An FPGA-Based Implementation of HW/SW architecture for CFAR Radar Target Detector

Ridha Djemal[1], Kais Belwafi[1], Walid Kaaniche[2] and Saleh A. Alshebeili[1]
1- Electrical Engineering Department College of Engineering
King Saud University, Box 800 CP 11421 KSA
2 - ENISO, Avenue 18 Janvier 1952 Sousse Tunisia
rdjemal@ksu.edu.sa

***Abstract*— This paper presents an efficient HW/SW Codesign FPGA-based architecture of B-ACOSD CFAR target detector in log normal distribution for radar system. All CFAR system modules are analyzed in order to identify the critical ones to be optimized so that the detection process will be conducted in real-time. To compel the design optimization of CFAR Architecture, we have considered the custom instruction approach offered by Altera environment. Furthermore HW/SW architecture of the CFAR detector is carried out where the NIOS II execute the software part and communicate via the Avalon switch fabric with the hardware modules represented by the custom logic components, on-chip memories, UART and JTAG interfaces. The proposed system-on-chip is validated and tested using the Stratix IV EP4SGX230KF4C2 of Altera operating at 250MHz. Using the HW/SW approach for our embedded target detection system, we improved the performance of the architecture compared to the pure software one with a total delay of 0.45 µs.**

I. Introduction

The received signal in a radar system is computed to extract necessary information on the targets related to the object type (target or clutter) and the locations of the identified objects. If the echo is associated with a clear or empty background, it can be simply compared with a fixed threshold and the target is detected whenever the signal exceeds this threshold. However, in real cases, the echo is accompanied with clutter that varies in time and position, and therefore, in the extraction of the target, the threshold should be calculated dynamically from the local background noise/clutter power and not be a constant. In this respect, adaptive signal processing with a variable detection threshold is required to decide if the cell under test represents a target [1].

Several constant false alarm rate (CFAR) techniques used for radar systems have been proposed in the literature, such as the application of cell averaging (CA) and ordered statistics (OS) [2,3]. For example, the OS-CFAR detector, for which an appropriate reference cell is used to estimate the background noise power level, has been proposed [4]. The OS-CFAR detector has a small additional detection loss over the CA-CFAR detector for homogeneous backgrounds but can resolve closely spaced interferences. However, it requires a longer processing time than the CA-CFAR detector, and in these terms, the CA-CFAR technique is the optimum CFAR approach for homogenous environments.

Other well-developed OS algorithms, such as the Greatest-of-CFAR (GO-CFAR) algorithm and the Smallest-of-CFAR (SO-CFAR) algorithm [5], the Censored Mean-Level Detector (CMLD) [6], and other OS algorithms [7,8], have been studied for different scenarios. However, the assumption of a homogenous environment is no longer valid when the number of targets changes. In such situations, the performance of the CA-CFAR processor is seriously degraded. Various classes of CFAR techniques have been proposed to enhance robustness against a non-homogeneous environment for different applications [9, 10] according to the background distribution. However, these implementations were pure software without any real-time analysis.

By contrast, a major concern in radar signal processing is to maintain the false alarm rate at a desired constant value. To achieve a CFAR, the processed target signal is compared with the cell under test with an adaptive threshold detector requiring good knowledge of the statistics (Rayleigh distribution, Weibull distribution, K-distribution or lognormal distribution) of the clutter. Another concern is to carry out target detection within a limited delay to satisfy real-time constraints, especially for high-resolution target detection. To accomplish this task, a design exploration of the solution space should be carried out for the CFAR detector.

Although the theory of CFAR radar detection has been well established, the hardware implementation for a real-time environment is still beyond currently available high-computational signal processing operations. Owing to the real-time constraints of target detection by a high-resolution radar system, system-on-chip (SoC) architecture is an attractive solution for the real-time CFAR processor. In SoC architecture, all components of a computer, such as the processor, glue logic and memories, are integrated onto a single chip and operate in an organized manner. Recent advances in field programmable gate array (FPGA) technology have made SoC fabrication faster and easier.

In this paper, a Nios II processor FPGA-based platform is used to implement Backward Automatic Censored Ordered Statistics Detector (B-ACOSD) CFAR algorithm. This detector should be able to operate robustly to detect automatically a target and to determine the number of interferences close to the target for a lognormal clutter distribution. The SoC architecture of the CFAR detector is implemented on an Altera Stratix IV board with HW/SW configuration based on the integration of the NIOS II core processor and custom logics described in VHDL language. The Avalon switch fabric is also integrated within the same FPGA to interconnect the system-on chip components detailed

978-1-4577-2209-7/11/$26.00 ©2011 IEEE

in section IV. The proposed CFAR system is a typical HW/SW example built in such way to achieve a processing delay of less than 500 ns, which is suitable for high-resolution radar applications in a desert environment [11].

The rest of this paper is organized as follows. In Section 2, the fundamentals of CFAR theory and related research on hardware realization for some types of CFAR algorithms are described. Section 3 presents mathematical formulas and algorithms for B-ACOSD. The SoC HW/SW FPGA-based design architecture for the detectors is explained in Section 4. Section 5 presents the co-simulation results and the realization of the target detection embedded system. In section 6, conclusions and future research plans are discussed.

II. Related Work

For a radar system, a detection method is needed to determine the power threshold above which any return can be considered as coming from a target. If the threshold is too low, then more targets are detected, but the number of false alarms is high. Conversely, if the threshold is too high, then fewer targets are detected but the number of false alarms is low. For most radar detectors, the threshold is set to realize a required probability of a false alarm P_{fa}. If the background of noise, clutter, and interference are constant in time and space, then a fixed threshold level that provides a specified probability of a false alarm can be chosen. However, under natural conditions, unwanted clutter and interference sources affect the noise level spatially and temporally. In this case, an adaptive threshold can be used, where the threshold level is raised and lowered to maintain a constant probability of a false alarm. This is called CFAR detection.

A typical CFAR processor is shown in Fig. 1. The input signals are set serially in a shift register. The content of the cells surrounding the cell under test (X_0) are processed using a CFAR processor to obtain the adaptive threshold T. The value of X_0 is then compared with the threshold to make the decision. The cell under test is declared a target if its value exceeds the threshold value.

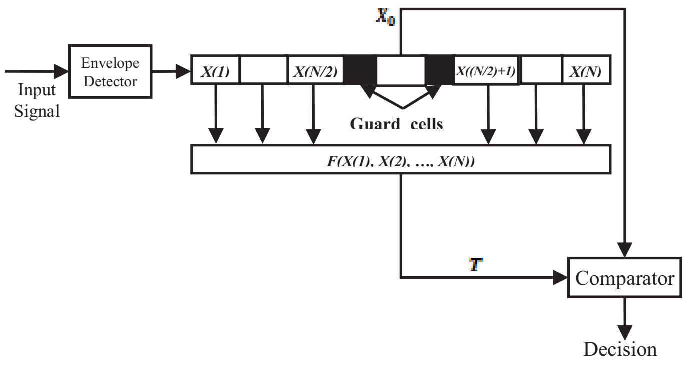


Fig. 1. Block diagram of a typical CFAR algorithm

The first and simplest CFAR detector is the CA-CFAR detector [3], for which the adaptive threshold is obtained from the arithmetic mean of the reference cells. Many CFAR algorithms have been recently developed. We can categorize a CFAR algorithm into one of three models according to the clutter power distribution and the interfering targets.

- When there is transition in the clutter power distribution, we can use, for example, greatest-of-selection logic for the CA-CFAR detector (GO-CFAR) [12] to control the increase in the probability of a false alarm. If one or more interfering targets are present, the GO-CFAR detector performs target detection poorly and it is suggested that an SO-CFAR algorithm employing smallest-of-selection logic is used instead for the CA-CFAR detector [13].
- When the clutter background is composed of homogeneous white Gaussian noise plus interfering targets, the CMLD can be used as a target detector. The CMLD censors target samples and estimates the noise level from the remaining noise sample. In addition, the trimmed mean-level CFAR (TM-CFAR) detector [4] implements trimmed averaging after ordering the samples in the window. When the number of interfering targets is not known a priori, the generalized CMLD (GCMLD), for which the number of interfering targets is determined and their corresponding samples are then sampled, can be used as well as the OS-CFAR detector, which chooses one ordered sample to represent the estimated noise level in the cell under test. If there is not only transition in the clutter power distribution but also interfering targets, a commonly used technique is the generalized two-level CMLD (GTL-CMLD) [14], which uses an automatic censoring algorithm of the unwanted samples when both interfering targets and extended clutter are present in the reference window of the cell under test.
- The last category deals with non-Gaussian clutter distribution. The lognormal distribution, Weibull distribution, gamma distribution, and K-distribution have been used to represent the envelope-detected non-Gaussian clutter distribution. Works on CFAR detection for Weibull clutter have been reported. For example, the maximum-likelihood CFAR (ML-CFAR) algorithm has been presented [15] and its performance has been analysed for the case in which both the scale and shape parameters are unknown. Furthermore, the optimal Weibull CFAR (OW-CFAR) algorithm where statistics test are expressed according to the estimate of the mean power of the Weibull clutter, has been proposed [16].

Theoretical developments 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 [16-18]. 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 [19]. In [20], CA-CFAR and OS-CFAR are combined and implemented in FPGA. In [21], 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.

Alsuwailem et al. [22] implemented an automatic censoring CFAR detector called Automatic Censored Cell Averaging (ACCA) ODV CFAR. However, the implementation does not consider the real time aspects where an offline validation is done without allowing interactive interaction with the architecture. Furthermore not standard interface is given in order to facilitate the communication with the Radar System environment. Winkler et al. [23] used SoC with reconfigurable processor inside for an automotive radar sensor. The processor is responsible for controlling the custom logic and IO tasks.

Recent automatic censoring detectors called ACOSD CFAR have introduced by Almarshad et al. [24]. The algorithm is able to make automatic censoring of unknown number of interfering targets in log-normal clutter. Because of increase in radar resolution, the log-normal distribution becomes more reliable to represent the amplitude of clutter than Rayleigh distribution. Meanwhile, the automatic censoring algorithms developed for Rayleigh clutter as presented in [11] and [22] cannot straightforwardly be extended to the case where clutter samples are drawn from log-normal distribution. In the rest of this paper, we will consider the ACOSD-CFAR proposed in [24] Algorithms for SoC Implementation

III. THE ACOSD DETECTION ALGORITHMS

In ACOSD CFAR algorithms, the detection consists of two steps: removing the interfering reference cells (censoring step) and the actual detection (detection step). Both steps are performed dynamically by using a suitable set of ranked cells to estimate the unknown background level and set the adaptive thresholds accordingly. This detector does not require any prior information about the clutter parameters nor do they require the number of interfering targets. In a CFAR processor, the radar outputs $\{X_i : i = 0,1,\dots,N\}$ are stored in a tapped delay line. The cell with the subscript $i = 0$ is the cell under test, where it contains the signal which should be detected as a target or not. The last N surrounding cells are the auxiliary cells used to construct the CFAR procedure. In the ACOSD CFAR, the N surrounding cells are ranked in ascending order according to their magnitudes to yield

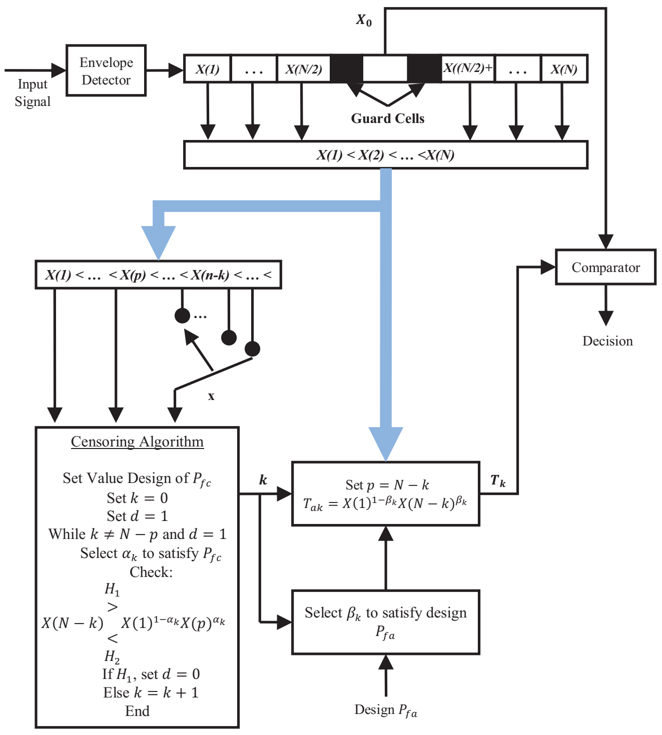
$$X(1) \leq X(2) \leq \cdots \leq X(p) \leq \cdots \leq X(N) \qquad (1)$$

After sorting, the sorted cells are then sent to detection stage.

In this stage B-ACOSD and F-ACOSD have different algorithm. In the B-ACOSD algorithm, sample $X(N)$ is then compared with the adaptive threshold T_{c0} defined as

$$T_{c0} = X(1)^{1-\alpha_0} X(p)^{\alpha_0} \qquad (2)$$

Where $X(p)$ is the p^{th} largest sample and α_0 is a constant chosen to achieve the desired probability of false censoring (P_{fc}). It is found that values of $p > N/2$ yield reasonable good performance in detection [26]. If $X(N) < T_{c0}$, the algorithm decides that $X(N)$ corresponds to a clutter sample without interference, and it terminates. If, on the other hand, $X(N) > T_{c0}$, the algorithm decides that the sample $X(N)$ is a return echo from an interfering target. In this case, $X(N)$ is censored and the algorithm proceeds to compare the sample $X(N-1)$ with the threshold:



X_0
Input Signal
Envelope Detector
X(1)
...
X(N/2)
X((N/2)+1)
X(N)
Guard Cells
X(1) < X(2) < ... < X(N)
X(1) < ... < X(p) < ... < X(n-k) < ... <
x
Comparator
Decision
Censoring Algorithm
Set Value Design of P_{fc}
Set $k = 0$
Set $d = 1$
While $k \neq N - p$ and $d = 1$
Select α_k to satisfy P_{fc}
Check:
$X(N-k) \underset{H_2}{\overset{H_1}{\gtrless}} X(1)^{1-\alpha_k} X(p)^{\alpha_k}$
If H_1, set $d = 0$
Else $k = k + 1$
End
k
Set $p = N - k$
$T_{ak} = X(1)^{1-\beta_k} X(N-k)^{\beta_k}$
T_k
Select β_k to satisfy design P_{fa}
Design P_{fa}

Fig. 2. Block diagram of the B-ACOSD algorithm

$$T_{c1} = X(1)^{1-\alpha_1} X(p)^{\alpha_1} \qquad (3)$$

to determine whether it corresponds to an interfering target or a clutter sample without interference.

At the $(k+1)^{th}$ step, the sample $X(N-k)$ is compared with the threshold T_{ck} and a decision is made according to the test,

$$X(N-k) \underset{H_0}{\overset{H_1}{\gtrless}} T_{ck}\,;\ 0 \leq k < N-p \qquad (4)$$

Where $T_{ck} = X(1)^{1-\alpha_k} X(p)^{\alpha_k}$.

Hypothesis H_1 represents the case where $X(N-k)$, and thus the subsequent samples $X(N-k+1)$, $X(N-k+2)$, … , $X(N)$ correspond to clutter samples with interference, while H_0 denotes the case where $X(N-k)$ is a clutter sample without interference. The successive tests are repeated as long as the hypothesis H_1 is declared true. The algorithm stops when the cell under investigation is declared homogeneous (i.e., clutter sample only) or, in the extreme case, when all the $N-p$ highest cells are tested; that is, $k = N-p$. Fig. 2 shows the block diagram of the B-ACOSD algorithm.

In detection step, the cell under test X_0 is compared with the threshold T_{ak} to decide whether a target is present or not according to

$$X_0 \underset{H_0}{\overset{H_1}{\gtrless}} T_{ak}\,;\ 0 \leq k < N-p+1 \qquad (6)$$

Hypothesis H_1 denotes the presence of target in the test cell, while hypothesis H_0 denotes there is no target.

In B-ACOSD CFAR, the threshold T_{ak} is defined as,

$$T_{ak} = X(1)^{1-\beta_k} X(N-k)^{\beta_k} \quad (7)$$

Where the value of β is selected so that the design probability of false alarm (P_{fa}), and k is the number of interfering targets found in censoring step.

Threshold Values

The threshold selection is a key element in the algorithms [26]. These thresholds should be selected in order to reach low probability of hypothesis test error in a homogeneous environment. Monte Carlo simulation with 500,000 independent runs was employed to obtain the threshold values by maintaining low value of P_{fa} and P_{fc}. Table 1 gives the threshold parameters α_k and β_k obtained using B-ACOSD with $P_{fa} = 0.001$ and $P_{fc} = 0.01$.

Table 1: Threshold parameters for B-ACOSD ($P_{fa} = 0.001$ and $P_{fc} = 0.01$)

(N,p)		*k*					
		1	*2*	*3*	*4*	*5*	*6*
(16,12)	α_k	2.596	2.038	1.709	1.443	-	-
	β_k	1.635	1.889	2.12	2.37	2.64	-

IV. HW/SW FPGA Based Design of The B-ACOSD CFAR Architecture

A. Generic HW/SW Architecture for Radar System

The B-ACOSD algorithm is implemented using the Stratix IV board of Altera integrating an FPGA in which we have embedded the Nios II processor in its fast version to execute the software modules described in ANSI C, where the hardware modules have been developed using the VHDL code through the custom instruction approach. In fact, to accelerate time-critical software algorithms, we proceed by adding custom instructions to extend the embedded Nios II processor instruction set. Custom instruction approach allow the designer to reduce a complex sequence of standard instruction to a single instruction implemented in hardware and represented as custom logic component adjacent to ALU of the Nios II processor. Furthermore, Custom instruction is used to optimize software inner loops and computation-intensive related to the CFAR application. In addition the custom instruction gives us the ability to tailor the system-on-chip architecture integrating Nios II core processor to meet the critical time requirements related to the B-ACOSD architecture.

The embedded CFAR architecture integrates the following devices as depicted in Fig.3.

- A software fast Nios II core processor with a 32-bit data path and 8-KB data cache, 16-KB instruction cache and a Master port.

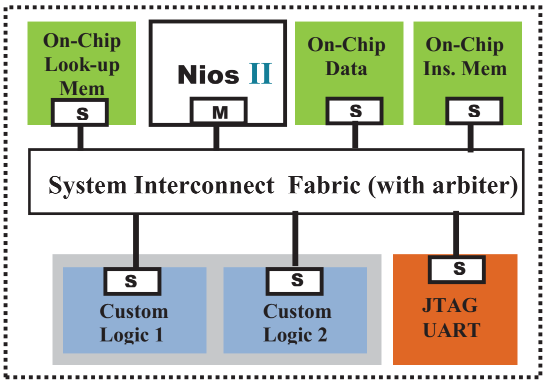


Fig.3. B-ACOSD Nios II-based embedded System:

- Custom logics hardware modules described in HDL language at RTL level and including only slave ports to communication with the Avalon bus of our embedded system.
- System Interconnect Fabric (SIF) which consists on the Avalon Memory Mapped bus to interconnect the custom instruction, NIOS II processor and all others interfaces within the same system-on-chip.
- JTAG UART interfaces with simplified configuration allowing target connection downloaded software and having an interface for on-chip trace data.
- On-chip memories with a size of 128kx32 and 64 kx16 and additional timers to monitor all timing aspect with regards to the CFAR architecture.

B. HW/SW Design flow

To design the B-ACOSD System on Chip, we propose to follow a typical HW/SW design flow as shown in Fig. 4.

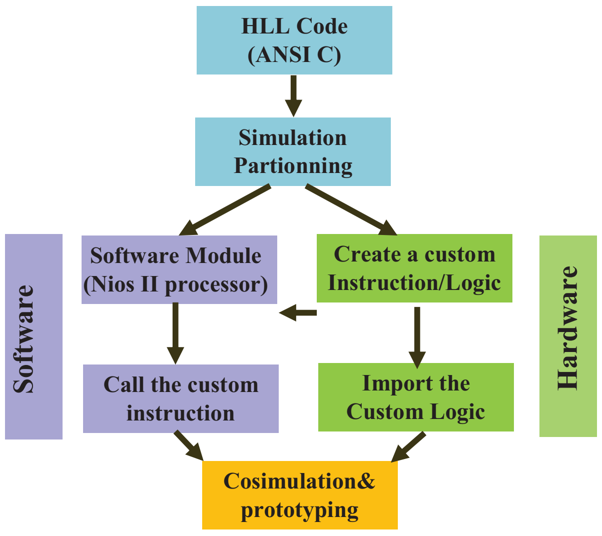

HLL Code (ANSI C)
Simulation Partionning
Software
Software Module (Nios II processor)
Create a custom Instruction/Logic
Hardware
Call the custom instruction
Import the Custom Logic
Cosimulation& prototyping

Fig.4. Typical HW/SW Design Flow

1. The first step consists on the design of a pure software architecture using a High Level Language (HLL) like ANSI C. The code is running over the Nios II processor within the FPGA using microC/OS II operating system in order to identify the critical component to be exported as hardware modules.

2. Create the custom instruction modules in HDL language such as VHDL or verilog and simulate before their integration within the design process.

3. Import the custom instructions into the design by adding them to the SOPC as an internal instruction accessible by the Nios II processor. These custom instructions can be called using ANSI C or assembly languages.

4. Co-simulate the system including Nios II, custom logics, and embedded memories with additional interfaces as depicted in Fig.3 using the stratix IV prototyping Board.

V. Experimental Implementation and Validation

The B-ACOSD CFAR architecture has been built around an FPGA single chip, with bloc diagram architecture similar than the one presented in Fig.3 where the architecture is mapped onto a HW/SW configuration. In this respect, the non critical tasks of our algorithms are executed over the Nios II core processor where the critical ones are exported as custom logic described in VHDL. First, we have implemented the architecture as a pure software core running over the Nios II processor and we have extracted the execution time for the three main components such as sorting, censoring and detection modules. In the second step, we have exported the critical components as a hardware architecture described in VHDL and interconnected to the Nios II processor via the Avalon interface. The following table gives the execution time of each component of the B-ACOSD architecture as a pure solution embedded into the Stratix IV board.

Table 2: Software Execution Time of B-ACOSD module

B-ACOSD Modules	Delays in µs
Sorting module	18
Censoring module	104
Detection module	21
Total delay	143

According to the computation results, we have found that the critical point is located in computing T_{ck} and T_{ak} which represents the threshold computation belonging to the censoring module and a part of the detection one. The sorting module represents also a critical task. We decide to export two custom instructions to integrate the sorting module and the censoring module with a part of the detection. The rest of the algorithm is dedicated for the Nios II core processor. In the mean time we tried to optimize the sorting technique as well as the look up table organization by approximating the resolution and decreasing the on-chip memory resources without increasing the constant false alarm rate of the B-ACOSD during the computation of the log function.

In terms of hardware, computation of the exponential equation (5) and (7) is hard and high cost, especially when related with floating point calculation. To reduce the hardware complexity and computational time, equation (5) and (7) are converted respectively into logarithmic form as follows:

$$\log T_{ck} = (1-\alpha_k)\cdot\log X(1) + \alpha_k\cdot\log X(p) \quad (8)$$
$$\log T_{ak} = (1-\beta_k)\cdot\log X(1) + \beta_k\cdot\log X(N-k) \quad (9)$$

In this form, power computation becomes a simple multiplication, and the multiplication becomes an addition. In addition, because the logarithmic computation in hardware is a complex and slow task, the logarithmic computation is simplified using a look-up table. The look-up table contains range of a lognormal distribution with $\mu = 1$ and $\sigma = 1.1$ as suggested in [26], based on real radar input data measurement. The proposed LUT support up to 2000 values of log for our implementation which allow a resolution following the number representation change to logarithmic form, test cell value was also converted accordingly. This logarithmic conversion also performed using the same look-up table mentioned above. The look-up table resides on 32K on-chip ROM inside FPGA. The data distribution resolution in the 32K on-chip ROM is 0.0610. MATLAB fixed-point B-ACOSD CFAR simulation with this resolution gives censoring results as good as its real-number simulation.

To accelerate the execution time, we have replaced the bubble sorting technique by a Parallel Range Computing (PRC) technique because it gives a reduced delay in software implementation, as depicted in table 3, and its hardware implementation allows s high degree of parallelism and an interesting timing.

Table 3: Execution time over the Nios II processor of different sorting techniques

Sorting technique	Bubble	Even/Odd	PRC
Nios II Execution time	18 µs	10 µs	1.28 µs

After integrating the overall architecture including the hardware and the software modules, we have simulated the design and evaluated their complexity before and after adding the custom instruction. In table 4, we presented the complexity of only the Nios II core processor downloaded within the FPAG to execute the A-COSD detector. Where table 5, indicated resources of the total system with their hardware and software parts in terms of combinational LUTs, dedicated logic registers and On-chip memories. We note the complexity of the design is increased reasonably by adding the custom logics as hardware components. However, the processing time is decreasing from 143µs to 0.45 µs which is a big delay saving.

Table 4: Nios II processor resources (pure software solution)

	Without Custom Inst.	With Custom Inst.
Comb.LUTs	1474 (0.8%)	1683 (1%)
Logic Reg.	1254(0.7%)	1257 (0.7%)
On-chip Memory	94528 (7%)	94528 (7%)

Table 5: B-ACOSD SoC resources (HW/SW Solution)

	Without Custom Inst.	With Custom Inst.
Comb. LUTs	3368 (2%)	4723 (3%)
Logic Reg.	2486(1%)	3074 (2%)
On-chip Memory	4547584 (31%)	4547400 (31%)

The FPGA implementation result for SoC with $N = 16$ and $p = 12$ shows that the NIOS processor can achieve a maximum operating frequency of 250 MHz. After a software implementation of the B-ACOSD, we have experienced a design exploration by exporting the critical components in terms of delays to custom logic blocs connected to the Nios II processor via the Avalon interface. The time delay of the total architecture decreases up to 0.45 μs. This processing time is below the real-time requirements 0.5 μs [24]. For demonstration purposes, the default configuration of the proposed system-on-chip employs 16-bit data samples, 16 reference cells, and 2 guard cells. The HW/SW architecture has been tested and verified by generating 256 data samples drawn from an exponential distribution. The data set is down loaded to 16 X 256 ROM. The output result is saved in 1 X 256 RAM.

VI. Conclusion

In this paper, a hardware software implementation of B-ACOSD CFAR target detector for lognormal clutter is reported. This proposed system on chip system has the advantages of being simple, and fast with low development cost. The performance of the prototype hardware setup proved the concept of the co-design within a reasonable time of design. We have considered the custom instruction approach to export and design the hardware components having critical delays. The proposed FPGA implementation integrates NiosII, custom logics, on-chip memories, Avalon switch fabric and additional interfaces. The proposed architecture allows the detection of each cell under test within a delay of 0.45 μs, below the real-time requirement of 0.5μs. The proposed architecture has been synthesized and validated using the Stratix IV development Kit (EP4SGX230KF4C2 device) over which we have measured all timing constraints.

References

[1] M. Barkat, Signal Detection and Estimation. Norwood, MA: Artech House, 2005.

[2] R.S. Johnson H.M. Finn, "Adaptive detection mode with threshold control as a function of sampled clutter-level estimates," RCA Review, vol. 29, pp. 414-463, Sept. 1968.

[3] H. Rohling, "Radar CFAR thresholding in clutter and multiple target situations," IEEE Trans. Aerospace and Electronics Systems, vol. 19, no. 4, pp. 608-621, Jul. 1983.

[4] S.A. Kassam P.P. Gandhi, "Analysis of CFAR processors in nonhomogenous background," IEEE Trans. Aerospace and Electronics Systems, vol. 24, no. 4, pp. 427-455, July 1988.

[5] H.A. Meziani and F. Soltani, "Performance analysis of some CFAR detectors in homogenous and non-homogenous Pearson-distributed clutter," Signal Processing, vol. 86, pp. 2115-2122, April 2006.

[6] G.M. Dillard J.T. Rickard, "Adaptive detection algorithms for multiple target situations," IEEE Trans. Aerospace and Electronics Systems, vol. 13, no. 4, pp. 383-343, Jul. 1977.

[7] A. Mezache and F. Soltani, "A novel threshold optimization of ML-CFAR detector in Weibull clutter using Fuzzy-neural networks," Signal Processing , vol. 87, pp. 2100-2110, Feb. 2007.

[8] M. Barkat T.Larouissi, "Performance Analysis of order-statistic CFAR detectors in time diversity systems for partially correlated chi-square targets and multiple target situations," Signal Processing, vol. 86, no. 7, pp. 1617-1631, July 2006.

[9] M.A. Khalighi and M. H. Bastani, "Adaptive CFAR processor for nonhomogenous environemnt," IEEE Trans. Aerospace and Electronics Systems, vol. 36, no. 3, pp. 889-897, Jul. 2000.

[10] P. Henttu, and M. Juntti H. Saarnisaari, "Iterative multidimensional impulse detectors for communications based on the classical diagnostic methods," IEEE Trans. Communication, vol. 53, no. 3, pp. 395-398, Mar. 2005.

[11] S. Alshebeili, S.M. Alhumaidi, and A. M. Obied Y.M. Seddiq, "FPGA-Based Implementation of a CFAR Processor using Batcher's sort and LUT arithmetic," in 4th International Design and Test Workshop (IDT), Riyadh-KSA, 2009, pp. 1-6.

[12] J.H. Sawyers V. G. Hansen, "Detectability loss due to greatest of selection in a cell-averaging CFAR," IEEE Trans. Aerospace and Electronics Systems, vol. 16, pp. 115-118, Jan. 1980.

[13] M. Weiss, "Analysis od some modified cell-averaging CFAR processors in multiple target situations," IEEE Trans. Aerospace and Electronics Systems, vol. 15, no. 1, pp. 102-114, Jan. 1982.

[14] S. D. Himonas, and P. K.Varshney M. Barkat, "CFAR detection for multiple target situations," IEE Proceeding, Part F: Radar and Signal Processin, vol. 136, no. 5, pp. 193-210, Oct.1989.

[15] R. Ravid and N. Levanon, "Maximum-likelihood CFAR for Weibull background," [16] R. Ravid and N. Levanon, “Maximum-likelihood CFAR for Weibull background,” IEE Proceeding, Part F: Radar and Signal Processing , vol. 139, no. 3, pp. 256-264, Jun. 1992.

[16] V. Anastassopoulos and G. Lampropoulos, "Optimal CFAR detection in Weibull clutter," [17] V. Anastassopoulos and G. Lampropoulos, “Optima IEEE Trans. Aerospace and Electronic System, vol. 31, no. 1, pp. 52-64, Jan. 1995.

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

[18] M.L.Bencheikh B. Magaz, "An Efficient FPGA Implementation of the OS-CFAR Processor," in International Radar Symposium, Wroclaw, 2008, pp. 1-4.

[19] R. Cumplido, C. Uribe and F. Del Campo R. Perez, "A versatile hardware architecture for a constant false alarm rat processor based on a linear insertion sorter," Digital Signal Processing, vol. 20, pp. 1733-1747, 2010.

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

[21] 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, no. 3-4, pp. 241-256, 2008.

[22] A. M. Alsuwailem, M.H. Alhowaish, S. A. Alshebeili, 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, no. 1, pp. 12-21, Feb. 2009.

[23] J. Detlefsen, U. Siart, J. Buchlert, and M. Wagner V. Winkler, "FPGA-based signal processing of an automotive radar sensor," in First European Radar Conference, Amsterdam, 2004, pp. 245-248.

[24] M. Barkat, and S. A. Alshebeili M. N. Almarshad, "A Monte Carlo simulation for two novel automatic censoring techniques of radar interfering targets in log-normal clutter," Signal Processing, vol. 88, no. 3, pp. 719-732, Mar. 2007.

[25] R. Djemal, "A real-time FPGA-based implementation of target detection technique in non-homogenous environement," in Design and Technology of Integrated System in Nanoscale Era (DTIS), Hammamet- Tunisia, 2010, pp. 1-6.

[26] R. Djemal and S. Alshebeili I. Rosyadi, "Design and Implementation of Real-time Automatic Censoring Systen on Chip for Radar Detection," in World Academic of Science, Engineering and Technology (WASET), Penang - Malaysia, 2009, pp. 318-324.