A Reconfigurable Smart Antenna Beamforming Architecture for Secure Localization Applications

Rathindra Nath Biswas
AJC Bose Polytechnic
Anurup Saha
Jadavpur University Faculty of Engineering and Technology
Swarup Kumar Mitra
MCKV Institute of Engineering Department of Electronics and Communication Engineering
Mrinal Kanti Naskar (mrinaletce@gmail.com)
Jadavpur University Faculty of Engineering and Technology https://orcid.org/0000-0001-9357-887X

Research Article

Keywords: Beamforming network, PSO (particle swarm optimization), CORDIC (coordinate rotation digital computer), FPGA (field programmable gate array), FSMD (finite state machine with datapath)

Posted Date: March 1st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1303539/v1

A Reconfigurable Smart Antenna Beamforming Architecture for Secure Localization Applications

Rathindra Nath Biswas [1], Anurup Saha [2], Swarup Kumar Mitra [3], Mrinal Kanti Naskar [2*]

[1] Department of ETCE, AJC Bose Polytechnic, Berachampa, 24 Parganas (N), India
[2] Department of ETCE, Jadavpur University, Kolkata, India
[3] Department of ECE, MCKV Institute of Engineering, Liluah, Howrah, India
E-mail: rathin02@gmail.com[1], sahaanurup24@gmail.com[2], swarup.subha@gmail.com[3], mrinaletce@gmail.com[2*]

Abstract

Beamforming of an adaptive array antenna often becomes effective for the precise localization of nodes in wireless sensor networks. In an erratic channel, a link with possibly a narrower beamwidth is always suitable for sending data to target nodes. It can also cancel interference from other nearby nodes while steering deep nulls and keeping the lower sidelobe level in the pattern. This paper presents a custom design for such a beamforming network on a reconfigurable architecture. It includes all the desired attributes of the beam on a reference template characterized by the parabolic function. It also considers a linear array with non-uniform element excitations. Using the Particle Swarm Optimization (PSO) algorithm, the weight vector containing the excitations of the array elements is iteratively optimized. It is then implemented adopting the concept of the Finite State Machine with Datapath (FSMD) model and using appropriate COordinate Rotation DIgital Computer (CORDIC) blocks. Its performance is tested on a dedicated Field Programmable Gate Array (FPGA) board with multiple fixed-point hardware-level simulations for beamforming accuracy and computational overhead. The results corroborate its competence comparable to existing systems.

Keywords: Beamforming network, PSO (particle swarm optimization), CORDIC (coordinate rotation digital computer), FPGA (field programmable gate array), FSMD (finite state machine with datapath).

1 Introduction

An adaptive array antenna system inherently has an attractive beamforming function (Godara 2004). It exploits spatial diversity to counter multipath interference by improving throughput and quality of service on an erratic channel (Liberti and Rappaport 1999). From this point of view, it is now the best choice to provide node localization services in wireless sensor networks (WSN) (Biswas et al. 2014). Such an attribute often becomes not only effective in relaying data packets with improved capacity but can also preserve their integrity and confidentiality by sending them over more stable and private links (Wang and Yang 2011). Thus, it can guarantee the lowest probability of errors in the distance/angle estimation process, which leads to the precise localization of the nodes in the network. However, malicious attacks altering transmission characteristics over a wireless medium can produce erroneous localization data (distance/angle) (Biswas et al. 2021). Conventional beamforming techniques that focus on *beam steering* (directs the main beam to the target node) or *null steering* (places deep nulls in interfering directions of neighboring nodes) depend solely on the accuracy of localization information. Therefore, they may not be as effective in hostile environments, and a new beamforming method is needed to provide a complete solution for secure data communications between nodes. However, to achieve this, an antenna pattern with a narrower *beamwidth* and lower *sidelobe level* (SLL) is always desirable. The beamforming network must also be reconfigurable to generate the beampattern coping with radio irregularities in a harsh environment (Hong et al. 2014). Again, a beamforming algorithm incorporating these criteria (i.e., beam steering, null steering, and lower SLL) can ensure an additional layer of protection in this situation. But this makes the pattern synthesis problem a multi-objective function, and a powerful heuristic search engine is often needed to solve it (Zardi et al. 2021). Optimizing one of the electrical parameters (e.g., element excitation, element position, or progressive phase shift) into a suitable array configuration can produce such a shaped beam pattern (Gies and Rahmat-Samii 2003). A digital platform is often preferable to implement the signal processing unit. It becomes simple to avoid the requirement of analog phase shifters in most cases (Jayaprakasam et al. 2017; Razavilar et al. 1999). In practice, several of these platforms, viz. Digital Signal Processor (DSP), General Purpose Microprocessor Unit (MPU), Application-Specific Integrated Circuits (ASICs) commonly exist in the literature. FPGA-based design technology is now best suited for a wireless sensor network infrastructure (Dikmese et al. 2011). Since an adaptive array antenna system typically operates at a higher frequency (on the order of a few GHz), it needs a high-speed parallel processor to compute the array weight vector for such applications (Nuteson et al. 2002). FPGA families can use their reconfigurable architectures to generate bundles of different shaped beams as needed from time to time in the data routing operation (Dikmese et al. 2010). In addition, it can meet resource constraints requiring relatively minimal hardware.

Nevertheless, only a few research has resulted in the realization of a beamforming architecture for the adaptive array antennas found so far, and they are mainly using conventional beamforming techniques (Choi and Shim 2000; Kucuk et al. 2009; Thiripurasundari et al. 2017).

Therefore, we have proposed the design and implementation of a novel PSO-based beamforming method in this work. Such a beamforming architecture can provide robust node localization by sending the necessary data over a link with a beam pattern having a prescribed beamwidth, SLL, and deep nulls steered in the interference directions. The PSO algorithm can determine the coefficient of excitation of the array elements to generate such a beam pattern according to the pre-specified attributes stipulated on a parabolic template. We then realized the architecture of this beamformer on the flexible platform of an FPGA chip. The results obtained with higher beamforming precision and lower hardware requirements validate its operational feasibility by tiny nodes with limited resources.

The organization of the rest of this paper is as follows. Section 2 summarizes the pioneering research work regarding the hardware implementation of beamforming in smart antennas on various digital platforms. Section 3 gives a brief overview of the PSO technique and the CORDIC algorithm. Section 4 and Section 5 respectively provide a detailed description of the design principles of the proposed PSO-based beamforming method and its implementation process on FPGA hardware. Section 6 discusses the simulation results, and Section 7 concludes the work by keeping track of its future extension.

2 Related works

Until now, some innovative research has only proposed a few suitable implementation methodologies of beamforming algorithms for smart antennas. Nevertheless, processing the digital signal with reduced computational complexity remains challenging. In addition, they overlooked incorporating the necessary and sufficient conditions to suppress noise/interference during signal transmission on an erratic channel. This literature review focuses on some new work that has explored the design aspects of adaptive array antenna beamforming on an appropriate platform over the past two decades.

As a beamforming algorithm, Thiripurasundari et al. 2017 proposed a modified CORDIC based on the QR Decomposition-Recursive Least-Squares (QRD-RLS) method named Mixed Scaling Rotation COordinate Rotation DIgital Computer (MSR-CORDIC). It is developed using the Verilog Hardware Description Language (HDL) and implemented on the Xilinx FPGA board. It shows a remarkable improvement over the conventional Space Code Correlator (SCC) algorithm for device utilization and execution time. Kim et al. 2004 also proposed an FPGA-based implementation of an adaptive array antenna system. It is suitable for uplink use in macro-cellular base stations. It contains a Dolph-Chebyshev beam and optional cancel beam, both stored in Read-Only Memory (ROM) for beam steering and null steering purposes. It ensures less overhead in the computation of the weight by using a simple convolution operation between these two pre-computed beams. Nuteson et al. 2002 showed a real-time implementation of an eight-channel smart antenna receiver using a back-end FPGA for calibration and signal processing. It can perform digital beamforming at the carrier frequency of the RF signal using bandpass/IF sampling for a high-speed analog-to-digital converter. Such combined technology conforms to the system complexity, speed, and precision by reducing the RF front-end electronics. Llamocca and Aloi 2018 presented a self-reconfigurable embedded system for a switched beam smart antenna implemented and tested on a Programmable System-on-Chip (PSoC). It generates a set of hardware configurations based on the desired beam patterns, varying the numerical formats into fully customized fixed-point arithmetic hardware. They are selected at runtime by a dynamic software-based manager. It can handle arithmetic overflows and user-specified resource constraints. Dikmese et al. 2010, 2011 also presented a comparative study to verify the feasibility of implementing a beamforming architecture on DSP and FPGA. They envisioned floating-point implementations for some CDMA compatible beamforming algorithms, namely Least Mean Square (LMS), Constant Modulus (CM), and Space Code Correlator (SCC). They also observed that the FPGA could compute weight vectors about 500 times faster than the DSP. It also had better utilization of resources. Hasanikhah et al. 2018, 2019 presented an FPGA implementation of a high-speed Space-Time Adaptive Processing (STAP) architecture. It uses QR Decomposition based on the Modified Gram–Schmidt (QRD-MGS) algorithm to calculate floating-point weight vectors. It shows that reducing vector size can minimize resource usage and power consumption while computation time increases slightly. Kucuk et al. 2009 proposed a DSP-based SCC beamformer that can estimate weight vectors with less time. It also offers comparable performance under the same Signal-to-Interference plus-Noise Ratio (SINR) level for the LMS and CM algorithms. Likewise, Dick et al. 2006 presented an FPGA realization of a flexible QRD-based beamforming method. It uses a System Generator model developed with MATLAB Simulink programming in its FPGA design flow. It enables runtime sharing of system parameters between the beamforming module and a host processor. In addition, Jarrah and Jamali 2013 suggested an FPGA architecture of the Direct Data Domain (D^3) algorithm for real-time applications.

However, all these hardware implementations of beamforming methods demonstrate the applicability of adaptive array antennas for 3G cellular/radar systems. But, they are not helpful in localization operations for WSNs deployed in a hostile radio environment. Because the precision of the distance/angle estimation always depends on the propagation characteristics of a wireless channel and erroneous information can extend the coverage to attackers/interferers. In this context, the proposed beamforming architecture is very robust to maintain data security,

having the flexibility to control beamwidth, nulls, and SLL values in the pattern. It also includes benefits like less hardware, less power, and minimal computational overheads needed for nodes with limited resources.

3 Design preliminaries

In WSNs, nodes remain vulnerable, and data packets become fragile while relayed over radio links. Therefore, attacks on localization systems can occur in two forms: (i) capturing data from a few benevolent nodes and (ii) modifying the radio environment (e.g., inserting obstacles or noise/jamming signals) in specific areas. In either case, the distance/angle estimation process gets disrupted. It ultimately renders the information erroneous. Adaptive beamforming methods can provide the fundamental solution to counter such physical layer attacks. However, they should have as simple formulas and flexible architecture as possible for their hardware implementation. This work presents the design and implementation of a beamforming method based on the PSO and CORDIC algorithms. Such a beamforming architecture can prevent attackers/interferers from injecting incorrect localization information, producing a beam pattern with lower SLL and narrower beamwidth.

3.1 Particle swarm optimization

Particle Swarm Optimization (PSO) is an evolutionary technique based on a socio-behavioral model of swarm intelligence. It has a simple structure and a higher degree of convergence. It is, therefore, now a better choice to optimize the objective of multidimensional, discontinuous, and complex problems (Robinson and Rahmat-Samii 2004).

In this method, each 'particle' refers to an individual agent within the swarm. And the position of each particle represents a probable solution to the problem. It usually formulates a problem-specific fitness function, $f\{X\}$ (also called the cost/error/objective function) that defines the level of precision of such a solution. Therefore, each particle should find its position with a higher fitness value on the search space. However, without any prior knowledge, it initializes the particles with a random location $[X]_{M\times N}$ and a random velocity $[V]_{M\times N}$ for their motions (M and N are the numbers of particles and the dimensions of the problem, respectively). It uses the social interaction between the particles during the search process. Thus, the particles must update their velocity and position based on their personal best location $[X_{pbest}]_{M\times N}$ and the global best point $[X_{gbest}]_{1\times N}$ iteratively, using equation (1) and equation (2) respectively as follows.

$$V = wV + c_1 rand()\left\{X_{pbest} - X\right\} + c_2 rand()\left\{X_{gbest} - X\right\} \quad (1)$$

$$X = X + V \quad (2)$$

Here, $rand()$ is a uniform random variable in the range $\{0, 1\}$ to introduce some natural randomness. In addition, the parameters c_1 and c_2 respectively specify the relative weights of X_{pbest} and X_{gbest}. It often uses an inertia weight (w) to improve its convergence performance by linearly reducing the velocity of the particles within a limit $\{0.9, 0.4\}$. It gradually emphasizes local exploitation rather than global exploration in the search process. It also applies the boundary conditions to improve and confine the movement of the particles in the desired domain of interest $\{X_{min}, X_{max}\}$. However, the process continues until a predefined termination criterion (t_{max}) is attainable. Algorithm 1 explains the standard PSO as below. The *input* parameters specify the set of data needed to start the process. Likewise, the desired data set obtained at the end of program execution represents *output* parameters.

Algorithm 1: Pseudo-code for Particle Swarm Optimization

Input: Number of particles (M), Number of dimensions (N), Position vectors for particles (X), Velocity vectors for particles (V), Dynamic ranges for position vectors: $\{X_{min}, X_{max}\}$, Dynamic ranges for velocity vectors: $\{V_{min}, V_{max}\}$, Inertia weight (w) and Maximum permissible search time (t_{max})
Output: Particle with the optimum position vector: X_{gbest}

1: **Initialize**: $M, N, V_{min} = X_{min}, V_{max} = X_{max}, f\{X_{gbest}\} \leftarrow \infty, w = 0.9, w_0 = \frac{0.5}{t_{max}}, m, n, t = 1, t_{max}$
2: **for** all particles m **do**
3: $f\{X_{pbest}\} \leftarrow \infty$
4: **for** all dimensions n **do**
5: $X \leftarrow rand() * \{X_{min}, X_{max}\}$; $V \leftarrow rand() * \{V_{min}, V_{max}\}$
6: **end for**
7: **end for**
8: **while** ($t \leq t_{max}$) **do**
9: **for** all particles m **do**
10: **if** ($f\{X\} < f\{X_{pbest}\}$) **then**
11: $X_{pbest} \leftarrow X$; $f\{X_{pbest}\} \leftarrow f\{X\}$

12: **end if**
13: **if** $(f\{X\} < f\{X_{gbest}\})$ **then**
14: $X_{gbest} \leftarrow X;\ f\{X_{gbest}\} \leftarrow f\{X\}$
15: **end if**
16: **for** all dimensions n **do**
17: $V \leftarrow w * V + c_1 * rand() * \{X_{pbest} - X\} + c_2 * rand() * \{X_{gbest} - X\};\ X \leftarrow X + V$
18: **if** $(X_{max} < X < X_{min})$ **then**
19: $X \leftarrow rand() * X_{max}$
20: **end if**
21: **if** $(V_{max} < V < V_{min})$ **then**
22: $V \leftarrow rand() * V_{max}$
23: **end if**
24: **end for**
25: **end for**
26: $w \leftarrow w - w_0; t \leftarrow t + 1$
27: **end while**
28: **return** X_{gbest}

3.2 Coordinate rotation digital computer

The COordinate Rotation DIgital Computer (CORDIC) is an efficient computational technique based on a rotation vector in 2-D coordinates (x, y). It can compute many mathematical functions such as real-valued/complex-valued, trigonometric, and hyperbolic with binary arithmetic. It can also use identical hardware for various signal processing applications by modifying only the initial conditions (Meher et al. 2009). Thus, it now becomes an ideal solution for flexible design technology based on FPGA.

In this method, the rotation vector rotates around the desired angle (α) in a specific coordinate system. It then decomposes this angle into a set of elementary angles (α_i) obtained by pseudo-micro-rotations. So, it computes this angle as the sum of a few constituent angles iteratively, using equation (3) as follows.

$$\alpha = \sum_{i=0}^{p-1} \sigma_i \alpha_i \tag{3}$$

Here, σ denotes the direction of the micro-rotations. It turns into a series of *shift* and *addition* operations by simply assigning the constituent angles to the power of 2 (e.g., choose $\alpha_i = tan^{-1}(2^{-i}), 2^{-i}$ or $tanh^{-1}(2^{-i})$ respectively for a circular, linear, or hyperbolic coordinate system). Equations (4)-(6) generally summarize this computational process in a circular coordinate system.

$$x_{i+1} = x_i + \sigma_i 2^{-i} y_i \tag{4}$$

$$y_{i+1} = y_i - \sigma_i 2^{-i} x_i \tag{5}$$

$$z_{i+1} = z_i - \sigma_i \tan^{-1}\left(2^{-i}\right) \tag{6}$$

Here, z is an accumulator for storing the angle of the micro-rotations. However, it avoids the effect of the scale factor always converging to 1.6467605 in a circular coordinate system by initializing the rotation vector to $x_0 = 0.61, y_0 = 0$. The iterative process continues until a predefined termination criterion (i_{max}) is achievable. Algorithm 2 explains the standard CORDIC in a circular coordinate system to compute the sine/cosine function below. It uses coordinates, rotation angles, and iterations to obtain an acceptable value for the sine and cosine functions. Therefore, they are considered the *input* and *output* variables, respectively.

Algorithm 2: Pseudo-code for realizing sine/cosine function in CORDIC

Input: X - coordinate of rotation vector (x), Y - coordinate of rotation vector (y), Accumulator for storing angle (z), Angle of rotation (α) and Maximum permissible iterations (i_{max})
Output: Sine function $\{sin\alpha\}: -y$; Cosine function $\{cos\alpha\}: x$

1: **Initialize**: $x = 0.61,\ y = 0,\ z = \alpha,\ i = 0,\ i_{max}$
2: **for** all iterations i **do**
3: **if** $(z \geq 0)$ **then**
4: $\sigma \leftarrow 1$
5: **else**
6: $\sigma \leftarrow -1$
7: **end if**

8: $x \leftarrow x + \sigma * 2^{-i} * y;\ y \leftarrow y - \sigma * 2^{-i} * x;\ z \leftarrow z - \sigma * tan^{-1}(2^{-i})$
9: **end for**
10: **return** $-y\ ;\ x$

---

4 Implementation methodology

This work focuses on the hardware implementation of beamforming in adaptive array antennas. This concept is an extended part of our previous work (Biswas et al. 2021). Thus, it assumes the same network architecture and protocols that supposed the smart antennas integration with a few mobile anchors. The anchors estimate *angle of arrival* (AoA) information from signals (s_j) received from nearby target nodes and schedule to convey them serially over point-to-point wireless links to localize corresponding nodes. From this point of view, the antenna pattern should be highly directional with a narrower beamwidth and lower sidelobe level for secure data transmission. It can be the primary beamforming requirement, but such a design may not always provide security against malicious attacks. Therefore, the flexibility of placing deep nulls in the direction of interference is crucial to extend the second layer of protection. However, this strategy makes the design problem very complicated. There is always a chance to deteriorate tradeoff issues. Again, the beamforming architecture must be reconfigurable, generating a new pattern as needed from time to time on the same antenna configuration.

In this work, the realization of a reconfigurable beamforming architecture producing the desired pattern with deep nulls steered towards interference is easily achieved using the CORDIC method and the PSO algorithm. For each target node, it operates in three successive phases. These are the formation of an appropriate reference template based on AoA information, construction of an array steering vector using the CORDIC block, and generation of an optimal pattern using the PSO module that only optimizes the excitation coefficients of the antenna elements in a linear array. Figure 1 illustrates such an adaptive array antenna beamforming architecture. We have described the proposed implementation methodology in detail as below.

4.1 Reference template

In WSNs, it is always possible to keep all adjacent nodes out of the radio links by placing deep nulls in their directions. Otherwise, any malicious/compromised node lying nearby can eavesdrop on access to localization data. Likewise, maintaining a lower SLL and narrower beamwidth would also be effective, while precise estimation of localization data remains challenging in a compromised radio environment. It can enhance gain and directivity to avoid packet drop (data loss) problems. A reference template should include these desired pattern attributes. Taking all neighboring nodes into account, each time a link is established along with one, assuming the others are the interferers. Thus, the corresponding reference template contains its AoA data as the desired direction (θ_d) for the beam orientation. And the rest of the AoA data is selected as the directions (θ_n) for the null positioning. Algorithm 3 explains the development of a reference template based on the AoA information of the target nodes. Since the generation of a reference template requires several data of the received signal and specifications for the design of the antenna pattern, it uses them as *input* variables. Likewise, it considers an antenna pattern with the desired attributes as *output* variables.

In this work, a parabolic function defines the reference template (AF_d), and we can express it using equation (7) as follows.

$$AF_d = \begin{cases} 1 - \left\{ \dfrac{2(\theta - \theta_d)}{FNBW} \right\}^2 & \text{if} \quad |\theta - \theta_d| < \dfrac{FNBW}{2} \\ \eta & \text{if} \quad \theta = \theta_n \\ SLL & \text{otherwise} \end{cases} \tag{7}$$

Here, the beamwidth between the first nulls ($FNBW$), the level of the sidelobes (SLL), and the depth of the nulls (η) are three user-defined variables to synthesize the desired pattern (Biswas et al. 2021).

Algorithm 3: Pseudo-code for generation of reference template

---

Input: AoA of the received signals (φ), Number of the received signals (D), Direction of the desired signal (θ_d), Directions of the interfering signals (θ_n), Dynamic ranges of the scanning angle (θ): $\{-90^0, 90^0\}$, Beamwidth between the first nulls ($FNBW$), Sidelobe level (SLL) and Depth of nulls (η)
Output: Beam pattern with the desired attributes (AF_d)

1: **Initialize**: $D,\ \eta,\ SLL,\ FNBW,\ j,k\ =\ 1, \theta\ =\ -90^0$

2: **for** all received signals j **do**
3: $\theta_d \leftarrow \varphi\,[1]$
4: **for** all scanning angles θ **do**
5: **if** $\left(|\theta - \theta_d| \leq \frac{FNBW}{2}\right)$ **then**
6: $AF_d \leftarrow 1 - \left\{2 * \frac{(\theta - \theta_d)}{FNBW}\right\}^2$
7: **else**
8: $AF_d \leftarrow SLL$
9: **end if**
10: **for** all interfering signals k **do**
11: **if** $(k \leq D - 1)$ **then**
12: $\theta_n \leftarrow \varphi\,[k + 1]$
13: **end if**
14: **if** $(\theta = \theta_n)$ **then**
15: $AF_d \leftarrow \eta$
16: **end if**
17: **end for**
18: **end for**
19: **return** AF_d
20: **for** all interfering signals k **do**
21: **if** $(k \leq D - 1)$ **then**
22: $\varphi\,[k] \leftarrow \varphi\,[k + 1]$
23: **end if**
24: **end for**
25: $\varphi\,[D] \leftarrow \theta_d$
26: **end for**

4.2 Array steering vector

The geometry of a linear and symmetrical array with uniform spacing (Δ) between the elements (N), as in Figure 1, characterizes the beamforming function (AF_p) defined by equation (8) as follows (Gross 2005).

$$AF_p = \sum_{n=1}^{N} W_n \cos\left[\left(n - \frac{1}{2}\right)\psi\right] \tag{8}$$

Here, $\psi = \beta\Delta(sin\theta - sin\theta_d)$ is the progressive phase shift component in a phased array that produces the pattern in the direction θ_d. Also, β is the phase of the antenna. W_n denotes the excitation coefficient of the n-th element. However, in matrix form, equation (8) can also be represented using equation (9) as follows.

$$AF_p = W^T a(\theta) \tag{9}$$

Here, $[W]_{N\times 1}$ and $[a(\theta)]_{N\times 1}$ are respectively the weight vector and the steering vector of the adaptive array antenna. The steering vector consists of sine and cosine functions. Therefore, a suitable CORDIC block can evaluate it. Algorithm 4 explains the creation of an array steering vector for this beamforming system. Since it involves CORDIC and antenna design parameters to generate an array steering vector, they are considered the *input* and *output* parameters.

Algorithm 4: Pseudo-code for construction of array steering vector

---

Input: Direction of the desired signal (θ_d), Maximum permissible iterations (i_{max}), Number of symmetrical array elements (N), Spacing between elements (Δ), Phase shift (β), Dynamic ranges of the scanning angle (θ): $\{-90^0, 90^0\}$
Output: Array steering vector (a)

1: **Initialize**: $\beta = \frac{2\pi}{\lambda}$, $\Delta = \frac{\lambda}{2}$, $N, i = 0, n = 1$, $\theta = -90^0, i_{max}$
2: $z \leftarrow \theta_d$; $Flag \leftarrow 0$; $x \leftarrow 0.61$; $y \leftarrow 0$
3: **for** all permissible iterations i **do**
4: **if** $(z \geq 0)$ **then**
5: $\sigma \leftarrow 1$
6: **else**
7: $\sigma \leftarrow -1$

8: **end if**
9: $x \leftarrow x + \sigma * 2^{-i} * y;\ y \leftarrow y - \sigma * 2^{-i} * x;\ z \leftarrow z - \sigma * tan^{-1}(2^{-i})$
10: **end for**
11: $sin\theta_d = -y;\ \psi_0 \leftarrow \beta * \Delta * sin\theta_d;\ Flag \leftarrow 1$
12: **for** all scanning angles θ **do**
13: $z \leftarrow \theta;\ sin\theta = -y;\ \psi_1 = \beta * \Delta * sin\theta;\ \psi \leftarrow \psi_1 - \psi_0;\ Flag \leftarrow 2$
14: **for** all vector elements n **do**
15: $z \leftarrow \left(n - \frac{1}{2}\right) * \psi;\ \cos\left[\left(n - \frac{1}{2}\right) * \psi\right] \leftarrow x;\ a \leftarrow \cos\left[\left(n - \frac{1}{2}\right) * \psi\right]$
16: **end for**
17: $Flag \leftarrow 1$
18: **end for**
19: **return** a

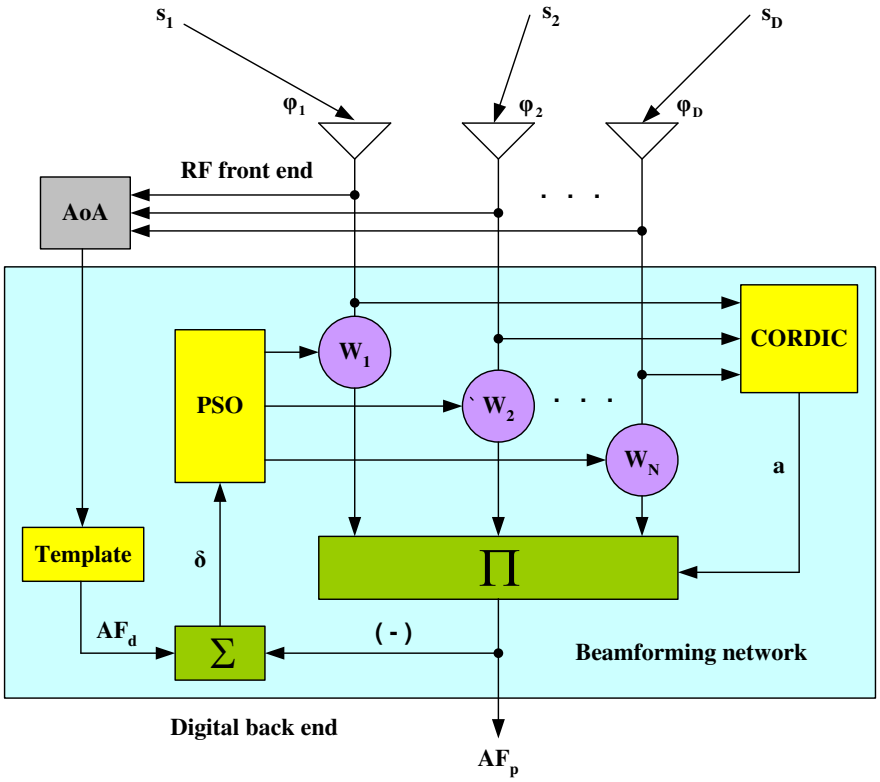

Figure 1: Smart antenna beamforming system

4.3 Optimal pattern

The beamforming system uses a well-known stochastic process called PSO for pattern optimization. It iteratively updates the excitation coefficients of the array elements (N). The optimization problem gets reduced to half of the actual/physical array dimensions ($2N$), choosing the symmetrical structure. Moreover, the boundary limit is set to a value in the range $\{0, 1\}$ for each vector element, simply giving a dynamic range of 1 in the optimization process. It formulates the fitness function (F) as the sum of all deviations (δ) obtained among the desired (AF_d) and the produced pattern (AF_p). It evaluates fitness value for the scanning angles ranging from -90^0 to 90^0 with an interval of 1^0 and expressed by using equation (10) as follows.

$$F = \min\left\{\sum_{\forall\theta}\delta(\theta)\right\} \tag{10}$$

Each deviation is usually assigned to a penalty value of 1 whenever it violates the design goal ($AF_p \leq AF_d$). It is also defined by equation (11) as follows.

$$\delta = \begin{cases} 1 & \text{if } AF_d - AF_p < 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Thus, the process always converges to a better design (having minimum penalty). It keeps a record of the best solution ever achieved. Algorithm 5 explains the optimal pattern generation of this beamforming system. Here, the PSO produces an optimal pattern using the template as a reference. Accordingly, it considers them as the *input* and *output* variables.

Algorithm 5: Pseudo-code for generation of the optimal pattern

---

Input: Array weight vector (W), Weight update vector (V), Number of vector set (M), Number of vector elements (N), Dynamic ranges of the vector elements: $\{0, 1\}$, Array steering vector (a), Dynamic ranges of the scanning angle: $\{-90^0, 90^0\}$, Desired beam pattern (AF_d), Inertia weight (γ) and Maximum permissible search time (t_{max})
Output: Beam pattern (AF_p) with the optimum weight vector (W_{gbest})

1: **Initialize**: $m,\ n,\ t\ =\ 1, \theta\ =\ -90^0, \gamma\ =\ 0.9,\ \gamma_0\ =\frac{0.5}{t_{max}}, f\{W_{gbest}\} =\ \infty, M, N, t_{max}$
2: **for** all vector set m **do**
3: $f\{W_{pbest}\}\ \leftarrow\ \infty$
4: **for** all vector elements n **do**
5: $W\ \leftarrow\ rand();\ V\ \leftarrow\ rand()$
6: **end for**
7: **end for**
8: **while** $(t\ \leq\ t_{max})$ **do**
9: **for** all vector set m **do**
10: $F\ \leftarrow\ 0$
11: **for** all scanning angles θ **do**
12: $AF_p\ \leftarrow\ 0$
13: **for** all vector elements n **do**
14: $AF_p\ \leftarrow\ AF_p\ +\ W\ *\ a$
15: **end for**
16: **if** $(AF_p\ >\ AF_d)$ **then**
17: $\delta\ \leftarrow\ 1;\ F\ \leftarrow\ F\ +\ \delta$
18: **end if**
19: **end for**
20: **if** $(F\ <\ f\{W_{pbest}\})$ **then**
21: $W_{pbest}\ \leftarrow\ W\ ;\ f\{W_{pbest}\}\ \leftarrow\ F$
22: **end if**
23: **if** $(F\ <\ f\{W_{gbest}\})$ **then**
24: $W_{gbest}\ \leftarrow\ W\ ;\ f\{W_{gbest}\}\ \leftarrow\ F$
25: **end if**
26: **for** all vector elements n **do**
27: $V\ \leftarrow\ \gamma * V\ +\ c_1 * rand() * \{W_{pbest}\ -\ W\} +\ c_2 * rand() * \{W_{gbest}\ -\ W\}$
28: **if** $(1\ <\ V\ <\ 0)$ **then**
29: $V\ \leftarrow\ rand()$
30: **end if**
31: $W\ \leftarrow\ W\ +\ V$
32: **if** $(1\ <\ W\ <\ 0)$ **then**
33: $W\ \leftarrow\ rand()$
34: **end if**
35: **end for**
36: **end for**
37: $\gamma\ \leftarrow\ \gamma\ -\ \gamma_0; t\ \leftarrow\ t\ +\ 1$
38: **end while**
39: **for** all scanning angles θ **do**
40: $AF_p\ \leftarrow\ 0$
41: **for** all vector elements n **do**
42: $AF_p\ \leftarrow\ AF_p\ +\ W_{gbest}\ *\ a$
43: **end for**
44: **end for**
45: **return** AF_p

---

5 Hardware implementation

We have implemented the beamforming system on an FPGA device. The FPGA enables parallel processing in practice. Unlike conventional processors, it always develops a reconfigurable and distributed arithmetic structure using the internal logic blocks. It avoids the instruction fetch and data load/store bottlenecks on execution. Thus, it is now the most popular and flexible design solution in the digital platform (Chu 2008).

The digital beamforming architecture consists of two functional blocks: (i) control unit and (ii) data unit. Therefore, it is a digital system, and we have developed it based on the Finite State Machine with Datapath (FSMD) design methodology. The Finite State Machine (FSM) acts as a controller, and the datapath performs digital signal (data) processing operations in this system. It translates the sequence of operations (statements) of the beamforming algorithm into *states* and represents them by state diagrams. A state diagram includes several *states* and *arcs* that depict operational flow in a series of *commands* and *actions* specified by arithmetic expressions. Such expressions involve external inputs, outputs, and other variables used in the algorithm.

In this work, we have realized the functionality of the beamforming processor with three separate modules corresponding to the reference template, CORDIC, and PSO, as in Figure 1. According to AoA information, the template module first produces a suitable pattern, AF_d, for using it as a reference. Then, considering the antenna design parameters ($N, \beta, \Delta,$ and θ_d), the CORDIC block forms the array steering vector, a. Finally, comparing each candidate design with the reference, the PSO unit evaluates the fitness function, F, and generates the optimal pattern, AF_p with the optimized weight vector, W_{gbest} after the prescribed iterations, t_{max}. The following sections provide a detailed description of the FSMD design method and the beamforming architecture.

5.1 Finite state machine with datapath

The datapath is a collection of registers (store the interim data), data processing units (compute arithmetic expressions), and routing networks (transfer data through buses). It performs a specific operation according to the control signal of the FSM and produces the internal status signal ($flag$). The FSM has a state register, input and output logic for a state transition. It utilizes the external commands and the status signal of the datapath as the input. Thus, it produces the control signals to specify the datapath operation. Also, it can generate an external status signal ($reset/halt$) that indicates the status of the system operation.

We have used three separate FSMD here to describe the behavioral model of the proposed beamforming algorithm. Figure 2 illustrates the state diagram to generate a reference template based on AoA information. It selects the first data, $\varphi[1]$ in the register containing AoA information as the direction of the main-lobe orientation (θ_d), and others as the directions of placing deep nulls (θ_n). It shifts the register, φ, left by one data position in each case of a new template generation. The process repeats until the last data comes in the first position. Therefore, the number of reference templates produced equals the number of data (D) available in the register at a particular time. It also defines the template by a specific SLL, null depth (η), and $FNBW$. A parabolic function characterizes the main lobe in this work.

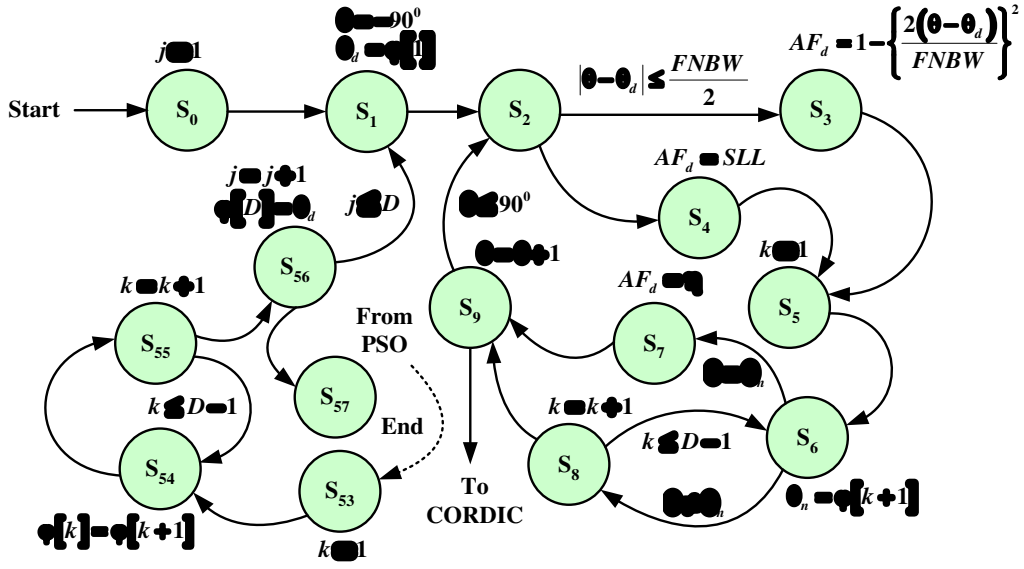

Figure 2: State diagram of reference template module

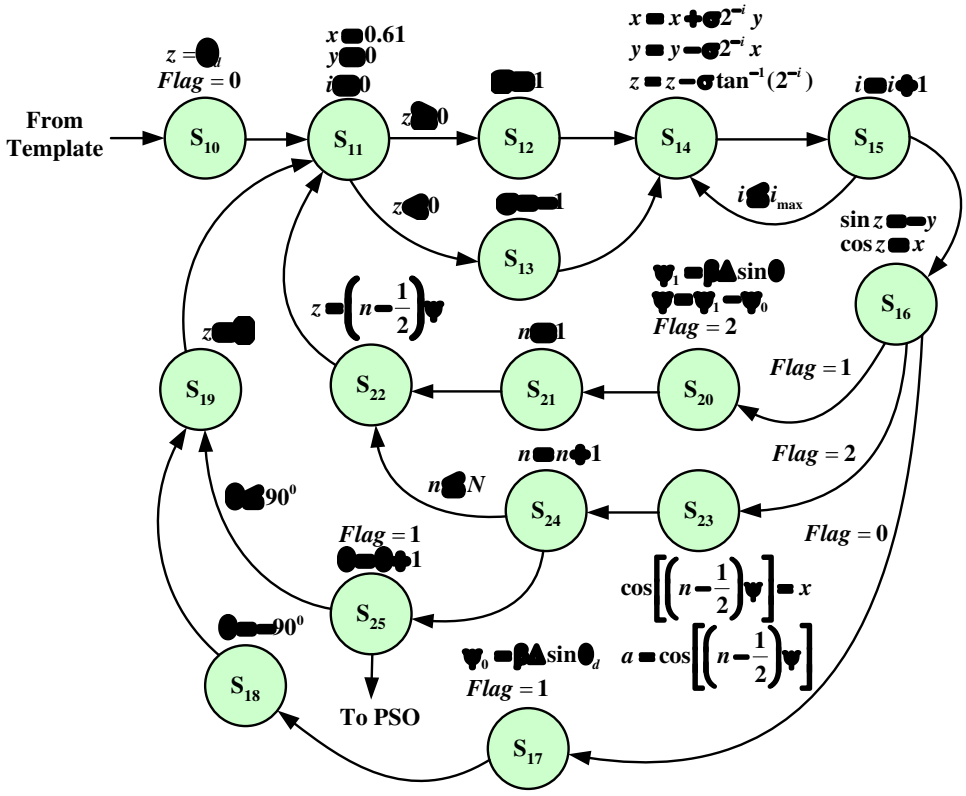

Figure 3: State diagram of CORDIC module

Figure 3 shows the state diagram to create an array steering vector. It uses three status signals ($flag$) to control the computation of sine and cosine functions on the CORDIC block. Thus, it computes the trigonometric sine functions, $sin\theta_d$, and $sin\theta$, respectively setting the $Flag$ register to a value of 0 and 1. It further computes the array steering vector described by the trigonometric cosine function, $\cos\left[\left(n-\frac{1}{2}\right)\psi\right]$, using array design parameters (β, Δ, and N) when the $Flag$ register content becomes 2.

Likewise, Figure 4 shows the state diagram to produce the optimal pattern on the PSO unit. It initializes the fitness values to infinity for both the personal and the global best position. Usually, it means a high value obtained with the worst design condition, yielding the deviation, δ, to 1 in all cases ($\infty \geq 181$). Also, it primarily defines the position and the velocity vectors by the variable, $rand()$, which represents a random distribution over the boundary limit of $\{0, 1\}$.

Table 1 also describes the functionalities of the beamforming processor in each state.

5.2 Proposed architecture

The FSMD often implements a digital system as the Register Transfer Level (RTL) circuit design (Vahid 2011). The RTL models a synchronous digital circuit that describes the flow of signals (data) between registers and performs the logical operations on such signals/data. It is very convenient to represent the circuit at a register level by hardware description language (HDL). Also, it is possibly easier to derive the gate-level representation and actual wire connectivity of the circuit from it. For the algorithm, an HDL (Verilog/VHDL) declares the variables with the registers and describes the statements (if-then-else, arithmetic expressions) with the combinational logic circuits. The system architecture is developed here as a combination of three modules: Template, CORDIC, and PSO. It uses an embedded clock to synchronize the operation of each module. Figure 5, Figure 6, and Figure 7 respectively illustrate the internal circuits of each module in RTL.

In RTL, a Boolean expression (binary arithmetic) containing variables, constants, and register contents as its input defines the data processing operations. A combinational logic circuit can implement it. It always operates on a clock-by-clock basis and makes a state transition of the state register. Thus, a data transfer occurs to a specific sequential logic circuit/register at the next rising edge of the clock. The output of the state register is used to control the function of the multiplexer and selects the desired data in a particular operation. The closed feedback loops preserve the current contents of the sequential logic circuits (registers) when they remain unchanged. Likewise, it realizes a data routing operation with a multiplexing circuit (multiplexer, tri-state buffers) that properly transfers the desired data to a specific register in each state. It implements any conditional data assignment statement (if-then-else) with a 2×1 multiplexer whose select input contains a Boolean (relational/logical) expression. When the

condition is satisfied, it passes the data in the *true* (logic 1) input port to the output port. Otherwise, it transfers the data of the *false* (logic 0) input port. Also, it executes any loop statement (for/while) in the algorithm via three sequential data operations: preliminary data assignment, conditional data assignment, and incremented/decremented data assignment. Thus, it realizes a loop counter composed of three-level hardware architectures. These are (i) a register (corresponding to the loop count variable) that loads with the initial data and updates through it in each count. (ii) A 2×1 multiplexer (having a relational expression as its select input) to check the condition for an update or keep the content of the register unchanged. And (iii) a multiplexer (containing the output of the state register as its select input) that transfers data to the sequential logic circuit/register in a suitable clock. While it uses the same loop counters in different modules, it reinitializes them at the separate clocks. An FPGA chip can store all constants and the parametric values to the user-defined variables into the Block Random Access Memory (BRAM). It also realizes the random function, $rand()$ with a 16-bit Linear Feedback Shift Register (LFSR). It executes all other data storage/transfer operations with the 12-bit registers in this work.

Thus, the template module contains four data registers $(AF_d, \varphi, \theta_d, \theta_n)$ and three number of loop counters (j, k, θ). Also, the CORDIC module includes eight data registers $(Flag, x, y, z, a, \sigma, \psi_0, \psi_1)$ and three number of loop counters (i, n, θ). The PSO module involves four number of loop counters (t, m, n, θ) and eleven data registers $(AF_p, \gamma, W, V, F, \delta, rand(), f\{W_{pbest}\}, f\{W_{gbest}\}, W_{pbest}, W_{gbest})$.

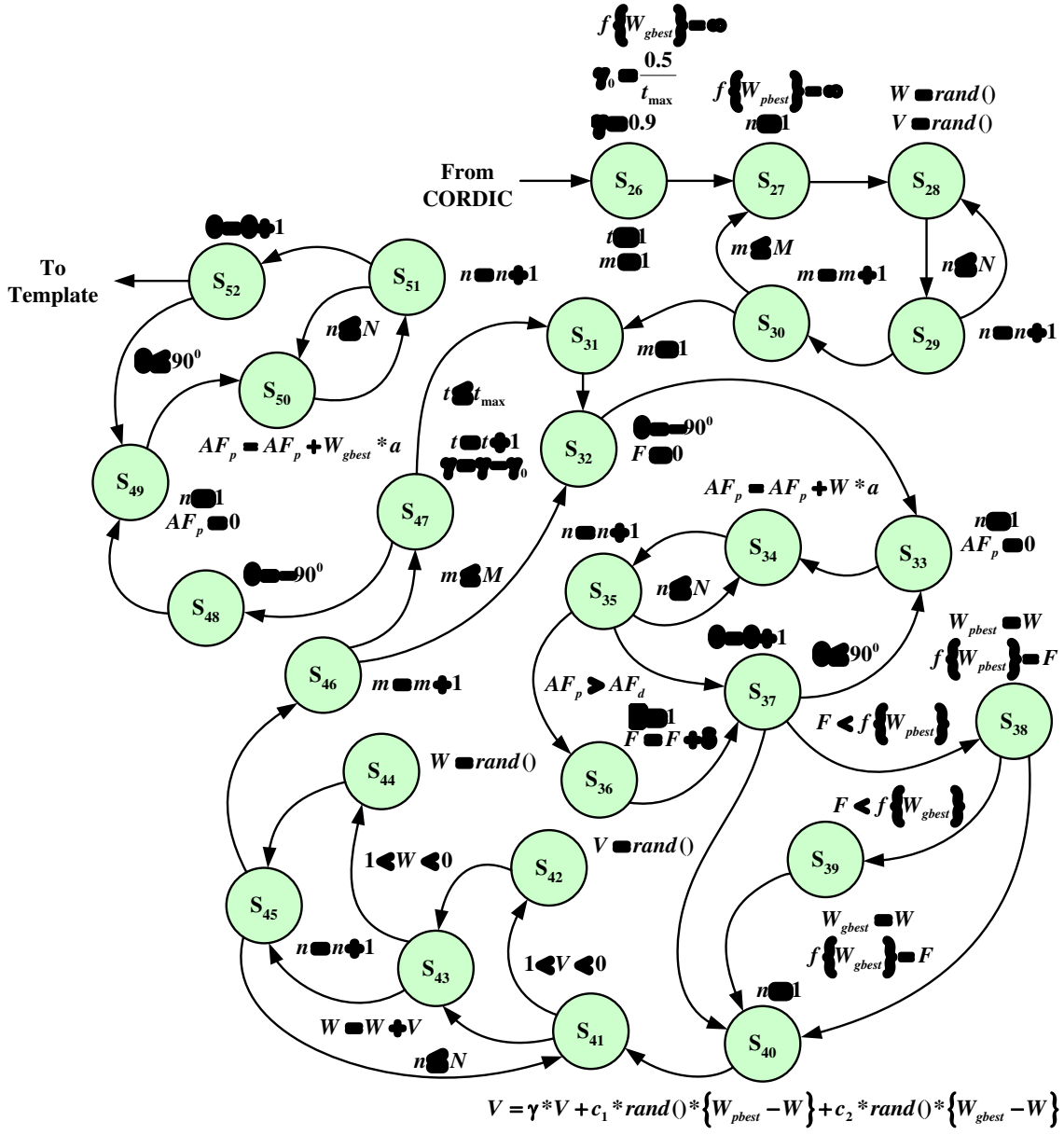

Figure 4: State diagram of the PSO module

Table 1: Beamforming processor behaviors in FSM

State	Command	Action	State	Command	Action
S_0	Reset/Start	Activate counter j	S_{29}	Clock	Update counter n
S_1	Clock, $j \leq D$	Activate counter θ; Move first content of register φ to register θ_d	S_{30}	Clock, $n > N$	Update counter m
S_2	Clock, $\theta \leq 90^0$	Subtract register θ_d from register θ; Compare difference with $\frac{FNBW}{2}$	S_{31}	Clock, $m > M$, $t \leq t_{max}$	Activate counter m
S_3	Clock, $\lvert\theta - \theta_d\rvert \leq \frac{FNBW}{2}$	Set register AF_d to $1 - \left\{\frac{2(\theta - \theta_d)}{FNBW}\right\}^2$	S_{32}	Clock, $m \leq M$	Set register F to 0; Activate counter θ
S_4	Clock, $\lvert\theta - \theta_d\rvert > \frac{FNBW}{2}$	Set register AF_d to SLL	S_{33}	Clock, $\theta \leq 90^0$	Set register AF_p to 0; Activate counter n
S_5	Clock	Activate counter k	S_{34}	Clock, $n \leq N$	Update register AF_p
S_6	Clock, $k \leq D - 1$	Move rest contents of register φ to register θ_n one by one	S_{35}	Clock	Update counter n
S_7	Clock, $\theta = \theta_n$	Set register AF_d to η	S_{36}	Clock, $n > N$, $AF_p > AF_d$	Set register δ to 1; Update register F
S_8	Clock, $\theta \neq \theta_n$	Update counter k	S_{37}	Clock, $AF_p \leq AF_d$	Update counter θ
S_9	Clock, $k > D - 1$	Update counter θ	S_{38}	Clock, $\theta > 90^0$, $F < f\{W_{pbest}\}$	Move register W to register W_{pbest} ; Update register $f\{W_{pbest}\}$
S_{10}	Clock, $\theta > 90^0$	Set $Flag$ register to 0; Move register θ_d to accumulator z	S_{39}	Clock, $F < f\{W_{gbest}\}$	Move register W to register W_{gbest} ;Update register $f\{W_{gbest}\}$
S_{11}	Clock	Activate counter i; Set register x to 0.61; Set register y to 0	S_{40}	Clock, $\theta > 90^0$, $F \geq f\{W_{pbest}\}$, $F \geq f\{W_{gbest}\}$	Activate counter n
S_{12}	Clock, $z \geq 0$	Set register σ to 1	S_{41}	Clock, $n \leq N$	Update register V
S_{13}	Clock, $z < 0$	Set register σ to -1	S_{42}	Clock, $1 < V < 0$	Move register $LFSR$ to register V
S_{14}	Clock, $i \leq i_{max}$	Update register x, y and z	S_{43}	Clock, $1 \geq V \geq 0$	Update register W
S_{15}	Clock	Update counter i	S_{44}	Clock, $1 < W < 0$	Move register $LFSR$ to register W
S_{16}	Clock, $i > i_{max}$	Check $Flag$ register content	S_{45}	Clock, $1 \geq W \geq 0$	Update counter n
S_{17}	Clock, $Flag = 0$	Load register ψ_0; Set $Flag$ register to 1	S_{46}	Clock, $n > N$	Update counter m
S_{18}	Clock	Activate counter θ	S_{47}	Clock, $m > M$	Update counter t; Update register γ
S_{19}	Clock, $\theta \leq 90^0$	Move register θ to accumulator z	S_{48}	Clock, $t > t_{max}$	Activate counter θ
S_{20}	Clock, $Flag = 1$	Load register ψ_1 ; Subtract register ψ_0 from ψ_1 ; Set $Flag$ register to 2	S_{49}	Clock, $\theta \leq 90^0$	Set register AF_p to 0; Activate counter n
S_{21}	Clock	Activate counter n	S_{50}	Clock, $n \leq N$	Update register AF_p
S_{22}	Clock, $n \leq N$	Set accumulator z to $(n - ½)\psi$	S_{51}	Clock	Update counter n
S_{23}	Clock	Move register x to register a	S_{52}	Clock, $n > N$	Update counter θ
S_{24}	Clock	Update counter n	S_{53}	Clock, $\theta > 90^0$	Activate counter k
S_{25}	Clock, $n > N$	Set $Flag$ register to 1; Update counter θ	S_{54}	Clock, $k \leq D - 1$	Shift contents of register φ left
S_{26}	Clock, $\theta > 90^0$	Set register γ to 0.9 and $f\{W_{gbest}\}$ to ∞; Activate counter m and t	S_{55}	Clock	Update counter k
S_{27}	Clock, $m \leq M$	Set register $f\{W_{pbest}\}$ to ∞; Activate counter n	S_{56}	Clock, $k > D - 1$	Move register θ_d to last content of register φ; Update counter j
S_{28}	Clock, $n \leq N$	Move register $LFSR$ to register W and V	S_{57}	Clock, $j > D$	Halt the process

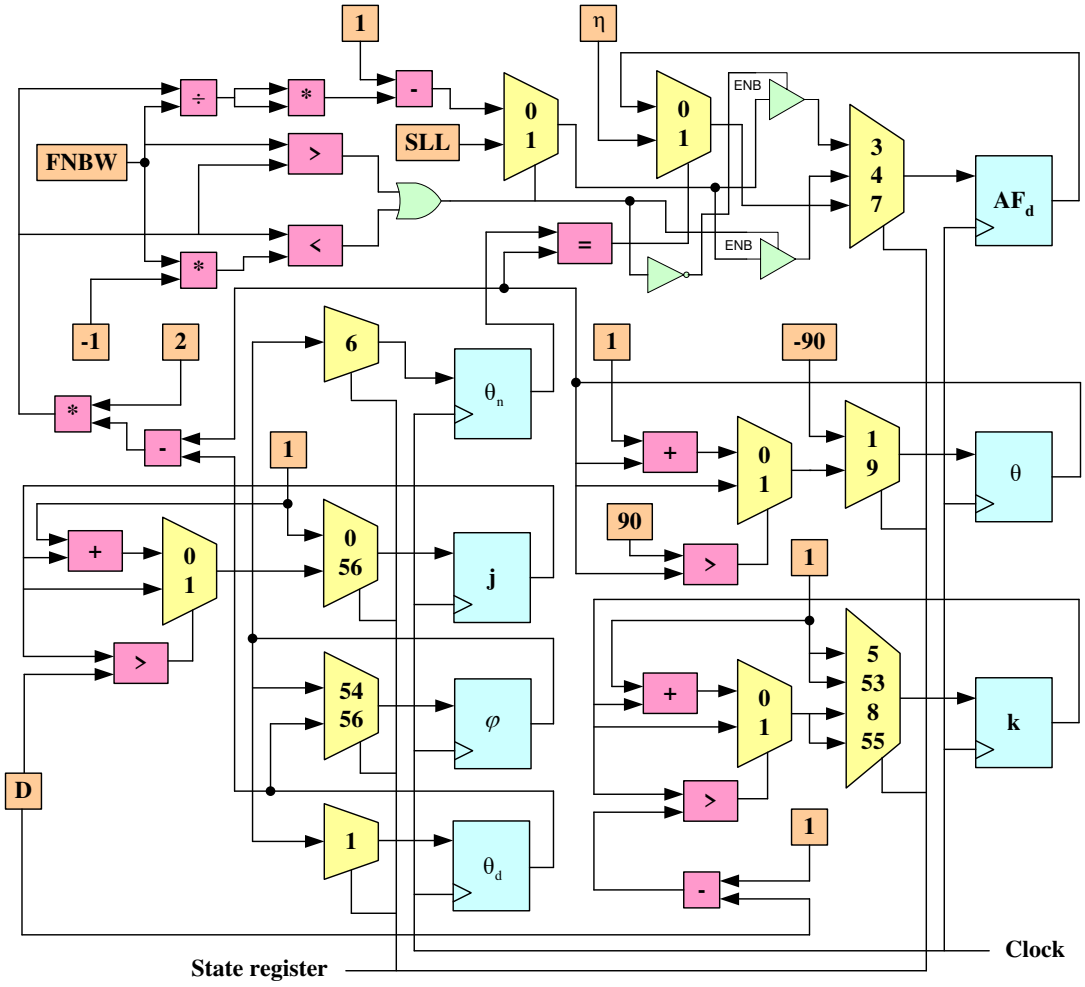

Figure 5: RTL schematic of the Template module

6 Performance evaluation

The beamforming architecture should always provide a stable link with a pattern of higher directivity for the target node. It must also produce a lower sidelobe level and deep nulls to keep all nearby nodes outside its radio coverage. Suitable benchmark functions are often desirable to evaluate its performance in a simulation environment. Therefore, we have performed a set of simulations on the Xilinx Virtex-5 (device: XC5VFX30T, frequency: 122.64 MHz) FPGA board for beamforming accuracy, FPGA resource utilization, power consumption, and computation time. We have chosen the simulation parameters by keeping compatibility with small wireless measurement systems such as the Crossbow MICAz mote that operates with an RF transceiver at 2.4 GHz and uses IEEE 802.15.4 MAC protocols. Table 2 shows all simulation parameters for this work. The Verilog hardware description language (HDL) maps the proposed design directly onto the FPGA board (Chu 2008).

6.1 Metrics

The performance depends on the accuracy of the beamforming process and the system complexities (utilization of hardware resources, computational overhead, and power usage) both. Hence, it is verified under four metrics as follows.

(a) *Beamforming accuracy* (Λ): It is a measure of exactness to produce the patterns according to the beamforming attributes defined on the template. In other words, it represents all deviations of the optimal pattern from the desired specifications. Equation (12) can express it (in percentage) as follows.

$$\Lambda = \left(1 - \frac{\sum_{\forall k} \delta_k}{K}\right) \times 100 \qquad (12)$$

Here, K is the total number of sample points taken to measure the deviations.

(b) *Hardware resource utilization* (υ): It is a quantitative measurement that indicates the number of resources used out of total resources available in the FPGA device and is usually expressed (in percentage) as a ratio of them using equation (13) as follows.

$$\upsilon = \frac{n_u}{n_t} \times 100 \tag{13}$$

Here, n_u and n_t represent the number of used resources and total available resources respectively in the system.

(c) *Computation time* (τ): It measures the total time needed for the system to execute weight vector computation and produce the optimal pattern. Therefore, it depends on the total number of clock cycles required to attain the convergence in the search process and expressed (in milliseconds) using equation (14) as follows.

$$\tau = t_{con} N_c t_c \tag{14}$$

Here, N_c, t_{con}, and t_c indicate the number of clock cycles per search, convergence time, and clock period, respectively.

(d) *Power consumption* (ρ_t): It is the measure of total power dissipation in the internal circuits usually arising in two forms: static power (caused due to leakage currents) and dynamic power (caused due to capacitive loads). So, it is expressed (in milliwatts) as the sum of these two power components using equation (15) as follows.

$$\rho_t = \rho_s + \rho_d \tag{15}$$

Here, ρ_s and ρ_d denote the static and dynamic power respectively in the system.

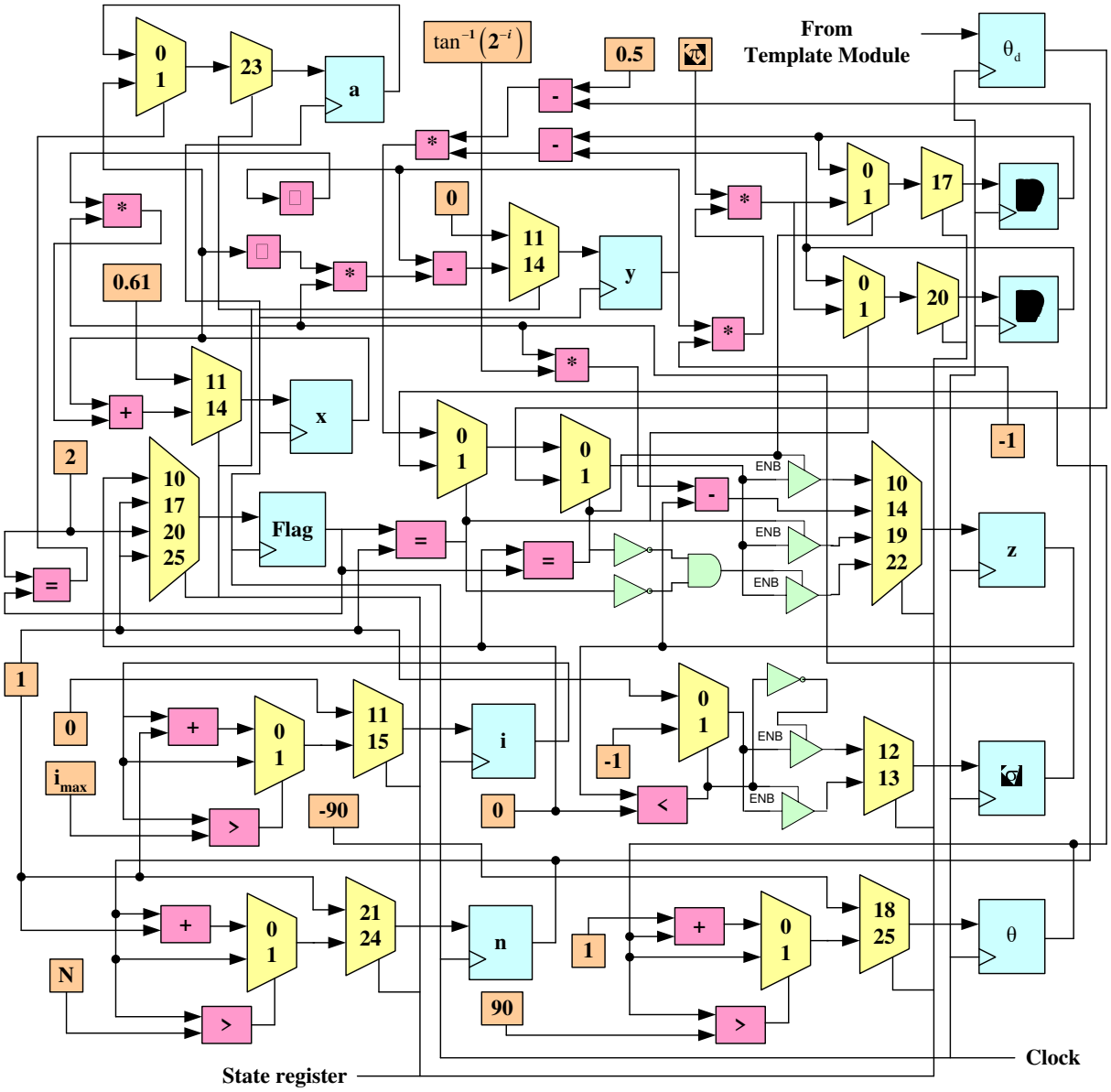

Figure 6: RTL schematic of the CORDIC module

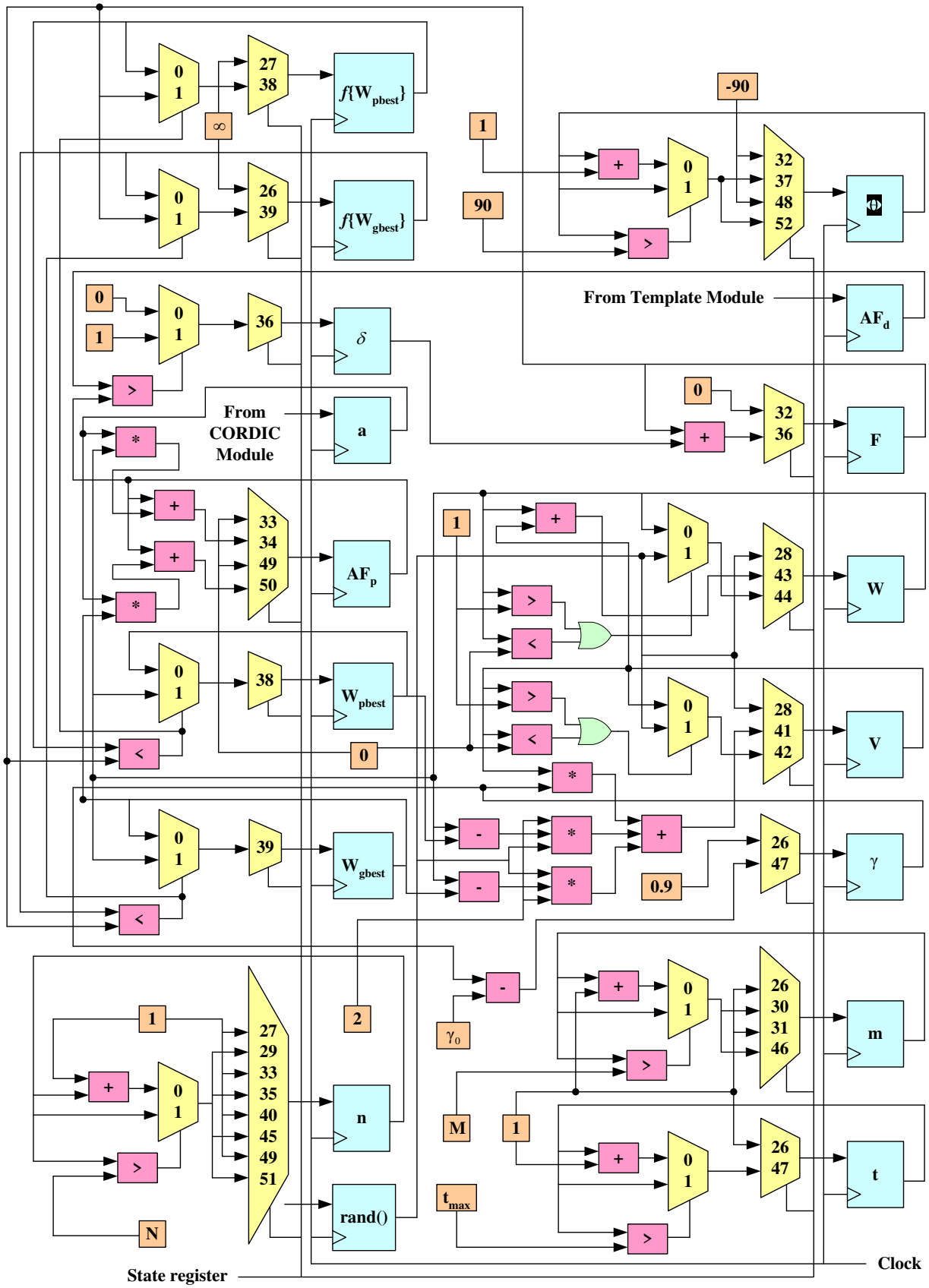

Figure 7: RTL schematic of the PSO module

Table 2: Simulation parameters

Parameters	Values
Number of symmetrical antennas (N) in array	10
Simulation frequency (f_s) on FPGA board	$122.64\ MHz$
Spacing (Δ) between the elements in array	0.5λ
Sidelobe level (SLL) for $-30\ dB$	0.0316
Beamwidth between the first nulls ($FNBW$)	20^0
Depth of the nulls (η) for $-100\ dB$	0.00001
Angle of arrivals (AoA) for a specific time	$-45^0, -30^0, -10^0, 20^0, 40^0$
Maximum permissible search time (t_{max}) in PSO	1000
Maximum permissible iterations (i_{max}) in CORDIC	10
Number of particles (M) in PSO	20

6.2 Results

We have tested the beamforming system for its performance by taking a set of simulations for the metrics mentioned above. Figure 8(a), Figure 8(b), Figure 8(c), Figure 8(d), and Figure 8(e) respectively show the optimal patterns for a specific AoA data. These figures explain that reducing SLL can improve the layer of protection in the node localization if null steering fails due to a wrong AoA estimation. Table 3 provides the corresponding optimized values for the elements of the weight vector. Figure 9 illustrates the convergence behaviors of the beamforming process in each case. Again, accuracy and convergence in a beamforming process may vary for different beam steering angles. Therefore, we have presented them as the Empirical Cumulative Distribution Function (ECDF) by considering 30 runs of the program and plotted in Figure 10 (a) and Figure 10 (b), respectively. Table 4 also lists the minimum and maximum levels of accuracy and convergence attained for these particular cases. We observed that the accuracy always decreases, and more iteration is required to finish the process when the magnitude of the beam steering angle becomes larger.

The data obtained from the experimentations on the Xilinx virtex 5 (version: ISE 14.7) FPGA board verifies the system complexities (space and time both). Table 5 and Table 6 respectively provide these data. It uses only fewer resources that remain almost consistent in all cases. Here, we have computed the execution time for a minimal number of iterations needed, and they are also within an acceptable limit. Table 7 gives the statistics of power consumption. It requires low power that remains constant for all cases.

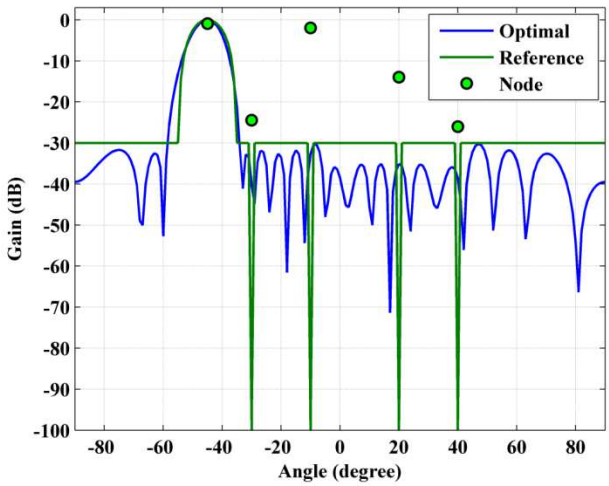

Figure 8(a): Optimal pattern for $\theta_d = -45^0$

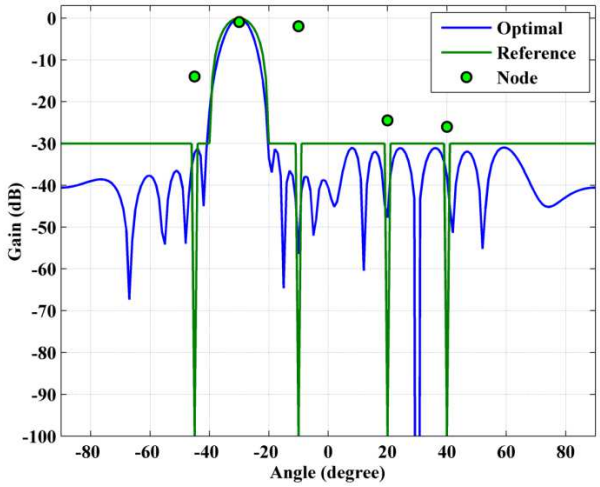

Figure 8(b): Optimal pattern for $\theta_d = -30^0$

The simulation results for resource usage, computation time, and power consumption are compared with some well-known methods and given in Table 8, Table 9, and Table 10, respectively. It clarifies that the proposed system often requires less power and can be realized with minimal hardware resources. Also, its execution time is comparable to the others.

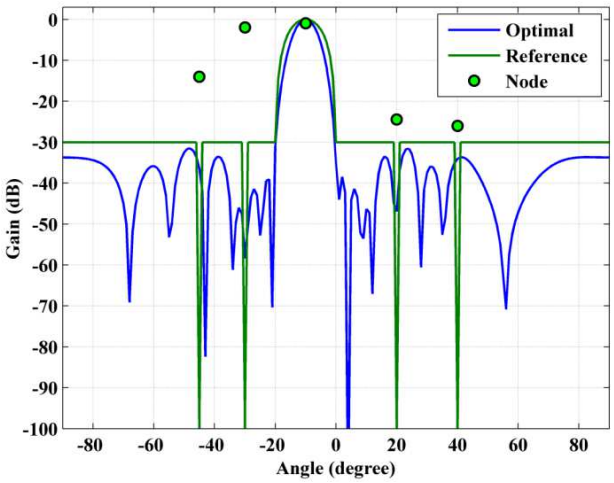

Figure 8(c): Optimal pattern for $\theta_d = -10^0$

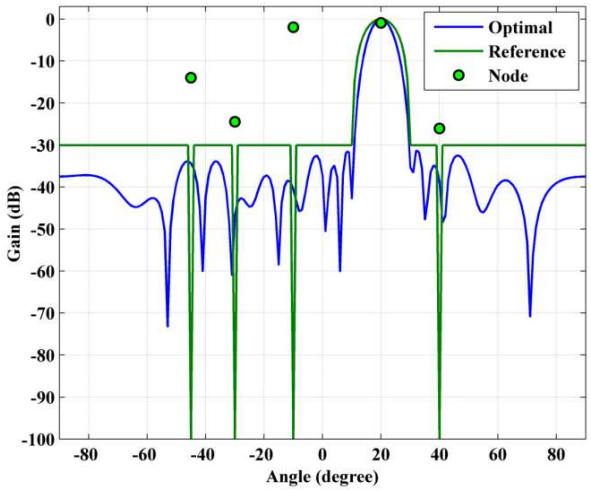

Figure 8(d): Optimal pattern for $\theta_d = 20^0$

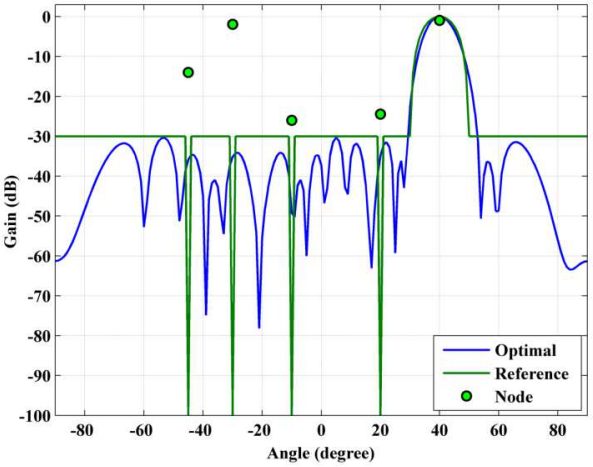

Figure 8(e): Optimal pattern for $\theta_d = 40^0$

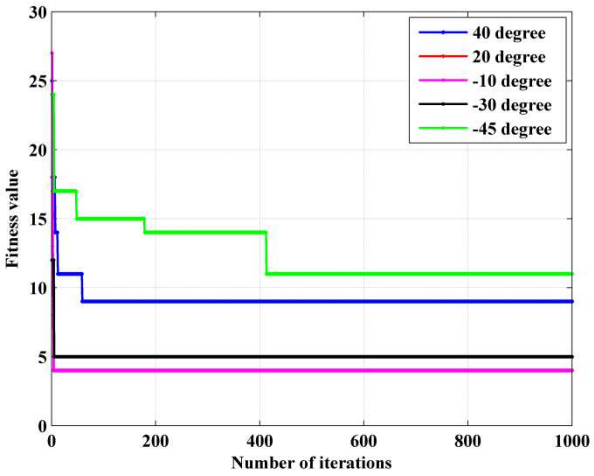

Figure 9: Convergence in the beamforming process

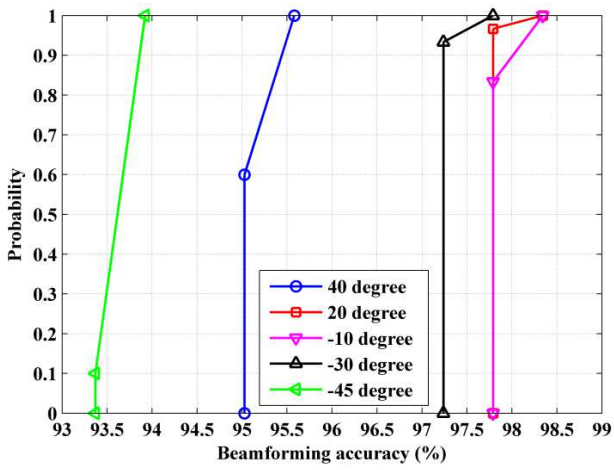

Figure 10 (a): ECDF of beamforming accuracy

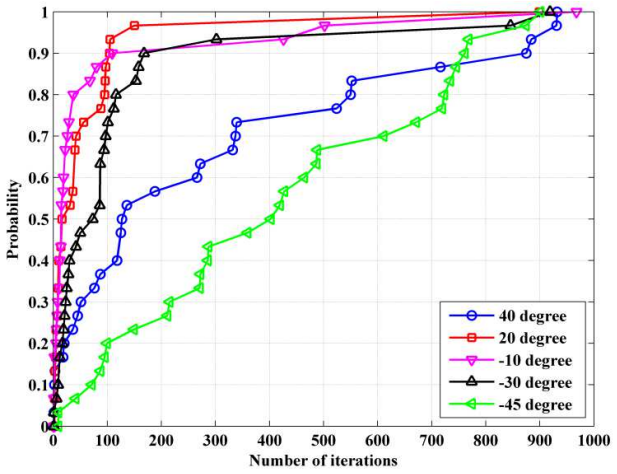

Figure 10 (b): ECDF of convergence in the beamforming process

Table 3: Optimum values of weight vector (W_{gbest})

Index	Directions of beam orientations (θ_d)				
	$\theta_d = -45^0$	$\theta_d = -30^0$	$\theta_d = -10^0$	$\theta_d = 20^0$	$\theta_d = 40^0$
n	W_n	W_n	W_n	W_n	W_n
1	0.9623	0.9110	0.9072	0.8864	0.9619
2	0.9190	0.8338	0.8897	0.8716	0.8959
3	0.8844	0.8030	0.7712	0.8391	0.8303
4	0.7528	0.7345	0.7075	0.7293	0.6914
5	0.6722	0.6203	0.5717	0.5760	0.6530
6	0.5950	0.4495	0.4968	0.5288	0.5393
7	0.4030	0.4417	0.3102	0.3836	0.4167
8	0.3415	0.2359	0.2028	0.3029	0.2521
9	0.2692	0.1952	0.1842	0.1789	0.2304
10	0.1860	0.0983	0.0680	0.1040	0.1422

Table 4: Accuracy and convergence of the beamforming process

Directions of beam orientations (θ_d)	Accuracy (Λ)		Iterations (t_{con})	
	Minimum	Maximum	Minimum	Maximum
$\theta_d = -45^0$	93.37	93.92	9	901
$\theta_d = -30^0$	97.24	97.79	1	919
$\theta_d = -10^0$	97.79	98.34	1	968
$\theta_d = 20^0$	97.79	98.34	2	899
$\theta_d = 40^0$	95.03	95.58	1	932

Table 5: FPGA resource utilization (υ)

Parameters	Directions of beam orientations (θ_d)				
	$\theta_d = -45^0$	$\theta_d = -30^0$	$\theta_d = -10^0$	$\theta_d = 20^0$	$\theta_d = 40^0$
Number of Slice Registers	724 out of 20480 (3 %)	724 out of 20480 (3 %)	724 out of 20480 (3 %)	724 out of 20480 (3 %)	724 out of 20480 (3 %)
Number of Slice LUTs	1079 out of 20480 (5 %)	1093 out of 20480 (5 %)	1092 out of 20480 (5 %)	1092 out of 20480 (5 %)	1091 out of 20480 (5 %)
Number of occupied Slices	447 out of 5120 (8 %)	442 out of 5120 (8 %)	435 out of 5120 (8 %)	396 out of 5120 (7 %)	453 out of 5120 (8 %)
Number of fully used LUT-FF pairs	654 out of 1149 (56 %)	667 out of 1150 (58 %)	674 out of 1142 (59 %)	668 out of 1148 (58 %)	646 out of 1169 (55 %)
Number of bonded IOBs	25 out of 360 (6 %)	25 out of 360 (6 %)	25 out of 360 (6 %)	25 out of 360 (6 %)	25 out of 360 (6 %)
Number of Block RAM/FIFO	1 out of 68 (1 %)	1 out of 68 (1 %)	1 out of 68 (1 %)	1 out of 68 (1 %)	1 out of 68 (1 %)
Number of BUFG/BUFGCTRLs	1 out of 32 (3 %)	1 out of 32 (3 %)	1 out of 32 (3 %)	1 out of 32 (3 %)	1 out of 32 (3 %)
Number of DSP48Es	7 out of 64 (10 %)	7 out of 64 (10 %)	7 out of 64 (10 %)	7 out of 64 (10 %)	7 out of 64 (10 %)

Table 6: Computation time (τ)

Parameters	Directions of beam orientations (θ_d)				
	$\theta_d = -45^0$	$\theta_d = -30^0$	$\theta_d = -10^0$	$\theta_d = 20^0$	$\theta_d = 40^0$
Clock cycles per search = Time elapsed (μs) × Device frequency (MHz) ($N_c = t_e f_s$)	88.76 ×122.64 = 10885	88.76 ×122.64 = 10885	88.76 ×122.64 = 10885	88.76 ×122.64 = 10885	88.76 ×122.64 = 10885
Total clock cycles ($N_c t_{con}$)	10885×9	10885×1	10885×1	10885×2	10885×1
Clock period (ns) (t_c)	8.15	8.15	8.15	8.15	8.15
Computation time (ms) = Total clock cycles × Clock period (ns) ($\tau = N_c t_{con} t_c$)	10885×9×8.15 = 0.798	10885×1×8.15 = 0.089	10885×1×8.15 = 0.089	10885×2×8.15 = 0.177	10885×1×8.15 = 0.089

Table 7: Power consumption (ρ_t)

Parameters	Directions of beam orientations (θ_d)				
	$\theta_d = -45^0$	$\theta_d = -30^0$	$\theta_d = -10^0$	$\theta_d = 20^0$	$\theta_d = 40^0$
Dynamic Power (mW)	6.66	6.66	6.66	6.66	6.66
Static Power (mW)	840.10	840.10	840.10	840.10	840.10
Total On-Chip Power (mW)	846.76	846.76	846.76	846.76	846.76

Table 8: Comparison of FPGA resource utilization

Beamforming systems	FPGA environment (Device, frequency)	Resource utilization (%)					
		LUTs	FFs	Slices	IOBs	BRAM	DSP48
PSO-CORDIC (Proposed)	Xilinx virtex 5 (XC5VFX30T, 122.64 MHz)	1093 out of 20480 (5 %)	724 out of 20480 (3 %)	453 out of 5120 (8 %)	25 out of 360 (6 %)	1 out of 68 (1 %)	7 out of 64 (10 %)
SCC (Dikmese et al. 2010, 2011)	Xilinx virtex 4 (XC4VLX60, 500 MHz)	43560 out of 53248 (82 %)	10727 out of 53248 (20 %)	26622 out of 26624 (100 %)	17 out of 448 (4 %)	--	8 out of 64 (13 %)
LMS (Dikmese et al. 2010, 2011)	Xilinx virtex 4 (XC4VLX60, 500 MHz)	21804 out of 53248 (41 %)	4271 out of 53248 (8 %)	13642 out of 26624 (51 %)	321 out of 448 (72 %)	--	16 out of 64 (25 %)
CM (Dikmese et al. 2010, 2011)	Xilinx virtex 4 (XC4VLX60, 500 MHz)	22026 out of 53248 (41 %)	4970 out of 53248 (9 %)	14105 out of 26624 (53 %)	321 out of 448 (72 %)	--	16 out of 64 (25 %)
MSR-CORDIC (Thiripurasundari et al. 2017)	Xilinx virtex 4 (XC4VLX60, 500 MHz)	37820 out of 53248 (71 %)	8597 out of 53248 (16 %)	22315 out of 26624 (83 %)	421 out of 448 (93 %)	49 out of 416 (11 %)	--
QRD-RLS (Dick et al. 2006)	Xilinx virtex 4 (XC4VLX60, 250 MHz)	5411 out of 53248 (10 %)	5916 out of 53248 (11 %)	3530 out of 26624 (13 %)	--	6 out of 416 (1 %)	13 out of 64 (20 %)
D^3 (Jarrah and Jamali 2013)	Xilinx Artix 7 (XA7A100T, 8.45 MHz)	42572 out of 63400 (67 %)	18251 out of 63400 (28 %)	12268 out of 15850 (77 %)	--	14 out of 135 (10 %)	64 out of 240 (26 %)
Frost (Llamocca and Aloi 2018)	Xilinx Zynq-7000 (XC7Z020, 100 MHz)	33537 out of 34648 (96 %)	34583 out of 69296 (49 %)	--	--	--	96 out of 116 (82 %)

Table 9: Comparison of computation time

Beamforming systems	FPGA environment (Device, frequency)	Computation time (ms)
PSO-CORDIC (Proposed)	Xilinx virtex 5 (XC5VFX30T, 122.64 MHz)	0.089
SCC (Dikmese et al. 2010, 2011)	Xilinx virtex 4 (XC4VLX60, 500 MHz)	0.022
LMS (Dikmese et al. 2010, 2011)	Xilinx virtex 4 (XC4VLX60, 500 MHz)	0.059
CM (Dikmese et al. 2010, 2011)	Xilinx virtex 4 (XC4VLX60, 500 MHz)	0.091
MSR-CORDIC (Thiripurasundari et al. 2017)	Xilinx virtex 4 (XC4VLX60, 500 MHz)	0.017
QRD-RLS (Dick et al. 2006)	Xilinx virtex 4 (XC4VLX60, 250 MHz)	0.057
D^3 (Jarrah and Jamali 2013)	Xilinx Artix 7 (XA7A100T, 8.45 MHz)	6
Frost (Llamocca and Aloi 2018)	Xilinx Zynq-7000 (XC7Z020, 100 MHz)	0.080
QRD-MGS (Hasanikhah et al. 2018, 2019)	Altera Arria 10 (10AX115, 210 MHz)	0.037
STAP (Hasanikhah et al. 2018, 2019)	Altera Arria 10 (10AX115, 210 MHz)	0.144

Table 10: Comparison of power consumption

Beamforming systems	FPGA environment (Device, frequency)	Power consumption (mW)
PSO-CORDIC (Proposed)	Xilinx virtex 5 (XC5VFX30T, 122.64 MHz)	846.76
D^3 (Jarrah and Jamali 2013)	Xilinx Artix 7 (XA7A100T, 8.45 MHz)	4284
QRD-MGS (Hasanikhah et al. 2018, 2019)	Altera Arria 10 (10AX115, 210 MHz)	2654
STAP (Hasanikhah et al. 2018, 2019)	Altera Arria 10 (10AX115, 210 MHz)	6369

7 Conclusion

This work presents the implementation of a novel beamforming circuit for adaptive array antennas. It works with three separate modules: Template, CORDIC, and PSO. Based on the AoA data, the Template unit produces a pattern having a prescribed SLL, FNBW, and null depth as a reference. The CORDIC unit produces the steering vector of the array based on the data of the antenna design parameters. The PSO unit produces an optimal value for the weight vector of the array elements based on the data from the fitness function, which yields an optimal pattern. It has a reconfigurable attribute that can provide a stable link to the target node with higher directive gain. Besides, it has a lower SLL and deep nulls placed in the directions of neighboring nodes, keeping them out of radio coverage. Thus, it is well suited for the precise localization of wireless nodes in a hostile environment. Here, we have used the concept of the FSMD design technique to realize it on a dedicated FPGA chip and verified its performance with multiple fixed-point hardware-level simulations of several metrics. It shows that minimal error in beamforming can often be obtainable using minimal hardware resources. However, this system is based on a behavioral model and reveals high latency. Thus, it is still possible to test its performance on a parallel and pipeline architecture in the future.

Declarations

Author contributions All authors have an equal contribution to this research work.

Funding This research work is not funded by any agency.

Data availability Data is only available on request.

Code availability Not available.

Compliance with ethical standards This research work retains all ethical publication policies.

Conflicts of interest The authors have no potential conflicts of interest.

References

Biswas RN, Mitra SK, Naskar MK (2014) A robust mobile anchor-based localisation technique for wireless sensor network using smart antenna. Int J Ad Hoc Ubiquitous Comput 15: 23- 37. https://doi.org/10.1504/IJAHUC.2014.059914

Biswas RN, Mitra SK, Naskar MK (2021) Wireless node localization under hostile radio environment using smart antenna. Wireless Pers Commun 116: 1815- 1836. https://doi.org/10.1007/s11277-020-07763-8

Choi S, Shim D (2000) A novel adaptive beamforming algorithm for a smart antenna system in a CDMA mobile communication environment. IEEE Trans Veh 49: 1793- 1806. https://doi.org/10.1109/TVT.2002.807584

Chu PP (2008) FPGA prototyping by Verilog examples. John Wiley & Sons Inc., New Jersey

Dick C, Harris F, Pajic M, Vuletic D (2006) Real-time QRD-based beamforming on an FPGA platform. In: 2006 Fortieth Asilomar Conference on Signals, Systems and Computers, IEEE. https://doi.org/10.1109/ACSSC.2006.354945

Dikmese S, Kavak A, Kucuk K, Sahin S, Tangel A (2011) FPGA based implementation and comparison of beamformers for CDMA 2000. Wireless Pers Commun 57: 233- 253. https://doi.org/10.1007/s11277-009-9855-4

Dikmese S, Kavak A, Kucuk K, Sahin S, Tangel A, Dincer H (2010) Digital signal processor against field programmable gate array implementations of space-code correlator beamformer for smart antennas. IET Microw Antennas Propag 4: 593- 599. https://doi.org/10.1049/iet-map.2009.0151

Gies D, Rahmat-Samii Y (2003) Particle swarm optimization for reconfigurable phase differentiated array design. Microwave Opt Technol Lett 38: 168- 175. https://doi.org/10.1002/mop.11005

Godara LC (2004) Smart antennas. CRC Press LLC, Boca Raton

Gross FB (2005) Smart antennas for wireless communications. McGraw-Hill Companies Inc., New York

Hasanikhah N, Amin-Nejad S, Darvish G, Moniri MR (2019) Comparison of practical methods for an efficient FPGA implementation of STAP. Int J Electron 106: 1113- 1126. https://doi.org/10.1080/00207217.2018.1553247

Hasanikhah N, Amin-Nejad S, Darvish G, Moniri MR (2018) Efficient implementation of space–time adaptive processing for adaptive weights calculation based on floating point FPGAs. J Supercomput 74: 3193- 3210. https://doi.org/10.1007/s11227-018-2369-7

Hong YWP, Lan PC, Kuo CCJ (2014) Signal processing approaches to secure physical layer communications in multi-antenna wireless systems. In: Gan WS, Kuo CCJ (Eds), Signal Processing. Springer International Publishing AG, New York, pp 1- 6

Jarrah A, Jamali M (2013) Software tool for efficient FPGA design of direct data domain approach for space-time adaptive processing. Electron Lett 49: 789-791. https://doi.org/10.1049/EL.2013.1307

Jayaprakasam S, Rahim SKA, Leow CY, Ting TO, Eteng A (2017) Multiobjective beampattern optimization in collaborative beamforming via NSGA-II with selective distance. IEEE Trans Antennas Propag 65: 2348 - 2357. https://doi.org/10.1109/TAP.2017.2684187

Kim M, Ichige K, Arai H (2004) Real-time smart antenna system incorporating FPGA-based fast DOA estimator. In: IEEE 60th Vehicular Technology Conference, IEEE. https://doi.org/10.1109/VETECF.2004.1399953

Kucuk K, Kavak A, Karakoc M, Yigit H, Ozdemir C (2009) A practical space-code correlator receiver for DSP based software radio implementation in CDMA2000. Wireless Pers Commun 49: 245- 261. https://doi.org/10.1007/s11277-008-9570-6

Liberti JC, Rappaport TS (1999) Smart antennas for wireless communications: IS-95 and third generation CDMA applications. Prentice Hall, New Jersey

Llamocca D, Aloi DN (2018) Self-reconfigurable implementation for a switched beam smart antenna. Microprocess Microsyst 60:1- 14. https://doi.org/10.1016/j.micpro.2018.03.009

Meher PK, Walls J, Juang TB, Sridharan K, Maharatna K (2009) 50 years of CORDIC: algorithms, architectures and applications. IEEE Trans Circuits Syst I Regul Pap 56: 1893- 1907. https://doi.org/10.1109/TCSI.2009.2025803

Nuteson TW, Stocker JE, Clark JS, Haque DS, Mitchell GS (2002) Performance characterization of FPGA techniques for calibration and beamforming in smart antenna applications. IEEE Trans Microw Theory Tech 50: 3043- 3051. https://doi.org/10.1109/TMTT.2002.805151

Razavilar J, Rashid-Farrokhi F, Liu KJR (1999) Software radio architecture with smart antennas: a tutorial on algorithms and complexity. IEEE J Sel Areas Commun 17: 662- 676. https://doi.org/10.1109/49.761043

Robinson J, Rahmat-Samii Y (2004) Particle swarm optimization in electromagnetics. IEEE Trans Antennas Propag 52: 397- 407. https://doi.org/10.1109/TAP.2004.823969

Thiripurasundari C, Sumathy V, Thiruvengadam C (2017) An FPGA implementation of novel smart antenna algorithm in tracking systems for smart cities. Comput Electr 65: 59- 66. http://doi.org/10.1016/j.compeleceng.2017.06.009

Vahid F (2011) Digital design with RTL design, VHDL, and Verilog. John Wiley & Sons Inc., New Jersey

Wang T, Yang Y (2011) Location privacy protection from RSS localization system using antenna pattern synthesis. In: 2011 Proceedings IEEE INFOCOM, IEEE. https://doi.org/10.1109/INFCOM.2011.5935061

Zardi F, Nayeri P, Rocca P, Haupt RL (2021) Artificial intelligence for adaptive and reconfigurable antenna arrays-a review. IEEE Antennas Propag Mag 63:28-38. https://doi.org/10.1109/MAP.2020.3036097