Journal of Engineering and Development, Vol. 15, No. 3, September (2011) **ISSN 1813-7822**

Design and Implementation of a Configurable Real-Time FPGA-Based Geometric Symmetry-CFAR Processer for Radar Target Detection

Asst. Prof. Dr. Waleed Khalid
Dep. of Electrical Engineering
Al-Mustansiriya University

Mohammed Hussein Ali
Dep. of Electrical Engineering
Al-Mustansiriya University

Abstract

A Constant False Alarm Rate (CFAR) processor is the signal processing algorithm that controls the rate at which target detection are falsely declared. In this paper, a configurable Field Programmable Gate Array (FPGA)-based hardware architecture for Geometric-symmetry (GS) - CFAR processer for radar target detection is presented. The proposed architecture of this algorithm has been designed using Matlab-Simulink 7.8(R2009a) to deal with parallel structure, so as to obtain system parameters and to test the flow of signal through the system. The design has been converted to behavioral VHDL coding style, as well as a VHDL test bench, Simulink HDL Coder tool has been used to realize hardware directly from Simulink design. The simulation waveforms are obtained using ModelSim Altera 6.1g. Synthesis reports and board programming files have been obtained using the QUARTUS II package. ALTERA-Cyclone III FPGA family with EP3C120F780C7 board has been used as target device for implementation purpose. The post place and route result show that the proposed design can achieve a maximum operating frequency of 115.77MHz which is close to the clock frequency of the prototyping board.

الخلاصة

معالج المعدل الثابت للانذار الكاذب(CFAR) هو عبارة عن خوارزمية لمعالجة الاشارة والسيطرة على النسبة التي يتم فيها اعلان كشف الهدف بالخطا. في هذه البحث تم استحدام مصفوفة البوابات المبرمجة موقعيا (FPGA) لبناء خوارزمية التماثل الهندسي لمعالج المعدل الثابت للانذار الكاذب حيث ان المعمارية المقترحة لهذة الخوارزمية تم تصميها اولا باستخدام برنامج المحاكاة(Matlab-Simulink) للتعامل مع التراكيب المتوازية وللحصول على معالم النظام واختبارمرور الاشارة, وبعد ذلك تم تحويل التصميم(Model) مباشرة الى لغة VHDL وكذلك تم توليد VHDL test bench باستخدام برنامج مشفر لغة الكيان الكتلي(Simulink HDL Coder tool). تم استخدام البرنامج ModelSim Altera 6.1g. للحصول على الاشكال الموجية وكذلك برنامج QUARTUS II لاغراض البناء والحصول على تقارير البناء وملفات البرمجة, و تم اختيار عائلة الالواح Cyclone III و تحديداً اللوح EP3C120F780C7 لغرض البناء الحقيقي, وقد بينت النتائج ان التصميم المقترح يمكن ان ينجز باقصى تردد تشغيلي مقداره (115.77 MHz) وهو قريب الى تردد عمل اللوح المستخدم.

1. Introduction:

The system that improve the radar ability to recognize the targets in the presence of clutter are the constant false alarm rate (CFAR) detectors. CFAR detectors are used to manage the false alarm rate and maintain it under a statistical design-specific value called the probability of false alarm, P_{fa} [1].

The CFAR detector designed for one type of distribution, such as Rayleigh or Weibull, cannot maintain the false-alarm rate constant for another type of distribution, or it can maintain the false-alarm rate constant sometimes but not always. In other words, the parametric CFAR detector cannot maintain the false-alarm rate constant in a real clutter environment. Therefore, the best way to solve this problem is to design a CFAR detector whose CFAR performance is distribution-free. This is the non-parametric CFAR detector [2].

Although software-based implementation is very flexible, the whole CFAR processing may degrade the performance of the processor. Hence, to accelerate the processing, it is proposed to realise the computationally intensive CFAR detector in field programmable gate array (FPGA) hardware. FPGAs are a form of programmable logic and have emerged as an attractive and ideal environment for hardware realizations for high speed algorithms and intensive computation applications. They offer design flexibility like software, but with time performance closer to application-specific integrated circuits (ASIC). Recent advances in FPGA technology have resulted in enormous possibilities for the implementation of sophisticated algorithms of high complexity, in a variety of important applications, by using low cost, high performance and high speed reconfigurable hardware. FPGAs have become one of the prevailing technologies for fast prototyping and implementation of complex digital systems [3]. In this work, a novel FPGA based design and realization of a geometric-symmetry CFAR detector is presented.

2. Background theory and related work:

A typical CFAR processor is shown in Figure. (1), the input signals (output samples of square law detector) are fed sequentially into a shift register. The content of the cells surrounding the cell under test (Y) are processed using a CFAR processor to get the adaptive threshold K_oZ. Then the value of Y is compared with the threshold to make the decision. The cell under test is declared as a target if its value exceeds the threshold value. In most radar detectors, the threshold is set in order to reach a required probability of false alarm.

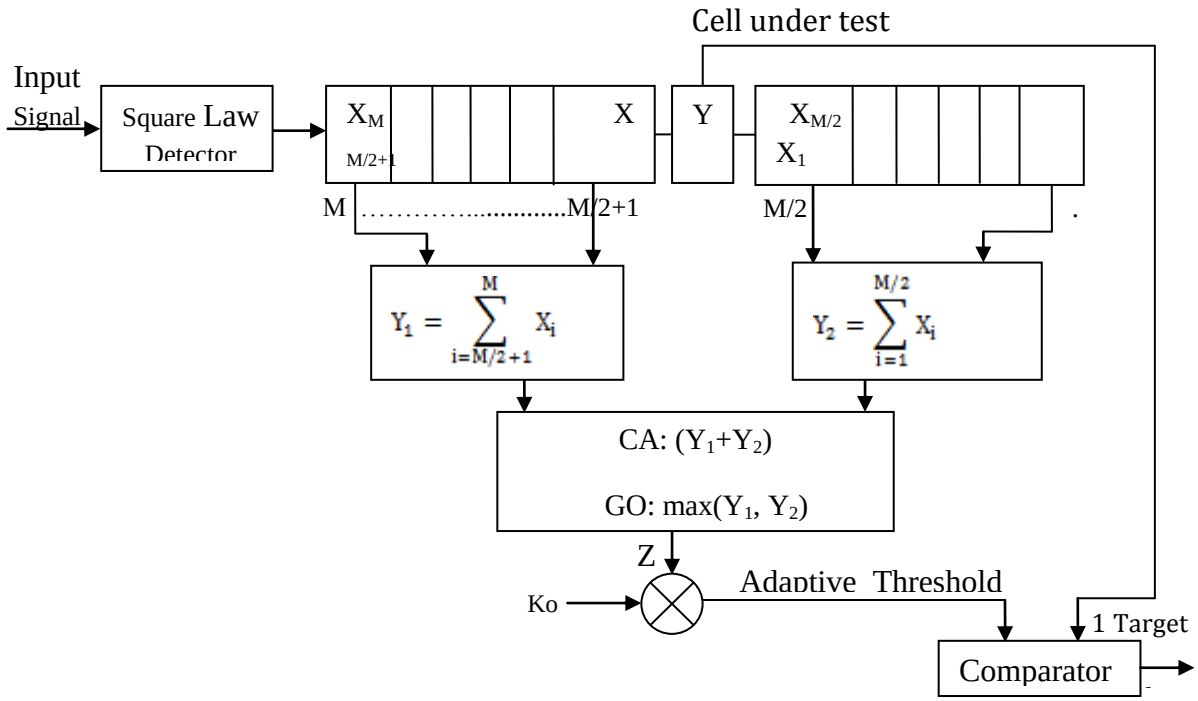


Figure 1: Block diagram of a typical CFAR algorithm

The developments of the theoretical aspects of CFAR detection are not followed by hardware implementation. There are few attempts considering hardware implementations of CFAR processors have been reported. In particular, configurable hardware architecture for adaptive processing of noisy signals for target detection based on CFAR algorithms has been presented in [4-6]. The architecture has been designed to deal with parallel/pipeline processing and to be configured for Max, Min, and Cell- Average (CA) CFAR algorithms. OS-CFAR was implemented using parallel structure in [7]. In [8], CA-CFAR and OS-CFAR are combined and implemented in FPGA. In [9], TM-CFAR algorithm has been realized using FPGA. However, all these implementation were for simple CFAR algorithms and only suitable for Gaussian distribution type of clutter. In [10], the proposed hardware considered the implementation of the automatic censored cell averaging ordered data variability CFAR (ACCA-ODV CFAR) algorithm. The authors of [10] proposed an efficient parallel architecture that managed to parallelize the sequential censoring process. However, a main disadvantage of [10] is the processing delay that meets the requirements of detection in Rayleigh distribution clutter**.** Furthermore not standard interface is given in order to facilitate the communication with the Radar System environment.

3. GS-CFAR Processor algorithm:

A signal processor that has constant false-alarm rate for any geometric-symmetry environment is shown in Figure (2). The decision random variable for this processor *d* is given [11]:

$$d = r_o \left[\sum_{i=1}^{M} \sqrt{r_i r_{-i}} \right]^{-1} \underset{H_o}{\overset{H_1}{\gtrless}} K \qquad (1)$$

The hypothesis H_o represents the case of noise alone, while hypothesis H_1represents the noise plus target signal case.

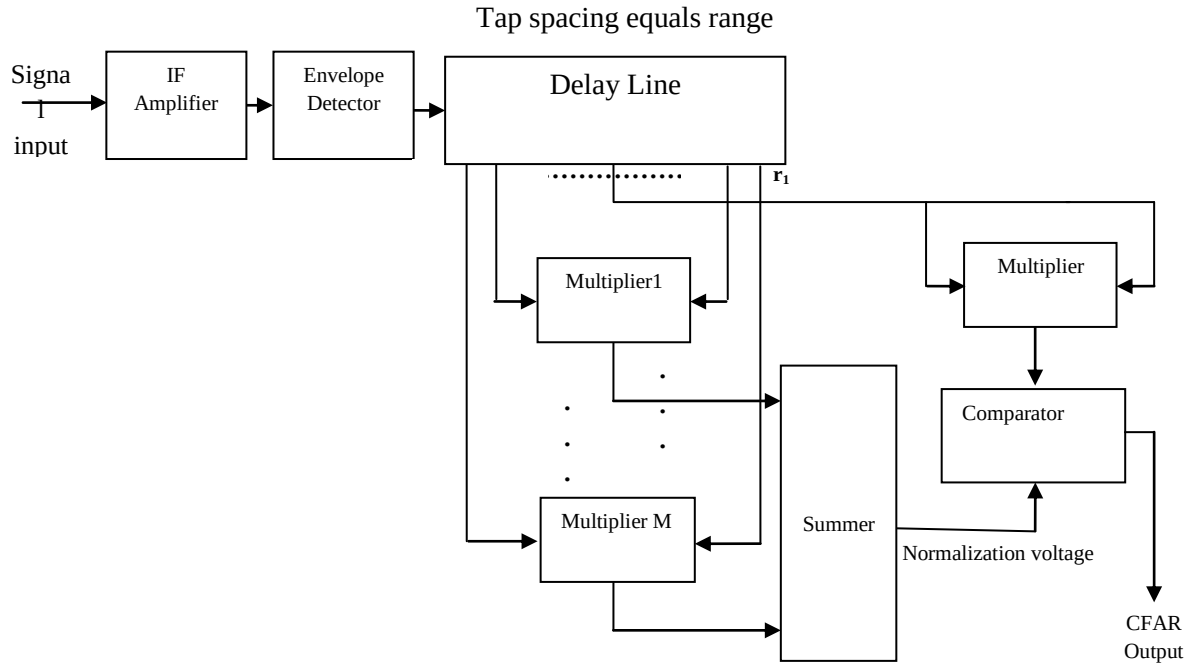


Figure 2: The block diagram of a geometric-symmetry CFAR signal processor.

A target-present decision is made if the value of *d* is greater than a threshold value *K*, and a target-absent decision is made otherwise. Assuming that the available set of independent identically distributed unit variance random variables is [x_i]. The set [r_i] is formed by multiplying each member of [x_i] by the constant ($a^{1/2}$), that is [11]:

$$r_i = \sqrt{a\beta_i}\,\mathrm{X}_i \qquad \mathrm{i} = 0,\pm1,\pm2,....,\pm M \qquad (2)$$

where a is scale parameter. For geometric-symmetry environment, the power of the *ith* cell multiplied by the power of the minus *ith* cell is the square of the power of the zeroth cell. From the generation viewpoint, the values of β_{-i} are given by [11]:

$$\beta_{-i} = \frac{1}{\beta_i}, \quad i = 1,2,\ldots\ldots,M \qquad (3)$$

Substitution of Equations (3) and (2) into Equation (1) demonstrates that the value of *d* is independent of the values of *a*. Thus, the processor is geometric-symmetry *CFAR*. Assume the *PDF* of X_0 & X_i or (r_0 & r_i) is Gaussian case, then *K* must be found.

By examination of the Equations (1), it can be seen that decision random variables of the signal processors in the form of a fraction with the numerator equal to r_o. The denominator, or normalizing voltage, is an estimator of the variance σ_o^2. The probability of a target present decision (detection probability) is given by [11]:

$$P_d = Pr[r_0 > KV] \qquad (4)$$

where V is the normalizing voltage of the signal processor, K is the threshold value of the signal processor, and P_d is the detection probability. The results of this section assume the interference additive and that the distribution function is Gaussian. It will be assumed that the appropriate target fluctuation model is Swerling I, so that the probability density function for the square-law detected voltage r_0 $\left(r_0 = r_i\big|_{i=0}\right)$ is given by [12]:

$$f_r(r_0) = \frac{e^{-r_0 / 2\sigma_0^2(1+SNR)}}{2\sigma_0^2(1+SNR)} \qquad o < r_o < \infty \qquad (5)$$

where SNR is a signal to noise ratio. It is also clear that the square law detector outputs from the other resolution cells are all mutually statistically independent. Thus, the density function for the output of the *ith* cell is the same in Equation (5) with SNR equal to zero, unless i is equal to zero. Therefore, the probability density function of the normalizing voltage is independent of that of r_0, so that [12]:

$$P_d = \int_0^\infty f_V(V)\left[\int_{KV}^\infty f_r(r_0)\, dr_0\right] dV \qquad (6)$$

where $f_v(\bullet)$ is the probability density function of the normalizing voltage of the processor. By substituting Equation (5) into Equation (6), the result is then recognized in the form of the characteristic function of V evaluated at a particular argument. Thus [11],

$$P_d = \int_0^\infty f_V(V) \int_{KV}^\infty \frac{e^{-r_0 / [2\sigma_0^2(1+SNR)]}}{2\sigma_0^2(1+SNR)}\, dr_0\, dV \qquad (7)$$

$$P_d = \int_0^\infty f_V(V)\, e^{-KV / [2\sigma_0^2(1+SNR)]}\, dV = \int_0^\infty f_V(V)\, e^{j\omega V}\, dV \qquad (8)$$

where $\omega = \frac{-K}{2\sigma_0^2}$, Equation (8) represents the characteristic function ψ_V of V then [11],

$$P_d = \psi_V(\omega)\Big|_{W=\frac{JK}{2\sigma_0^2(1+SNR)}} = \psi\left[\frac{JK}{2\sigma_0^2(1+SNR)}\right] \qquad (9)$$

so it is enough to find the *PDF* of *V*and then to find ψ_V for it to get P_d.

For GS-CFAR processor; the normalizing voltage is the sum of *M* independent random variables. Each of these random variables is obtained from the multiplication of two independent Rayleigh random variables, each with different variance. Since the adaptive threshold voltage T_h is,

$$T_h = K\sum_{i=1}^{M}\sqrt{r_i r_{-i}} = KV \qquad (10)$$

So $V = \sum_{i=1}^{M}\sqrt{r_i r_{-i}} = \sum_{i=1}^{M} t_i$ where $t_i = \sqrt{r_i r_{-i}}$

The density function of this product may then be obtained as [11]:

$$f(t_i) = \frac{t_i}{\sigma_i^2 \sigma_{-i}^2} K_0\left(\frac{t_i}{\sigma_i \sigma_{-i}}\right) \qquad (11)$$

where K_o is the modified Bessel function of the second kind and zero order. The probability of false alarm can be found as [11]:

$$\therefore Pf_a = \left[\psi_{r_i}\left(\frac{K}{2}\right)\right]^M \qquad (12)$$

or $$\psi_{r_i}\left(\frac{K_2}{2}\right) - (Pf_a)^{1/M} = F(K) \qquad (13)$$

The root of *F(K)* will be the threshold(*K*) for the processor. The complete derivation of this process in [13].

It is clear that when the signal-to-noise ratio is zero, this last expression gives the false-alarm probability so the required threshold value may be computed.

4. GS-CFAR Processor Architecture:

The proposed processor comprises of the following main modules: tapped delay line , multiplier, threshold setting device, comparator and threshold factor as shown in Figure (3). The block of threshold setting device contains the square root block, parallel adder and block for variance estimation .Figure (4) represents the flow chart of the GS-CFAR algorithm. For illustration, a reference window of length M=8and16 is taken into consideration. The serial-in parallel-out tapped delay line consists of M reference cells, surrounding the test cell, which is located at the centre tap. Samples of the reference window are divided into M/2 symmetrical leading (right side) and lagging (left side) groups. The raw data samples of the signal to be processed is received from the radar system in digital form and fed to the tapped delay line in a serial manner. The length L of the shift register is given by L= M+1 cells . The input signal block of the Figure (3) contains a Gaussian random variable generator to generate input clutter to system to accept changes in the environmental conditions.

The MATLAB package has already a Gaussian random generator from signal processing block set, or through the instruction *randn*. Statistical tests on that variable show a negligible level of correlation between one call of this function and the other even for small lengths of random sequences.

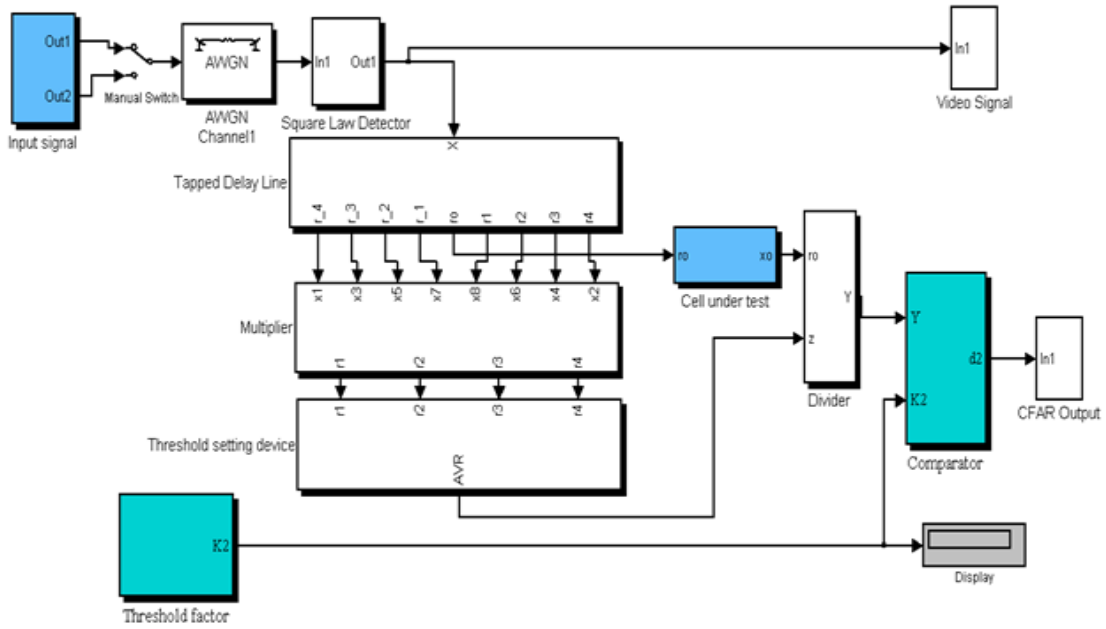


Figure 3: The GS-CFAR Processor.

To demonstrate the general CFAR characteristics, some typical signal situations are generated which are considered to be characteristic for radar applications. Figure (5a) shows noise interference and target situation when the CFAR procedure is applied, while Figure (5b) shows the output of the CFAR processor. The output of CFAR is made by a simple comparison of the output from divider with the values of threshold factor (K) .

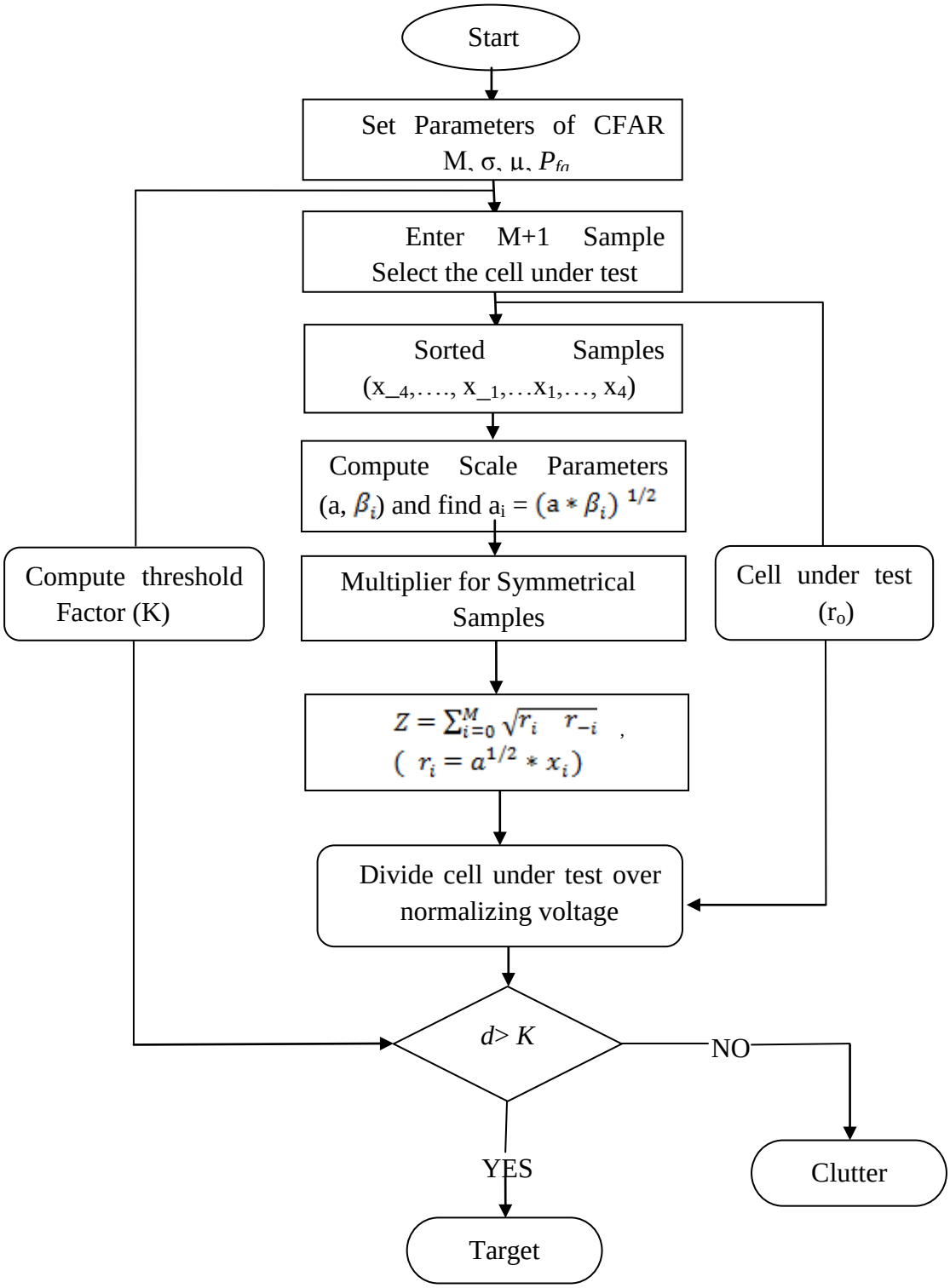


Figure 4: The flow chart of GS-CFAR Processor.

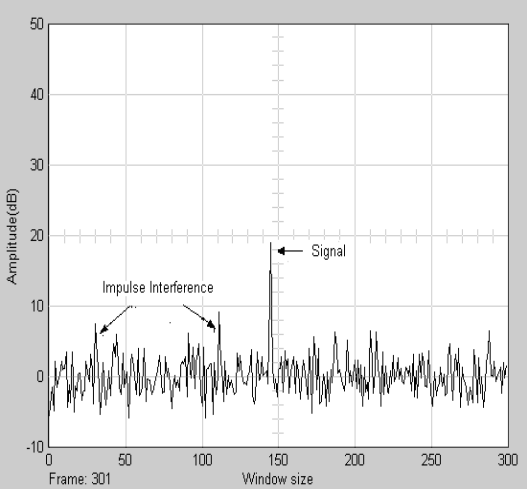


Figure5a: Input Signal

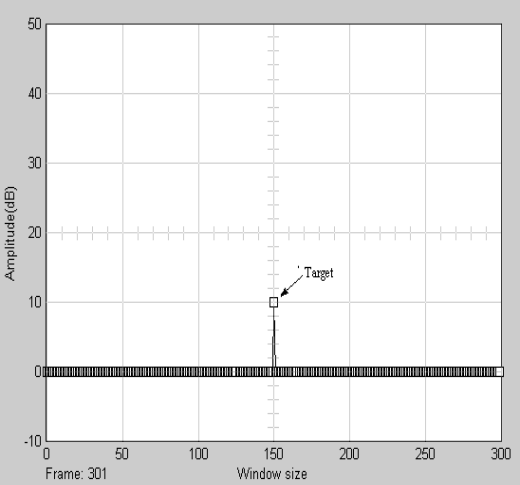


Figure 5b:Output Signal

5. Simulink HDL Coder and design procedure for GS-CFAR processor:

Simulink HDL Coder is high level design tool which generates HDL code from Simulink models and Stateflow finite state machines. Simulink HDL coder also provides interfaces to combine manually-written HDL codes, HDL Cosimulation blocks and RAM blocks in its environment, as well as Simulink HDL coder provides two types of linkage between the model and generated HDL code. ***Code-to-model*** and ***Model-to-code*** are hyperlinks which let the user to view the blocks or subsystems from which the code was generated and generated code for any block in the model respectively. The design procedure and implementation for Geometric- Symmetry CFAR Processor is done via using Simulink HDL Coder tool for generating VHDL code . The steps for design procedure and implementation are [14]:

1. Create a simulink design for GS-CFAR processor using blocks supported by HDL coder and Verifying, as shown in the Figure (3) .
2. Create control file(m file) for configurations and include(target language, project name of Quartus, family name Cyclone III, device name EP3C3…, and name of folder that includes the generated VHDL and its test bench . A code generation control file is attached to the design and executed when code generation process is invoked. If no specifications are provided, a default code generation file is created.
3. Set model parameters with the HDL coder (Solver options and Tasking option fixed-step type, discrete and single tasking).
4. Run the Simulink HDL Coder compatibility checker for the designs' suitability for HDL code generation.

5. Run Simulink HDL coder to generate HDL files for the design as well as a testbench in VHDL which are bit-true and cycle accurate.
6. Use testbench and HDL simulation tools to test the generated design.
7. Export HDL codes to synthesis and layout tools for real hardware implementation. Simulink HDL coder also generates required scripts for the Synplify family of synthesis tools and ModelSim from Mentor Graphics.
8. After generating VHDL netlist and test bench of geometric-symmetry CFAR, they are conveyed to QUARTUS II synthesis script for synthesis purposes and to Mentor Graphics ModelSim Altera 6.1g for simulation purposes.

6. FPGA Implementation results:

The GS- CFAR processor has been designed, synthesized and simulated using Altera Quartus II software targeting Cyclone III FPGA. It provides a complete design environment for designing system on a programmable chip. It offers a very rich library of parameterized modules that can be utilized to construct different processing units used in this design. The designed CFAR detector is modular, which enables the designer to test the various modules individually. Figure (6) shows the top level design file for the implementation of the Geometric- symmetry CFAR processor and pins location after successful compilation, while Figure(7) shows the block diagram of register transfer level(RTL) of Geometric- symmetry CFAR processor. Finally Figure (8) shows input/output waveforms of all system using ModelSim Altera 6.1g. Notice that, the reference signal appeares and it can be used for comparison with output data, and clock enable output (ce_out) appears. More details about the results in [13].

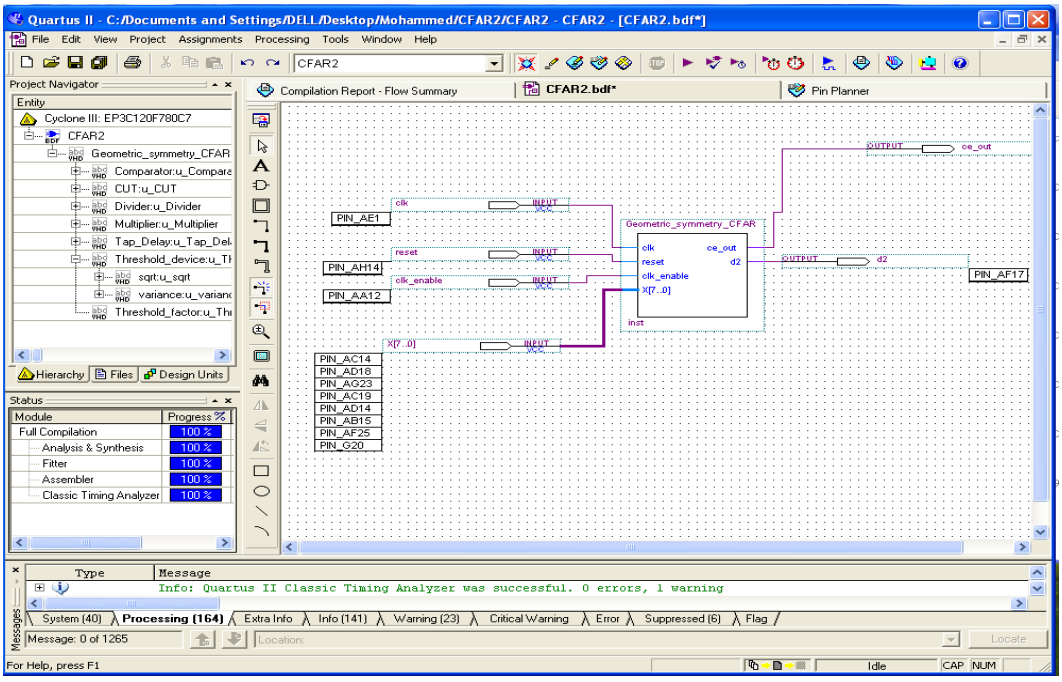


Figure 6: Top level design of Geometric- symmetry CFAR processor.

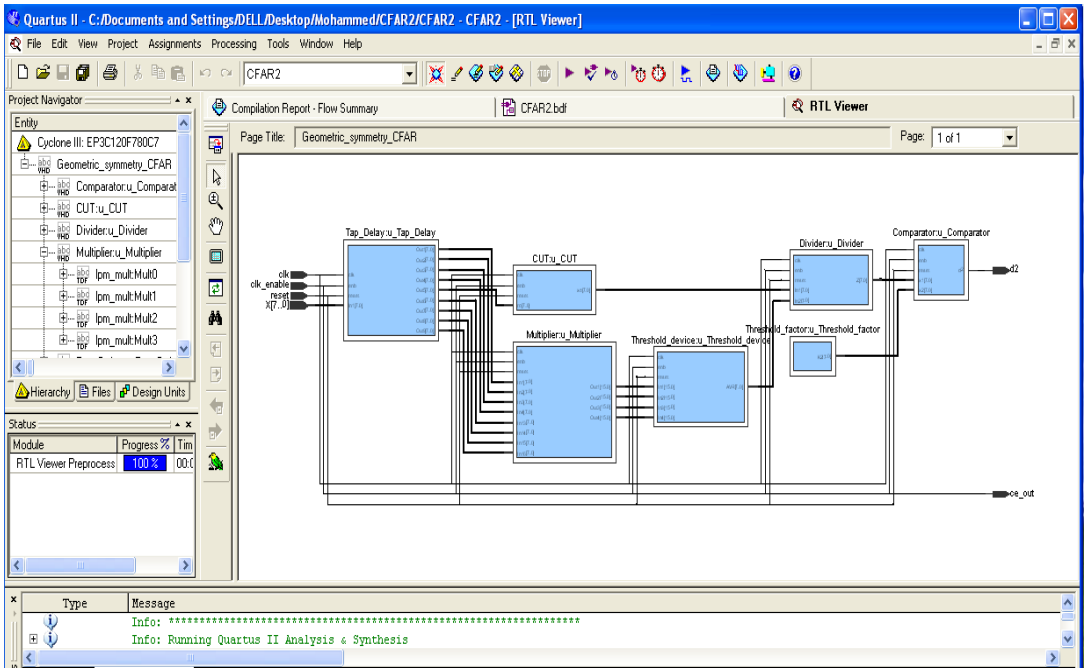


Figure 7: RTL Geometric- symmetry CFAR processor.

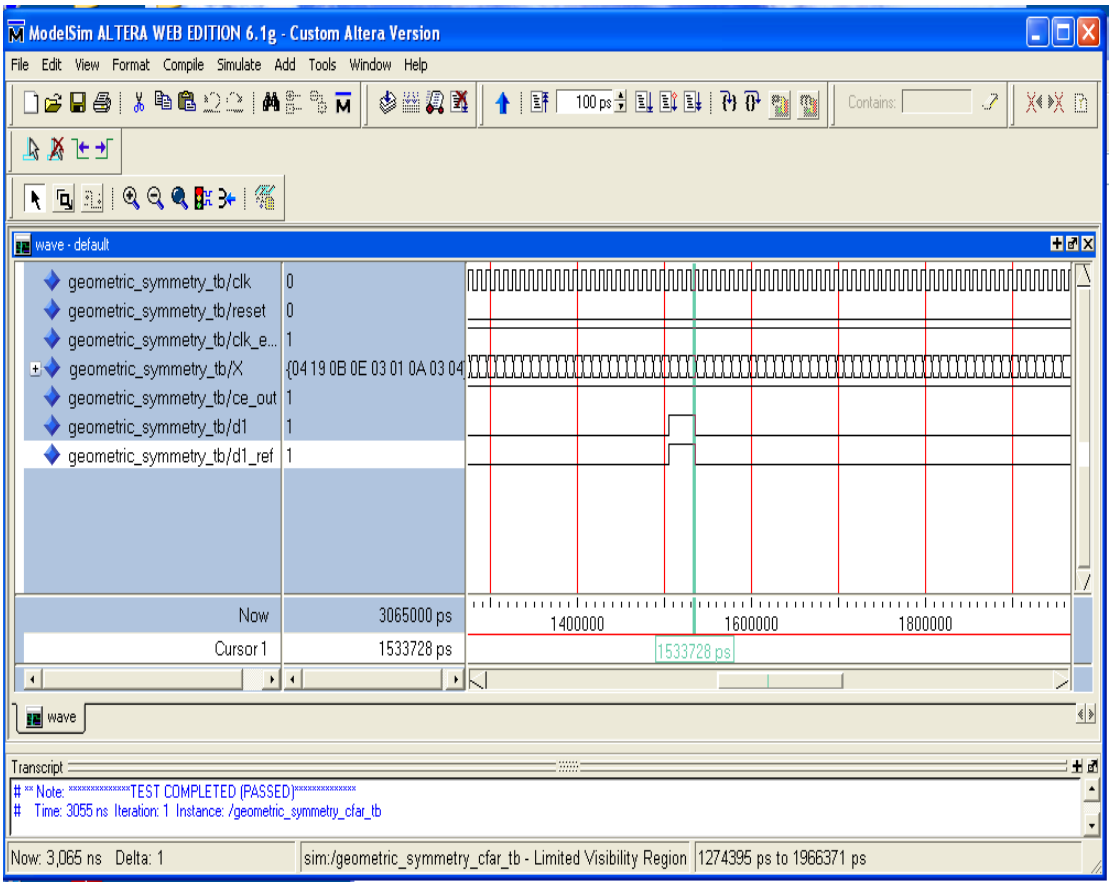


Figure 8: Input/output simulation waveforms of Geometric symmetry CFAR Processor.

The Cyclone III board provides three input clocks: 125 MHz, 50 MHz and external clock specified by the user. Therefore, the speed of design should be checked when it is implemented on Cyclone III. The FPGA implementation result shows that the processor can achieve a maximum operating frequency of 115.77 MHz, which is very close to the clock frequency of the board (125 MHz) as shown in the reports (Figures 9 and 10). The synthesis reports for the Geometric symmetry CFAR processor can be summarized in Table (1).

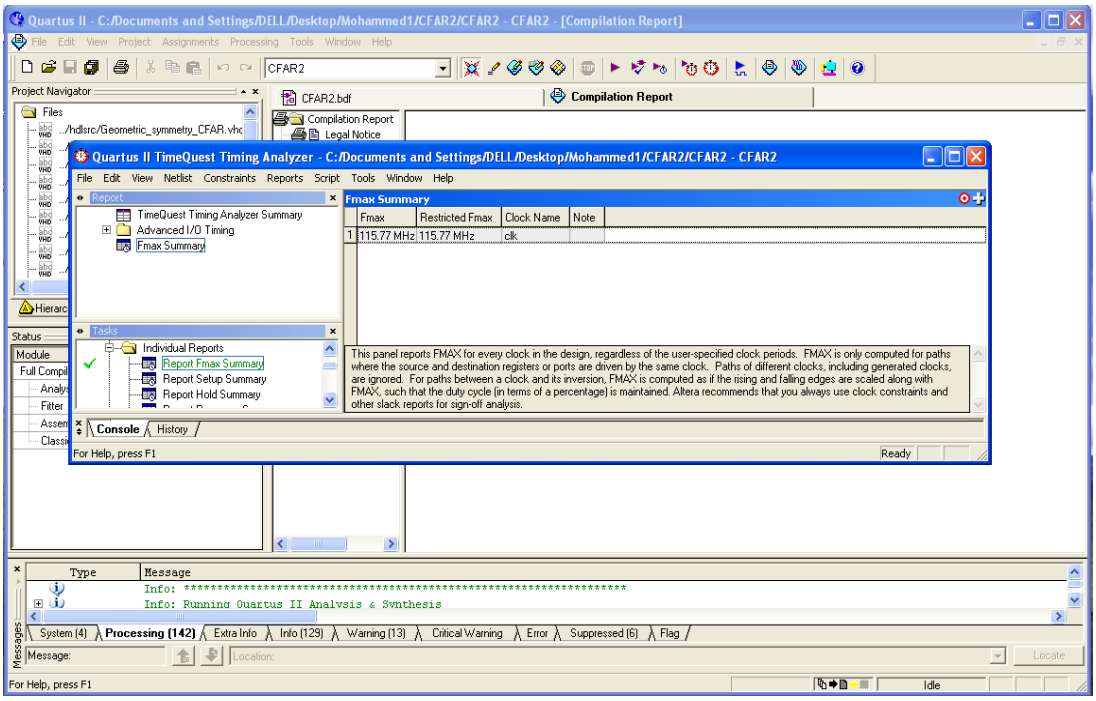


Figure 9: report for maximum frequency.

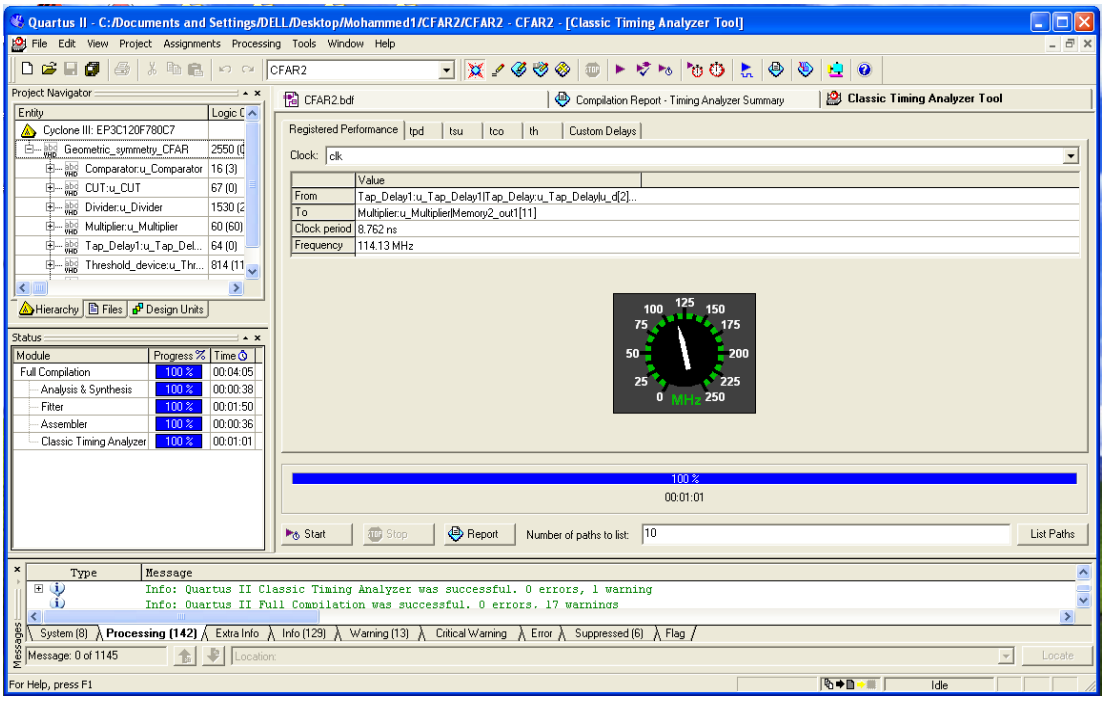


Figure 10: Report for maximum frequency by timing analyses tool.

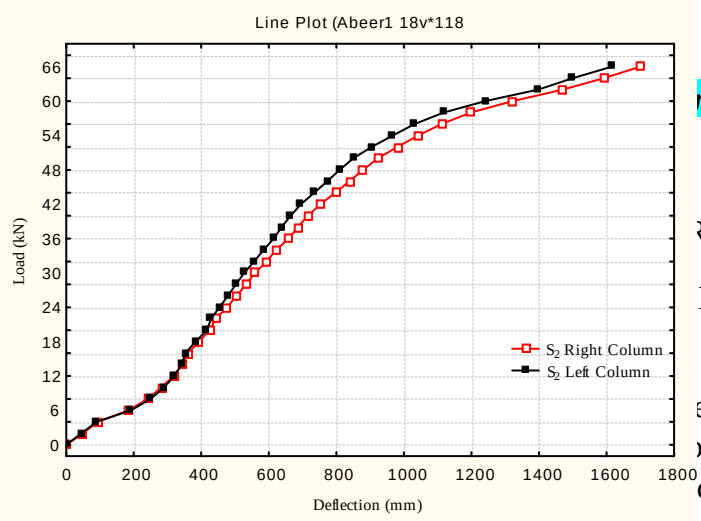
Tables (1)

Synthesis summary for the Geometric - symmetry CFAR processor targeted for CycloneIII (EP3C12F780C7) device	
Resources	**Utilization**
Total logic elements	2550/119088(2%)
Total registers	112/119088(1%)

7. Conclusions:

In this paper, the GS- CFAR processor has been designed, synthesized and simulated using Altera Quartus II software. The proposed architecture is implemented using Cyclone III (EP3C120F780C7) FPGA. FPGA proves to be an efficient hardware target for realizing the proposed CFAR processor and the implementation results demonstrate that hardware-based GS- CFAR processor provides high computational speeds. However, the total processing time for executing a single run for detecting a target depends on the number of reference cells and maximum clock frequency of the FPGA chip.

From Table (1), the system only needs little FPGA resources, and the proposed architecture allows detection of each cell under test within a delay of 8.762ns. The processing speed is not directly affected by the data words width received from the radar system, but increasing the number of samples around the cell under test will lead to increment the processing time, and the complexity of the design. Finally decreasing the word length of fixed point data as minimum as possible, will leads to reduce The FPGA resources.



Radar Signal Processing. McGraw-Hill,2005.

ll Radar Clutter", Peter Peregrinus Ltd., London,

es, S.J.Wilton, O. Mencer, W. Luk and P.Y.K. puting: Architectures and Design Methods", IEE chniques, Vol. 152, No. 2, pp. 193-207, 2005.

4. R. Cumplido, C. Torres, and S. Lopez, "A configurable FPGA-based Hardware Architecture for Adaptive Processing of Noisy Signals for Target Detection Based on Constant False Alarm Rate (CFAR) Algorithms," Global Signal Processing Conference, Santa Clara, CA, pp 214-218, September 2004.

5. R. Cumplido, C. Torres, and S. Lopez, "On the implementation of an efficient FPGA-based CFAR processor for target detection," 1st International Conference on Electrical and Electronics Engineering, Acapulco, México. pp. 214-218. 24-27, June 2004.

6. T. R. Saed, J. K. Ali, and Z. T. Yassen, "An FPGA-based implementation of CA-CFAR processor," Asian Journal of Information Technology, vol. 6, no. 4, pp. 511-514, 2007.

7. B. Magaz and M.L. Bencheikh, "An efficient FPGA implementation of the OS-CFAR processor", International Radar Symposium, Wroclaw, 21-23, pp. 1-4, May 2008.

8. P. Ramesh Babu and R. Prasanthi, "Analysis of CFAR techniques and FPGA realization for radar detection", Proceedings of International Conference on Intelligent Knowledge Systems, Troy, Turkey, 16-20, pp. 1-9, August 2004.

9. A. M. Alsuwailem, S. A. Alshebeili, and M. Alamar, "Design and implementation of a configurable real-time FPGA-based TM-CFAR processor for radar target detection", Journal of Active and Passive Electronic Devices, vol. 3, pp. 241-256, 2006.

10. A. M. Alsuwailem, S. A. Alshebeili, M.H. Alhowaish, and S.M Qasim, "Field programmable gate array-based design and realization of automatic censored cell averaging constant false alarm rate detector based on ordered data variability", IET Circuits, Devices, & Systems, vol. 3, pp. 12-21, 2009.

11. R. Nitzberg, "Constant-False-Alarm-Rate Signal Processing for Several Types of Interference", IEEE Trans. Vol. AES-8, No. 1, PP.27-34 January 1972.

12. P.Swerling, " Probability of Detection for Fluctuating Targets ", IRE Trans., IT-6: (2), PP.269-308, April, 1960.

13. M. H. Al-zubaidi,"Design and Implementation Of Unknown-Slope And Geometric-Symmetry statistics of Constant False Alarm Rate Using FPGA ",M.Sc Thesis, College of Engineering, Al-Mustansira University, 2010.

14. A.U. Irturk., " General architecture design Utility and Synthesis Tool for Optimization", Ph.D. Thesis, University of California, San Diego, USA, 2009