An FPGA Based Implementation of a CFAR Processor Applied to a Pulse-Compression Radar System

Slobodan SIMIĆ[1], *Milenko ANDRIĆ*[1], *Bojan ZRNIĆ*[2]

[1] Military Academy, University of Defense in Belgrade, Gen. Pavla Jurišića Šturma 33, 11000 Belgrade, Serbia
[2] Dept. of Defense Technology, Ministry of Defense, Nemanjina 15, 11000 Belgrade, Serbia

simasimic01@gmail.com, asmilenko@beotel.net, bojan.zrnic@vs.rs

Abstract. *A hardware architecture that implements a CFAR processor including six variants of the CFAR algorithm based on linear and nonlinear operations for radar applications is presented. Since some implemented CFAR algorithms require sorting the input samples, the two sorting solutions are investigated. The first one is iterative, and it is suitable when incoming data clock is several times less than sorting clock. The second sorter is very fast by exploiting a high degree of parallelism. The architecture is on-line reconfigurable both in terms of CFAR method and in terms of the number of reference and guard cells. The architecture was developed for coherent radar with pulse compression. Besides dealing with surface clutter and multiple target situations, such radar detector is often faced with high side-lobes at the compression filter output when strong target presents in his sight. The results of implementing the architecture on a Field Programmable Gate Array (FPGA) are presented and discussed.*

Keywords

CFAR, FPGA, pulse compression, radar, self-clutter.

1. Introduction

The quality of a radar system is quantified with a variety of figures of merit, depending on the function being considered. In analyzing detection performance, the fundamental parameters are the probability of detection P_D and the probability of false alarm P_{FA}. If other system parameters are fixed, increasing P_D always requires accepting a higher P_{FA} as well. Standard radar threshold detection assumes that the interference level is known and constant. On the other hand, this allows accurate setting of a threshold that guarantees a desired P_{FA}.

In practice, interference levels are often variable. In order to obtain predictable and consistent performance, the radar system designer would usually prefer a constant false alarm rate (CFAR). To achieve this, the actual interference power must be estimated from the data in real time, so that the detector threshold can be adjusted to maintain the specified P_{FA}, [1] and [2]. CFAR detection is a set of techniques designed to provide predictable detection and false alarm behavior in realistic interference scenarios.

Several variants of the CFAR algorithm have been proposed in the radar literature to deal with different scenarios in radar applications. These different techniques have been developed in order to increase target detection probability P_D under several environment conditions, especially to deal with two of them: regions of clutter transitions and multiple target situations. Although the theoretical aspect of CFAR detection is advanced, in radar related literature there are few reported practical hardware implementations of CFAR processors because the high computational requirements involved in applications such as radar signal processing.

Thanks to the advent of dedicated hardware multipliers in FPGAs, these devices now challenge general-purpose programmable DSPs for signal processing tasks in many DSP applications. FPGAs have grown from simple, clear logic components to complex circuits. A whole microcomputer system on a chip, including microprocessors, dedicated DSP hardware, memory, and speed input-output components, can be realized by them. Hardware resources that are available and the speed that can be achieved are often much greater than the requirements of the typical problems that are before them. For example, in [21] authors refer that proposed architecture requires 84 milliseconds to process a radar data set of 4096 × 4096 samples, which is 30× times faster than the required theoretical processing time of 2.5 seconds needed for that application parameters.

Simplifying the development is now a key issue. Today FPGA companies not only sell the chips on which the user's designs being implemented, but can also provide many of the fundamental building blocks needed to create these designs. Bearing in mind this, we have modified CFAR detector architecture proposed in [21]. The design is implemented on low cost FPGA and applied to a laboratory pulse compression radar system. In modification, the goal was to introduce on-line reconfigurability both in terms of CFAR method and in terms of the number of reference and guard cells.

The rest of the paper is organized as follows. In Section 2, the related works are highlighted through general CFAR theory review and CFAR hardware implementations review. Section 3 describes in detail the proposed CFAR processor design and implementation. The experimental setup and application of implemented CFAR to a pulse-compression radar system is discussed in Section 4. Section 5 brings experimental results and discussion. Finally, Section 6 presents concluding remarks with some suggestions for further work.

2. Related Works

2.1 CFAR Theory Review

Fig. 1 shows a general block diagram of a generic range CFAR processor. This processor consists of a reference window with $2n$ cells that surround the cell under test. Each cell stores an input sample and such values are right shifted when a new sample arrives. Some $2m$ guard cells are incorporated in order to avoid interference problems in the noise estimation. The spacing between reference cells is equal to the radar range resolution (usually the pulse/ sub-pulse width). The reference cells are used to compute the Z statistic and, depending on the technique, this operation can be linear or nonlinear. The Z statistic and a scaling factor α are used to obtain the threshold. This scaling factor depends on the estimation method applied and the false alarm required according to the application. It is also related to the interference distribution in the radar environment. The resulting product αZ is used as the threshold value that is compared with the cell under test (CUT), to determine whether the CUT is declared a target.

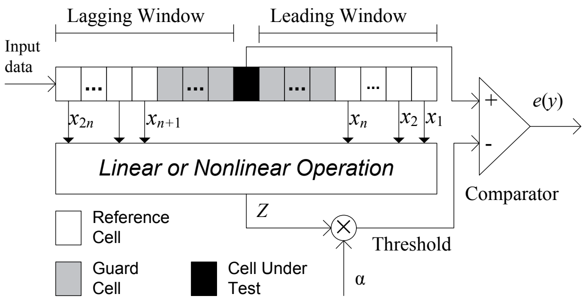


Fig. 1. Generic CFAR processor.

For any radar measurement that is to be tested for the presence of a target, one of two hypotheses can be assumed to be true:

H_0: The measurement is the result of interference only.
H_1: The measurement (y) is the combined result of interference (g) and echoes from the target (d).

This can be modeled by:

$$\begin{aligned} \text{H0: } & y = g, \\ \text{H1: } & y = d + g. \end{aligned} \tag{1}$$

Z statistic computed across the reference cells represents estimated interference power. The required threshold is then estimated as a multiple of Z. The decision criterion is represented by:

$$e(y) = \begin{cases} H_1, & CUT \geq \alpha Z \\ H_0, & CUT < \alpha Z \end{cases} \tag{2}$$

If the values of the CUT exceed the Z statistic, then a target presence is declared, i.e. the CFAR processor outputs 1 if a target is present, otherwise it outputs 0:

$$e(y) = \begin{cases} 1, & CUT \geq \alpha Z \\ 0, & CUT < \alpha Z \end{cases} \tag{3}$$

Fin and Johnson [3] developed a theory based on arithmetic mean of the nearby resolution cells of CUT. This is known as Cell Averaging CFAR (CA-CFAR). CA-CFAR was shown to be not efficient in nonhomogenous environment or in the presence of interfering targets [2]. Other related approaches calculate separate averages of the cells to the left and right of the CUT and then use the greatest or smallest of these two power levels to define the local power level. These techniques are referred to as greatest-of CA-CFAR (CA-GO CFAR) proposed in [4] and smallest-of CFAR (CA-SO CFAR) proposed in [5]. All of these CFAR techniques require linear operations such as getting the maximum, minimum, or average of a set of values. However, GO-CFAR detection performance in multiple target situations is poor and SO-CFAR has undesired effects when interfering targets are located in both halves of the reference cells [2].

To improve the detection performance on these situations, order statistics techniques (OS-CFAR) were proposed in [6] reporting better overall performance results. In [7], two modified OS-CFAR processors that require less processing time than the OS-CFAR processor were proposed. The generalized order statistics processors (GOS-CA-CFAR, GOS-GO-CFAR and GOS-SO-CFAR) are proposed in [8]. These processors achieve better detection performance in the presence or in the absence of interference, so they are more robust than the processors proposed in [6] and [7]. The censored cell-averaging CFAR (CCA-CFAR) is used for case of multiple target situations and it is the first trimmed mean CFAR (TM-CFAR) [9] where the ordered range samples are trimmed only from the upper end. All these techniques require nonlinear operations like sorting a set of values and selecting one on a specific position before performing a linear operation.

Therefore, the method for calculating the Z statistics can be based on linear or non-linear operations on data samples from the reference windows. The most common linear processors are the CA-CFAR, GO-CFAR and SO-CFAR. Basic operation in these processors is the arithmetic mean calculating of the amplitude contained in the Y_1 lagging cells and Y_2 leading cells from the CUT. Then, the CA processor estimates the arithmetic mean, the GO and SO take the major and minor values of Y_1 and Y_2, respectively.

The mathematical model of these three linear operations for the Z statistic is given by (4):

$$Z = \begin{cases} (Y_1 + Y_2)/2, \\ \max(Y_1, Y_2), \\ \min(Y_1, Y_2). \end{cases} \tag{4}$$

Common nonlinear processors are the OSCA-CFAR, OSGO-CFAR and OSSO-CFAR; and their generalized form called GOSCA-CFAR, GOSGO-CFAR and GOSSO-CFAR processors. These order statistics processors rank orders the reference window data samples and then select the *k*th element of the ordered list. The GOSCA-CFAR, GOSGO-CFAR and GOSSO-CFAR processors, perform the selection of the k-th ($Y_{(1)}$) and i-th ($Y_{(2)}$) sorted value from the lagging and leading cells, respectively. Then, the Z statistic is calculated in a similar way to the linear processors as shown in (5):

$$Z = \begin{cases} (Y_{(1)} + Y_{(2)})/2, \\ \max(Y_{(1)}, Y_{(2)}), \\ \min(Y_{(1)}, Y_{(2)}). \end{cases} \tag{5}$$

A different approach to obtain CFAR, based on clutter map, exploits the local homogeneity of radar environment, in which the detector output of each range resolution cell is averaged over several scans in order to obtain an estimate of the background level [10]. In recent years, distributed detection systems based on multiple detectors with data fusion have been widely considered [11–15]. This is due to number of advantages over the centralized detection system employing a single sensor.

2.2 CFAR Hardware Implementations Review

First implemented analog, CFAR detection is today almost exclusively performed with digital signal processing hardware and software. Through the 90s, real-time radar DSP systems were built using discrete logic. Many systems were built using custom devices designed to perform a particular function. The use of those custom devices allowed DSP systems to become very small with high performance. However, they were difficult and expensive to develop, often requiring several design iterations before the device was fully operational. If such a system needs to be modified, the custom devices need to be redesigned, incurring significant expense. Systolic architectures for CFAR processors based on custom VLSI chips are proposed in [16–18]. These systems were very difficult to develop and modify, but in order to achieve the required system performance, it was the only option available.

Digital technology has advanced to the point where several implementation alternatives exist that make the processor more programmable and, hence, easier to design and change. The introduction of the FPGA in the 80s heralded a revolution in the way real-time DSP systems were designed. FPGAs are integrated circuits that consist of a large array of configurable logic elements that are connected by a programmable interconnect structure. FPGAs can also incorporate thousands of multipliers that can be clocked at rates up to a billion and half operations per second, and memory blocks, microprocessors, and serial communication links that can support multigigabit-per-second data transfers. High-performance FPGAs store their configuration in volatile memory, which loses its contents when powered down, making the devices infinitely reprogrammable. Architecture for three versions CFAR processors (CA, CA-GO, CA-SO) on FPGA is presented in [19]. This architecture implements the average computations with two accumulating processing elements and a configurable threshold processing element. An example of OS-CFAR implementation on FPGA, using Virtex II V2MB100 development kit is presented in [20].

A versatile processing architecture that implements six variants of the CFAR algorithm based on linear and nonlinear operations for radar applications is presented in [21]. In [22] 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. Software components and Altera's Nios-II processor accelerated by developed hardware co-processors are used to realize higher-order nonlinear operation like automatic censoring of sorted data. A Real-time implementation approach of a distributed CFAR detection with noncoherent integration is proposed in [23].

Bearing in mind that modern radars usually work in dynamic electromagnetic environment, CFAR detector architecture should be adaptive as more as possible. In [21] authors proposed a versatile architecture in terms of variants of the CFAR algorithm. So, changing CFAR method on-line is possible, keeping number of reference and guard cells constant. This was a base for an upgrade we done. We modified this architecture making it on-line reconfigurable both in terms of CFAR method and in terms of the number of reference and guard cells. An example where it is important is ground surveillance radar with pulse compression [25]. Besides dealing with surface clutter and multiple target situations, such radar detector is often faced with high side-lobes at the compression filter output, so-called 'self-clutter', when strong target presents in his sight. It is shown that changing CFAR processor architecture can be crucial for making correct decision in this scenario.

3. Implementation of a CFAR Processor

3.1 Sorting Block

Sorting data from reference window is critical operation, because of its non-linear nature. Hardware implementations using different kinds of sorting architectures have been presented in literature. In this design we investigated two standard solutions. The first one is iterative, and it is suitable when incoming data clock is several times less than sorting clock. The second architecture is very fast by exploiting a high degree of parallelism. System is designed and implemented by Xilinx's System Generator™ and ISE™ v14.7.

The basic building block for both sorting architectures is a compare-swap element (CSE) that compares two input values and swaps the values at the output, if required. A compare-swap element is depicted in Fig. 2. It can operate fully combinatory, Fig. 2(a), but it is common to add the pipeline registers after each of outputs in order to reduce their critical path and latency, Fig. 2(b).

Fig. 3 shows the modified parallel architecture for suffix sorting proposed in [24]. The proposed sorter has parallel input load instead serial one suggested in [24]. This eliminates need for priority decoder and sorting cell control logic in the CFAR processor proposed in [21]. Hardware design shown in Fig. 3 is actually an iterative implementation of even-odd bubble sorting technique and consists of $n/2$ CSEs run in parallel to build the even stage, and the remaining $(n/2–1)$ CSEs are used for the odd stage. The input of the first stage is read directly from the leading/lagging windows registers. An underlying block is fully combinatory CSE shown in Fig. 2(a).

For correct operation, sorting clock, clk_{sort}, must be several times higher than incoming data clock, so the sorting operation can be done in one data clock cycle clk_{data}. *Read/Sort* block works at sorting clock, and when it is '0' the read operation is performed in one clk_{sort} cycle. Considering that architecture has 2 stages of comparators working fully combinatory, sorting operation is done in maximal $n/2$ sorting clock cycles where n is number of sorting elements. Hence clk_{data} should be $n/2+1$ times smaller than clk_{sort}.

It is possible to reduce critical path adding pipeline registers after the first (even) stage. Nevertheless, in this version sorting operation is lengthened to maximal n sorting clock cycles. In this case clk_{data} should be $n+1$ times smaller than clk_{sort}.

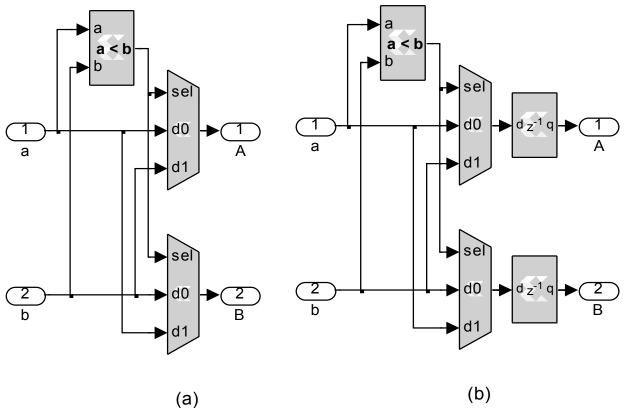


Fig. 2. The compare-swap element: (a) fully combinatory, (b) pipelined.

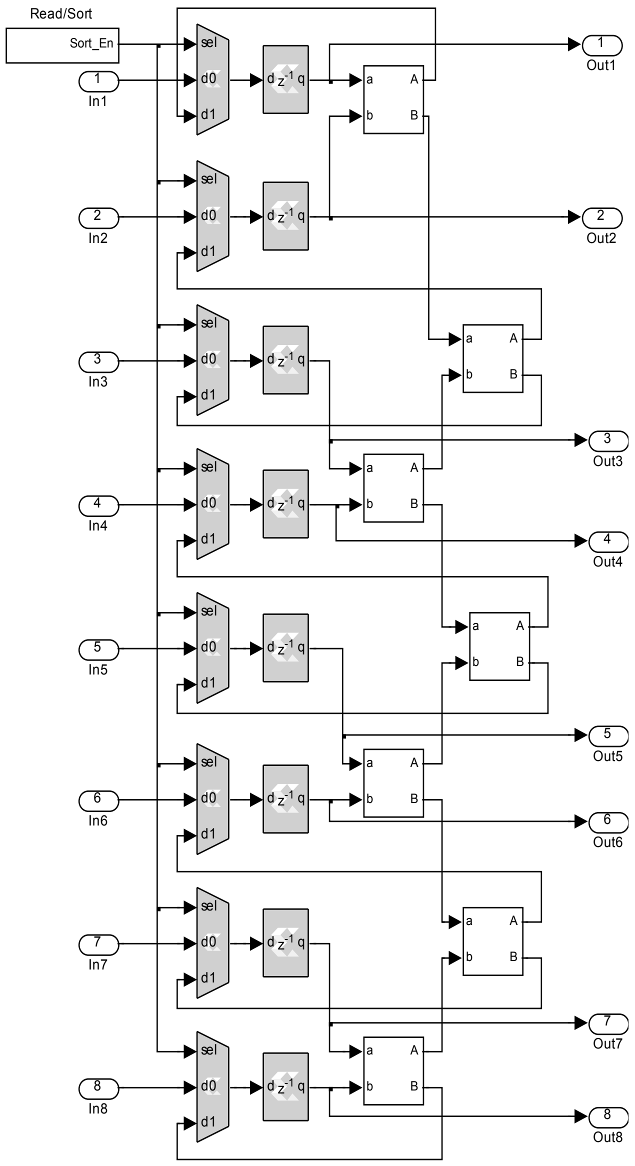


Fig. 3. The iterative even-odd sorting block for 8 data.

This algorithm can be parallelized, by introducing more CSEs, so that compare-swap operations from all the iterations are performed in one clock cycle. Such architectures are known as *sorting networks*. The sorting network that corresponds to the previous scheme requires $(n-1)\times n/2$ CSEs, so it is not practical for large n.

Common architectures include more efficient Batcher's even-odd networks. Fig. 4 illustrates an even-odd network for eight input operands. The network could operate fully combinatory, but it is common to use pipelined CSEs depicted in Fig. 2(b) in order to reduce their critical path and latency thus resulting in a better throughput. Then a set of n samples can be sorted in $\log_2 n\times(\log_2 n+1)/2$ clock cycles ($3\times4/2=6$ in the shown example). The hardware cost of such a sorting network is $(\log_2 n)^2\times n/2+1$ CSEs. Extra pipeline registers in Fig. 4 are inserted in order to equalize latency across the data lines providing a continuous stream sorting.

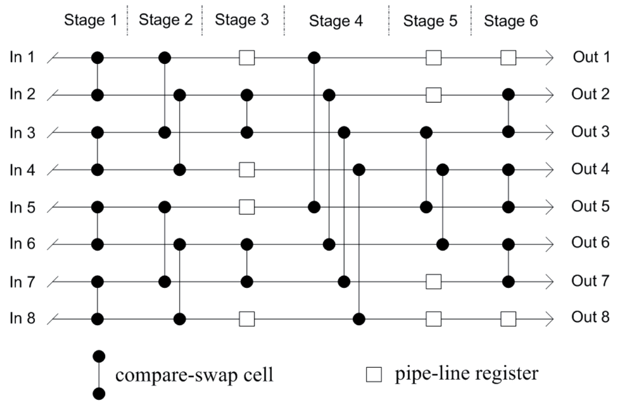


Fig. 4. Batcher's even-odd sorting network.

3.2 CFAR Processor Architecture

The proposed CFAR processor architecture has two shift registers for leading and lagging window each consisting of n reference cells and m guard cells, parallel sorting arrays for reference cells, and one shift register for the CUT, which is at the middle of these registers.

Leading and lagging windows are the key elements of the architecture for the realization of ability of on-line changing n and m. An example is shown in Fig. 5, where a user selectable 4–8–16-cells lagging window with up to 4 user selectable guard cells is presented. For linear CFAR techniques, number of reference cells is set changing *Lag_end* position by the control input *sel_Nref* and the multiplexer signed as *MuxNrefLin* in Fig. 5. Values of 0, 1 and 2 of *sel_Nref* set *Lag_end* at the output of the 4th, 8th and 16th data register. For nonlinear CFAR techniques, number of reference cells is set inserting zeros in some registers which produces zeros at the outputs unused in sorting. When the control input *sel_Nref* has value of 0, comparator *Comp1* has value of 1 and multiplexer *Mux1* produces zeros at the output inducing zeros in all registers after 4th. These zeros have no effect on sorting operation, so that only the first 4 are relevant. When the control input *sel_Nref* has value of 1, comparator *Comp1* has value of 0 and multiplexer *Mux1* passes data from previous register at the output. Now comparator *Comp2* has value of 1 and multiplexer *Mux2* produces zeros at the output inducing zeros in all registers after 8th. Four registers are added at the end of reference cells registers in order to provide guard cells. Number of guard cells is set changing *Lag_out* position by the control input *sel_Ng* and the multiplexer signed as *MuxNg* in Fig. 5.

Also, the proposed CFAR processor architecture, Fig. 6, has two n-input–1-output multiplexers that perform the rank operation for the lagging and leading sorting arrays. Given that the reference cells values are ordered, the k-th and i-th value can be selected by the control signals *sel_k* and *sel_i* respectively. The result of this selection is the $Y_{(1)}$ and $Y_{(2)}$ values needed in the nonlinear operations shown in (5).

For the linear operations presented in (4), it is required to add all values stored in the reference cells registers of the leading/lagging windows for computing the average. In order to perform this operation, it is not necessary to add all values each time that one value from the reference cells registers is inserted and deleted. Once a value is inserted and other one deleted, the preceding result can be used to compute the next result without adding all values. Only by adding and subtracting the newest and oldest values respectively, the next result is obtained. This whole operation can be performed by an accumulator, which computes the average of Y_1 and Y_2 values on each window. Accumulators are signed as *PE_leading* and *PE_lagging*. The PE accumulator consists of an adder, which receives the incoming value, a subtracter, which selects the oldest value stored in the reference cells of leading/lagging window, a register to store the accumulated value. A left shifter that performs the division needed to compute the average value is added later, in the next block, Fig. 6. Because of the left shifter, the number of reference cells must be a power of two. This does not restrict the usability of this architecture as the number of reference cells used in practical applications is usually a power of two [1].

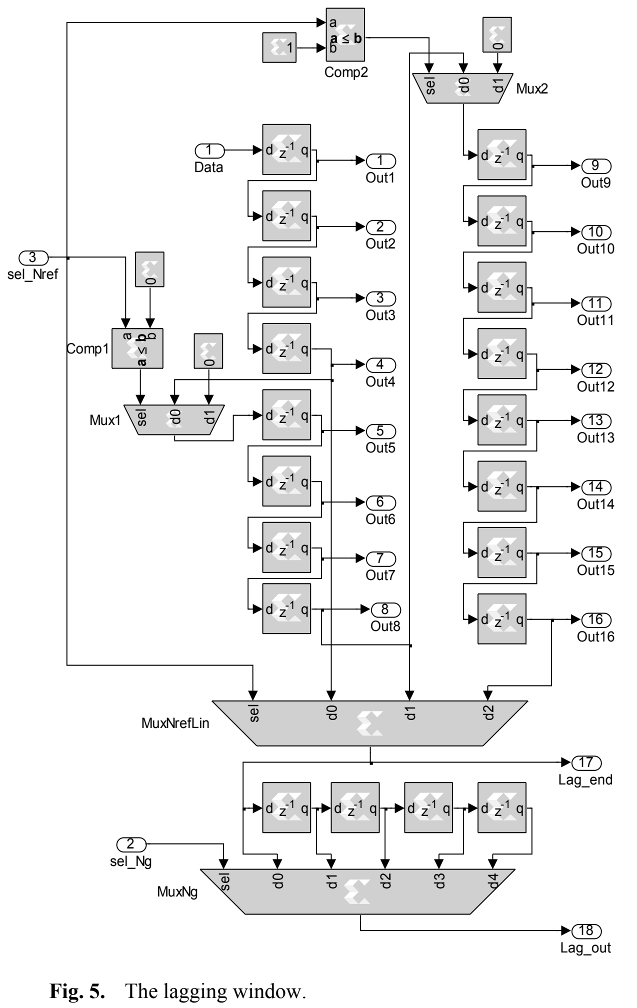


Fig. 5. The lagging window.

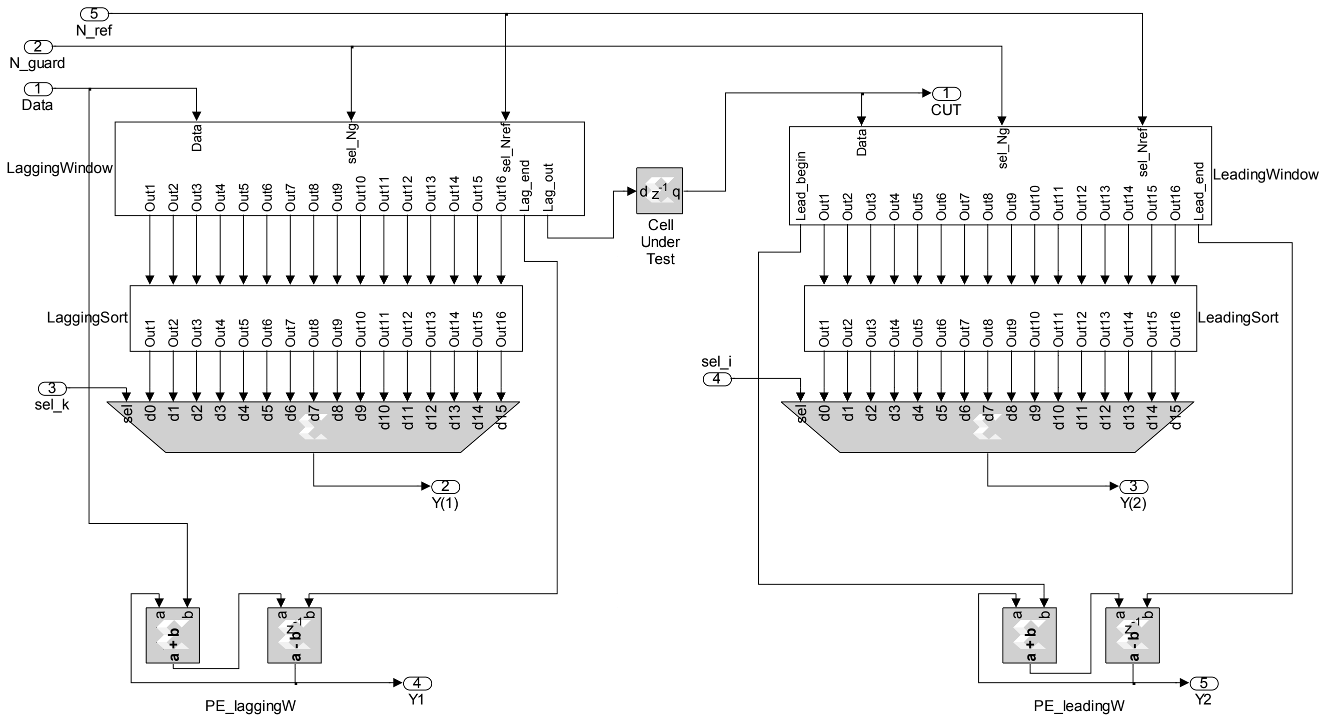


Fig. 6. CFAR processor hardware architecture.

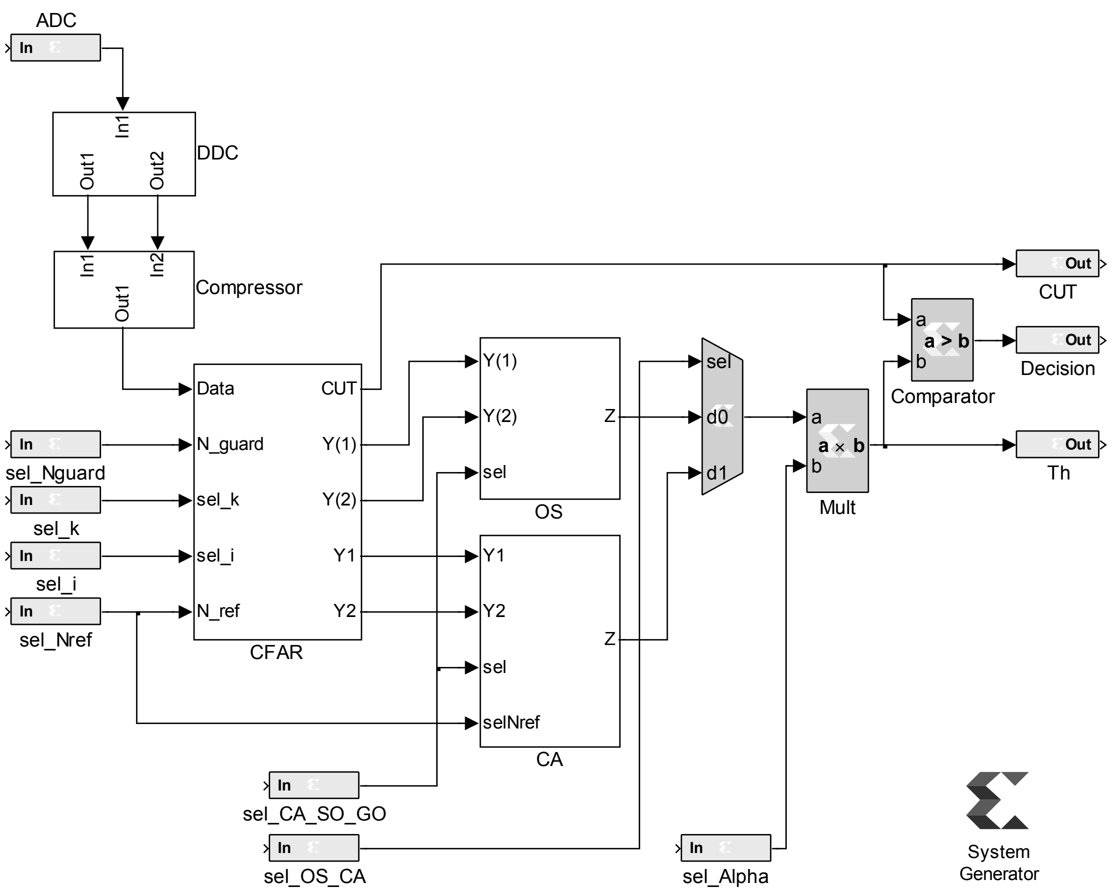


Fig. 7. CFAR processor applied to a pulse compression radar system.

Sorting method	Ref. cells amount (2n)	Slices count (max. 3,758)	Speed [MHz] (max. 350)	Process. time [clock cycles]	Throughput [MSps]	Efficiency [Mbps/Slice]
Iterative, fully comb. CSEs	8	165	189.7	3	63.2	6.08
	16	276	182.9	5	36.6	2.08
	32	514	181.6	9	20.2	0.64
Iterative, pipelined CSEs	8	168	343	5	68.6	6.56
	16	281	326.1	9	36.2	2.08
	32	527	315.7	17	18.6	0.56
Even-odd sorting network	8	144	317.4	1	317.4	35.30
	16	355	299.6	1	299.6	13.44
	32	983	283	1	283	4.64

Tab. 1. Resources utilization for a Xilinx's Spartan-6 XC6SLX25 FPGA device.

4. Application to a Pulse-Compression Radar System

The achievable combinations of P_D and P_{FA} are determined by signal and interference statistic, especially the signal-to-interference ratio. When multiple targets are present in the radar scope, additional considerations of resolution and side lobes arise in evaluating detection performance. For example, if two targets cannot be resolved by radar, they will be registered as a single object. If side lobes are high, the echo from one strongly reflecting target may mask the echo from a nearby but weaker target, so that again only one target is registered when two are present. Resolution and side lobes in range are determined by the radar waveform. In ground surveillance pulse-Doppler radar systems, range-Doppler processing is usually applied. It means that detection is accomplished in several Doppler channels. In Doppler channels close to zero, ground clutter is a dominant part of interference. In higher Doppler channels, thermal noise and self-clutter are prevailing.

The proposed radar processor architecture has digital down converter (DDC), compression filter and CFAR processor, Fig. 7. First two blocks are described in [26] and [27]. The architecture was modeled using the VHDL and Xilinx's System Generator™. The behavioral VHDL description of this design is placed and routed on a Spartan 6 device by the ISE™ 14.7 tool. Currently, the whole project is on XC6SLX25 device. Beside the main part described in the previous section which calculates Y_1, Y_2, $Y_{(1)}$ and $Y_{(2)}$, CFAR processor has parts for CFAR method selection, Z statistic computing and threshold computing multiplying Z and α. Architecture is on-line reconfigurable by control inputs *sel_Nguard*, *sel_k*, *sel_i*, *sel_Nref*, *sel_CA_OS*, *sel_CA_SO_GO* and *sel_Alpha*. The default configuration of the CFAR processor uses 16-bit for data, 32 reference cells and 8 guard cells and k-th and i-th rank-order sample equals 12 (3/4 of leading/lagging register length). The value used for scaling factor was $\alpha = 2$.

5. Results and Discussion

Proposed architectures are compared in terms of hardware resources, throughput and efficiency. The throughput is defined as a number of processed samples (CUTs) per second, whereby samples are 16 bits wide. The throughput was calculated using (6), whereas the efficiency with (7).

$$Throughput = \frac{1\ \ Sample}{Clock\ \ time \times Clock\ \ cycles}, \tag{6}$$

$$Efficiency = \frac{Throughput}{Number\ of\ \ slices}\ . \tag{7}$$

The first proposed architecture (with iterative even-odd sorter) produces an output result on each data clock cycle after the latency period. The latency period is proportional to the number of reference cells, and the number of the guard cells around the CUT. The latency arises at the start of processing since the pipeline or shift register must be full in order to output a result. System clock must be $n/2$+1 (fully combinatory version) apropos n+1 (pipelined version) times higher than incoming data clock, so the sorting operation can be done in $n/2$+1 apropos n+1clock cycle.

The second proposed architecture (with even-odd sorting network) produces an output result on each system clock cycle after the latency period. Once the data stream starts, after $2n+2m+1+N_{sort}$ clock cycles, the CFAR architecture produces a valid output each clock cycle. So data processing time in this case equals one clock cycle.

The architecture was synthesized for a Xilinx's XC6SLX25 Spartan 6 FPGA device using different number of reference cells. Tab. 1 summarizes the results in terms of FPGA hardware resources utilization including four guard cells at each side of the CUT and excluding DDC and compressor. All these three configurations use 16-bit to represent input data.

With a greater configuration of the CFAR detector of 16-bits of data, 32 reference cells and 8 guard cells, the first architecture, with non-pipelined CSEs stages in sorting windows, achieved a throughput of 20.2 MSps, with a clock frequency of 181.6 MHz, 514 used slices and 9 clock cycles resulting in efficiency of 0.64 Mbps/Slice on XC6SLX25 device. Adding pipeline registers between even and odd CSE stages in the second architecture reduced critical path resulting in speed increasing to 315.7 MHz. Nevertheless, in this version sorting operation is lengthened to 17 sorting clock cycles, so there is no significant changes in achieved throughput (18.6 MSps) and efficiency (0.56 MSps/Slice). The third architecture, with full parallel sorting networks, produces results in each clock cycle, so achieved throughput is an order of magnitude increased (283 MSps). On the other hand, architecture is implemented using 983 slices resulting in efficiency of 4.64 Mbps/Slice.

Hence, the first two versions are suitable when incoming data clock is several times less than sorting clock because of lower space occupancy. In this case it is of order 20 MSps (at $2n = 32$) which is sufficient if signal bandwidth is smaller than 20 MHz, i.e. radar range resolution is not less than 7.5 m. In the state of the art HRR radars (High Resolution Range) where the radar range resolution is below 1 m, signal bandwidth is order of hundreds MHz, throughput is order of hundreds MSps, so the third CFAR architecture is the only option. Implemented on this low cost platform, this hardware design achieves throughput of 283 MSamples/s (at $2n = 32$) which is sufficient if signal bandwidth is smaller than 283 MHz, i.e. radar range resolution is not less than 0.53 m.

To validate the results of the proposed CFAR processor, the experimentally generated data were passed through the pulse compressor and six selected variants of the CFAR algorithm. Input data, i.e. uncompressed radar video signal, were obtained from a waveform generator. It is a typical choice using long pulses in radars which has small transmitting power, e.g. ground surveillance portable radars [25]. In order to achieve resolution of order tens to hundred meters, bandwidth of order MHz is used. In this case we use chirp signal with duration $T = 100$ µs and bandwidth $B = 1$ MHz. Signal is sampled at frequency of 1 MHz, so $TB = 100$ samples per pulse are obtained and compressor has 100 complex coefficients. CFAR input data, i.e. range profiles were obtained from the output of the radar compressor. The results from the output of the CFAR processor (default configuration) are sent to DAC and shown on digital scope. Fig. 8 shows the resulting threshold calculated by the linear (Fig. 8(a)) and nonlinear (Fig. 8(b)) CFAR detectors implemented in hardware using as input the radar receiver range profile when single target is present. It is a typical situation in pulse compression radars in higher Doppler channels when a strong target is in radar scope. Sharp main lobe at zero time represents this target. Side-lobes at the output of radar compression filter, often named as 'self clutter', are dominant part of interference. Side-lobes near the main and side-lobes at the edge (far side-lobes) are emphasized, so they can be considered as regions of clutter transitions. At first glance, there are not significant differences among the thresholds in the middle side-lobes region. In the near side-lobes region, linear techniques, CA and CA-GO, overvalue self-clutter, while both smallest-of techniques, CA-SO and OS-SO, undervalue it. In the far side-lobes region, all techniques undervalue the greatest side-lobe occurring false alarms. Conclude OS-CA CFAR exerts the best agreement in shape with self clutter. As problem with far side-lobes exists in all these CFAR processors, some kind of side-lobes shaping should be applied in the receiver. Mismatching the receiver is commonly technique [27].

To compare the results of the implemented CFAR processor to the theoretical, the six selected variants of the CFAR algorithm were modeled in software using MATLAB® and compared to the measured data at the output of the proposed hardware design. The results are similar for all processors. There are small differences between the thresholds calculated in software and hardware caused primarily by ADC and DAC. As an example, the comparison between real and theoretical thresholds when OS-CA CFAR is applied is shown in Fig. 9. Measured and simulated data manifest high similarity in shape. Difference is made by presence of noise in measured data. It can be caused by filter imperfection in DAC board. A modified configuration of CFAR with $2n = 8$ reference cells and $2m = 0$ guard cells is used. It is clear that this configuration valuate far side-lobes quite better than default one, so there is no false alarms.

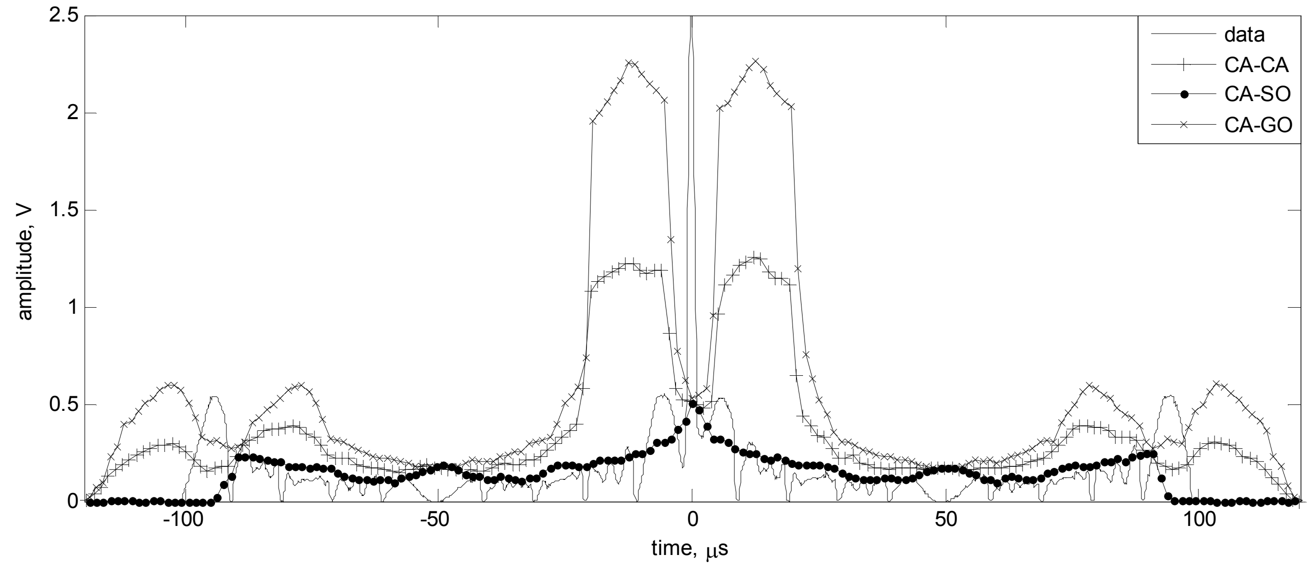
Fig. 10 clarifies the way we obtain Fig. 8 and Fig. 9. There is the radar receiver range profile and threshold calculated by CFAR processor on the oscilloscope screen. Control pins from Fig. 6 are changed choosing CFARs one by one. Obtained data from the scope are stored on personal computer and reproduced by MATLAB®.

6. Conclusion

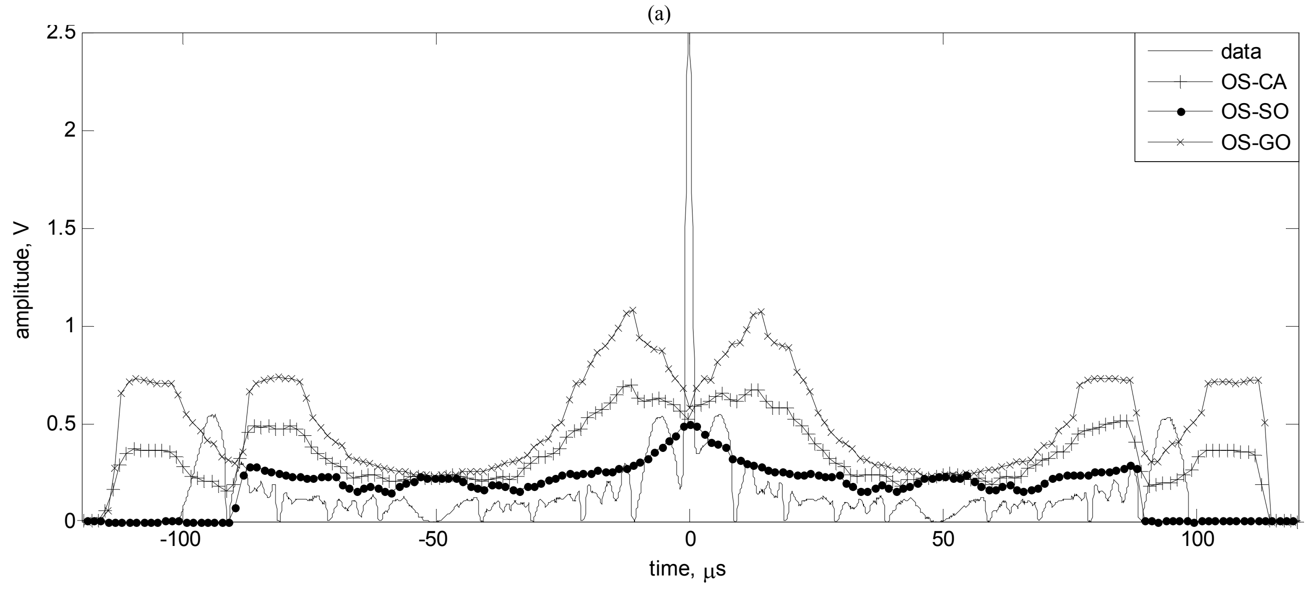
A hardware design that implements a CFAR processor using higher-level blocks and system-level hardware design tools, actually Xilinx's System Generator™ for Matlab Simulink™, is presented. The availability of such varied libraries of functions and the blank canvas of the FPGA brings great power to even the smallest design team. They no longer have to rely on internal experts in certain areas, allowing them to concentrate on the overall design.

Hence, thanks to the upgrade of low cost FPGA technology and hardware design tools, we designed one generalized CFAR processor for various types of CFAR methods instead of several specialized ones.

The system has been implemented on a Spartan6 FPGA by Xilinx achieving real-time execution times and minimum levels of error between the ideal results and the real ones obtained from the hardware implementation. The hardware resources utilization and speed has been analyzed



(a)



(b)

Fig. 8. Radar receiver range profile and thresholds calculated by the (a) linear and (b) nonlinear CFAR detector implemented in hardware.

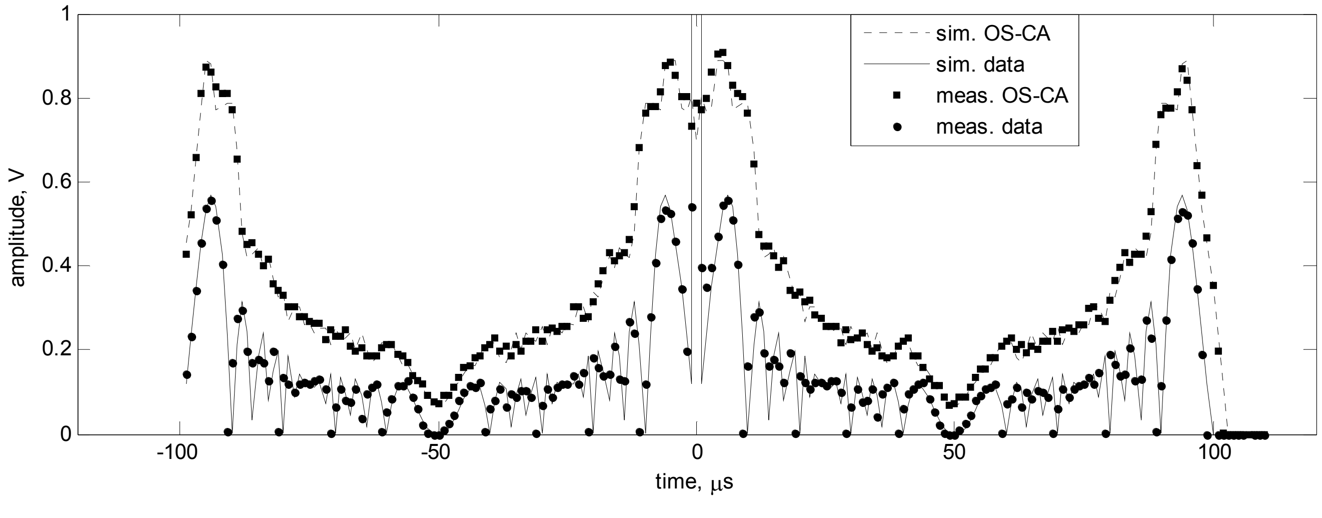


Fig. 9. Measured and simulated data when OS-CA CFAR with $2n = 8$ reference cells and $2m = 0$ guard cells is applied.

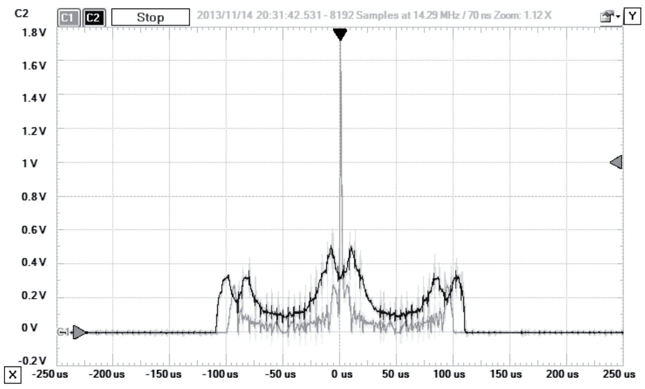


Fig. 10. Radar receiver range profile and threshold on the oscilloscope screen.

modifying the number of the used reference cells. The results show that this module can be successfully used in the development of high performance multi-mode radar/ sonar systems as a part of radar signal processor in chip.

The design exploration is performed for the CA and OS CFAR techniques to satisfy high-resolution target detection, with range resolution of 0.53 m when 32 reference cells and 8 guard cells are used. Proposed hardware design has the advantages of being simple and fast with a low development cost. Also, the performance of the prototype hardware setup proved the concept of the design within a reasonable design time.

References

[1] SKOLNIK, M. I. *Introduction to Radar Systems*. 3rd ed. New York: McGraw-Hill, 2001.

[2] GANDHI, P. P., KASSAM, S. A. Analysis of CFAR processors in nonhomogeneous background. *IEEE Transactions on Aerospace and Electronic Systems*, 1988, vol. 24, no. 4, p. 608 - 621.

[3] FINN, H. M., JOHNSON, R. S. Adaptive detection mode with threshold control as a function of spatially sampled clutter-level estimates. *RCA Review*, 1968, vol. 29, p. 414 - 464.

[4] HANSEN, G. V., SAWYERS, J. H. Detectability loss due to greatest-of selection in a cell averaging CFAR. *IEEE Transactions on Aerospace and Electronic Systems*, 1980, vol. 16, no. 1, p. 115 - 118.

[5] WEISS, M. Analysis of some modified cell-averaging CFAR processors in multiple-target situations. *IEEE Transactions on Aerospace and Electronic Systems*, 1982, vol. 18, no. 1, p. 102 to 114.

[6] ROHLING, H. Radar CFAR thresholding in clutter and multiple target situations. *IEEE Transactions on Aerospace and Electronic Systems*, 1983, vol. 19, no. 4, p. 608 - 621.

[7] ELIAS-FUSTE, A. R, GARCÍA, G. M., REYES-DAVO, E. Analysis of some modified order statistic CFAR: OSGO and OSSO CFAR. *IEEE Transactions on Aerospace and Electronic Systems*, 1990, vol. 26, no.1, p. 197 - 202.

[8] YOU, H. Performance of some generalised modified order statistics CFAR detectors with automatic censoring technique in multiple target situations. *IEE Proceedings - Radar, Sonar and Navigation*, 1994, vol. 141, no. 4, p. 205 - 212.

[9] RICKARD, J. T., DILLARD, G. M. Adaptive detection algorithms for multiple target situations. *IEEE Transactions on Aerospace and Electronic Systems*, 1977, vol. 13, no. 4, p. 338 - 343.

[10] NITZBERG, S. Clutter map CFAR analysis. *IEEE Transactions on Aerospace and Electronic Systems*, 1986, vol. 22, no. 4, p. 419 to 421.

[11] ELIAS-FUSTE, A. R., BROQUETAS-IBARS, A., ANTEQUERA, J. P., YUSTE, J. C. M. CFAR data fusion center with inhomogeneous receivers. *IEEE Transactions on Aerospace and Electronic Systems*, 1992, vol. 28, no. 1, p. 276 - 285.

[12] UNER, M. K., VARSHNEY, P. K. Distributed CFAR detection in homogeneous and nonhomogeneous backgrounds. *IEEE Transactions on Aerospace and Electronic Systems*, 1996, vol. 32, no. 1, p. 84 - 97.

[13] BARKAT, M., VARSHNEY, P. K. Decentralized CFAR signal detection. *IEEE Transactions on Aerospace and Electronic Systems*, 1989, vol. 25, no. 2, p. 141 - 149.

[14] MEZIANI, H. A., SOLTANI, F. Decentralized fuzzy CFAR detectors in homogenous Pearson clutter background. *Signal Processing*, 2011, vol. 91, no. 11, p. 2530–2540.

[15] IVKOVIĆ, D., ANDRIĆ, M., ZRNIĆ, B. A new model of CFAR detector. *Frequenz* (accepted for publication Oct. 24, 2013.) [Online] Cited 2014-01-31. DOI: 10.1515/freq-2013-0087

[16] HWANG, J. N., RITCHEY, J. A. Systolic architectures for radar CFAR detectors. *IEEE Transactions on Signal Processing*, 1991, vol. 39, no.10, p. 2286 - 2295.

[17] HAN, D. S. VLSI architectures for CFAR based on order statistic. *Signal Processing*, 1997, vol. 62, no. 1, p. 73 - 86.

[18] BEHAR, V. P., KABAKCHIEV, C. A., DOUKOVSKA, L. A. Adaptive CFAR PI processor for radar target detection in pulse jamming. *Journal of VLSI Signal Processing*, 2000, vol. 26, no. 3, p. 383 - 396.

[19] TORRES, C., CUMPLIDO, R., LOPEZ, S. Design and implementation of a CFAR processor for target detection, In *Proceedings of the 14th International Conference on Field Programmable Logic, FPL04. Lectures Notes on Computer Science,* 2004, vol. 3203, p. 943 – 947.

[20] MAGAZ, B., BENCHEIKH, M. L. An efficient FPGA implementation of the OS-CFAR processor. In *Proceedings of the 9nd International Radar Symposium*. Wroclaw (Poland), 2008, p. 1 - 4.

[21] PEREZ-ANDRADE, R., CUMPLIDO, R., FEREGRINO-URIBE, C., DEL-CAMPO, F. M. A versatile hardware architecture for a constant false alarm rate processor based on a linear insertion sorter. *Digital Signal Processing*, 2010, vol. 20, no. 6, p. 1733 to 1747.

[22] DJEMAL, R., BELWAFI, K., KAANICHE, W., ALSHEBEILI, S.A. A novel hardware/software embedded system based on automatic censored detection for radar systems. *International Journal of Electronics and Communications (AEÜ)*, 2013, vol. 67, no. 4, p. 301 - 312.

[23] BENSEDDIK, H. E., HAMADOUCHE, M., KHOUAS, A. FPGA-based real-time implementation of distributed system CA-CFAR and clutter MAP-CFAR with noncoherent integration for radar detection. In *Proceedings of the 2nd International Symposium on Modeling and Implementation of Complex Systems*. Constantine (Algeria), 2012, p. 61 - 67.

[24] MARTINEZ, J., CUMPLIDO, R., FEREGRINO, C. An FPGA-based parallel sorting architecture for the Burrows Wheeler transform. In *Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs, ReConFig 2005.* Puebla City (Mexico), Sept. 2005. DOI: 10.1109/RECONFIG.2005.9

[25] SIMIĆ, S., ZEJAK, A. J., GOLUBIČIĆ, Z. Range sidelobe reduction in the portable battlefield surveillance radar. In *Proc. of the 10th Internat. Conf. on Telecommunications in Modern Satellite, Cable and Broadcasting Services*. Niš (Serbia), 2011, p. 571–574.

[26] GOLUBIČIĆ, Z., SIMIĆ, S., ZEJAK, A. J. Design and FPGA implementation of digital pulse compression for band-pass radar signals. *Journal of Electrical Engineering*, 2013, vol. 64, no. 3, p. 191–195.

[27] SIMIĆ, S., ZEJAK, A. J., GOLUBIČIĆ, Z. Hardware implementation of DIRLS mismatched compressor applied to a pulse-Doppler radar system. *Microprocessors and Microsystems*, 2013, vol. 37, no. 4-5, p. 381–393.

About Authors ...

Slobodan SIMIĆ was born in 1976. He received his B.Sc. in Electrical Engineering from the Military Technical Academy of Belgrade (Sep. 2000), M.Sc. degree from the Faculty of Electrical Engineering, University of Belgrade (2006), Ph.D. degree from the Faculty of Technical Sciences, University of Novi Sad (2013). He is a Teaching Assistant at the Department of Electronic Systems, Military Academy, University of Defense in Belgrade. His research interests include digital signal processing and digital design. He published over 40 papers in national and international conferences proceedings and journals.

Milenko ANDRIĆ was born in Pljevlja in 1972, Montenegro. He received the B.Sc. degree in Electronics Engineering from the Military Technical Academy, Serbia in 1995. He received M.Sc. and Ph.D. in Electrical Engineering in 2001 and 2006, respectively. Currently, he is an associate professor at the Department of Military Electronics Engineering and he also works as a scientific researcher at the Electronic Systems Laboratory, Military Academy in Belgrade. His main research interests are in the fields of stochastically process in telecommunication and radar engineering, pattern recognition, methods for signals analysis and digital signal processing. He has published more than 50 papers in national and international conferences and journals.

Bojan ZRNIĆ graduated in 1992 on the Military Technical Academy in Belgrade with B.Sc. degree in Electrical Engineering. He received M.Sc. and Ph.D. in Electrical Engineering in 1998 and 2001, respectively. Currently, he is a Head of the Defense Technology Department in Serbian MOD. He is also visiting professor at the Serbian Military Academy on the radar and EW subjects. His research work includes radar signals and systems. He published over 70 papers in national and international conferences proceedings and journals.