Int. J. Electron. Commun. (AEÜ) 67 (2013) 301–312

Contents lists available at SciVerse ScienceDirect

International Journal of Electronics and Communications (AEÜ)

journal homepage: www.elsevier.com/locate/aeue

A novel hardware/software embedded system based on automatic censored target detection for radar systems

Ridha Djemal [a,*], Kais Belwafi [b], Walid Kaaniche [b], Saleh A. Alshebeili [c]

[a] Department of Electrical Engineering, King Saud University, Riyadh 11421, Saudi Arabia
[b] Department of Electrical Engineering, ENISo Sousse, Tunisia
[c] KACST Technology Innovation Centre in RF and Photonics (RFTONICS), King Saud University, Riyadh 11421, Saudi Arabia

ARTICLE INFO

Article history:
Received 7 June 2012
Accepted 11 September 2012

Keywords:
CFAR
FPGA
Radar
Embedded system
Co-design

ABSTRACT

This paper presents a practical design exploration for a new application related to real-time, high-resolution target detection for radar systems. In this paper, an embedded architecture that combines the hardware and software components in a single platform is experienced using a field programmable gate array FPGA-based PC-board. The detection process utilises three techniques: namely, automatic censored ordered statistics detection (ACOSD), cell averaging (CA) and ordered statistics (OS) CFAR techniques, all of which operate in parallel to increase the accuracy of the detection and to reduce the false-alarm rate for both homogeneous and non-homogeneous environments. A prototype of the embedded system detector has been implemented for homogeneous and non-homogeneous environments on Stratix IV FPGA Board. The prototype operates at 200 MHz and performs real-time target detection with an execution delay of 0.27 μs, which is less than the critical time (0.5 μs) for high-resolution detection.

1. Introduction

Significant studies of CFAR techniques have been carried out over the past decade to address detection problems depending on the environments in which radar systems operate. In these systems, target detection techniques require linear and nonlinear operations. The sorting technique is an example of a nonlinear operation, which consists of ranking, in ascending or descending order, the range cells according to their magnitude to yield N-ordered samples. Furthermore, the censoring technique applied to choose one ordered sample to represent the estimated noise level in the cell under test is an example of a nonlinear operation. However, some target detectors, such as the cell-averaging CFAR (CA-CFAR) and the greatest-of-selection CFAR (GO-CFAR) detector, are used to control the increase in the probability of false alarm. The architecture of the GO-CFAR detector [1] is based on linear operations by calculating the arithmetic mean of the amplitude within the window cells. All of the target detection techniques have been developed to increase the target detection probability under several environment conditions, especially those related to the region of clutter transitions and multiple-target situations.

Several CFAR processors have been proposed for use in radar systems: namely the cell averaging (CA) and the ordered-statistics (OS) processors. The CA-CFAR processor is the optimal CFAR processor in homogeneous environments. However, the assumption of a homogeneous environment is no longer valid when the number of users changes abruptly (the presence of multiple-access interference) and/or when there is fading. In such situations, the performance of the CA-CFAR processor is seriously degraded. Various classes of CFAR techniques have been proposed to enhance the robustness of this processor against non-homogeneous environments for different applications [2]. In particular, OS-based CFAR detectors have been introduced [3,4] and have proven to provide good performance in the presence of multiple-access interference (MAI). In the OS-CFAR detector, an appropriate reference cell is used to estimate the background noise power level. Even if, in homogeneous backgrounds, the OS-CFAR detector has a small additional detection loss compared with the CA-CFAR detector, it can resolve closely spaced interferences. However, the OS-CFAR detector requires a longer processing time than the CA-CFAR detector. With advances in certain technologies (DSP, ASIC, FPGA, and embedded systems), it is now possible to carry out hardware-based implementations of such complex CFAR systems with an increased capability and an acceptable time delay.

The theoretical aspect of CFAR target detection has been well developed, and a few attempts to implement CFAR processors have been reported in the literature. A parallel-pipelined hardware implementation of a CA-CFAR-based target detection system in a noisy environment using the TMS320C6203 DSP and FPGA devices has been reported [5]. The processing time achieved for

* Corresponding author. Tel.: +966 14676804.
E-mail address: rdjemal@ksu.edu.sa (R. Djemal).

1434-8411/$ – see front matter © 2012 Elsevier GmbH. All rights reserved.
http://dx.doi.org/10.1016/j.aeue.2012.09.001

this implementation was approximately 420 ms using 32 reference cells with 8 guarded cells. Another example of OS-CFAR implementation, using the Virtex-II-V2MB100 development kit, led to an execution time of the detection algorithms within 0.48 ms for a data set of 800 samples using only 16 reference cells, resulting in a 0.6-μs delay in cell computation [6]. These delays can be sufficient in a homogeneous environment but are not suitable for high-resolution detection, which requires less than 0.5 μs per cell. Another parallel-pipeline implementation using the CA-CFAR, GO-CFAR and SO-CFAR techniques was presented in [7]. This architecture uses 12 bits of data, and its operating frequency is 120 MHz in a XC2V250 Virtex-II FPGA device. In [8], the authors presented an FPGA-based implementation with an OS-CFAR processor using 16 bits for data processing. The proposed architecture was implemented on XCV400E-Virtex FPGA device with a maximum clock frequency of 205 MHz. A so-called ES-CFAR expert system is presented in [9] based on the sensing of the clutter environment; by means of a set of rules along with a voting scheme, this system selects the most appropriate CFAR processor to produce decisions that will outperform a single processor. This expert system is based on five separate CFAR processors: CA-CFAR, GO-CFAR, OS-CFAR, TM-CFAR and the ES-CFAR system. The latter of which is based, on the use of knowledge-based signal processing algorithms [10]. A versatile processing architecture that allows switching between six CFAR algorithms and operating parameters is presented in [11]. Other hardware implementations have been presented by Alsuwailem using other techniques related to Gaussian and Rayleigh distributions [12,13]; in these studies, the validation was performed off-line, and the real-time estimation of the performance of the system was not reported. Recently, it has been reported that a pure hardware implementation based on simulation at the RTL level of the forward ACOSD target detector of the Stratix-II development board can operate at up to 130 MHz with a delay of 0.29 μs for a log-normal distribution [14,15]. However, the proposed delay provides only an approximate delay, as no interface is defined to interconnect the detector with its environment. Therefore, the performance of the system should be updated after integrating delays related to standard interfaces and wrappers.

This work presents a new design exploration and an improvement on previous studies by integrating three algorithms such that the final decision of the target is achieved based on the results appearing in a specific display, called the target spot display, thus reducing the probability of false targets. The CA, OS and ACOSD techniques have been proposed to enhance the robustness of CFAR detectors against both homogeneous and non-homogeneous environments. Given a homogeneous background, when the reference cells contain independent and identically distributed (IID) observations governed by a Gaussian distribution, the CA-CFAR algorithm is recommended. In various non-homogeneous environments where multiple targets occur with clutter edges in the reference window, the order statistics (OS) detector is chosen, yielding good performance. Another case is also considered: the situation in which the number of interfering targets is unknown in a log-normal clutter, which occurs in many practical situations and for which the F-ACOSD algorithm is considered. The factors motivating this work are summarised as follows:

- In a homogeneous environment, it is easy to estimate the threshold adaptively with reduced computational cost using CA-based techniques.
- In a non-homogeneous environment, the OS detector features clutter with a Gaussian P_{df}, which results in a Rayleigh-distributed amplitude at a lower resolution. However, if we consider a high resolution with low grazing angles and a horizontal polarisation, the distribution should be log-normal.
- In the case of high resolution, low grazing angles and horizontal polarisation at high frequencies, the amplitude statistics of clutter returns deviate from a Rayleigh to a log-normal distribution.
- The adaptive threshold is based on ranked samples of reference cells to reduce cell loss and improve the detection probability of a log-normal-based distribution.

In the rest of this paper, we will consider a novel hardware/software implementation of the CA, OS and ACOSD-CFAR algorithms [15,16] for SoC implementation validated using FPFA on a PC board. The custom instruction approach will be considered to implement critical components, such as accelerators around the embedded system, including a Nios-II softcore CPU connected to the target components via the Avalon interface. All critical components will be exported as custom logic to operate as accelerators working in parallel, while the remaining non-critical parts of the CFAR processor are executed by the Nios-II softcore CPU within the same system-on-chip. The integration of the softcore CPU allows the designer to more easily perform validation and timing measurements within the FPGA with high accuracy based on internal timers. Furthermore, the co-design approach makes it easier to evaluate the performance of the CFAR processor and to modify the design partitioning accordingly.

This paper is organised as follows: Section 2 provides the theoretical foundation of the different CA, OS and ACOSD CFAR techniques. Section 3 presents the hardware/software approach applied for the CFAR processor and the details of the proposed architecture. In Section 4, a brief discussion of the performance improvements is presented, and the obtained HW/SW results are compared with those obtained for the software implemented. Finally, Section 5 presents concluding remarks and some directions for future research in the field.

2. CFAR Algorithms

2.1. CFAR basics and high-resolution requirements

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

Several types of CFAR techniques based on the method used to obtain the adaptive threshold from the reference window cells have been reported in the literature [11] for different backgrounds. Of these techniques, we are interested in the CA and OS CFAR techniques in the case of homogeneous and non-homogeneous environments and in the ACOSD techniques in the case of high-resolution target detection. This requires a great computational load to be processed in real time with a limited computation delay. In this respect, the delay is fixed by the pulse width $T < 0.5$ μs, which represents a pulse width for several practical applications when clutter is viewed in the desert at high resolution and at low grazing angles ($\phi < 5^\circ$), regardless of the radar resolution [17]. Because of the complexity of the ACOSD technique, particular attention is given to the implementation of this CFAR technique, which has not yet been tested with a real architecture.

2.2. The CA-CFAR (distribution is exponential)

The cell-averaged (CA) CFAR technique is one of the most common CFAR detection schemes used in adaptive threshold estimation for target detection. The adaptive threshold

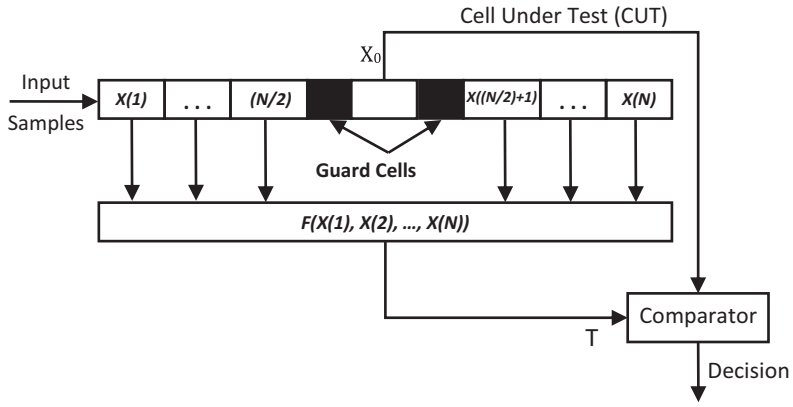

Fig. 1. Block diagram of a typical CFAR algorithm.

is obtained from the arithmetic mean of the reference cells and is used to maintain the false-alarm probability at a desired level. For homogeneous background noise, as well as independent and identically distributed reference cell outputs, the arithmetic mean is the maximum likelihood estimate. This means that the detection threshold is designed to adapt to changes in the environment. To avoid corrupting this estimate with power from the CUT itself, cells immediately adjacent to the CUT are normally ignored (and referred to as "guard cells"). As proposed by Finn and Johnson [18], the cell under test is considered as a target if it is greater than the sum of all of its adjacent cells multiplied by a constant T to establish a threshold, as shown in Fig. 1. Hypothesis H_1 indicates the presence of a target in the test cell, while H_0 indicates that there is no target. Other related approaches calculate separate averages for the cells to the left and right of the CUT and then use the greatest or least of these two power levels to define the local power level. These techniques are referred to as the greatest-of CFAR (GO-CFAR) and least-of CFAR (LO-CFAR), respectively, and can improve the detection process when immediately adjacent to areas of clutter.

2.3. The OS-CFAR

There are other algorithms, such as the order statistics CFAR, which compute an order statistic of the reference cells to compute the threshold. The OS-CFAR proposed by Rohling [4] provides inherent protection against serious performance degradation in the presence of non-homogeneous samples. In fact, the OS-CFAR processor estimates the noise power simply by selecting the Kth largest cell in the reference window; the threshold is obtained from one of the ordered samples of the reference window. The range samples are first ordered according to their magnitudes, and the statistic Z is taken to be the Kth largest sample. Fig. 2 shows a block diagram of the OS-CFAR algorithm for a sliding reference window of size $N = 16$ and a threshold sample order of $K = 3N/4 = 12$, which is known to be the optimum value for a probability of false alarm $P_{fa} = 10^{-5}$ in the presence of two clusters.

Although the improvement in the development of the theoretical aspects of CFAR detection is advanced and very promising, the practical hardware aspect of CFAR detection is still not adequate for the required high computational signal processing operations. Even though an FPGA implementation has been proposed [6], the timing has not yet been reported, and we did not find any information regarding this implementation and the high-resolution requirements in terms of timing and frequency (Fig. 3).

2.4. ACOSD algorithms

The ACOSD algorithm and its sub-algorithms consist of two steps: removal of the interfering reference cells (censoring step) and detection. Both steps are performed dynamically with 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 does it require information about the number of interfering targets.

In a CFAR processor, the input samples $\{X_i: i = 0, 1, \ldots, N\}$ are stored in a tapped delay line. The cell with the subscript $i = 0$ is the cell under test and contains the signal that needs to be classified as

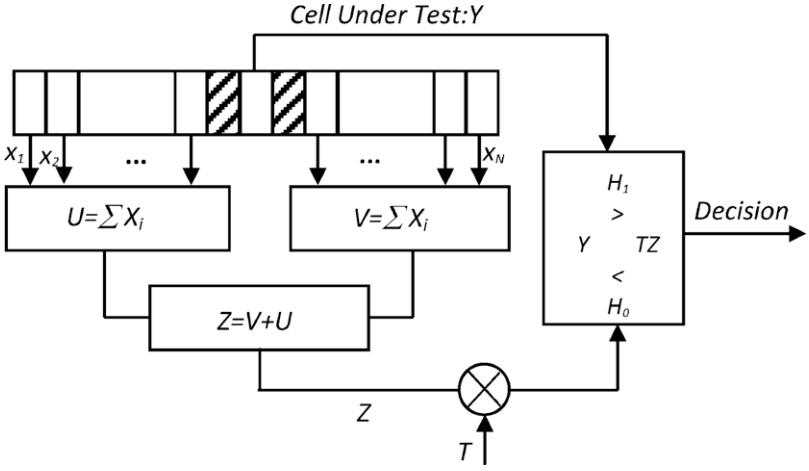

Fig. 2. Block diagram of the CA-CFAR algorithm.

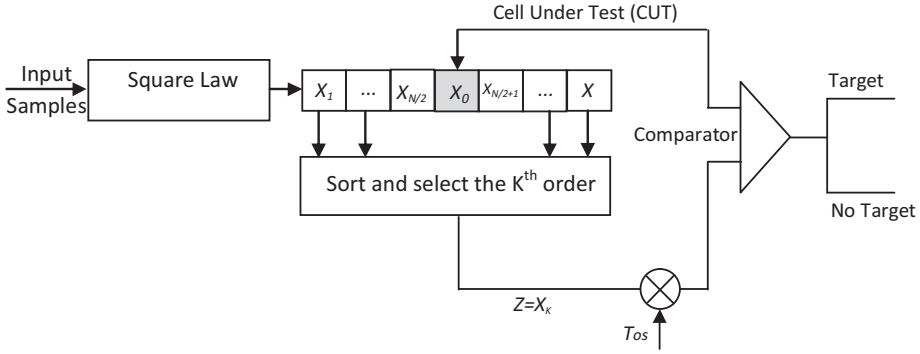

Fig. 3. Block diagram of the OS-CFAR algorithm.

either the actual target or not the target. The last N surrounding cells are the auxiliary cells used to construct the CFAR procedure. In the ACOSD algorithm, the N surrounding cells are ranked in ascending order according to their magnitudes to yield

$$X(1) \le X(2) \le \ldots X(p) \le \ldots \le X(N) \tag{1}$$

The sorted cells are then sent to the detection component, which has two different algorithms: backward ACOSD (BACOSD) and forward ACOSD (FACOSD).

2.4.1. The B-ACOSD algorithm

In the B-ACOSD algorithm, sample $X(N)$ is compared with the adaptive threshold T_{c0}, defined as

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

where $X(p)$ is the pth largest sample and α_0 is a constant chosen to achieve the desired probability of false censoring P_{fc}. For CFAR applications, it is recommended that a value of p greater than $N/2$ be used to obtain reasonable performance [16,19].

If $X(N) < T_{c0}$, the algorithm decides that $X(N)$ corresponds to a clutter sample without interference and terminates. If, by contrast, $X(N) > T_{c0}$, 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

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

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,

$$X(N-k) \underset{H_0}{\overset{H_1}{\gtrless}} T_{ck}; \quad 0 \le k < N-p \tag{4}$$

where

$$T_{ck} = X(1)^{\alpha_k} X(p)^{1-\alpha_k} \tag{5}$$

Hypothesis H_1 represents the case in which $X(N-k)$ and, thus, the subsequent samples $X(N-k+1)$, $X(N-k+2)$,...,$X(N)$ correspond to clutter samples with interference, whereas H_0 denotes the case in which $X(N-k)$ is a clutter sample without interference. The successive tests are repeated as long as hypothesis H_1 is declared true. The algorithm terminates when the cell under investigation is declared homogeneous (clutter sample only) or, in the extreme case, when all of the $(N-p)$ highest cells are tested; that is, $k = N-p$. Fig. 4 shows a block diagram of the B-ACOSD algorithm.

In the detection step, the cell under test X_0 is compared with the threshold T_{ak} to determine whether a target is present according to

$$X_0 \underset{H_0}{\overset{H_1}{\gtrless}} T_{ak}; \quad 0 \le k < N-p+1 \tag{6}$$

Hypothesis H_1 denotes the presence of a target in the test cell, whereas hypothesis H_0 denotes the absence of a target. In B-ACOSD CFAR, the threshold T_{ak} is defined as follows:

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

where the value of β_k is selected according to the design probability of false alarm $P_{f\alpha}$ for k interfering targets found in the censoring step.

2.4.2. The F-ACOSD algorithm

The F-ACOSD algorithm starts by comparing sample $X(p+1)$ with the threshold $\hat{T}_{c0}$ given by

$$\hat{T}_{c0} = X(1)^{\hat{\alpha}_0} X(p)^{1-\hat{\alpha}_0} \tag{8}$$

where $\hat{\alpha}_0$ is a constant chosen to achieve the desired (P_{fc}) for the F-ACOSD algorithm. In contrast to the B-ACOSD algorithm, if $X(p+1) > \hat{T}_{c0}$, the algorithm decides that $X(p+1)$ is a return echo from an interfering target, and it terminates. If, by contrast, $X(p+1) < \hat{T}_{c0}$, the algorithm decides that the sample $X(p+1)$ corresponds to a clutter sample without interference, then the detector compares the sample $X(p+2)$ with the threshold given by

$$\hat{T}_{c1} = X(1)^{\hat{\alpha}_1} X(p)^{1-\hat{\alpha}_1} \tag{9}$$

This comparison is performed to determine whether the sample corresponds to an interfering target or a clutter sample without interference. At the $(k+1)$th step, the sample $X(p+k+1)$ is compared with the threshold $\hat{T}_{c0}$, and a decision is made according to the test,

$$X(p+k+1) \underset{H_0}{\overset{H_1}{\gtrless}} \hat{T}_{ck}; \quad 0 \le k < N-p \tag{10}$$

where

$$\hat{T}_{ck} = X(1)^{\hat{\alpha}_k} X(p+k)^{1-\hat{\alpha}_k} \tag{11}$$

Hypothesis H_1 represents the case in which $X(p+k+1)$ and, thus, the subsequent samples $X(p+k+2)$,...,$X(N)$ correspond to clutter samples with interference, while H_0 denotes the case in which $X(p+k+1)$ is a clutter sample without interference. The successive tests are repeated as long as hypothesis H_0 is declared true. The

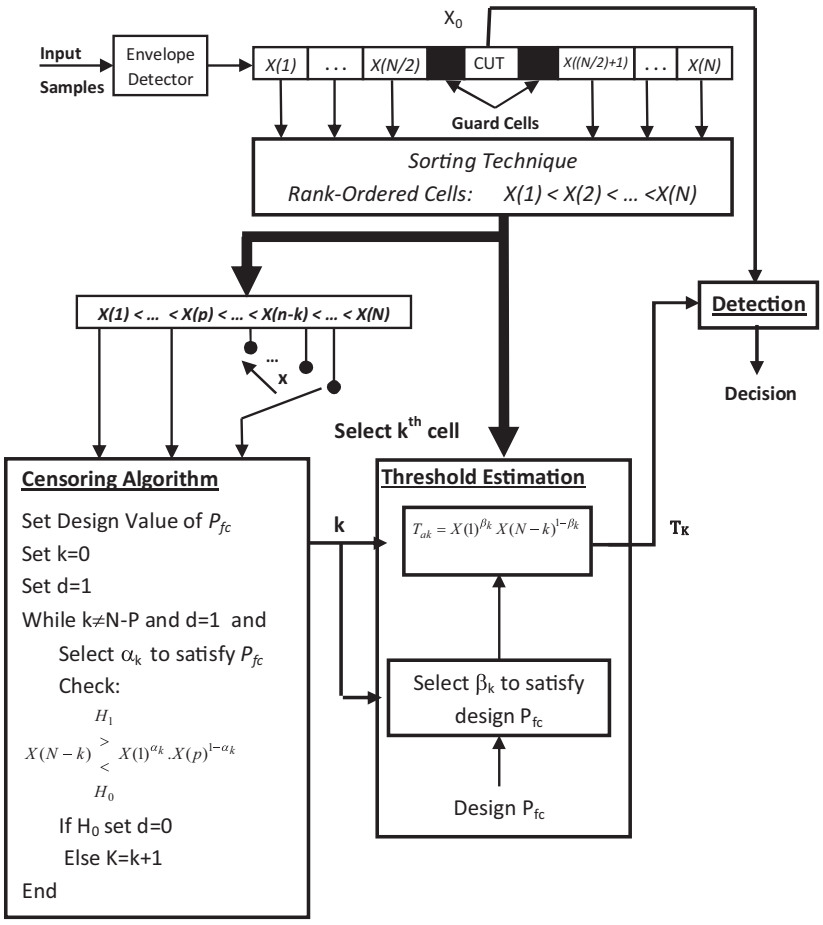

Fig. 4. Block diagram of the B-ACOSD algorithm.

algorithm terminates when the cell under investigation is declared non-homogeneous (i.e., clutter plus interference sample) or, in the extreme case, when the entire $N-p$ highest cells are tested; that is, $k=N-p$.

The F-ACOSD algorithm is operated in a manner similar to that of the B-ACOSD algorithm, although the implementation of the former differs with respect to how the censoring is performed and how the threshold T_{ak} is calculated in the detection process. In the detection step, the CUT X_0 is compared with the threshold $\hat{T}_{c0}$ to determine whether a target is present according to

$$X_0 \underset{H_0}{\overset{H_1}{\gtrless}} \hat{T}_{ak}; \quad 0 \le k < N-p+1 \tag{12}$$

Hypothesis H_1 denotes the presence of a target in the test cell, while hypothesis H_0 denotes the absence of a target. In the F-ACOSD algorithm, the threshold $\hat{T}_{c0}$ is defined as

$$\hat{T}_{ak} = X(1)^{\hat{\beta}_k} X(p+k)^{1-\hat{\beta}_k} \tag{13}$$

where the value of β_k is selected according to the design probability of false alarm $P_{f\alpha}$ for k interfering targets found in the censoring step.

2.4.3. Threshold values

Threshold selection is a key element in ACOSD algorithms [16]. Thresholds should be selected to reach a low probability of error in a homogeneous environment. We used a Monte Carlo simulation with 500,000 independent runs to obtain threshold values by maintaining desired values of $P_{f\alpha}$ and P_{fc}. Table 1 presents the threshold parameters α_k and β_k obtained using the B-ACOSD algorithm with $P_{fa}=0.001$ and $P_{fc}=0.01$. Table 2 shows the values of $\hat{\alpha}_k$ and $\hat{\beta}_k$ for the F-ACOSD algorithm with the same probabilities.

3. HW/SW embedded system architecture for CFAR detectors

3.1. The HW/SW system-on-chip design flow

The CPU is a critical control function required for system-level integration of the proposed CA, OS and ACOSD techniques. To compile our proposed system on a programmable chip, we have used the Quartus-II tool, which is an integrated synthesis and place-and-route engine used to implement such embedded systems in FPGA logic. The FPGA hardware design flow is summarised in Fig. 5.

- Functional simulation:
  From the design specifications, we perform a functional simulation using a Matlab Monte Carlo simulation to verify the algorithms and define the statistical parameters to be used later in the HDL description.
- Soft Implementation of the proposed CFAR techniques:
  This step consists of the design of a pure software architecture using a high-level language (HLL), such as ANSI-C. The code runs on the Nios-II processor within the FPGA using a MicroC/OS-II operating system and the Quartus II tool to identify the critical components to be exported as hardware modules.

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

(N,p)		K												
		1	2	3	4	5	6	7	8	9	10	11	12	13
(16, 12)	α_k	2.596	2.038	1.709	1.443	–	–	–	–	–	–	–	–	–
	β_k	1.635	1.889	2.12	2.37	2.64	–	–	–	–	–	–	–	–
(36, 24)	α_k	2.538	2.154	1.953	1.812	1.7	1.601	1.523	1.443	1.369	1.3	1.225	1.153	–
	β_k	1.35	1.465	1.566	1.65	1.73	1.8	1.87	1.94	2.02	2.1	2.18	2.265	2.345

Table 2
Threshold parameters for F-ACOSD (P_{fa} = 0.001 and P_{fc} = 0.01).

(N,p)		K												
		1	2	3	4	5	6	7	8	9	10	11	12	13
(16, 12)	$\hat{\alpha}_k$	1.442	1.465	1.535	1.745	–	–	–	–	–	–	–	–	–
	$\hat{\beta}_k$	2.64	2.37	2.12	1.889	1.635	–	–	–	–	–	–	–	–
(36, 24)	$\hat{\alpha}_k$	1.15	1.152	1.154	1.158	1.16	1.167	1.174	1.191	1.21	1.264	1.311	1.467	–
	$\hat{\beta}_k$	2.345	2.265	2.18	2.1	2.02	1.94	1.87	1.8	1.73	1.65	1.566	1.465	1.35

- Partitioning of the HW and SW components:
  Based on the simulation results of the software version of the CFAR techniques, the critical functions are identified, and some of them are exported as a hardware block to be integrated later with the rest of the design for co-simulation purposes.
- RTL modelling of the hardware components, simulation and verification:
  All hardware components are described in the VHDL language at the RTL level and simulated using ModelSim and the Quartus-II tool to verify their functionalities before their integration in the CFAR-based system-on-chip.
- Design integration, synthesis and co-simulation:
  All custom instruction components are imported into the design by adding them to the SOPC as device-specific primitives accessible by the Nios-II processor. These custom instructions can be called using ANSI-C or assembly languages. Then, the overall system, with its HW and SW components, is co-simulated using Nios-II, custom logics, and embedded memories with additional interfaces. During synthesis, many optimisations are performed to meet the required timing and performance constraints.
- Test the FPGA on a PC board:
  In this step, the FPGA is programmed, and the system-on-chip is tested on a PC board; the performance is verified to determine whether the performance specifications are met. In addition, dynamic timing analysis using the internal timers of the Nios-II core processor is performed.

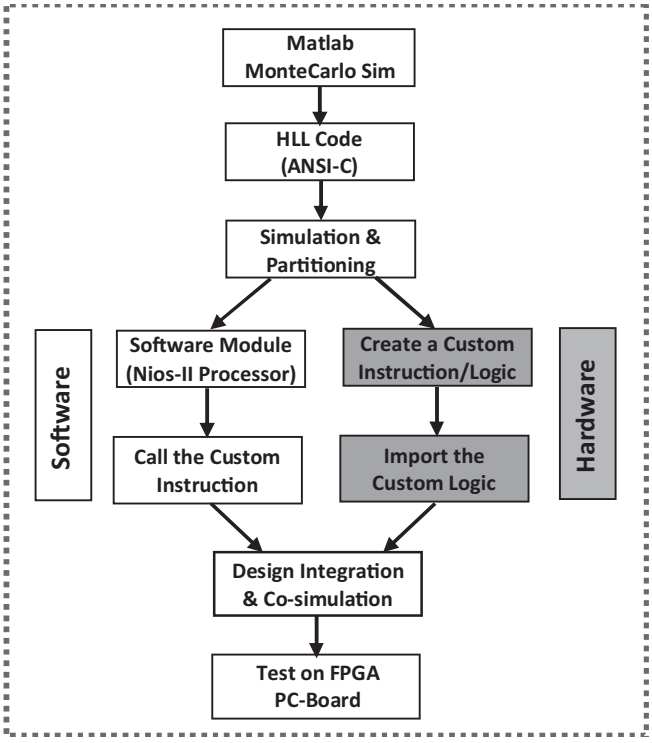

Fig. 5. Typical HW/SW design flow.

3.2. The Nios-II-based Software architecture for CFAR detectors

To initiate the design of the target detectors, we implemented all ACOSD, CA and OS detectors as a pure SW solution to evaluate their performances and identify critical components. The first system configuration incorporates the Nios-II softcore CPU into the FPGA with on-chip memories and a JTAG-UART interface interconnected using the Avalon System Fabric, as depicted in Fig. 6. The Nios-II is used to execute all proposed AINSI-C codes related to the CFAR detectors and to measure the execution time using their internal timers. This step is performed to obtain a first evaluation of the design and thus identify the critical parts that violate the timing specifications; it is also performed to propose a new architectural partitioning to remedy the deficiencies in performance and thus satisfy the high-resolution timing requirements.

The timing presented in Table 3 is provided by the Nios-II softcore CPU, the internal timers of which allow an accurate measurement of the detector execution timing. It is clear that all delays

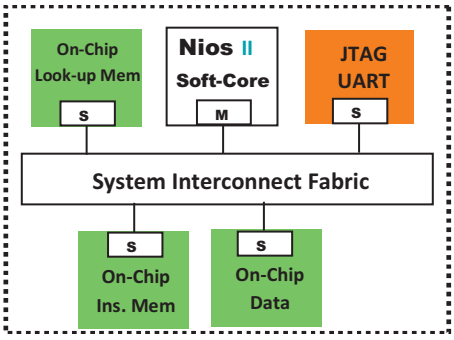

Fig. 6. SW architecture of CFAR detectors.

Table 3
Software execution time of CFAR detectors.

Detector	Modules	Delays in μs
CA	Sorting module	18
	Censoring and detection module	0.96
	Total delay	18.96
OS	Sorting module	18
	Censoring and detection module	0.5
	Total delay	18.5
B-ACOSD	Sorting module	18
	Censoring module	104
	Detection module	21
	Total delay	143
F-ACOSD	Sorting module	18
	Censoring module	87
	Detection module	20
	Total delay	125

are quite large compared with the target value (0.5 μs). Through this software implementation, we conclude that this solution is not suitable for this type of application, and, consequently, we should explore HW/SW solutions for integration in the same embedded system.

3.3. HW/SW CFAR embedded system architecture

To increase the performance of the CFAR-based embedded system, we extended the previous architecture shown in Fig. 6. By integrating more hardware components as co-processors or accelerators and custom instructions embedded within the Nios-II processor, we allow the designer much flexibility to easily modify the design to optimise performance according to a given set of specifications (see Fig. 7). The new design organisation provides HW and SW components that work around the Nios-II processor, which is the only master device in the proposed system developed in the HDL language; therefore, it can be customised and synthesised in a programmable FPGA device [20]. The Nios-II softcore CPU supports a micro-operating system (MicroC-OS) that compiles and executes the software modules of the application described in ANSI-C. The remaining hardware modules of our embedded system have been developed using the VHDL language.

3.3.1. Software components

The software implementation of CA, OS and ACOSD detectors is based on the execution of the soft-IP, which is described in AINSI-C code running on the Nios-II core processor. A Nios-II softcore 32-bit RISC microprocessor is described in the HDL language with 8 KB of data cache, 16 KB of instruction cache and a master port. Thus, all peripherals can be targeted as needed. The Nios-II softcore comes in three instruction set architecture (ISA)-compatible versions:

- The fast version is optimised only for speed.
- The standard version is balanced for speed and size.
- The economy version is optimised only for size.

In our architecture, we considered the fast version of the Nios-II softcore processor to obtain the best performance for our detectors. We also extended the instruction set of the Nios-II softcore processor by adding new custom instructions dedicated to target-detection purposes.

3.3.2. Hardware components

After identifying the critical components by measuring the execution time for each component using the software solution, as presented in Table 3, we proceeded by implementing the CA, OS and a part of the ACOSD as co-processors described and simulated in VHDL at the RTL level. In addition, to increase the parallelism for the proposed system, we considered the custom instruction approach as a new alternative to execute many functions as new instructions within the Nios-II processor. This approach allows the designer to take full advantage of the flexibility of the FPGA to accelerate the execution of certain critical functions to meet system requirements by reducing a complex sequence of standard instructions to a single instruction implemented in the hardware and represented as custom logic.

Furthermore, custom instruction is used to optimise the software inner loops and computation-intensive procedures related to the CFAR application, which provides us with the ability to tailor the system-on-chip architecture by integrating the Nios-II core processor to meet the critical time requirements related to the CFAR architecture. Even if the Nios-II processor can integrate up to 256 custom instructions, only two instructions are defined as being related to the sorting technique required in both the OS and ACOSD detectors, and the look-up technique is a critical operation required by the ACOSD detector.

3.3.2.1. The sorting custom login instruction. A fundamental issue associated with sorting techniques in CFAR detectors is how this technique affords a ranked cell for a given window with reduced sorting timing. The underlying principle is that of operating on a newly inserted cell in a sorted list such that the amount of computation will be reduced. In addition, we try to benefit from the parallel

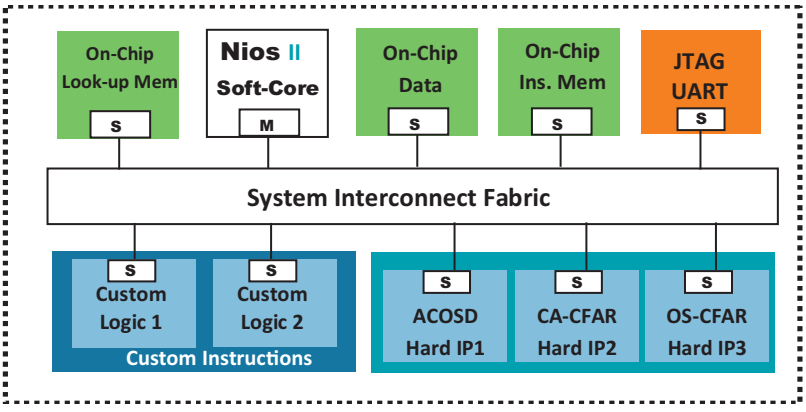

Fig. 7. CFAR Nios-II-based embedded system.

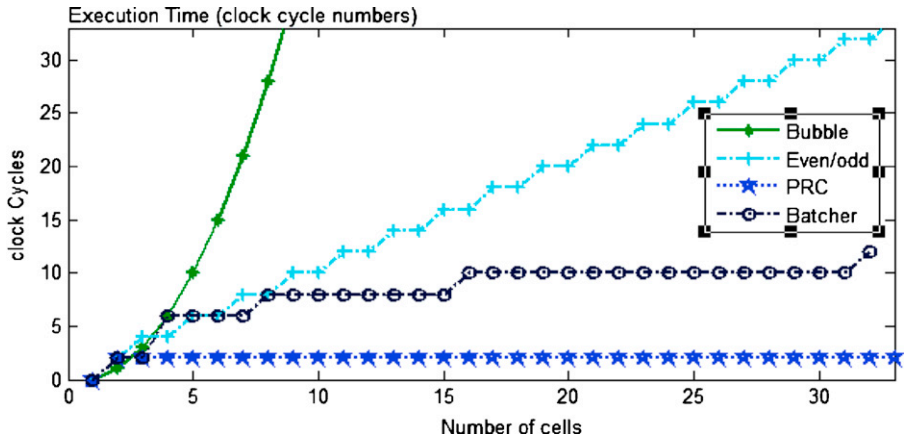

Fig. 8. Sorting technique performance.

execution of the sorting technique even if this technique requires an increase in the complexity of the sorter (FPGA resources). Fig. 8 presents a comparison of the performance of four sorting techniques: bubble sorting, even/odd bubble sorting, batcher sorting [21] and parallel rank computing techniques [22].

- The even-odd sorting technique consists of $N/2$ comparators run in parallel to build the even stage, and the remaining $(N/2-1)$ comparators are used for the odd stage. This technique ends the sorting of all N cells within a particular window using N stages within N clock cycles.
- The batcher sorter technique requires $T = (\log_2(N))^2 - \sum_{i=0}^{\log_2(N)} i$, where N is the number of cells and T is the number of clock cycles. For example, for a window size of $N = 16$, this technique requires 6 clock cycles to sort all of the data.
- The PRC technique is based on the insertion of a new value in a sorted table. It considers the maximum parallel degree, where all comparators are implemented in parallel for all cells. Only two clock cycles are required for this technique; thus, we have replaced the bubble sorting technique with the PRC technique. Furthermore, the PRC technique allows for a sorting delay, which is independent of the number of cells N.

From a timing point of view, the parallel range computing technique is faster than the bubble sorting technique; thus, we have replaced the bubble sorting technique with the parallel range computing (PRC) technique [19] because it provides a reduced delay in software implementation, as shown in Table 4.

Table 4 shows the software implementation timing for these techniques using the Nios-II processor. It is clear that the PRC technique yields the fastest execution time. Consequently, we decided to implement this technique in VHDL, and we integrated it as a custom logic instruction 1 to benefit from the parallelism and hardware performance of the system and achieve a competitive timing.

3.3.2.2. The look-up table custom instruction logic 2. Another critical function identified during the software implementation consists of the computation of the threshold according to Eqs. (5) and (6).

Table 4
Execution time using the Nios-II processor for different sorting techniques (pure SW solution).

Sorting technique	Bubble	Even/odd	Batcher	PRC
Nios-II execution time	18 μs	10 μs	3.84 μs	1.28 μs

To reduce the complexity of these functions, these equations are converted into their logarithmic forms as follows:

$$\log\ T_{ck} = (1-\alpha_k)\cdot \log X(1) + \alpha_k \cdot \log X(p) \quad (14)$$

$$\log\ T_{ak} = (1-\beta_k)\cdot \log X(1) + \beta_k \cdot \log X(N-k) \quad (15)$$

In this form, power computation becomes a matter of simple multiplication, which then becomes a matter of addition. In addition, because 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 a range of log-normal distributions with $\mu = 1$ and $\sigma = 1.1$, as suggested in [16], based on real radar input data measurements. The proposed LUT supports up to 2000 log values for our implementation, which allows the resolution to follow the change in number representation to the logarithmic form; the test cell value was also converted accordingly. This logarithmic conversion was also performed using the same look-up table mentioned above. The look-up table resides in a 32-K on-chip ROM inside the FPGA. The data distribution resolution in the 32-K on-chip ROM is 0.0610. A Matlab fixed-point ACOSD CFAR simulation with this resolution provides censoring results that are as good as those afforded by a real-number simulation. This is also implemented as a second custom logic instruction.

3.3.2.3. The hardware ACOSD accelerator (Hard IP1). The Hard IPs presented in Fig. 7 consist of pure hardware modules described in the HDL language (VHDL) at the RTL level and integrate slave ports to interconnect the system via the Avalon interface [23]. These IPs operate in parallel with the co-processor to accelerate the computation of the integrated detectors. Three hard IPs are developed:

- Hard IP 1 is dedicated to the remaining parts of the design of the ACOSD detector.
- Hard IP 2 integrates the hardware architecture of the CA detector.
- Hard IP 3 consists of the hardware architecture of the OS detector.

All of these IPs are operated in parallel as slave components, where the Nios-II processor is responsible for sending and receiving data to/from all of the detectors. It is also responsible for all communication management in the embedded system.

4. Implementation results

The embedded system architecture for the CA, OS and ACOSD detectors was developed on a Stratix IV FPGA-based board. The Nios-II processor, built around the Stratix IV device, was considered

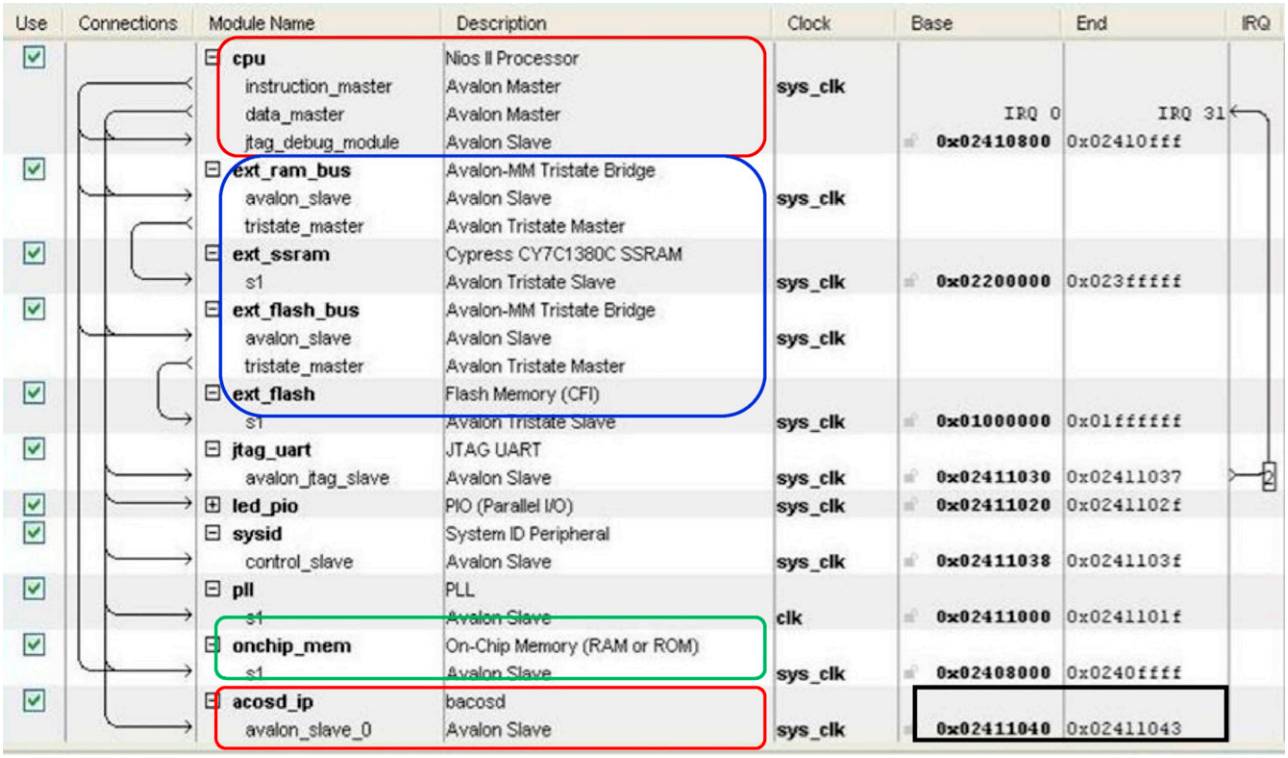

Fig. 9. Embedded system organisation of an ACOSD-CFAR detector.

for prototyping because of its low cost and its flexibility in design exploration. Furthermore, the master clock frequency for prototyping reached 200 MHz, which allows for the fast execution of target detectors to meet the system timing requirements. Under these circumstances, the HW/SW solution for the architectural exploration seems to be suitable for this type of real-time application. Thus, the software component of the proposed architecture is based on the generation of a fast version of a 32-bit Nios-II processor with an 8-KB data cache, a 16-KB instruction cache and a master port to interconnect the Avalon interface, which represents the bus system of the proposed embedded architecture.

The hardware of the system integrates two custom logic instructions and three hard IPs modelled in VHDL at the RTL level and synthesised using the Quartus II environment. In addition, this system includes on-chip memories with sizes of 128KBx32 and 64KBx16 and additional timers to monitor all timing aspects with regard to the CFAR architecture. Finally, a JTAG UART interface is embedded with a simplified configuration, allowing target connection for software downloading.

For validation purposes, the Nios-II processor is responsible for generating and sending all data to all detectors. For CA, the data satisfy an exponential distribution to emulate the homogeneous environment, where the noise is log-normal-distributed and the data are Rayleigh-distributed for the OS and ACOSD detectors. This Nios-II is also responsible for all communications between IPs and on-chip memories, as well as for timing measurements with great accuracy using their internal timers. In addition, the Nios-II checks for the false-alarm rate of each detector and presents the results on a display in real time. Fig. 9 presents an example of an embedded system integrating a Nios-II processor with custom logic instructions and a master port, the ACOSD hardware accelerator, and all on-chip memories.

Table 5
Resources utilisation of the Stratix IV FPGA device.

Complexity of HW/SW	Nios-II	Custom logic 1	Custom logic 2	Hard IP1	Hard IP2	Hard IP3	On-chip memories	Total
LUTs	5671(3%)	1254 (<1%)	152 (<1%)	2260 (<1%)	172 (<1%)	1585 (<1%)	×	11,094 (6%)
D-FF	6617 (4%)	512 (<1%)	0	3497(2%)	0	512	×	11,192 (6.2%)
On-chip memory	×	×	×	×	×	×	4,607,168 32%	4,607,168 32%

Table 6
Time savings using the custom instructions and hardware accelerators approach.

	Timing of B-ACOSD (pure SW) in μs	Timing of FACOSD (pure SW) in μs	Timing of F/B-ACOSD (HW/SW) in μs
Sorting	1.28	1.28	0.02
Censoring	104	87	0.17
Detection	21	20	0.08
Total delay	126.28	108.28	0.27

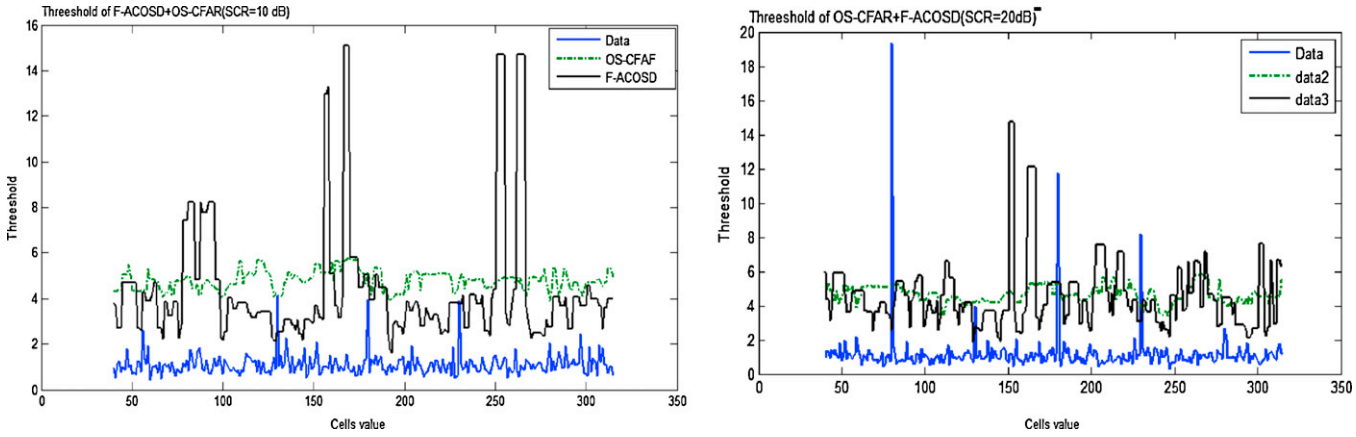

Fig. 10. Threshold variation for SCR = 10 dB and SCR = 20 dB.

```
Simulator4 Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (08/01/12 03:43 )
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

*******************************************************************************
*********************--- Results Taken From the Board --- **********************
************************--- F-ACOSD+CA-CFAR algorithm --- ************************
*************************** Frequency= 200 Mhz ********************************
*******************************************************************************
|||      ------              Data Generated SNR=10 db    (1000 Sample)          |||
|||-----------------------------------------------------------------------------|||
|||                           Total Test Target= 1000                           |||
|||                       Num_Decision_FACOSD  = 20          2.0./.            |||
|||                       Num_Decision_CA      = 0           0.0./.            |||
|||                       Num_Decision_OS      = 4          0.4./.             |||
||| Totla Time:      0.269415 (melli-second)                                    |||
||| Decision Time:   0.27     (micro-second)                                    |||
|||                                                                             |||
-------------------------------------------------------------------------------
*******************************************************************************
|||          -----------   Data Generated SNR=20 db    (1000 Sample)           |||
-------------------------------------------------------------------------------
|||                           Total Test Target= 1000                           |||
|||                       Num_Decision_FACOSD  = 24           2.4./.           |||
|||                       Num_Decision_CA      = 9           0.9./.            |||
|||                       Num_Decision_OS      = 16           1.6./.           |||
||| Totla Time:      0.265210 (melli-second)                                    |||
||| Decision Time:   0.27     (micro-second)                                    |||
-------------------------------------------------------------------------------
*******************************************************************************
|||          -----------   Data Generated SNR=30 db    (1000 Sample)           |||
-------------------------------------------------------------------------------
|||                           Total Test Target= 1000                           |||
|||                       Num_Decision_FACOSD  = 25           2.5./.           |||
|||                       Num_Decision_CA      = 18           1.8./.           |||
|||                       Num_Decision_OS      = 20           2.0./.           |||
||| Totla Time:      0.266310 (melli-second)                                    |||
||| Decision Time:   0.27     (micro-second)                                    |||
-------------------------------------------------------------------------------
*******************************************************************************
|||          -----------   Data Generated SNR=40 db    (1000 Sample)           |||
-------------------------------------------------------------------------------
|||                           Total Test Target= 1000                           |||
|||                       Num_Decision_FACOSD  = 21          2.1./.            |||
|||                       Num_Decision_CA      = 20           2.0./.           |||
|||                       Num_Decision_OS      = 20           2.0./.           |||
||| Totla Time:      0.266110 (melli-second)                                    |||
||| Decision Time:   0.27     (micro-second)                                    |||
-------------------------------------------------------------------------------
*******************************************************************************
*********************************--- End  --- *********************************
*******************************************************************************
```

Fig. 11. Real-time target detection for CA, OS and ACOSD detectors.

Table 5 summarises the results in terms of FPGA resource utilisation, including LUTs, registers and on-chip memories. The last column presents the total resources of the three detectors, which are reasonable.

To demonstrate the performance of the HW/SW architecture, we evaluated the gain in timing compared with that obtained by the pure SW architecture. Table 6 presents these timings for both the FACOSD and BACOSD detectors using the custom instruction approach and the remaining parts as a hardware accelerator. We noticed that the delays associated with the sorting and censoring modules are decreases by 64-fold and by more than 500-fold respectively, leading to significant time savings in the overall architecture. The timing of the ACOSD detectors is more critical than that of the other proposed detectors, which is why we dedicate more attention to the evaluation of these delays. Meanwhile, the CA and OS detectors exhibit much smaller delays and do not require additional optimisation during the integration of their architectures.

The embedded system architecture is prototyped using the Stratix IV FPGA-based board by integrating both HW and SW components in the same FPGA, with $N = 16$ and $p = 12$. The Nios-II processor operates at a frequency of 200 MHz. All critical components of the design are designed using custom instructions or are described in VHDL and connected to the Avalon bus system as hardware accelerators. As shown in Table 6, the total delay of the F-ACOSD and B-ACOSD is equal to 0.27 μs. This processing time is still below the real-time system requirements, fixed at 0.5 μs. In addition, using the PRC technique and a high degree of parallelism, we considered a design featuring 8 and 32 cells in the window of cells and found exactly the same timing of 0.27 μs.

It is worth noting that, for a non-homogeneous environment, the ACOSD and OS CFAR detectors have been designed under the assumption that the radar clutter is log-normal-distributed, whereas the target is Rayleigh-distributed. This assumption is well correlated with high-resolution radar, low grazing angles and horizontal polarisation at high frequency. On the other hand, for a homogeneous environment, the radar clutter is considered to have a Gaussian P_{df}, which results in a Rayleigh-distributed amplitude. Fig. 10 presents the adaptive threshold variation extracted from our proposed embedded system via the high-resolution target detection afforded by the Nios-II processor, with a log-normal distribution for the clutter and a Rayleigh distribution for the target. Two detectors are represented: the F-ACOSD and the OS CFAR for signal-to-clutter ratio SCR = 10 and 20 dB, respectively. We can confirm that the performance of the ACOSD is significantly better than that of the OS, especially when the SCR is low. Starting from SCR = 20 dB, the performances of these two detectors become very similar.

Fig. 11 presents the results extracted in real time from the embedded system, where the Nios-II displays the decisions of all cells in the proposed detectors. It should be noted that for a lower SCR less than 10 dB, the P_{fa} is greater than 10^{-3}. If the SCR increases, the detection becomes better and the resolution is improved where the target value of P_{fa} is satisfied.

5. Conclusion

In this paper, an efficient HW/SW embedded system design of CFAR detectors for homogeneous and non-homogeneous environments is presented. The design exploration is performed for the CA, OS and ACOSD CFAR techniques to satisfy high-resolution target detection, low grazing angles and horizontal polarisation at high frequencies with a timing limitation of no more than 0.5 μs. This proposed system-on-chip has the advantages of being simple and fast with a low development cost. The performance of the prototype hardware setup proved the concept of the co-design within a reasonable design time. We considered the custom instruction approach to export and design the hardware components, with critical delays integrated in the Nios-II processor. Furthermore, we considered hardware accelerators connected to the Avalon interface as slave devices controlled by the Nios-II, which integrates a real-time operating system MicroC-OS to facilitate the execution of software components and to guarantee the management of the communication for the overall system. Our proposed architecture operates over a window size of 16 cells with two guarded cells, with a frequency of 200 MHz. The architecture performs the detection for the cell under test within a delay of 0.27 μs, which is below the time constraint fixed by high-resolution detection. The proposed architecture has been synthesised and validated using the Stratix IV development kit (EP4SGX230KF4C2 device), with which we have measured the overall architecture complexity and the associated timing constraints.

We also check the false alarm rate for different values of signal-to-clutter ratio SCR from 10 to 20 dB, and we find the results in conformity with those obtained by Monte-Carlo simulations. Indeed, the probability of false alarm P_{fa} remains below 10^{-3} when the SCR exceeds the 20 dB, which is quite reasonable result.

Acknowledgement

This work reported in this paper is supported by the National Plan for Science and Technology (NPST) at the King Saud University (project number: ADV-170-2-08).

References

[1] Barkat M. Signal detection and estimation. MA: Artech House; 2005.
[2] Laroussi T, Barkat M. Performance analysis of order-statistic CFAR detectors in time diversity system for partially correlated Chi-square targets and multiple target situations: a comparison. Signal Process 2006;86(7):1617–31.
[3] Rickard JT, Dillard GM. Adaptive detection algorithms for multiple target situations. IEEE Trans Aerosp Electron Syst 1977;AES-13(4):338–43.
[4] Rohling H, Radar CFAR. Thresholding in clutter and multiple target situations. IEEE Trans Aerosp Electron Syst 1983;AES-19(4):608–21.
[5] Cumplido R, Torres C, Lopez S. 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. 2004. p. 214–8.
[6] Magaz B, Bencheikh ML. An efficient FPGA implementation of the OS-CFAR processor. In: International Radar Symposium. 2008. p. 1–4.
[7] Torres C, Cumplido R, Lopez S. Design and implementation of a CFAR processor for target detection. In: 14th International Conference on Field Programmable logic, FPL04, Lectures Notes on Computer Science, vol. 3203. 2004. p. 943–7.
[8] Wei B, Sharif MY, Binnie TD, Almaini AEA. Adaptive PN code acquisition in multi-path spread spectrum communications using FPGA. In: IEEE international Symposium on Signals, Circuits and Systems ISSCS, vol. 2. 2007. p. 1–4.
[9] Wick MC, Baldygo WJ, Brown RD. Expert system constant false alarm rate (CFAR) processor. U.S. Patent 5 499 030, 12 March 1996.
[10] Capraro GT, Farina A, Griffiths H, Wicks MC. Knowledge-based radar signal and data processing: a tutorial review. IEEE Signal Process 2006:18–29. Mag.23.
[11] Cumplido R, Uribe C, Perez F, Del Campo R. A versatile hardware architecture for a constant false alarm rate processor based on a linear insertion sorter. Digit Signal Process 2010;20(6):1733–47.
[12] Alsuwailem AM, Alshebeili SA, Alhowaish MH, Qasim SM. Field programmable gate array-based design and realization of automatic censored cell averaging constant false alarm rate detector based on ordered data variability. IET Circ Dev Syst 2009;3(1):12–21.
[13] Alsuwailem AM, Alshebeili SA, Alamar M. Design and implementation of a configurable real-time FPGA-based TM-CFAR processor for radar target detection. J Active Passive Electron Dev 2008;3(3–4):241–56.
[14] Djemal R, Alshebeili S, Rosyadi I. Design and implementation of real-time automatic censoring system on chip for radar detection. Penang, Malaysia: World Academic of Science, Engineering and Technology (WASET); 2009. pp. 318–324.
[15] Djemal R. A real-time FPGA-based implementation of target detection technique in non-homogenous environment. Hammamet, Tunisia: Design and Technology of Integrated System in Nanoscale Era; 2010. pp. 1–6.
[16] Almarshad MN, Barkat M, Alshebeili SA. A Monte Carlo simulation for two novel automatic censoring techniques of radar interfering targets in log-normal clutter. Signal Process 2008;88(3):719–32.
[17] Farina A, Russo A, Studer FA. Coherent radar detection in log-normal clutter. IEE Proc Commun Radar Signal Process 1986;133(1):39–54.

[18] Finn HM, Johnson RS. Adaptive detection mode with threshold control as a function of spatially sampled clutter-level estimates. RCA Rev 1968;29(3):414–64.
[19] Farrouki A, Barkat M. Automatic censoring CFAR detector based on ordered data variability for non-homogeneous environments. IEE Proc Radar Sonar Navig 2005;152(1):43–51.
[20] Ben Atitallah A, Kadioniak P, Ghozzi F, Nouel P, Masmoudi N, Levi H. An FPGA implementation of HW/SW codesign architecture for H263 video coding. Int J Electron Commun 2007;61(9):605–20.
[21] Seddiq YM, Alshebeili SA, Alhumaidi SM, Obied AM. FPGA-based implementation of a CFAR processor using Batcher's sort and LUT arithmetic. In: 4th International Design and Test Workshop. 2009.
[22] Perez-Andrade R, Cumplido R, Del Campo FM, Feregrino-Uribe C. A versatile linear insertion sorter based on a FIFO scheme. In: IEEE Computer Society Annual Symposium on VLSI. 2008.
[23] Djemal R, Belwafi K, Kaaniche W, Alshebeili S. An FPGA-Based Implementation of HW/SW architecture for CFAR Radar Target Detector. In: International Conference on Microelectronics)ICM). 2011.

Ridha Djemal received the Ph.D. degree in Microelectronics from the Institut National Polytechnique de Grenoble (France) in 1996. He is working as a Professor at National Institute for Applied Sciences and Technology (ISSAT) Sousse- Tunisia. He is an adjunct professor at the EE department, college of Engineering of King Saud University since 2007. His current area of interests includes Telecommunication networks, image and video processing, Formal verification for IP interfaces and System on Chip (SoC).

Kais Belwafi was born in Mezzouna-Tunisia, on May, 10, 1985. He received the Engineer degree from ISSATS, Sousse-Tunisia on 2010, and the Master degree in Communicant and Intelligent System in 2012. He is currently PhD student in ENIS, Sfax-Tunisia. His research interests have been in the areas of HW/SW Codesign, and real-time embedded systems applications.

Walid Kaaniche was born in Sfax-Tunisia, on December, 17, 1976. He received his Engineering degree, Master degree and PhD thesis in electrical engineering from the National College of Engineering in Sfax-Tunisia. He has also a Master degree in international trade from the College of Commerce in Sfax. Actually, he is an Assistant Professor at the National College of Engineering of Sousse-Tunisia. Much of his research work has been devoted to the design of embedded systems and their applications.

Saleh A Alshebeili is professor and chairman (2001–2005) of Electrical Engineering Department, King Saud University. He has more than 20 years of teaching and research experience in the area of communications and signal processing. Dr Alshebeili is member of the board of directors of Prince Sultan Advanced Technologies Research Institute (PSATRI), the Vice President of PSATRI (2008–2011), the director of Saudi-Telecom Research Chair (2008–2012), and the director (2011–Present) of the Technology Innovation Center, RF and Photonics in the e-Society (RFTONICS), funded by King Abdulaziz City for Science and Technology (KACST). Dr Alshebeili has been in the editorial board of Journal of Engineering Sciences of King Saud University (2009–2012). He has also an active involvement in the review process of a number of research journals, KACST general directorate grants programs, and national and international symposiums and conferences.