_of_1920+1;
//                    end
                end
          end

        blk_mem_gen_0 line_1 (
          .clka(clk),    // input wire clka
          .ena(enable),      // input wire ena
          .wea(enable),      // input wire [0 : 0] wea
          .addra(address_counter),  // input wire [10 : 0] addra
          .dina(inp_pixel),    // input wire [11 : 0] dina
          .douta(line_1_wire)  // output wire [11 : 0] douta
```

```
    );
     blk_mem_gen_1 line_2 (
       .clka(clk),    // input wire clka
       .ena(enable),      // input wire ena
       .wea(enable),      // input wire [0 : 0] wea
       .addra(address_counter),  // input wire [10 : 0] addra
       .dina(line_1_wire),    // input wire [11 : 0] dina
       .douta(line_2_wire)  // output wire [11 : 0] douta
     );
     blk_mem_gen_2 line_3 (
       .clka(clk),    // input wire clka
       .ena(enable),      // input wire ena
       .wea(enable),      // input wire [0 : 0] wea
       .addra(address_counter),  // input wire [10 : 0] addra
       .dina(line_2_wire),    // input wire [11 : 0] dina
       .douta(line_3_wire)  // output wire [11 : 0] douta
     );
     blk_mem_gen_3 line_4 (
       .clka(clk),    // input wire clka
       .ena(enable),      // input wire ena
       .wea(enable),      // input wire [0 : 0] wea
       .addra(address_counter),  // input wire [10 : 0] addra
       .dina(line_3_wire),    // input wire [11 : 0] dina
       .douta(line_4_wire)  // output wire [11 : 0] douta
     );
     blk_mem_gen_4 line_5 (
       .clka(clk),    // input wire clka
       .ena(enable),      // input wire ena
       .wea(enable),      // input wire [0 : 0] wea
       .addra(address_counter),  // input wire [10 : 0] addra
       .dina(line_4_wire),    // input wire [11 : 0] dina
       .douta(line_5_wire)  // output wire [11 : 0] douta
     );
     blk_mem_gen_5 line_6 (
       .clka(clk),    // input wire clka
       .ena(enable),      // input wire ena
       .wea(enable),      // input wire [0 : 0] wea
       .addra(address_counter),  // input wire [10 : 0] addra
       .dina(line_5_wire),    // input wire [11 : 0] dina
       .douta(line_6_wire)  // output wire [11 : 0] douta
     );


endmodule
```

III. VERILOG HDL FAST ALGORITHM MODULE ON VIVADO

```
module data_receive_and_detection (
                input clk,
                input enable,
                input rstb,

                output reg [4:0] bright,
                output reg [4:0] dark,
                output reg [11:0] line_4_3,

                input wire [11:0] inp_pixel1,
                input wire [11:0] inp_pixel2,
                input wire [11:0] inp_pixel3,
                input wire [11:0] inp_pixel4,
                input wire [11:0] inp_pixel5,
                input wire [11:0] inp_pixel6,
                input wire [11:0] inp_pixel7

            );
      parameter threshold=10;
      reg [11:0] line_1_0;        reg [11:0] line_2_0;      reg [11:0] line_3_0;      reg [11:0] line_4_0;
      reg [11:0] line_1_1;        reg [11:0] line_2_1;      reg [11:0] line_3_1;      reg [11:0] line_4_1;
      reg [11:0] line_1_2;        reg [11:0] line_2_2;      reg [11:0] line_3_2;      reg [11:0] line_4_2;
      reg [11:0] line_1_3;        reg [11:0] line_2_3;      reg [11:0] line_3_3;      reg [11:0] line_4_4;
      reg [11:0] line_1_4;        reg [11:0] line_2_4;      reg [11:0] line_3_4;      reg [11:0] line_4_5;
      reg [11:0] line_1_5;        reg [11:0] line_2_5;      reg [11:0] line_3_5;      reg [11:0] line_4_6;
      reg [11:0] line_1_6;        reg [11:0] line_2_6;      reg [11:0] line_3_6;      reg [11:0] line_4_7;
      reg [11:0] line_1_7;        reg [11:0] line_2_7;      reg [11:0] line_3_7;

      reg [11:0] line_5_0;        reg [11:0] line_6_0;      reg [11:0] line_7_0;
      reg [11:0] line_5_1;        reg [11:0] line_6_1;      reg [11:0] line_7_1;
      reg [11:0] line_5_2;        reg [11:0] line_6_2;      reg [11:0] line_7_2;
      reg [11:0] line_5_3;        reg [11:0] line_6_3;      reg [11:0] line_7_3;
      reg [11:0] line_5_4;        reg [11:0] line_6_4;      reg [11:0] line_7_4;
      reg [11:0] line_5_5;        reg [11:0] line_6_5;      reg [11:0] line_7_5;
      reg [11:0] line_5_6;        reg [11:0] line_6_6;      reg [11:0] line_7_6;
      reg [11:0] line_5_7;        reg [11:0] line_6_7;      reg [11:0] line_7_7;

      reg comp_1_d;       reg comp_1_b;
      reg comp_2_d;       reg comp_2_b;
      reg comp_3_d;       reg comp_3_b;
      reg comp_4_d;       reg comp_4_b;
      reg comp_5_d;       reg comp_5_b;
      reg comp_6_d;       reg comp_6_b;
      reg comp_7_d;       reg comp_7_b;
      reg comp_8_d;       reg comp_8_b;
      reg comp_9_d;       reg comp_9_b;
      reg comp_10_d;       reg comp_10_b;
      reg comp_11_d;       reg comp_11_b;
      reg comp_12_d;       reg comp_12_b;
      reg comp_13_d;       reg comp_13_b;
      reg comp_14_d;       reg comp_14_b;
      reg comp_15_d;       reg comp_15_b;
      reg comp_16_d;       reg comp_16_b;

               // output reg [3:0] dark1,
               // output reg [3:0] dark2,
               // output reg [3:0] bright1,
               // output reg [3:0] bright2,

initial
     begin
     comp_1_d=0;    comp_1_b=0;    line_1_0=0;  line_2_0=0;  line_3_0=0;  line_4_0=0;  line_5_0=0;
line_6_0=0;  line_7_0=0;
     comp_2_d=0;    comp_2_b=0;    line_1_1=0;  line_2_1=0;  line_3_1=0;  line_4_1=0;  line_5_1=0;
line_6_1=0;  line_7_1=0;
     comp_3_d=0;    comp_3_b=0;    line_1_2=0;  line_2_2=0;  line_3_2=0;  line_4_2=0;  line_5_2=0;
line_6_2=0;  line_7_2=0;
```

```
    comp_4_d=0;    comp_4_b=0;    line_1_3=0;  line_2_3=0;  line_3_3=0;  line_4_3=0;  line_5_3=0;
line_6_3=0;  line_7_3=0;
    comp_5_d=0;    comp_5_b=0;    line_1_4=0;  line_2_4=0;  line_3_4=0;  line_4_4=0;  line_5_4=0;
line_6_4=0;  line_7_4=0;
    comp_6_d=0;    comp_6_b=0;    line_1_5=0;  line_2_5=0;  line_3_5=0;  line_4_5=0;  line_5_5=0;
line_6_5=0;  line_7_5=0;
    comp_7_d=0;    comp_7_b=0;    line_1_6=0;  line_2_6=0;  line_3_6=0;  line_4_6=0;  line_5_6=0;
line_6_6=0;  line_7_6=0;
    comp_8_d=0;    comp_8_b=0;    line_1_7=0;  line_2_7=0;  line_3_7=0;  line_4_7=0;  line_5_7=0;
line_6_7=0;  line_7_7=0;
    comp_9_d=0;    comp_9_b=0;
    comp_10_d=0;   comp_10_b=0;
    comp_11_d=0;   comp_11_b=0;
    comp_12_d=0;   comp_12_b=0;
    comp_13_d=0;   comp_13_b=0;
    comp_14_d=0;   comp_14_b=0;
    comp_15_d=0;   comp_15_b=0;
    comp_16_d=0;   comp_16_b=0;
    bright=0;
    dark=0;
    end

     always @(posedge clk)
       begin
         line_1_0<=inp_pixel1;   line_1_1<=line_1_0;   line_1_2<=line_1_1;    line_1_3<=line_1_2;
         line_2_0<=inp_pixel2;   line_2_1<=line_2_0;   line_2_2<=line_2_1;    line_2_3<=line_2_2;
         line_3_0<=inp_pixel3;   line_3_1<=line_3_0;   line_3_2<=line_3_1;    line_3_3<=line_3_2;
         line_4_0<=inp_pixel4;   line_4_1<=line_4_0;   line_4_2<=line_4_1;    //line_4_3<=line_4_2;
         line_5_0<=inp_pixel5;   line_5_1<=line_5_0;   line_5_2<=line_5_1;    line_5_3<=line_5_2;
         line_6_0<=inp_pixel6;   line_6_1<=line_6_0;   line_6_2<=line_6_1;    line_6_3<=line_6_2;
         line_7_0<=inp_pixel7;   line_7_1<=line_7_0;   line_7_2<=line_7_1;    line_7_3<=line_7_2;

         line_1_4<=line_1_3;     line_1_5<=line_1_4;   line_1_6<=line_1_5;    line_1_7<=line_1_6;
         line_2_4<=line_2_3;     line_2_5<=line_2_4;   line_2_6<=line_2_5;    line_2_7<=line_2_6;
         line_3_4<=line_3_3;     line_3_5<=line_3_4;   line_3_6<=line_3_5;    line_3_7<=line_3_6;
         line_4_4<=line_4_3;     line_4_5<=line_4_4;   line_4_6<=line_4_5;    line_4_7<=line_4_6;
         line_5_4<=line_5_3;     line_5_5<=line_5_4;   line_5_6<=line_5_5;    line_5_7<=line_5_6;
         line_6_4<=line_6_3;     line_6_5<=line_6_4;   line_6_6<=line_6_5;    line_6_7<=line_6_6;
         line_7_4<=line_7_3;     line_7_5<=line_7_4;   line_7_6<=line_7_5;    line_7_7<=line_7_6;

       end

     always @(posedge clk) begin // comparison  begin
             if(line_4_3+threshold<line_1_3) //1
                comp_1_d<=1; end

     always @(posedge clk)  begin
             if (line_4_3>line_1_3+threshold)
                comp_1_b<=1; end

     always @(posedge clk)  begin
             if (line_4_3+threshold<line_1_4) //2
                comp_2_d<=1;  end

     always @(posedge clk) begin
              if (line_4_3>line_1_4+threshold)
                comp_2_b<=1;    end
```

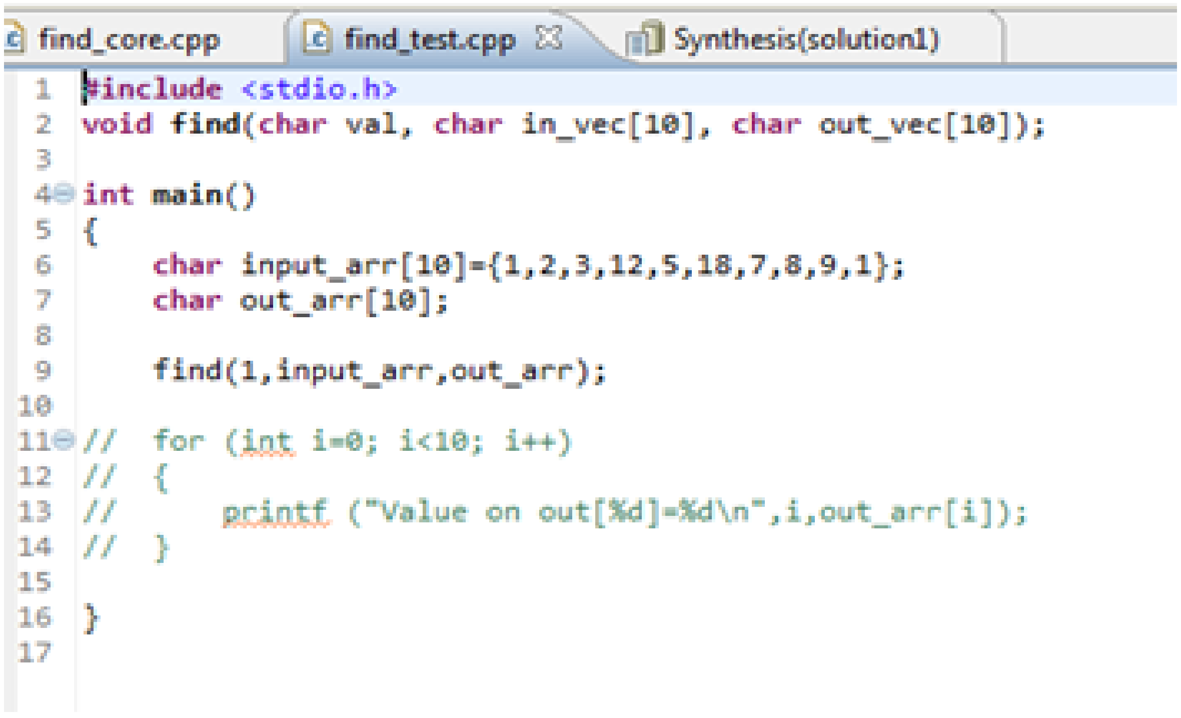
```
always @(posedge clk) begin
              if (line_4_3+threshold<line_2_5) //3
                comp_3_d<=1; end

always @(posedge clk) begin
              if (line_4_3>line_2_5+threshold)
                 comp_3_b<=1; end

always @(posedge clk) begin
              if (line_4_3+threshold<line_3_6) //4
                 comp_4_d<=1; end

always @(posedge clk)  begin
              if (line_4_3>line_3_6+threshold)
                 comp_4_b<=1; end

always @(posedge clk)   begin
              if (line_4_3+threshold<line_4_6)  // 5
                 comp_5_d<=1;  end

always @(posedge clk)  begin
              if (line_4_3>line_4_6+threshold)
                 comp_5_b<=1;  end

always @(posedge clk)  begin
               if (line_4_3+threshold<line_5_6) //6
                 comp_6_d<=1;  end

always @(posedge clk) begin
               if (line_4_3>line_5_6+threshold)
                 comp_6_b<=1;  end

always @(posedge clk)  begin
               if (line_4_3+threshold<line_6_5) //7
                 comp_7_d<=1;  end

always @(posedge clk)  begin
              if (line_4_3>line_6_5+threshold)
                 comp_7_b<=1; end

always @(posedge clk)  begin
              if (line_4_3+threshold<line_7_4) //8
                 comp_8_d<=1;  end

 always @(posedge clk) begin
           if (line_4_3>line_7_4+threshold)
          comp_8_b<=1; end

 always @(posedge clk) begin
           if (line_4_3+threshold<line_7_3) //9
          comp_9_d<=1; end

 always @(posedge clk) begin
           if (line_4_3>line_7_3+threshold)
          comp_9_b<=1; end
```

```
        always @(posedge clk) begin
                    if (line_4_3+threshold<line_7_2) //10
                   comp_10_d<=1;  end

        always @(posedge clk) begin
                     if (line_4_3>line_7_2+threshold)
                   comp_10_b<=1; end

        always @(posedge clk) begin
                    if (line_4_3+threshold<line_6_1) //11
                   comp_11_d<=1; end

        always @(posedge clk) begin
                    if (line_4_3>line_6_1+threshold)
                   comp_11_b<=1; end

        always @(posedge clk) begin
                    if (line_4_3+threshold<line_5_0) //12
                      comp_12_d<=1;end

        always @(posedge clk) begin
                      if (line_4_3>line_5_0+threshold)
                     comp_12_b<=1; end

        always @(posedge clk) begin
                      if (line_4_3+threshold<line_4_0) //13
                     comp_13_d<=1; end

        always @(posedge clk) begin
                      if (line_4_3>line_4_0+threshold)
                     comp_13_b<=1; end

        always @(posedge clk) begin
                       if (line_4_3+threshold<line_3_0) //14
                     comp_14_d<=1; end

        always @(posedge clk) begin
                      if (line_4_3>line_3_0+threshold)
                     comp_14_b<=1; end

        always @(posedge clk) begin
                      if (line_4_3+threshold<line_2_1) //15
                     comp_15_d<=1; end

        always @(posedge clk) begin
                      if (line_4_3>line_2_1+threshold)
                     comp_15_b<=1;end

        always @(posedge clk) begin
                      if (line_4_3+threshold<line_1_2) //16
                     comp_16_d<=1;    end

        always @(posedge clk) begin
                      if (line_4_3>line_1_2+threshold)
                         comp_16_b<=1;
//                       bright<=0;
//                       dark<=0;
```

```
                    end

        always @(negedge clk)  // count for both bright and dark
          begin
            bright<=comp_1_b+comp_2_b+comp_3_b+comp_4_b+comp_5_b+comp_6_b+comp_7_b+
comp_8_b+comp_9_b+comp_10_b+comp_11_b+comp_12_b+comp_13_b+comp_14_b+comp_15_b+comp_16
_b;

//bright2<=comp_9_b+comp_10_b+comp_11_b+comp_12_b+comp_13_b+comp_14_b+comp_15_b+comp_16
_b;
           // bright<=bright1+bright2;


dark<=(comp_1_d+comp_2_d+comp_3_d+comp_4_d+comp_5_d+comp_6_d+comp_7_d+comp_8_d)+(comp_
9_d+comp_10_d+comp_11_d+comp_12_d+comp_13_d+comp_14_d+comp_15_d+comp_16_d);

//dark2<=(comp_9_d+comp_10_d+comp_11_d+comp_12_d+comp_13_d+comp_14_d+comp_15_d+comp_16_
d);
           //dark<=dark1+dark2;
          end


        always @ (negedge clk)
          begin
             if((bright>=9)||(dark>=9))
               begin
                line_4_3<=4095;
                 if(bright>=9)
                 bright<=0;
else if(dark>=9)
                 dark<=0;
               end
else
                line_4_3<=line_4_2;
          end

        always @ (negedge clk)
          begin
            comp_1_d<=0;    comp_1_b<=0;
            comp_2_d<=0;    comp_2_b<=0;
            comp_3_d<=0;    comp_3_b<=0;
            comp_4_d<=0;    comp_4_b<=0;
            comp_5_d<=0;    comp_5_b<=0;
            comp_6_d<=0;    comp_6_b<=0;
            comp_7_d<=0;    comp_7_b<=0;
            comp_8_d<=0;    comp_8_b<=0;
            comp_9_d<=0;    comp_9_b<=0;
            comp_10_d<=0;   comp_10_b<=0;
            comp_11_d<=0;   comp_11_b<=0;
            comp_12_d<=0;   comp_12_b<=0;
            comp_13_d<=0;   comp_13_b<=0;
            comp_14_d<=0;   comp_14_b<=0;
            comp_15_d<=0;   comp_15_b<=0;
            comp_16_d<=0;   comp_16_b<=0;


          end
```

```
//	data_shifter data_shifter(
//	.clk(clk),
//	.enable(enable),
//	.rstb(rstb),
//	.inp_pixel(inp_pixel),
//	.line_1_6_out(line_6[0]),
//	.line_2_5_out(line_5[0]),
//	.line_3_4_out(line_4[0]),
//	.line_4_3_out(line_3[0]),
//	.line_5_2_out(line_2[0]),
//	.line_6_1_out(line_1[0]),
//	.last_data_7(line_7[0])
//	);

endmodule
```

IV. HLS CODE AND ITS' VERILOG RTL IMPLEMENTATION RESULTS



find_core.cpp | find_test.cpp | Synthesis(solution1)

```
#include <stdio.h>
void find(char val, char in_vec[10], char out_vec[10]);

int main()
{
    char input_arr[10]={1,2,3,12,5,18,7,8,9,1};
    char out_arr[10];

    find(1,input_arr,out_arr);

//  for (int i=0; i<10; i++)
//  {
//      printf ("Value on out[%d]=%d\n",i,out_arr[i]);
//  }

}
```

```
// ==============================================================
// RTL generated by Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC
// Version: 2018.3
// Copyright (C) 1986-2018 Xilinx, Inc. All Rights Reserved.
//
// ===========================================================

`timescale 1 ns / 1 ps

(*
CORE_GENERATION_INFO="find,hls_ip_2018_3,{HLS_INPUT_TYPE=cxx,HLS_INPUT_FLOAT=0,HLS_INPUT_FIXED
=0,HLS_INPUT_PART=xc7z045ffg900-
2,HLS_INPUT_CLOCK=10.000000,HLS_INPUT_ARCH=others,HLS_SYN_CLOCK=4.824000,HLS_SYN_LAT=21,HLS_S
YN_TPT=none,HLS_SYN_MEM=0,HLS_SYN_DSP=0,HLS_SYN_FF=15,HLS_SYN_LUT=18,HLS_VERSION=2018_3}"
*)

module find (
        ap_clk,
        ap_rst,
        ap_start,
        ap_done,
        ap_idle,
        ap_ready,
        val_r,
in_vec_address0,
in_vec_ce0,
in_vec_q0,
        out_vec_address0,
        out_vec_ce0,
        out_vec_we0,
        out_vec_d0
);

parameter    ap_ST_fsm_state1 = 3'd1;
parameter    ap_ST_fsm_state2 = 3'd2;
parameter    ap_ST_fsm_state3 = 3'd4;

input   ap_clk;
input   ap_rst;
input   ap_start;
output   ap_done;
output   ap_idle;
output   ap_ready;
input  [7:0] val_r;
output  [3:0] in_vec_address0;
output   in_vec_ce0;
input  [7:0] in_vec_q0;
output  [3:0] out_vec_address0;
output   out_vec_ce0;
output   out_vec_we0;
output  [7:0] out_vec_d0;

reg ap_done;
reg ap_idle;
reg ap_ready;
reg in_vec_ce0;
reg out_vec_ce0;
reg out_vec_we0;

(* fsm_encoding = "none" *) reg   [2:0] ap_CS_fsm;
wire    ap_CS_fsm_state1;
wire   [3:0] idx_1_fu_75_p2;
reg   [3:0] idx_1_reg_104;
wire    ap_CS_fsm_state2;
wire   [63:0] tmp_fu_81_p1;
reg   [63:0] tmp_reg_109;
wire   [0:0] exitcond_fu_69_p2;
reg   [3:0] idx_reg_58;
wire    ap_CS_fsm_state3;
wire   [0:0] tmp_1_fu_86_p2;
reg   [2:0] ap_NS_fsm;

// power-on initialization
initial begin
#0 ap_CS_fsm = 3'd1;
end

always @ (posedge ap_clk) begin
   if (ap_rst == 1'b1) begin
      ap_CS_fsm <= ap_ST_fsm_state1;
   end else begin
      ap_CS_fsm <= ap_NS_fsm;
   end
end

always @ (posedge ap_clk) begin
   if ((1'b1 == ap_CS_fsm_state3)) begin
      idx_reg_58 <= idx_1_reg_104;
   end else if (((ap_start == 1'b1) & (1'b1 == ap_CS_fsm_state1))) begin
      idx_reg_58 <= 4'd0;
   end
end

always @ (posedge ap_clk) begin
   if ((1'b1 == ap_CS_fsm_state2)) begin
      idx_1_reg_104 <= idx_1_fu_75_p2;
   end
end

always @ (posedge ap_clk) begin
   if (((exitcond_fu_69_p2 == 1'd0) & (1'b1 == ap_CS_fsm_state2))) begin
      tmp_reg_109[3 : 0] <= tmp_fu_81_p1[3 : 0];
   end
end

always @ (*) begin
   if (((exitcond_fu_69_p2 == 1'd1) & (1'b1 == ap_CS_fsm_state2))) begin
      ap_done = 1'b1;
   end else begin
      ap_done = 1'b0;
   end
end

always @ (*) begin
   if (((ap_start == 1'b0) & (1'b1 == ap_CS_fsm_state1))) begin
      ap_idle = 1'b1;
   end else begin
      ap_idle = 1'b0;
   end
end

always @ (*) begin
   if (((exitcond_fu_69_p2 == 1'd1) & (1'b1 == ap_CS_fsm_state2))) begin
      ap_ready = 1'b1;
   end else begin
      ap_ready = 1'b0;
```

```
    end
end

always @ (*) begin
    if ((1'b1 == ap_CS_fsm_state2)) begin
in_vec_ce0 = 1'b1;
    end else begin
in_vec_ce0 = 1'b0;
    end
end

always @ (*) begin
    if ((1'b1 == ap_CS_fsm_state3)) begin
        out_vec_ce0 = 1'b1;
    end else begin
        out_vec_ce0 = 1'b0;
    end
end

always @ (*) begin
    if ((1'b1 == ap_CS_fsm_state3)) begin
        out_vec_we0 = 1'b1;
    end else begin
        out_vec_we0 = 1'b0;
    end
end

always @ (*) begin
    case (ap_CS_fsm)
        ap_ST_fsm_state1 : begin
            if (((ap_start == 1'b1) & (1'b1 == ap_CS_fsm_state1))) begin
                ap_NS_fsm = ap_ST_fsm_state2;
            end else begin
                ap_NS_fsm = ap_ST_fsm_state1;
            end
        end
        ap_ST_fsm_state2 : begin
            if (((exitcond_fu_69_p2 == 1'd1) & (1'b1 == ap_CS_fsm_state2))) begin
                ap_NS_fsm = ap_ST_fsm_state1;
            end else begin
                ap_NS_fsm = ap_ST_fsm_state3;
            end
        end
        ap_ST_fsm_state3 : begin
            ap_NS_fsm = ap_ST_fsm_state2;
        end
default : begin
            ap_NS_fsm = 'bx;
        end
    endcase
end

assign ap_CS_fsm_state1 = ap_CS_fsm[32'd0];

assign ap_CS_fsm_state2 = ap_CS_fsm[32'd1];

assign ap_CS_fsm_state3 = ap_CS_fsm[32'd2];
assign exitcond_fu_69_p2 = ((idx_reg_58 == 4'd10) ? 1'b1 : 1'b0);

assign idx_1_fu_75_p2 = (idx_reg_58 + 4'd1);

assign in_vec_address0 = tmp_fu_81_p1;

assign out_vec_address0 = tmp_reg_109;

assign out_vec_d0 = tmp_1_fu_86_p2;

assign tmp_1_fu_86_p2 = ((in_vec_q0 == val_r) ? 1'b1 : 1'b0);

assign tmp_fu_81_p1 = idx_reg_58;

always @ (posedge ap_clk) begin
    tmp_reg_109[63:4] <= 60'b000000000000000000000000000000000000000000000000000000000000;
end

endmodule //find
```

V. HSL VERILOG RTL IMPLEMENTATION RESULTS OF THE CODE GIVEN IN FIGURE 20