VOL. 14, NO. 1, JANUARY 2019 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

TRUE PARALLELISM STRUCTURAL HARDWARE IMPLEMENTATION OF LAZER JAMMING SYSTEM USING FPGA-SOC

Hussein Ibrahim and Muataz H. Salih
School of Computer and Communication Engineering, Universiti Malaysia Perlis (UniMAP) Perlis, Malaysia
E-Mail:husseinsarhan45@yahoo.com

ABSTRACT

The current trend in the system development and the competition among manufacturers motivate both the designers and developers to improve the performance of systems and decrease the power consumption but not at the cost expense of those systems. Today, the Field Programmable Gate Array-based embedded systems is considered as the preferred computational platforms due to multi key features of these platforms including the reconfigurability, flexibility, short-time to marketing, and etc. The advanced technologies and the facilities presented via these technologies pushed towards real-time and multi-functional systems, which encourages the designers and developers to replace the single processing unit by those units with the ability to process multiple data per time. Many mechanisms can be applied over the field programmable gate array platforms to provide the ability for multi-processing and the true parallelism. Therefore, in this paper, the advantages of applying the true parallelism over the Altera Nios II Embedded Cyclone V (DE1-SoC) board are presented. The true parallelism was used to design and implement a laser missile frequency jamming system. The true parallelism is combined with the FPGA features, which improves designed system in many aspects such as increasing the system throughput, decreasing the system cost, the power consumption of the system, and the system complexity. This paper provides a presentation of the system modules, the functionality of each of these modules, and the results obtained from the LCD and Seven Segment of the (DE1-SoC), (DE2-115) board.

Keywords: embedded processor, FPGA system design, true parallelism, jamming system.

INTRODUCTION

In system development, the most core aspects that need addressing are enhancing the overall performance of the system as well as making the system respond in real time. Integrating these features can put a lot of constraints to complex systems; therefore, more sophisticated mechanisms are in high demand for embedding such features within these systems. Performing complexity is much easier with the embedded systems when compared to the general-purpose systems[1].

Rapid improvements in systems development have allowed for the development of systems that are more specifically related to human lives. As a consequence, it is now vital to give such systems the capacity to process real-time data and take accurate decisions in order to solve specific problems. The frequency jamming system considered a part of this category, where such a system's computational platform has to be efficient, accurate, and robust. In addition, systems with such functionalities, like the jamming system, should be scalable and portable.

Nowadays FPGA-based systems could combine the advantages of both of DSPs and ASIC which resulted in systems capable for rapid development cycles, high flexibility, high reliability, easy upgrading, and moderate costs[2]. This research used the FPGA to design and implement its functional units to gain the advantage of these features and even more. The harnessing of the FPGA resources in this project resulted in higher performance and better throughput since the true parallelism has been used to implement it.

Despite its long clinical success, processing platforms of the frequency jamming system have a number of problems in use which is represented in their complexity and the time required to process signals. Such expositions are unsatisfactory because the delay of processing is examined as a nature result for the system design and the delay occurring in the processing modules. Because these main factors, the platforms used to implement the frequency jamming functionality are not capable of providing the requirements for effective and fast processing systems for multiple missiles attack from different directions. The delay time required for processing data, the complexity of the system and the limitation in the frequency range are the core factors of this complication.

Additionally, design and implementation of embedded true parallelism jammer system using FPGA-SoC for low design complexity, is required to overcome delay of processing for multiple missiles threat.

Additionally, supported modules are designed and implemented such as the input signal unit manager that can synchronise four input signals, the signal emulator module that produce test signals to check the functionality of the system when there are no available signals, the output data buffer module that is responsible for storing the captured and processed signals for later studies, and the address generation unit which generates addresses for the processed data so that they can be stored within the output data buffer.

Since the code is divided into discrete portions, every part is considered an independent process concurrently executed using various modules. In this case, true parallelism was used. This means that the calculation of an interpolated value for every grid cell in the lattice is independently treated from all the other cells' computation of values. First, the jammer analyses the spectrum of the signal that was received from the frequency hopped transmitter. Then, the signal's features are extracted. The

features are then taken to the frequency synthesiser so that the synthesiser produces the same hopping frequency. Lastly, the narrow-band interference signals are sent to the display units on board like the Seven Segment and LCD.

RELATED WORK

In almost all segments of the industry, the general-purpose multicore processors are being adopted, including embedded space and signal processing. The demand for higher performance and general-purpose programmability has increased significantly. Performance can be improved through parallel processing by adding more parallel resources while simultaneously maintaining power characteristics that can be managed. The applications of multicore processors are diverse and many. Designs may vary from simple conventional multiprocessor machines to those that include a "sea" of programmable arithmetic logic units (ALUs). This paper covers few of the attributes that are common across all multicore processor implementations and these attributes are illustrated with the current and future commercial multicore designs. The focused characteristics include power/performance, application domain, memory system, processing elements and accelerators/integrated peripherals [3].

For real-time processing applications, FPGAs are commonly employed as implementation platforms due to their structure that can efficiently exploit temporal and spatial parallelism. Such parallelisation depends on the system's processing mode and associated hardware constraints. Sometimes, the designer is forced to reformulate the algorithm due to such constraints [4]. This article showcases parallelism associated with processing algorithms, which generally exists in two major forms [4]: temporal and spatial parallelism. FPGA implementations possess the capability to be parallel, employing a mixture of both the forms. For example, to exploit both forms, configuration of the FPGA could be done by distributing and partitioning the resulting sections into multiple pipelines such that each of these can process data concurrently. In practice, achieving such parallelisation relies on the system's processing mode and hardware constraints. This could force the designer to face hardware issues like concurrency.

Omar F. Yousif *et al* (2015) used a concurrent structure and the spatial parallelism mechanism over embedded systems. They employed the FPGA platform (NEEK board) that could be act as an implementation environment for this system. This led to achieving enriched proposed system that possessed core features such as decreased system complexity and low cost as within this system, a concurrent structure was employed. The ability to duplicate tasks can be achieved through spatial parallelism, which can be employed via specific modules. Here, the system signals ranged from 1 Hz to 200 MHz. The Phase Locked Loop was manipulated to remove master clock limitation of Nios II Embedded Evaluation Kit, as well as allows the system to cover a wide spectrum of signals. Multiple frequencies as per time processed through laser projective frequency jamming system. Because of high operating frequency (200 MHz), FPGA resource usage and lower complexity (small size (2604)) logic elements, the implementation has achieved acceptable throughput. Also, scalability of the embedded concurrent computing architecture done through the structural design methodology with the entire system growing [5].

A cruise missile is a pilotless, dispensable, continuously powered, self-guided, air-breathing vehicle that can fly like an airplane. It has aerodynamic surfaces as a support, and is designed to deliver nuclear or a conventional device. It is a guided missile that is employed to destroy terrestrial targets. It can remain in the atmosphere and flies at almost a constant speed for the major part of its flight path. However, the enemy can side track and destroy this cruise missile by deploying anti-cruise missile (ACM). So, it is clear that the threat of anti-cruise missile (ACM) needs to be eliminated for successful completion of the cruise missile's operation or mission [6]. In 2016, Manish Debnath, discussed about a small tracking system, where a few small air-to-air missiles were attached within the main cruise missile that could destroy any incoming ACM. So, to put it simple, this small air-to-air missile would act as an anti-missile to destroy any incoming ACM threatening the main cruise missile.

The infrared guided missile can be considered as a core weapon in the military field since it can use the reflected energy from the intended targets. According to [7]has proposed a scheme to jamming this kind of missiles. Where the IR missiles has tracking and missile guidance modules, the jamming module which was presented here is including signal processing unit and phase detector as well as tracking loop and was designed using MATLAB SIMULINK. The signal processing module detects the frequency using a Band Pass Filter (BPF) while the functionality of the phase detector is to capture the signal modulation phase and in the same time generates a signal indicator for the error. Finally, the tracking unit is in simple words represents a DC motor.

Another mechanism has been introduced by[8] where researchers developed a jamming method for false target deceptive in order to facing the missile-born Synthetic-aperture radar (SAR). In this research, the proposed jammer unit functionality is to determine the phase signal of the missile-born SAR then modulating the Doppler frequency phase. After determining the signal phase, the proposed architecture calculated and modulated the Doppler frequency phase in azimuth direction to all of the ranges of the jamming signal bin phase as well as the delay corresponding time. Deceptive jamming Image of False Vessel Target. To generate the jamming signal, the system determined every range of the bins of jamming signal and then re-transmitted these signal to the directions of the missile borne SAR. The system could counter some difficulties when implementing on the TMS kit.

Parallelism

From 2003 to 2005, a dramatic change was observed that seized processor manufacturing as well as

the whole of the semiconductor industry. Enhancing processors' computing performance by just screwing up the clock frequency had become insufficient. Before that period, steadily increasing the clock frequency allowed achieving improvement in terms of both architectural and technology aspects.

Shorter switching times and smaller channel lengths in devices were a result of scaling technology processes, as well as led to measures like out-of-order processing and instruction-level-parallelism. All this resulted in processor pipelines having high fill rates, which were guarantors to meet Moore's law [9][3][10]. Interconnected components were employed to construct all computing systems, based on the abstraction level for which a system could be viewed. These components could include gates, transistors, complete processors registers, memories or arithmetic units. There are basically two fundamental ways to compose components at all levels of abstraction for constructing parallel computing structures: one involving spatial parallelism and the other temporal parallelism. Figure-1 presents the Parallel Structures.

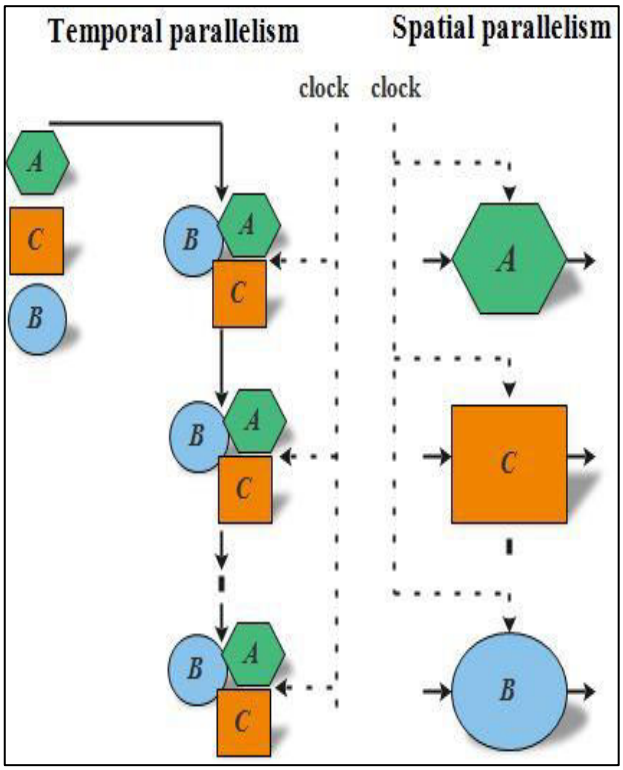

Figure-1. The parallel structures.

A. Spatial parallelism

The key concepts related to this feature are latency and bandwidth. The ideal parallel computation architecture that can be used in systems is the one which has large logic blocks of independent computation which resulting in execution these blocks in a concurrent manner. In spatial parallelism, the mechanism stipulates that each function is divided into several parts and then each part will be processed by a different processing element (PE) [11]. Here the component used to carry out the processing task is not subdivided but is instead replicated, so that each unit of information is processed by a its own dedicated component.

To exploit this form of parallelism, the units of information processed by the original (non-parallel) component must be partitionable. In other words, the task space must be parallelised. The main difference between the spatial parallelism and the temporal parallelism (which is the second main form of parallelism) is that in spatial parallelism there are multiple similar subtasks are executed in simultaneously manner[12]. The spatial parallelism can be look at as a parallel composition of the data which results in partitioning the data spatially over the available processing units. Harnessing the spatial parallelism will affect the system performance since that the functionality units will be processed in concurrent manner[13] and Figure-2 shows the parallel pipeline model.

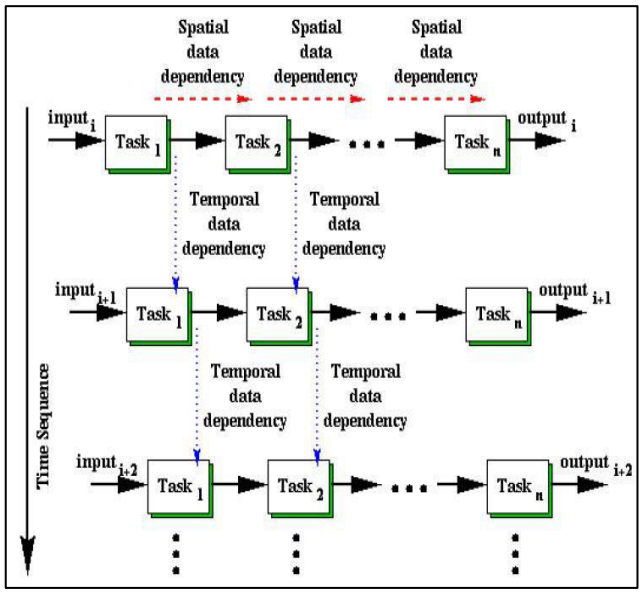

Figure-2. The parallel pipeline model[14].

The performance can be improved by duplicating the hardware for processing multiple tasks within a specific period of time. For a system that needs multiple functions for processing, the spatial parallelism can be the solution with merits as it allows processing each function simultaneously through different processing cores. However, this can happen only if there are no resource conflicts as the flexibility to perform spatial computation can be availed through the reconfigurable fabric. Figure-3 presents both the temporal and spatial parallelism [15],[16]. Function units are considered as a set of code blocks in the systems' hierarchical structure. Multiple threads are present in each one of these code blocks, which correspond to its related instructions. This is possible only if there is no interruption situation during the execution process of making the spatial parallelism's extraction feature from a program as an easy issue. After completing the compilation step, the back-end compiler can be used to partition the program into multi-threads.

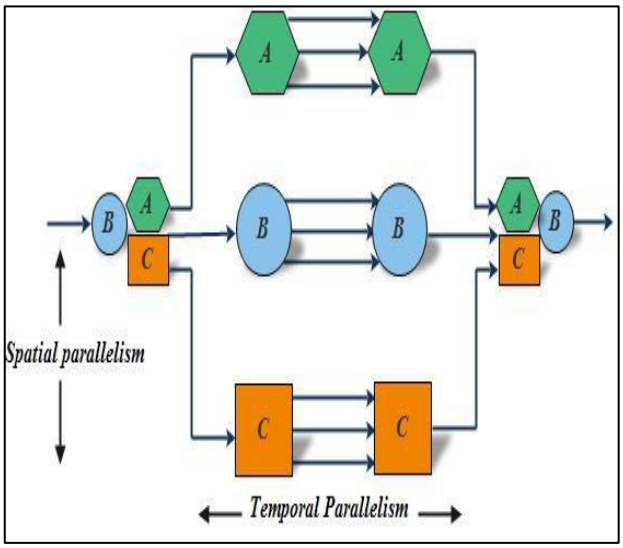

Figure-3. Spatial and temporal parallelism.

The spatial parallelism feature can be exploited for use in many levels of the system construction, which may range from interconnects employed within the system to a whole functional unit. To improve the systems' cost-effectiveness, the Ethernet-based interconnects can be employed as a potential solution [17]. Exploitation of the spatial parallelism at the communication level and interconnects is then done. This proposed scheme employs end-to-end or multiple physical links for getting results for systems having lower-cost components and a scalable feature. Good results have been obtained in terms of improving the throughput over low-cost by employing the spatial parallelism principle in communication levels and interconnects that use multiple (1GBit/s) links.

The authors in [18] have harnessed the spatial parallelism to perform a fully stochastic simulation employing the FPGA architecture. The speed of the presented architecture is greater when compared to the existing simulator, which has been designed and implemented on FPGA for more than 12–30 times. The time to detect the next reactions in the chemical systems is reduced through the spatial parallelism concept, both in terms of coarse-grain replication and fine-grain pipelining. In this architecture, if there is an occurrence of an event, then the corresponding reaction is detected by employing multiple functional units. Each of these is accountable for a specific number of reactions, where all of these would update their status in parallel.

B. In the hardware part, as proposed by [19], the spatial parallelism can be employed side-by-side with the algorithm levels. A fast motion and detection (ME/DE) algorithm was employed together with the spatial parallelism hardware architecture. To improve the throughput, both the four levels of the Multi-View Coding (MVC) and parallel ME/DE modules are harnessed. Moreover, two data dependencies classifications can arise because of employing the spatial parallelism on systems design. The first type is known as the intra-task data dependency, which arises when there is a need for exchanging system data among multiple subtasks for executing in a parallel manner. The data exchange process itself can occur when integrating the sub-results, during algorithms execution, or sometimes in both. Inter-task dependency is the other type of data dependency which shows re-organisation and transfer of data to be passed to the next functionality unit in the pipeline that has already applied a fast motion and detection (ME/DE) algorithm jointly with the spatial parallelism hardware architecture.

C. Temporal parallelism

Temporal parallelism involves partitioning the processing task into a number of steps, which when applied sequentially to each unit of information, produce the same results as the original task. In other words, the task is partitioned in time, with each step of the task being applied to a separate unit of information. A typical example is "assembly line" manufacturing. The application of temporal parallelism in computing produces pipelined structures. The time taken to produce any one result using a pipeline is in practice slightly longer than in a non-pipelined system but the rate at which results are produced is increased in proportion to the number of steps into which the original task is divided.

The overall time of processing considered as a core factor in many applications and in such fields the parallelism can be acquired via decomposing the problem in the time domain. Where the main system function can be considered as the fundament a unit of the work and each sub-task of the function is assigned to a processing unit[21]. In another word, the temporal parallelism divides the task into multiple sub-tasks and assign a processing segment to each subtask to execute it in simultaneously way as in pipelining. In sometimes this parallelism form is called pipelining since it includes execution of subtasks of a single task in a cascade fashion which means a multiple successive processing unit are contained within the system to process data and all of these processing units are capable of working in the same time in an overlapped manner. It can be said that when the execution of a specific piece of data by a particular processing unit is finished, those data will be moved to the next processing unit to complete the processing operation while the first processing unit will restart its functionality with different set of data[22]. Figure-4: show the temporal parallelism.

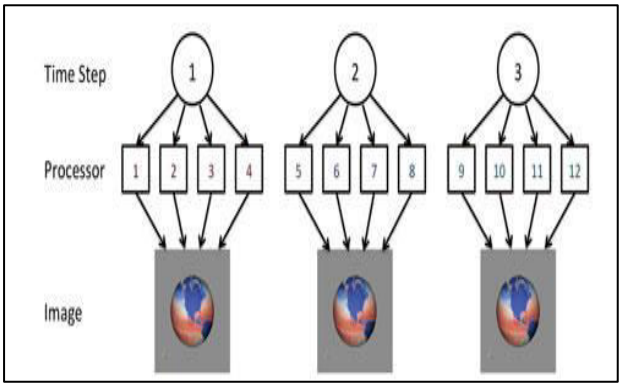

Figure-4. The temporal parallelism[13].

Figure-4 shows that the temporal parallelism can also reduce the communication and even the synchronization in parallel simulation environment using the hardware description language[23] where the HDL has a drawback represented in low runtime performance and to harnessed the temporal parallelism in this side, the whole simulation can be divided into slices and each one of these slices can be executed by independent simulator. On the other hand, the temporal parallelism perhaps requires more amount of memory since that each process will load an entire functionality unit.

Frequency jamming system structural approach

The phases that this project is passed over to see the light is shown as a flowchart in the Figure-5, these phases or volumes are explained in a brief manner in this section.

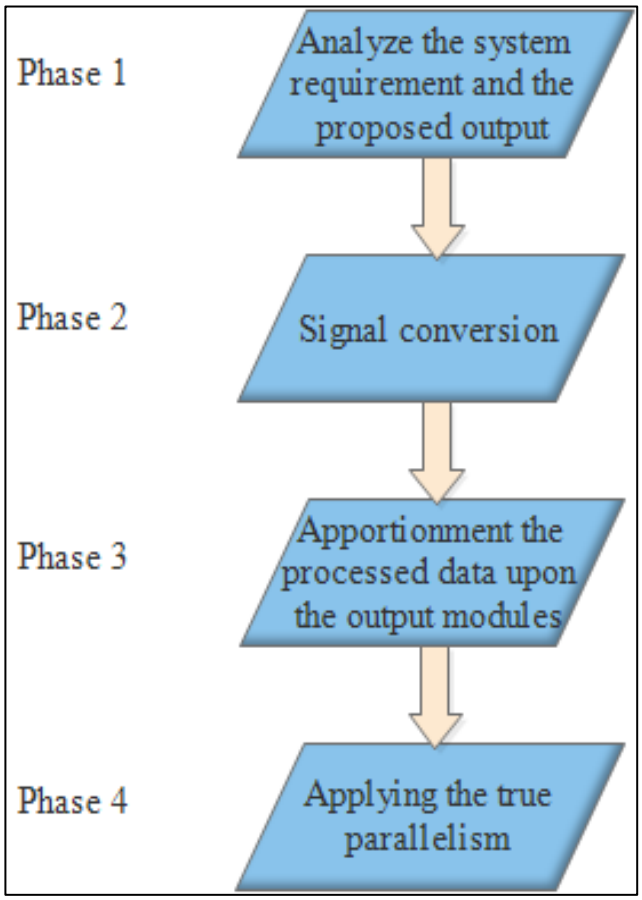

Figure-5. Phase state diagram.

A. Phase-1 Analyze the system requirements and the proposed output

This phase considered as the fundamental phase and it has the following actions:

a) The core modules that influence on the power and the performance of the processing units where studied, discussed, and determined.
b) Analyze the frequency detection approaches and figures out the fit approach based on the true parallelism.
c) Construct the main modules of the design depending on the system requirements.
d) The reference frequency was manipulated via the Phase Locked-Loop (PLL) to fit the system requirement.
e) The FPGA CAD tool has been used to test each module individually and structurally.

B. Phase-2 Signal conversion

This phase included the following activities:

a) To provide better performance for the laser missile frequency jamming, the form of the processed data has been converted to meet other system modules requirement.

C. Phase-3 Apportionment the processed data upon the output modules

The activities within this phase were as follow:

b) The processed and converted data has been distributed to the output modules in this phase.
c) The control over the defused platforms has been done within this phase.

D. Phase-4 Applying the true parallelism

a) All the system modules where implemented and tested via the Altera Quartus II software.
b) All the constructed processing modules and peripherals were connected to the FPGA chip.
c) Applying the true parallelism mechanism upon the overall system to perform multiple signals processing at each time.

METHODOLOGY OF RESEARCH

To design an embedded system with successful functionality, the right platform should be selected since it's considered as a key factor in success and failure alongside with the development tools. The Altera® (DE1-SoC), Cyclone V edition presents solution to decide critical decisions with the minimal investment. Depending on that, to perform the full functionality of this project (design and implementation of embedded true parallelism jammer system using DE1-SoC for low design complexity), the FPGA Cyclone V chip was used. The control unit and others sub-modules have been generated and verified, the next step is to create a top-level module connecting each module in order to assign the proper pins assignment. All the components that used in the system must be declared in the top-level module as show in Figure-6.

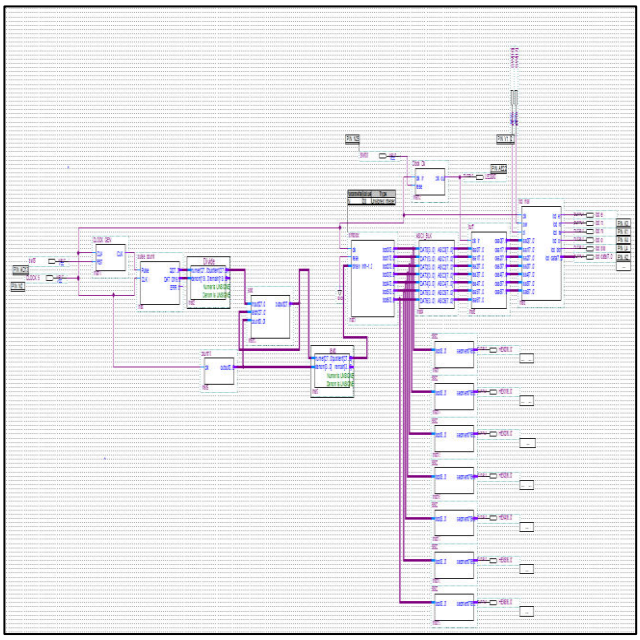

Figure-6. System top-level design.

All the components that used in the system must be declared in the top-level module. Once the top-level module has completely constructed and compiled with full success, the Quartus II compiler will generate a. sof file which is used to be download to the DE1-SoC board. Before the. sof file been downloaded to the board, all the pins need to be assigning to the correct location based on the DE1-SoC user manual. The proposed hardware design for this project was downloaded over the FPGA chip after compiling the code using the Altera® software Quartus II. The development / training kit was used to facilitate the structure deployment. The hardware equipment used in this project is the Altera® FPGA Cyclone V (5CSEMA5F31C6) chip as a portable component over the Nios II Embedded Board Development and Education board.

The Frequency Jamming system design harnessed some core concepts like true parallelism, locality and reuse in embedded application in order to integrate the desired improvement aspects like efficiency and performance.

A. Pulse detection

This project's initial stage was focused on detecting the frequencies used by multiple missiles to guide them to their target, which were emitted by multiple missile launchers. The frequencies are detected to figure out the total number of pulses associated with this signal.

A General-Purpose Input / Output unit (GPIO) is employed that has clock signal input of 50MHz to provide signal in this project. This unit can amplify the range through design manipulation of the PLL to get up to 1.6 GHz, the maximum value achievable through the PLL [24].

The circuit's main principle is to determine the time period width for the input signal. For this, the input signal provided from the GPIO is received first to continue with the counting of the number of pulses embedded within that input. The signal is later processed within the time period of the board's master clock. Figure 7 displays the RTL involved in the integrated synthesis of the Frequency Detection Circuit.

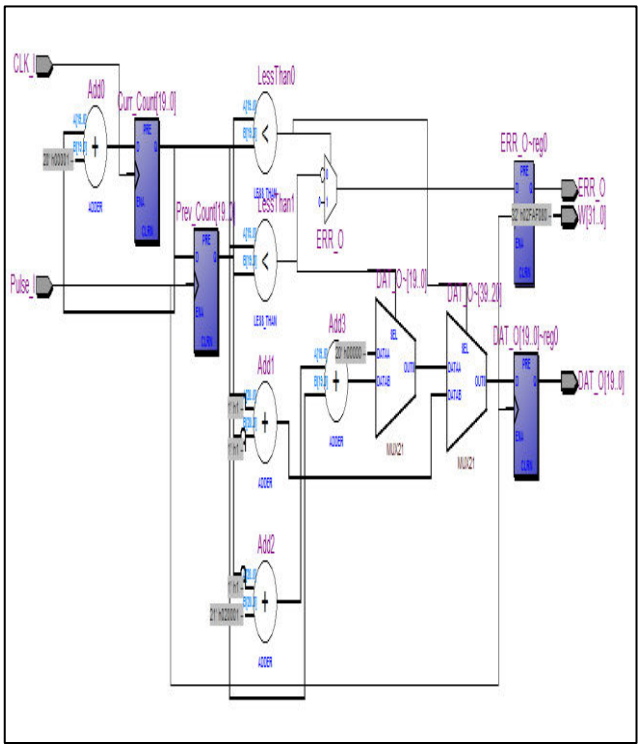

Figure-7. RTL View for pulse counter.

This module ability to counts frequencies up to six digits so the signals declared within the code was of size 24 bits in order to be converted smoothly in the next steps of the module to be represented as a BCD form. With each clock cycle the time period of the pulse will be calculated and an internal parameter will be increased then this parameter will be assigned at the end to the output. The module also handled the count overflow situation which can happen when the module is running for a long period.

B. Signal conversion module

As obtained from the previous stages, the frequencies are in the binary form. To meet viewing requirements, the BCD form is used to view the results. This module's core performance is focused on transforming the binary form to the BCD form as given in Figure-8. The Double Dabble (DD) algorithm was employed to achieve this.

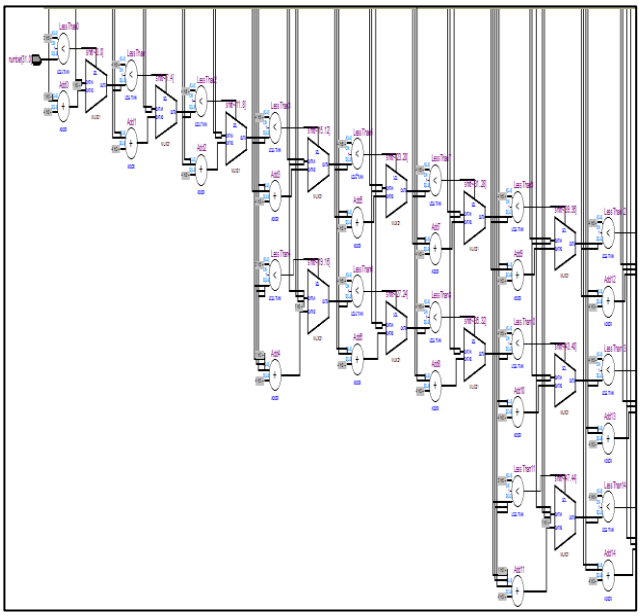

Figure-8. The internal logic circuits of one-digit binary-to-BCD conversion.

The DD algorithm, also called as Shift and Add 3, is an extensively used algorithm for transforming the binary form to the BCD notation. The basic principle behind this algorithm is explained below:

First, it is well known that each BCD digit is made of 4-bits. The proposed frequency jamming system can view up to 6-digits; so, the next step is building a register for storing both the binary data taken from the frequency detection part as well as from its BCD representation. Thus, the detected frequency represented by BCD was 32-bit in size while the detected frequency was of 36-bit binary size (4-bit for each digit). This module's initial step was focused on initialising the register value to zero. The algorithm result is represented in the form of a BCD and is divided into ones, tens, 100s, 1,000s, 10,000s, and 1,00,000s units. After initialisation of the register is complete, the next step was shifting the binary number to the left with one bit. This could result in two situations where either the number value is (³5), then the next step would be to add (3) to the shifted number and completely shift to the left with one bit, which goes on until the complete input data is shifted or the binary number value in any BCD columns is (£4) where the procedure would be to completely shift the number to the left with one bit [25]. The original binary input is stored in the right side of the register while the BCD result is to the left.

After getting a general idea of the employed algorithm in this module, a detailed description of this module is illustrated here. To store the output BCD number, a temporal variable was selected and initialised to zero. The input binary form is then loaded into the same temporal variable. Since the ones unit cannot go beyond the value of four, it is passed for the first three iterations. The procedures are cycled for 32 times to perform a BCD output, which is based on the total number of binary inputs. After shifting of one unit is complete, the shifting of tens unit is initiated while still checking and shifting for the (³5) condition. To finish up the converting process, the procedure continues in the same steps and then the output is set into the final results at the end.

C. Distribution module

A common scheme employed in logical circuits for transferring a lot of data to multiple units through a single set of wires is the multiplexing technique. In most cases, this functionality relies on the sequence timing of the signals to be sent, which should be in a correct manner and simultaneously should identify the final destination of the signals. The basic idea behind the use of multiplexing module is to send multiple BCD digits to 7-segment. This approach employs the same mechanism as that for sending multiplexed data to the LCD. The 6-BCD digits that were already converted in the last stage are then multiplexed in this module for sending a single digit at a time via 3-control signals, as demonstrated in Figure-9. At the same time, the selected digit that has to be sent should be in the ASCII form to enable viewing on the LCD.

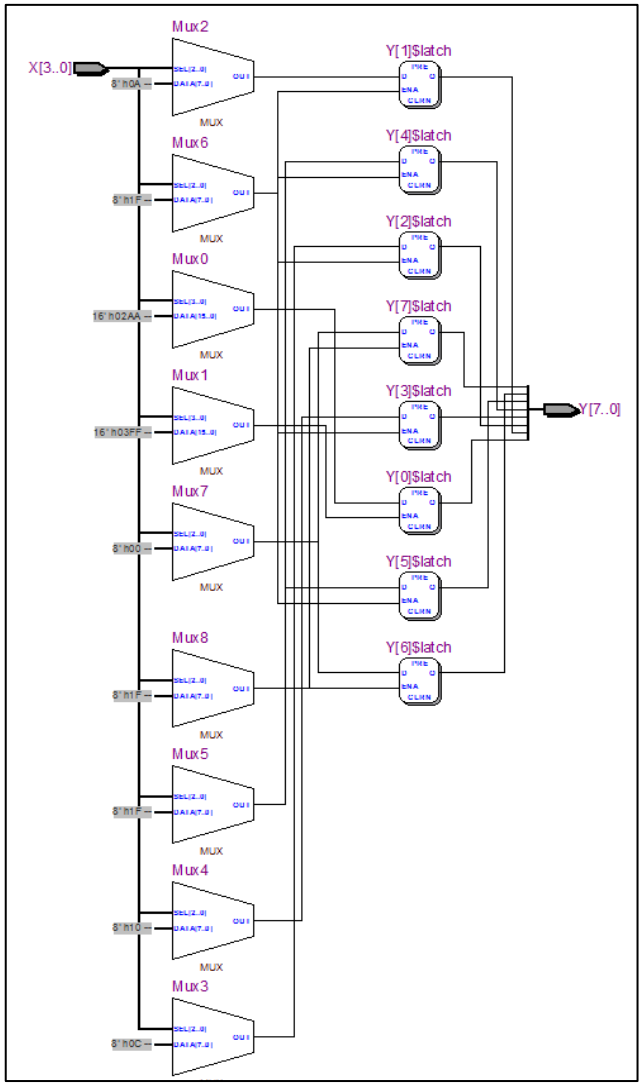

Figure-9. Distribution module.

The initial step in this module is to make the output be in the high impedance state (Z) such that only one entity writes to the bus. Next step is to control which BCD-digit will be sent to the LCD view unit and this was done via two control signals. At each state of the 6-states of the control signals, only one digit will be assign to a temporal signal. Now, the selected digits at each state of the temporal signal will be assigned to the final output but in the ASCII form and not in the BCD form.

D. Concurrent Jammer Module

Many researchers and designers have been continuously trying to improve performance by enhancing throughput and speeding up the computation power. After completion of core functionality designing and testing for a system through a simulation CAD tool, the next step was to evaluate the contribution of our system for multiple signals processing capability. For processing, more than a signal at a time, we have exploited the true parallelism concept. This module deals with collection of other modules to form the main module and then the functions of each one of the system modules are reproduced. Exploiting this concept has offered several benefits such as the ability to process more than a signal each time, improving the overall performance of the system, reducing the FPGA used resources, and scaling up the system throughput. The concept of true parallelism was employed in splitting the system's multi-functional units into several parts, where each sub-functionality unit is processed through different modules.

E. View sub-system

This section focuses on configurability, which allows viewing results on 7-segment and LCD. Also, it can be used for more than one board.

A. LCD

The first stage involved viewing characters (Freq=), the design of the second stage was aimed at displaying the 8 digits representing the detected frequency, and the final stage involves displaying the characters (Hz) on the LCD.

The display step on the LCD module initiates when the entry mode is activated for the LCD and is ready to receive data. The second state machine showed the first four letters (FREQ) of the word "Frequency" where the special character (=) was employed to get a final result in the form as FREQ= 25000.

B. Seven segment

This action is needed to prepare the 7 segments for receiving the commands (Read).

The name 7 segment is based on the fact that it has 7 LEDs that form the number 8 should all of them are 'on'. To form numbers from 0 to 9 as well as some alphabet letters, the 7 LEDs can be individually lit.

The LED has 2 parts: the positive part called the anode and the negative part called the cathode. If a positive voltage is applied to the anode and the cathode is put to the ground (0 V), then lighting up of the LED occurs. One of the LEDs' sides must be common to all of them (either the cathode or anode) and the other should be individual (7 data lines) for individually lighting up the 7-segments and make use of just the minimum pins as possible. The display having one anode is known as the common anode and the other is known as the common cathode.

A binary coded decimal (BCD) is a 4-bit number representing the numbers 0–9 (0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001). A BCD to 7 segment decoder allows mapping each of these 10 codes to 7bit codes (one bit for each segment) for controlling the display.

The decoder 4 inputs employ one input of type std_logic_vector(3downto0) employing std_logic_vector or a type of std_logic. The output will be a std_logic_vector(6downto0). The selected signal assignment statement is used for the decoding.

RESULTS

In this section, we have emphasised on the performance of our proposed laser missile frequency jamming system while presenting the obtained results though this system over the (DE1-SoC) board. We opted to split the results in this section to present the results of each module separately from the results of the overall system, which are shown on the LCD and 7-Segment to highlight the enhanced strength of the system. The obtained results from the verification volume, through the use of simulation CAD tool (Quartus II web edition) presented by Altera, are illustrated below.

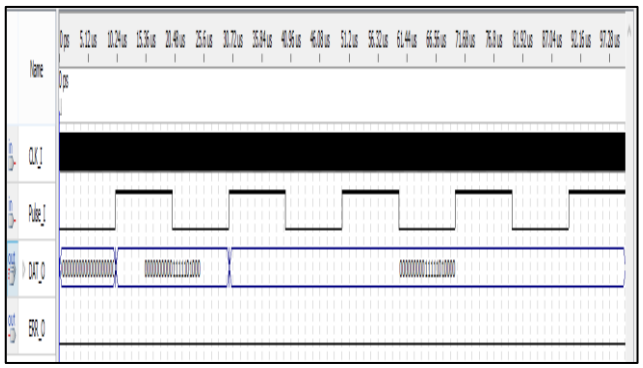

Figure-10. The detected frequency for the input signal 25000 KHz.

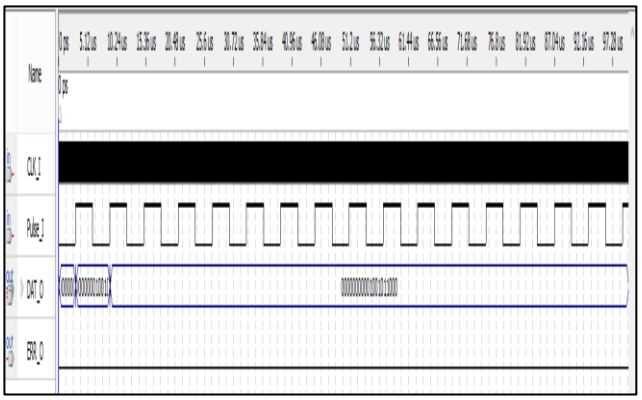

Figure-11. The detected frequency for the input signal 166 KHz.

Table-1 presents a brief view about the binary representation for each result obtained from this module.

Table-1. Frequency capturing calculations.

FRQ.	**Pulse**	**on bits** (20)	**pulse, width** (ns)
25000	2000	00000000011111010000	20
166666	300	00000000001001011000	20

As it can be seen that the detected frequency is represented in the binary form and in simple calculating process, it can be found that the detected frequency here is (25000 Hz) where the Figure above shows that the result is (00000000000000000110000110101000). For the signal conversion module, the table below shows the detected input frequency in both binary and BCD format:

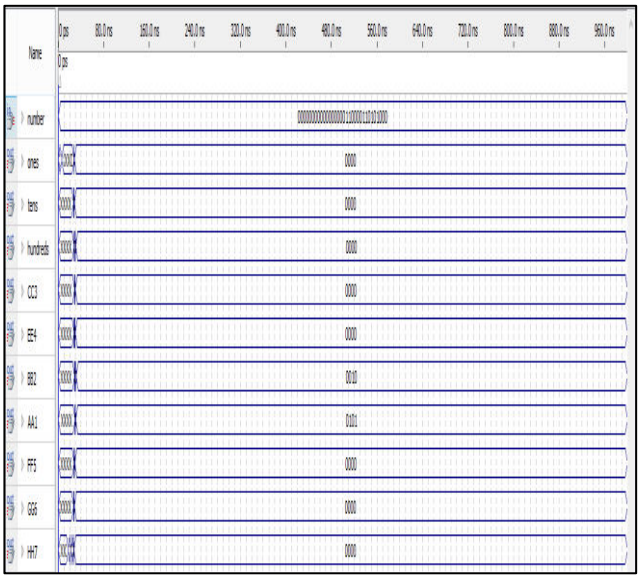

Figure-12. Binary coded decimal representation for signal.

Table-2. BCD Representation of signal (25000KHz).

Signals									
Binary	Decimal	Binary coded decimal							
00000000000000000110000110101000	25000	FF5	EE4	CC3	BB2	AA1	Hundred	tens	ones
		0000	0000	0000	0010	0101	0000	0000	0000

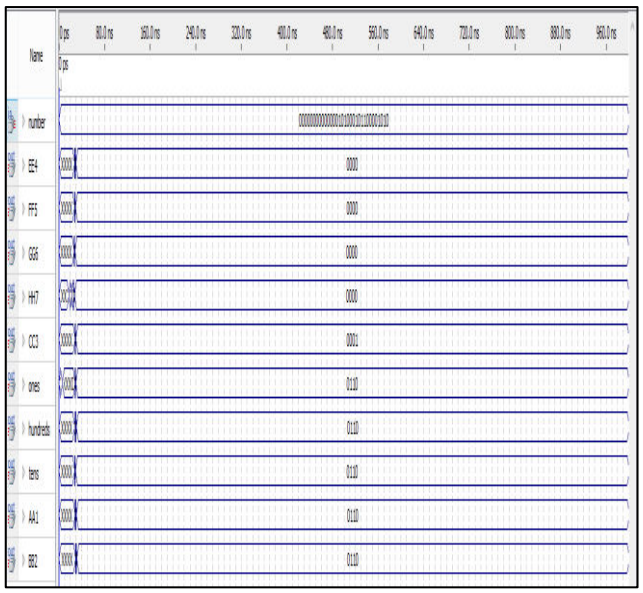

Figure-13. Binary coded decimal representation for signal.

Table-3. BCD representation of signal (166666KHz).

Signals									
Binary	Decimal	Binary coded decimal							
00000000000000101000101100001010	166666	FF5	EE4	CC3	BB2	AA1	hundred	tens	Ones
		0000	0000	0001	0110	0110	0110	0110	0110

The Altera CAD tool Quartus II web edition software was employed for getting the above results during the verification phase. The results shown over the LCD and 7-Segment are all outlined in this section.

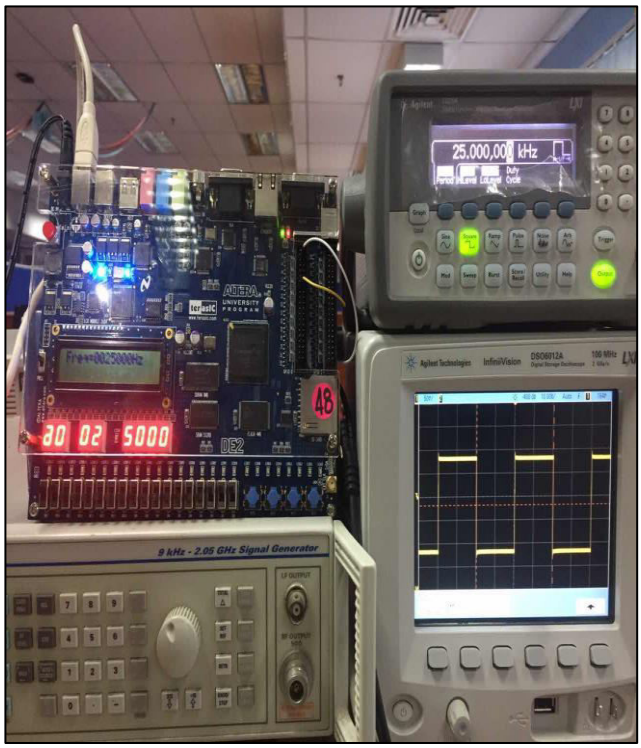

Figure-14. System outcome for the input (25000Hz) on DE2.

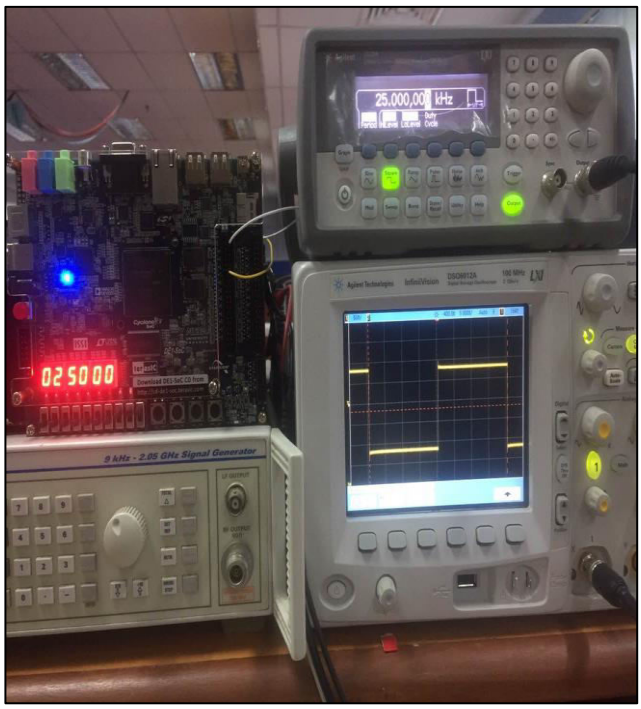

Figure-15. System outcome for the input (25000Hz) on DE1-SoC.

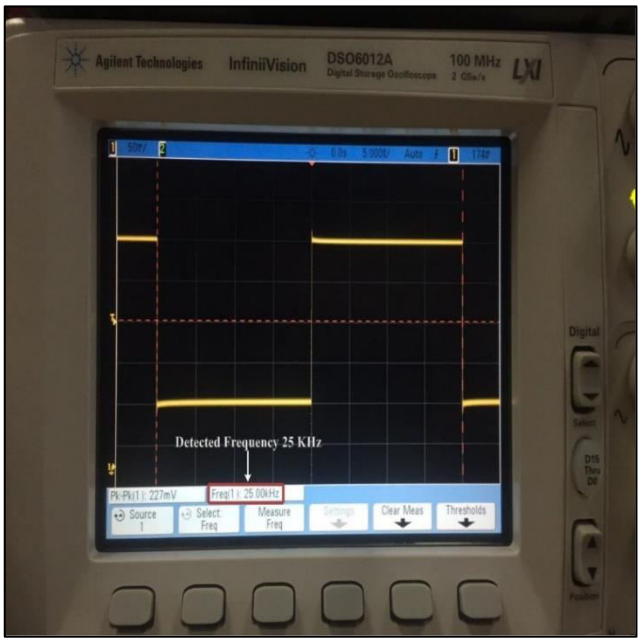

Figure-16. Oscilloscope view for the signal (25000Hz).

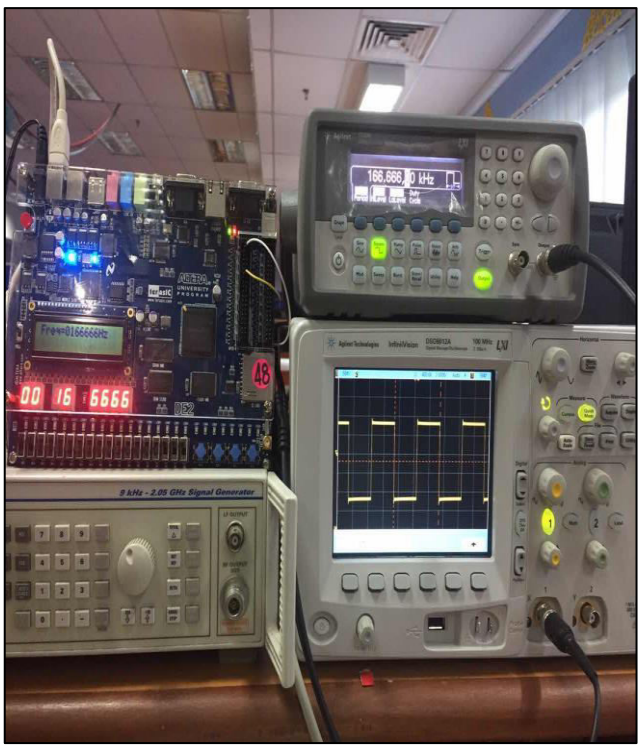

Figure-17. System outcome for the input (166,666Hz) on DE2.

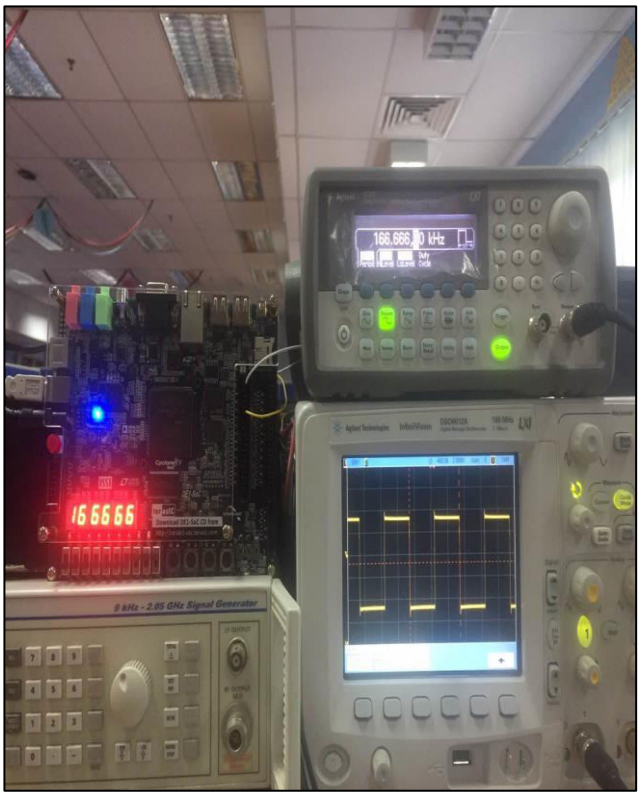

Figure-18. System outcome for the input (166,666Hz) on DE1-SoC.

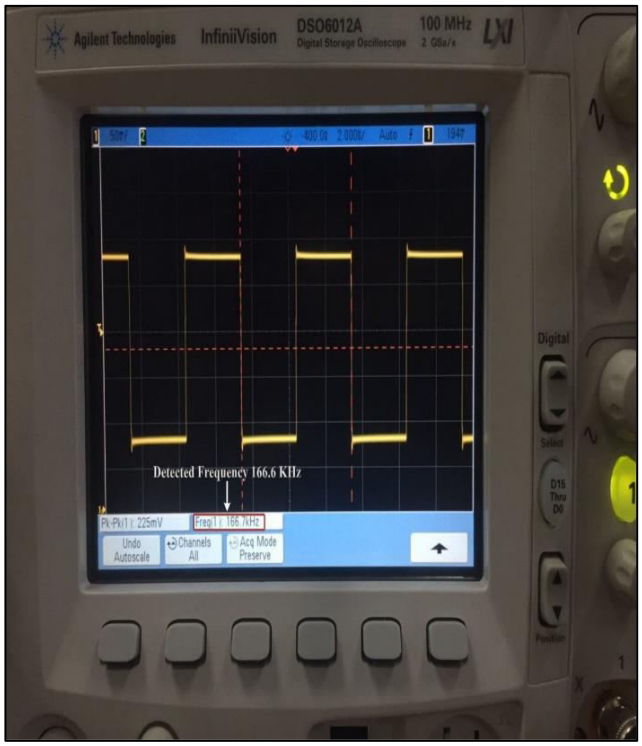

Figure-19. Oscilloscope view for the signal (166,666Hz).

The system response when four laser missile launchers will attack a target from four proposed directions as shown and what are the detected frequencies emitted from these launchers.

CONCLUSIONS

Based on the obtained results, it can be clearly concluded that applying the true parallelism concept within our proposed system conferred the system modules with the ability to process multiple data per time for executing processes involving multiple outputs, while reducing the overall system complexity and increasing the modules utility. Overall, our results demonstrated that applying true parallelism helped achieve high throughput over low-cost and reduced power consumption and system complexity, which seem to be a promising direction. Finally, to attain even better throughput and performance, the design of the VHDL code and logic circuits for this project can be enhanced further.

ACKNOWLEDGEMENT

The authors would like to thank the Ministry of Education Malaysia (MOE) for providing the FRGS research grant (Ref: FRGS/2/2014/ICT06/UNIMAP/02/3).

REFERENCES

[1] M. P. M. Chawan, B. Patle, V. Cholake, and S. Pardeshi. 2012. Parallel Computer Architectural Schemes. In International Journal of Engineering Research and Technology. 1(9) (November-2012).

[2] R. Griessl *et al*. 2014. A Scalable Server Architecture for Next-Generation Heterogeneous Compute Clusters.

[3] G. Blake, R. G. Dreslinski, and T. Mudge. 2009. A survey of multicore processors. IEEE Signal Processing Magazine. 26(6): 26-37.

[4] C. T. Johnston, K. T. Gribbon, and D. G. Bailey, "Implementing Image Processing Algorithms on FPGAs Abstract.

[5] O. F. Yousif, M. H. Salih, L. A. Hassnawi, M. A. M. Albreem, M. Q. Seddeq, and H. M. Isam, "Design and Implementation Computing Unit for Laser Jamming System using Spatial Parallelism on FPGA," no. 9003, 2015.

[6] M. Debnath. 2016. Protection of Cruise Missile from the Threat of Anti- Cruise Missile (ACM) by Using Small Air-to-Air Missile (AAM). 3(5): 99-103.

[7] G. Kim, B. Kim, T. Bae, Y. Kim, S. Ahn, and K. Sohng. 2010. Implementation of a Reticle Seeker Missile Simulator for Jamming Effect Analysis. pp. 1-4.

[8] X. He, J. Zhu, J. Wang, D. Du, and B. Tang. 2015. False target deceptive jamming for countering

missile-borne SAR. Proc. - 17th IEEE Int. Conf. Comput. Sci. Eng. CSE 2014, Jointly with 13th IEEE Int. Conf. Ubiquitous Comput. Commun. IUCC 2014, 13th Int. Symp. Pervasive Syst. (9140): 1974-1978.

[9] D. E. Culler, J. P. Singh, and A. Gupta. 1999. Parallel computer architecture - a hardware / software approach.

[10] T. G. Mattson, B. Sanders, and B. Massingill. 2005. Patterns for Parallel Programming. p. 355.

[11] H. E. Egeth, C. L. Folk, and P. A. Mullin. 1989. Spatial parallelism in the processing of lines, letters, and lexicality. Object perception: {S}tructure and process. pp. 19-52.

[12] D. Kumar, R. K. Behera, and K. S. Pandey. 2013. Concept of a Supervector Processor: A Vector Approach to Superscalar Processor, Design and Performance Analysis. 228(2): 225-228.

[13] B. Nouanesengsy *et al.*2013. A Model for Optimizing File Access Patterns Using Spatio-temporal Parallelism. Proceedings of the 8th International Workshop on Ultrascale Visualization. 4:1-4:8.

[14] A. Choudhary *et al*. 2000. Design, implementation and evaluation of parallel pipelined STAP on parallel computers. IEEE Transactions on Aerospace and Electronic Systems. 36(2): 528-548.

[15] D. H. Koh, K. K., Hwang, H. K., Kim, P. G., Lee, S. H., Cho, S. K., Kim, S. S., and Yoon. 1993. Isolated left main coronary ostial stenosis in Oriental people operative. Journal of the American College of Cardiology. 21(2): 369-373.

[16] Z. Guo, W. Najjar, F. Vahid, and K. Vissers. 2004. A quantitative analysis of the speedup factors of FPGAs over processors. FPGA'04: Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays. pp. 162-170.

[17] S. Passas, G. Kotsis, S. Karlsson and A. Bilas. 2008. Exploiting spatial parallelism in ethernet-based cluster interconnects. IPDPS Miami 2008 - Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, Program and CD-ROM.

[18] D. B. Thomas. 2013.A Fully Pipelined Fpga Architecture For Stochastic Simulation of Chemical Systems Imperial College London email: dt10@ic.ac.uk Hideharu Amano Keio University.No. 1.

[19] B. Zatt, M. Shafique, F. Sampaio, L. Agostini, S. Bampi, and J. Henkel. 2011. Run-time adaptive energy-aware Motion and Disparity Estimation in Multiview Video Coding. 2011 48th ACM/EDAC/IEEE Des. Autom. Conf. pp. 1026-1031.

[20] A. N. Choudhary, S. Das, N. Ahuja, and J. H. Patel. 1990. A reconfigurable and hierarchical parallel processing architecture: performance results for stereo vision. [1990] Proceedings. 10th Int. Conf. Pattern Recognit. Ii: 389-393.

[21] T. W. Crockett. 1995. Parallel rendering. World.No. 95.

[22] A. A. Freitas and S. H. Lavington. 1997. Mining very large databases with parallel processing, vol. 9. Springer Science & Business Media.

[23] M. Ciesielski. 2011. Temporal parallel simulation: A fast gate-level HDL simulation using higher level models. 2011 Design, Automation & Test in Europe. In: 2011 Design, Automation & Test in Europe (pp. 1-6). IEEE. pp. 1-6.

[24] A. Corporation. 2015. Altera Phase-Locked Loop (Altera PLL) IP Core User Guide Altera PLL IP Core Parameters - General Tab.

[25] S. Gao. 2012. An Improved BCD Adder Using 6-LUT FPGAs. pp. 13-16.