Design and Implementation of Real-Time Automatic Censoring System on Chip for Radar Detection

Imron Rosyadi, Ridha A. Djemal, and Saleh A. Alshebeili

Abstract—Design and implementation of a novel B-ACOSD CFAR algorithm is presented in this paper. It is proposed for detecting radar target in log-normal distribution environment. The B-ACOSD detector is capable to detect automatically the number interference target in the reference cells and detect the real target by an adaptive threshold. The detector is implemented as a System on Chip on FPGA Altera Stratix II using parallelism and pipelining technique. For a reference window of length 16 cells, the experimental results showed that the processor works properly with a processing speed up to 115.13MHz and processing time0.29 µs, thus meets real-time requirement for a typical radar system.

Keywords—CFAR, FPGA, radar.

I. INTRODUCTION

Radar is an electromagnetic system that detects, locates, and recognizes target objects. Radar transmits electromagnetic signal and then receives echoes from target objects to get their location or other information. The received signal is frequently accompanied by noise and clutter. The disturbances may cause serious performance issues with radar systems by concluding these signals as targets.

To make a right censoring decision, the receiver is desired to achieve constant false alarm rate (CFAR) and maximum probability of target detection. Modern radars usually detect the targets by comparing with adaptive thresholds based on a CFAR processor. In this processor, threshold is determined dynamically based on the local background noise/clutter power.

The CFAR detectors have been widely used in radar signal processing applications to detect the targets from noisy background. Gini et al. noted a list of more than 120 papers about CFAR detection [1]. Though the theoretical aspect of CFAR detection is developed well, the practical hardware applications are not. The computational requirements in radar signal processing should be met by advanced technologies that employ high parallel computational technique.

II. BACKGROUND THEORY

In a radar system, it is needed to determine the power threshold which any return can be considered from a target. In most radar detectors, the threshold is set in order to reach a required probability of false alarm rate. In natural environment, unwanted clutter and interference sources changes spatially and temporally. In this situation, an adaptive threshold should be employed, where the threshold level is changed to maintain a constant probability of false alarm. This method is known as constant false alarm rate (CFAR) detection.

A typical CFAR processor is shown in Fig. 1. The input signals are set serially in a shift register. The content of the cells surrounding the cell under test \(X_{s}\) are processed by a CFAR processor to get the adaptive threshold \(T\). Then \(X_{s}\) is compared with \(T\) to make the decision. The cell under test is declared as a target if its value exceeds the threshold value.
parallel/pipeline processing for Max, Min, and Cell-Average (CA) CFAR algorithms. OS-CFAR was implemented using parallel structure in [18]. In [19], CA-CFAR and OS-CFAR are combined and implemented in FPGA. TM-CFAR was implemented in [20]. All of these implementation were for simple CFAR algorithms suitable for Gaussian distribution type of clutter.

An automatic censoring CFAR detector called Automatic Censored Cell Averaging (ACCA) ODV CFAR was designed by Alsuwailem et al. [21] as a simply IP core without flexible communication with others radar system peripherals. Winkler et al. [22] used SoC with reconfigurable processor inside for an automotive radar sensor. The processor is responsible for controlling the custom logic and IO tasks. Simple OS-CFAR is incorporated in the system.

A recent automatic censoring called B-ACOSD CFAR have introduced by Almarshad et al. [23]. The algorithm is able to make automatic censoring of unknown number of interfering targets in log-normal clutter. Because of increase in radar resolution, the log-normal distribution becomes more reliable to represent the amplitude of clutter than Rayleigh distribution. Meanwhile, the automatic censoring algorithms developed for Rayleigh clutter as presented in [10] and [21] cannot straightforwardly be extended to the case where clutter samples are drawn from log-normal distribution.

III. B-ACOSD CFAR ALGORITHM

B-ACOSD CFAR has been proposed in [23]. The algorithms consist of two steps: removing the interfering reference cells and the actual detection. Both steps are performed dynamically by using a suitable set of ranked cells to estimate the unknown background level and set the adaptive thresholds accordingly. This detector does not require any prior information about the clutter parameters nor do they require the number of interfering targets.

In a CFAR processor, the radar outputs \( \{X_i; \ i = 0, 1, \ldots, N\} \) are stored in a tapped delay line. The cell with the subscript \( i = 0 \) is the cell under test, where it contains the signal which should be detected as a target or not. The last \( N \) surrounding cells are the auxiliary cells used to construct the CFAR procedure.

In censoring step, the procedures first rank the outputs of all reference range cells in ascending order according to their magnitudes to yield

\[
X(1) \leq X(2) \leq \cdots \leq X(p) \leq \cdots \leq X(N).
\]

In the algorithm, sample \( X(N) \) is compared with the adaptive threshold \( \tau_N \) defined as

\[
\tau_N = X(1)^{\frac{p}{p-1}} X(p)^{\frac{p-1}{p}}.
\]

\( X(p) \) is the \( p^{th} \) largest sample and \( \tau_N \) is a constant chosen to achieve the desired probability of false censoring (\( P_{\text{false}} \)). It is found that values of \( p > N/2 \) yield reasonable good
performance in detection [26].

If \( X(N) < T_{\text{cl}} \), the algorithm decides that \( X(N) \) corresponds to a clutter sample without interference, and it terminates. If, on the other hand, \( X(N) > T_{\text{cl}} \), 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

\[
T_{\text{cl}} = X(1)^{1-\varepsilon_2} X(p)^{\frac{\varepsilon_2}{2}},
\]

(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 \( T_{\text{ck}} \) and a decision is made according to the test,

\[
\begin{align*}
X(N - k) &> T_{\text{ck}} \\
H_1 \bigg/ \bigg. \\
X(N - k) &< T_{\text{ck}} \\
H_0
\end{align*}
\]

(4)
where

\[
T_{\text{ck}} = (X(1)^{1-\varepsilon_k} X(p)^{\frac{\varepsilon_k}{2}}).
\]

(5)

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

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

\[
\begin{align*}
H_2 &> \\
X_0 &< T_{\text{ck}} \\
H_0
\end{align*}
\]

(6)
Hypothesis \( H_2 \) denotes the presence of target in the test cell, while hypothesis \( H_0 \) denotes there is no target.

In B-ACOSD CFAR, the threshold \( T_{\text{ck}} \) is defined as,

\[
T_{\text{ck}} = X(1)^{1-\varepsilon_k} X(p)^{\frac{\varepsilon_k}{2}},
\]

(7)
where the value of \( \varepsilon_k \) is selected so that the design probability of false alarm (\( P_{\text{fa}} \)), and \( k \) is the number of interfering targets found in censoring step.

The values of \( \alpha \) and \( \beta \) for each detected interference are defined by Monte Carlo simulation with 500,000 independent runs by maintaining low value of \( P_{\text{fa}} \) and \( P_{\text{fa}} \), respectively. The threshold parameters for \( (N, p) = (16, 12) \), \( P_{\text{fa}} = 0.001 \), and \( P_{\text{fa}} = 0.01 \) are presented in Table I.

<table>
<thead>
<tr>
<th>( k )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha_k )</td>
<td>2.60</td>
<td>2.04</td>
<td>1.71</td>
<td>1.44</td>
<td>-</td>
</tr>
<tr>
<td>( \beta_k )</td>
<td>1.64</td>
<td>1.89</td>
<td>2.12</td>
<td>2.37</td>
<td>2.64</td>
</tr>
</tbody>
</table>

IV. B-ACOSD CFAR SOC ARCHITECTURE

A. System Overview

Design flow for the system architecture development consists of some steps started with Monte Carlo simulation using Matlab®. Monte Carlo simulation will define values of \( \alpha_k \) and \( \beta_k \) for some values of \( N \) and \( p \). The B-ACOSD CFAR algorithm for specific value of \( N \) and \( p \) then is executed under Matlab® fixed-point simulation to improve performance and maintain its accuracy when implemented in a digital system. The next step is HDL coding and test bench simulation which be done with ModelSim® from Mentor Graphics.

Integration of all components in a SoC is done using SOPC-Builder®. The software provides useful and easy tools to connect many logic elements ranging from processor, memory, to user-logic elements. Quartus II® from Altera was employed for synthesis, timing simulation and downloading the design.
into FPGA chip. Hardware Abstraction Layer needed by the system was coded, compiled, and downloaded using NIOS-II IDE®.

The SoC consists of five main modules: Processor, On-Chip ROM, B-ACOSD CFAR Detector, Input/ROM interface, and Output/RAM interface, as shown in Fig. 3. The SoC utilize Nios II soft-core as the processor. Nios II is a 32-bit embedded-processor architecture designed specifically for the Altera family of FPGAs. Nios II has some key features such as custom instructions and easy custom peripherals management. Nios II’s core type is used because it is suitable to maintain a balance between performance and cost.

Input signal from an envelope detector sent directly to the Avalon bus through an input interface or be stored first in a 16-MBytes flash ROM provided by a development board and connected to the system through flash ROM driver. The censoring results are stored in a 2-MBytes external SSRAM controlled by a SSRAM driver.

Since computation of the exponential equation (5) and (7) is hard and high cost, the equations are converted respectively into logarithmic form as follows,

\[ \log T_{\text{ref}} = \left( 1 - c_i \right) \cdot \log X(1) + c_i \cdot \log X(p) \]  
\[ \log T_{\text{test}} = \left( 1 - \beta_i \right) \cdot \log X(1) + \beta_i \cdot \log X(N - k). \]  

In those forms, power computation becomes a simple multiplication, and the multiplication becomes an addition. Because the logarithmic computation in hardware is a complex and slow task, the logarithmic computation is simplified using a look-up table. The look-up table contains range of a lognormal distribution with \( \mu = 1 \) and \( \sigma = 1.1 \) as suggested in [23], based on real radar input data measurement. Following the number representation change to logarithmic form, test cell value was also converted accordingly.

The look-up table resides on 32K on-chip ROM inside FPGA. The data distribution resolution in the 32K on-chip ROM is 0.0610. MATLAB fixed-point B-ACOSD CFAR simulation with this resolution gives censoring results as good as its real-number simulation.

B. B-ACOSD Detector Architecture

The CFAR detector comprises five main modules: I/O interface, shift register, sorting module, censoring module, and comparator as shown in Fig. 4. The shift register consists of \( N \) reference cells, \( G \) guard cells, and the test cell \( X_0 \). The test cell is surrounded symmetrically by its reference cells and guard cells. The total length of the shift register is given as

\[ L = N + G + 1. \]  

In a register with length \( L \), each datum streamed in input serially will need \( (L + 1)/2 \) clocks to be a cell test. The shift register output data are sent the value of \( N \) reference cells in parallel manner to the sorting module. The test cell from the shift register is delayed for some clocks until the threshold value of its value is computed. After sorting is done sequentially for these \( N \) cells, some of the sorted data are subjected to an automatic censoring mechanism. This mechanism will define the threshold value needed to decide the test cell as a target or not. The decision is made by comparing the threshold value with the test cell value.

The sorting operation plays a crucial role since it consumes long computation time, and constitutes a bottleneck in the field of real-time signal processing applications. In this work, the sorting circuit is based on a parallel bubble-sort that implemented over an array of compare-swap units as illustrated in Fig. 5 for six inputs.

The bubble sorting algorithm compares every two elements,
and then decides which one is the greater. The compare-swap circuit task is to compare the two inputs elements and to swap these inputs if the first element is greater than the second. This operation is repeated for each pair of adjacent elements till the end of the entire data array. The number of stages needed \( N \) elements is \( N \) stages. The advantage of this circuit is its small logic size and that the maximum value can be found with only \( N \) stages [24].

The B-ACOSD algorithm was represented as a flow chart as shown in Fig. 6. From the sorted reference cells, the number of interfering cells should be decided first by comparing \( T_{ck} \) and \( X(N - k) \) for \( k < (N - p) \). The number of interfering target will define the threshold value \( T_{ck} \) to be compared to the test cell \( X_k \).

![B-ACOSD Censoring Flow Chart](image)

The threshold \((T_{ck})\) resulted in the masking process was finally compared to the test cell \( X_k \). In this single comparator, if value of \( X_k \) is greater than the threshold \( T_{ck} \), then the CFAR-out is 1, else CFAR-out is 0. CFAR-out equal to 1 means the test cell is a target, else the test cell is not a target.

### V. SIMULATION AND REALIZATION

The overall SoC B-ACOSD CFAR SoC consists of NIOS-II processor, on-chip ROM, B-ACOSD CFAR Detector, Input/ROM interface, and Output/RAM interface have been implemented on Stratix II EP2S60F672C3N FPGA chip. On an Altera Development Board, the config.d FPGA chip is then connected to SSRAM, Flash ROM, and other I/O’s. Since the SoC design is modular, each module can be test individually. To verify the design, we performed four type simulations: MATLAB® simulation, MATLAB® fixed-point simulation, ModelSim® functional simulation, and Quartus II® timing simulation. The timing simulation diagram for CFAR system with \( N = 16 \) and \( p = 12 \) is shown in Fig. 7. The diagram showed that for each input datum streamed to the System on Chip, the censoring output will be reached after 34 clocks. Table III detailed the clock needed for each module.

![Timing Diagram](image)
An input should take \((L + 1)/2\) clocks to be positioned in the center of tapped delay line surrounding by \(N\) reference cells and \(G\) guard cells. Sorting module needs \(N\) clock to sort \(N\) reference cells. The clock needed by censoring module is independent to the number of reference cells. The pipeline architecture in censoring module only needs one clock for the parallel multiplier, one clock the parallel adder, one clock for the first comparator, one clock for masking, and one clock for the final comparator.

The FPGA implementation result for SoC with \(N = 16\) and \(g = 12\) shows that the processor can achieve a maximum operating frequency of 115.13 MHz. Because the clock needed for one CFAR detection is 34 clocks, this implies that the processing time to perform a single run is 0.29 \(\mu\)s. This processing time is below the real-time requirements 0.5 \(\mu\)s [23].

The B-ACOSD CFAR SoC also efficiently utilizes the FPGA hardware resources. The SoC only needs 9% combinational ALUT’s, 13% dedicated logic registers, 21% of memory elements, and 19% of DSP elements as shown in Table IV.

### VI. Conclusion

In this work, a novel hardware implementation of a CFAR detector for radar target detection is presented. The proposed architecture is implemented as a System on Chip integrating a Nios II Core processor, on-chip ROM, and B-ACOSD CFAR IP core using Avalon Switch Fabric. The architecture exploits parallelism and pipelining techniques to meet the real time requirement of the radar detection.

As a future work, the proposed architecture will be extended to direct implementation in the complete radar system. The other types of CFAR algorithm such as F-ACOSD and ACCA CFAR will be incorporated to the SoC to realize a generic CFAR detector architecture.

### REFERENCES


Imron Rosyadi was born in Banyumas, Indonesia, on September 24, 1979. He received the B.Sc. degree in electrical engineering from Gadjah Mada University, Indonesia, in 2003. Currently, he is a M.Sc. student in electrical engineering King Saud University, Saudi Arabia. Since March 2009, he was engaged in a research project with Prince Sultan Advanced Technology Research Institute (PSATRI) King Saud University, Saudi Arabia. Since 2004, he has been with Jenderal Soedirman University, Indonesia as an Assistant Lecturer. His main research interests are in the area of statistical signal processing, artificial intelligence application in signal processing, and biomedical engineering.

Ridha Ahmed Djemal received the Ph.D. degree in microelectronics (integrated circuit design) from Grenoble Institute of Technology, France, in 1996. Currently, he is an Associate Professor in the Department of Electrical Engineering at the King Saud University of Saudi Arabia, since September 2007, his research has been in the area of System Design, Image and video processing and encryption and Formal verification for System on Chip (SoC). The main theme of his research works are in the area of Design and Implementation of video application having a standard interfaces and Integrating the formal Verification in the Design flow. This work represents a very successful collaborative effort with the TELNET Firm in Tunisia and TansEda in France. The main theme of his research works are in the area of Design and Implementation of video application having a standard interfaces and Integrating the formal Verification in the Design flow. This work represents a very successful collaborative effort with the TELNET Firm in Tunisia and TansEda in France. Currently, he is also a member of team working on the Hardware Design Implementation of Image Encryption and Radar Recognition (CFAR) Algorithms in the Electrical Engineering Department of the King Saud University. Since October 2005, he is also an associate professor in Electronic Engineering Department, Applied Sciences and Technology Institute of Sousse (ISSAT Sousse), Tunisia.

Saleh Abdullah Alshebeili received the Ph.D degree in electrical engineering from University of Toronto, Ontario, Canada in 1992. Currently, he is a Professor of Electrical Engineering Department, King Saud University. He has more than 15 years of teaching and research experience in the area of communications and signal processing. Much of his research work has been devoted to the development of statistical estimation and modeling techniques. Such techniques play an important role in the analysis and interpretation of empirically gathered data that commonly encounter investigators in practice. He has published more than 49 papers in specialized international conferences and 37 papers in prestigious referred journals.