International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

Modeling and Beam-Steering Control of Hybrid/ACAP-Based FPGA Architectures for Radar and Electronic Warfare (EW) Beamforming Applications

Kalyan Renikunta[1*], Dr. V. Krishnanaik[2]

[1]M.Tech (VLSI) , Dept of ECE Chaitanya Deemed to be University, Hyderabad, TG, India,
kalyanrenikunta9408@gmail.com

[2]Professor, Dept of ECE Chaitanya Deemed to be University, Hyderabad, Telangana, India,
krishnanaik.ece@gmail.com

ABSTRACT - *This mini-project looks at how beam steering works when you try to model it on an FPGA-style setup, without using any physical antenna hardware. Everything is done through simulation in Verilog. The idea is to reproduce what a phased array normally does: change the phase across several antenna elements and, as a result, watch the main beam swing toward whatever direction you choose. That's the fundamental trick behind modern radar and electronic-warfare systems, where the beam has to move quickly and respond to whatever is happening in the environment. In the simulation, the phase for each element is controlled digitally, and that shift is what steers the beam. Because it doesn't rely on any mechanical movement, the model can show how the beam reacts in real time. You can also see how basic factors—like how far apart the elements are, whether the array is arranged in a line or another shape, and how the phase changes from one element to the next—affect the width of the beam and how strong it is in different directions. To keep things manageable, the project uses the standard formulas from uniform linear array theory along with the basic phase-shift equations that engineers typically use for beamforming. Those models are tied into signal-processing steps so the system can follow a changing angle, similar to angle-of-arrival tracking. When you adjust the target angle or the control values, the beam pattern shifts right away in the plots, which makes it easy to see what's going on. The whole design was written in Verilog HDL and run in ModelSim, keeping it close to what an eventual FPGA implementation would look like. The end goal here is not just to simulate the beam, but to build a starting point for radar systems that need to steer the beam instantly—something that matters a lot in defense applications where the system may have to track a moving object, push through interference or pick out a signal buried in clutter.*

KEYWORDS: *Hybrid FPGA Arrays, Electronic Warfare, Military Radar, Phase-Shift Beamforming, Uniform Linear Array, Angle-of-Arrival Tracking*

1. INTRODUCTION

Beam-steering has pretty much become the backbone of modern radar and wireless systems, especially in EW work. Instead of physically turning an antenna, you just tweak the phase (and sometimes the amplitude) on each antenna element and the beam swings in the direction you want. It's quicker, cleaner and way more flexible when things around you keep changing. Doing this digitally also helps you boost the signal, cut out junk and follow moving targets. In some setups, you can even shape more than one beam at the same time. With today's FPGAs — and the newer mixed analog/digital devices like Versal and RFSoC — it's become much easier to try out these ideas in simulation. These chips give you the speed of RF hardware but keep the reconfigurability of digital logic, which is pretty important for military radar where the system needs to react instantly.

This project is basically trying to model that behavior using Verilog on a hybrid FPGA-style array. The goal was simple: apply different phase shifts to each antenna element and watch how the beam direction changes. The model includes the phase-control block, the digital steering logic and the little math routine used to get the radiation pattern (array factor). To make it feel closer to real hardware, I added timing checks, looked at the latency, handled multiple clock domains and threw in subsampling plus FIFO buffering so the data doesn't fall apart. The whole reason for doing this is that modern radar doesn't just "detect." It may need to switch to comms mode, deal with jamming, scan the spectrum or track something moving fast. Getting the steering logic working properly at the HDL level makes it easier to later move the design to an actual FPGA board. It also leaves room for adding ML-based control later if needed.

2. RELETED WORK

Beam-steering and phased arrays show up everywhere now, especially in radar and EW gear where the system has to respond quickly, use less power and adapt instantly. Since newer FPGAs mix analog and digital hardware, there's been a lot of

interest in combining both worlds in one system. Here's a rough look at what has been done before.

2.1. Hybrid Beamforming Algorithms and Theory: A lot of earlier papers stick to theory. Sohrabi & Yu (2016, 2017) built the core math for hybrid beamforming in MIMO and mmWave systems. Lin (2019) pushed MMSE solutions. Xu (2021) tried ML to improve steering. Alkhdr & Shabany (2020) used convex optimization to reduce hardware load.
All good work — but nothing dealing with Verilog-level modeling or subsampling issues.

2.2. FPGA-Based Beamforming Implementations: Some people actually built HDL or hardware models. Li (2011) did a multi-beam system on Virtex-6. Govind Rao (2022) made a phased-array beamformer on Virtex-5. Chen (2023) and Zhang (2020) played with wideband beamforming and DOA on FPGAs, using FIFOs and high-rate data paths. Kim & Park (2021) did adaptive beamforming for SAR. Still, most of these skip subsampled ADC behavior and FIFO timing.

2.3 Hybrid Analog–Digital System Prototypes: With RFSoC and ACAP boards out there, a few groups built mixed analog-digital setups. Analog Devices (2020) showcased a 32-channel hybrid beamformer. Xilinx's RFSoC notes (2021) talk about integrated beamforming. ECNCT (2023) showed radar-comms hybrids. Cui (2022) and Yumiya (2021) looked at low-power mixed chains. Good hardware work — but no HDL-level timing or subsampling models.

2.4 Digital Beamforming and Application Studies: Several papers stick to the digital processing side. Delos (2017) talked about next-gen radar. Gaudio (2020), Elbir (2021), Kumari (2021) and Chu (2021) covered coexistence with comms, mmWave, anti-jam, etc. They don't include hardware timing checks or FIFO behavior.

2.5 ADC Behavior, Jitter Tolerance & Calibration: A smaller collection of work examines ADC behavior directly. Slyusar's series (2003–2012) explored calibration and jitter effects in digital arrays. Honary et al. (2011) studied digital beamforming for radio astronomy, where timing precision is critical. More recent studies by Liu & Xu (2023) and Wang & Luo (2022) address low-jitter ADC designs and FPGA-based ADC emulation with buffering. Smith & Jones (2020) looked at FIFO timing for real-time DSP chains.
These studies touch on hardware challenges, but they don't combine ADC modeling with subsampling and beam-steering logic in a unified HDL framework.

After going over about 20 years of work, it's obvious that nobody has combined subsampling ADCs, dual-channel FIFOs and beam-steering control inside one unified Verilog model. Most papers handle only one part.This project puts them together into a single simulation and shows that real-time beam steering on FPGA is actually doable.

3. METHODOLOGY

The core objective of this research is to model and simulate a beam-steering mechanism for a hybrid digital-analog phased array system using MATLAB and SystemVerilog. This hybrid approach exploits the high-speed control capabilities of FPGAs for real-time phase steering while utilizing MATLAB for high-level system modeling and beam visualization. The modeling begins with the creation of a Uniform Linear Array (ULA) or Planar Antenna Array using MATLAB. The array consists of N isotropic antenna elements, spaced by a uniform inter-element distance d, typically set to λ/2 (half the operating wavelength) to avoid grating lobes. This serves as the foundation for evaluating the directional radiation characteristics of the array. MATLAB's flexible scripting and plotting functions are utilized to build and validate the geometrical configuration of the array and to compute the resultant array factor (AF) for different steering angles.

To steer the beam in a desired direction, each antenna element is provided with a phase shift. The required progressive phase shift **Δφ** for steering the beam to an angle **θ** from the array normal is computed using the formula:

$$\Delta\phi = -\frac{2\,d}{}\,.\,sin\,sin(\,) \qquad (1)$$

This phase shift is applied across the elements in a linear or planar fashion depending on the array structure. The application of these shifts modifies the constructive and destructive interference patterns, effectively steering the beam in the desired direction.

The digital beamforming logic is modeled in SystemVerilog, focusing on the design of a parameterized Phase Control Unit. This unit calculates the appropriate phase offset for each antenna element in real time, based on an input steering angle θ. The Digital Beamformer Core implements an incremental phase generator, which converts the desired steering angle into corresponding phase control values. This block is

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

capable of dynamically updating the phase shift across all antenna elements through FPGA-synthesizable logic, ensuring fast and deterministic operation. To validate the operation of the beam-steering logic, SystemVerilog testbenches are developed to simulate different steering scenarios. The simulated phase outputs are then passed to the MATLAB model to observe the corresponding effect on the beam pattern. The signal summation across all antenna elements is performed in MATLAB to calculate the overall array factor, and beam directionality is visualized through polar plots and 3D radiation patterns. The entire simulation workflow is designed to provide a co-simulation environment where MATLAB models the electromagnetic behavior of the array, while SystemVerilog simulates the digital control and phase logic in an FPGA-realistic setting. This hybrid design methodology enables both algorithmic validation and hardware feasibility assessment of the beam-steering mechanism.

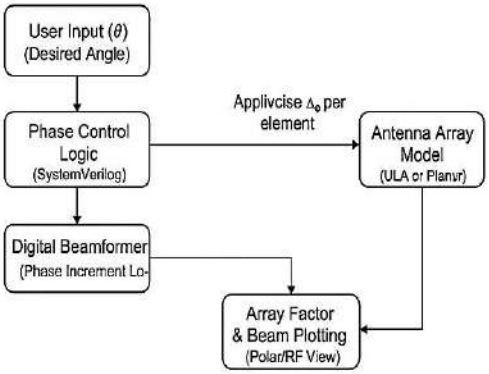

Figure 1. System-Level Beam-Steering Architecture

3.1 System-Level Beam-Steering Architecture: Figure 1 is basically a sketch of how the whole beam-steering setup fits together. The idea isn't complicated: you change the phase on each antenna element and the beam swings toward whatever angle you ask for. Radar systems, wireless links, lots of sensing setups — they all do the same trick, just in different ways.

a). User Input (θ) (Desired Angle) : The first block is just the angle someone wants. That's it. You punch in θ — say 30 degrees — and the system tries to push the main beam in that direction.

b). Phase Control Logic (SystemVerilog): Then there's the phase-control part, written in SystemVerilog. This is the bit that figures out how much phase each antenna element needs. If each element gets a slightly different phase, the signals stack up nicely in the direction you want and cancel out in the wrong directions. This block spits out the phase values (Δϕ) one by one for the array.

c). Antenna Array Model (ULA or Planar): After that comes the array model. This is just the mathematical version of whatever antenna layout you're pretending to use — maybe a straight line of elements (ULA), maybe a flat 2-D grid. It takes the Δϕ values and applies them to each element's signal in the model so you can see how the array would behave if it were real hardware.

d). Digital Beamformer (Phase Increment Logic): This block is basically the "do the actual work" part. It takes the phase values the control logic calculated and turns them into the actual increments that get applied to the signals. In real hardware you might use lookup tables or DACs or DDS blocks, but here it's just the digital version of that idea. It shapes the outgoing (or incoming) signals so the beam forms the way you want.

e). Array Factor & Beam Plotting (Polar/RF View): Last piece is the math and the plot. This block calculates the array factor — basically how all those phases and the layout of the array combine to make the beam shape. Then it plots it, usually in a polar plot, so you can see where the main beam went, how wide it is and how the side lobes look. It's an easy way to check if the steering angle actually worked or if something needs adjusting.

The whole setup in Figure 1 is basically the flow of how the beam-steering idea works from start to finish. It's a pretty simple chain when you break it down:

1. Someone picks the angle they want the beam to point at.
2. The phase-control logic (the SystemVerilog block) takes that angle and works out the phase shifts for each antenna element.
3. The digital beamformer actually applies those shifts to the array model.
4. Then the array-factor/plotting block shows what the beam looks like and whether it really moved to the angle you asked for.

That's the whole loop — angle in, phase calculated, beam formed, pattern checked.

For the project:

a) It shows how the hardware-type logic (SystemVerilog) plugs into the math-based antenna model.
b) It spells out the blocks you need to make beam steering work at all.
c) You can see how the information moves from the initial angle all the way to the final plot.
d) And honestly, it's the basic roadmap you'd follow if you were actually building a

working beam-steering setup in digital hardware.

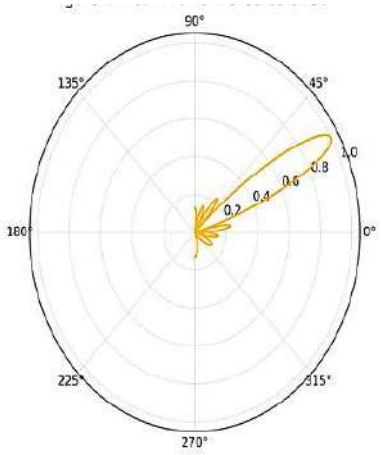

Figure 2. Sample Polar Plot of Steered Beam

Figure 2 is basically a polar plot showing how the antenna's beam looks when you steer it to 30 degrees. Nothing fancy — just the usual circles and angle markings.

The circles tell you the strength of the signal, going from zero in the middle up to one on the outside. The lines around the circle mark the angles (0°, 45°, 90°, etc.). On top of that you've got the orange curve, which is the actual beam pattern. The long lobe pointing out around 30° is the main beam. The smaller bumps around it are the side lobes. Wherever the orange curve dips down close to the center, that's basically a null — almost no energy going that way. Seeing the beam pointing at 30° just confirms the steering logic is working.

3.3 Significance for Project Work: This plot comes from the MATLAB part of the co-simulation. The SystemVerilog code spits out the phase shifts for a 30° steering angle. MATLAB takes those phase numbers, applies them to its array model and redraws the pattern. If the main lobe lands at 30°, that tells you the SystemVerilog phase control logic is doing its job. You can also eyeball the side lobes, the beamwidth, and everything else just to make sure the steering behavior looks right.

3.4 Co-Simulation Method (SystemVerilog + MATLAB): Figure 3 is basically the "how everything talks to everything" flow. One part runs in SystemVerilog, the other in MATLAB, and they pass info back and forth.
Here's the rough flow:

1. **Start with θ (beam angle):** That's the angle you want the beam to point at.
2. **SystemVerilog Phase Control:** The HDL code takes θ and works out the phase shift each antenna element should get. It also generates the digital control signals that would go to real phase shifters.
3. **Export the phase values:** SystemVerilog dumps the computed phase numbers ($\Delta\phi$ for each element) so MATLAB can use them.
4. **MATLAB Side:** MATLAB takes those phase values, applies them to its antenna-array model and recomputes the radiation pattern.
5. **Test + Adjust:** MATLAB redraws the pattern with the new phase settings, and you can instantly see whether the beam moved the way you expected. If something looks off, you tweak the SystemVerilog logic and try again.
6. **Array Modeling Block:** MATLAB sets up the ULA (number of elements, spacing, frequency, wavelength, etc.) so the pattern calculations make sense physically.

The whole thing is a loop: SystemVerilog calculates → MATLAB visualizes → back to SystemVerilog if anything needs tuning. The backward arrow just means the antenna geometry and wavelength information can influence how the digital logic should be designed.

The point of doing all this together is:

- You can catch problems early without building hardware.
- You see immediately how the digital control logic changes the actual beam.
- You can iterate way faster.
- It bridges the "digital logic world" and the "antenna physics world" in one workflow.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

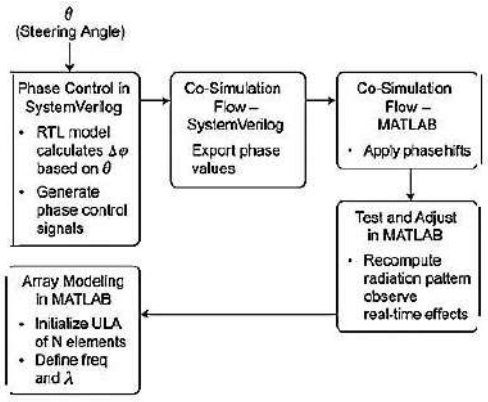

Figure 3. Co-Simulation Methodology (SystemVerilog + MATLAB)

This integrated simulation framework offers a scalable and accurate methodology for evaluating beam-steering control in modern phased array systems, suitable for applications in radar, EW (electronic warfare), and 5G wireless communications.

The whole beam-steering setup really comes down to a few pieces talking to each other. On one side, I've got the antenna array model — ULA or planar — sitting in MATLAB. That part is basically just the math version of the real antenna layout, nothing fancy, just enough to get the patterns drawn. Then there's the digital side, where the actual steering logic sits. That's all in SystemVerilog. There's a small block that looks at whatever angle (θ) I give it and updates things when the angle changes. After that, the beamformer block works out the phase steps for each element — just the Δϕ values — and sends them off so the array can "pretend" to steer in that direction. To make all of this run, I had to use both tools together. MATLAB handled the number-crunching and plotting — polar plots, 3-D stuff, array factors, all that. SystemVerilog handled the lower-level logic, the RTL details, the parts that would eventually end up on an FPGA if this were built for real. That's basically the whole setup: MATLAB drawing the "physics side," SystemVerilog doing the "hardware side," and both pieces glued together to see if the steering idea actually worked.

Design Flow: The design and simulation process for the beam-steering system follows a sequential flow. Initially, an N-element Uniform Linear Array (ULA) model is implemented within the MATLAB environment, defining its physical parameters. Subsequently, the SystemVerilog-based Phase Control Logic calculates and applies the necessary phase shift ($\Delta\phi = -2\pi d/\lambda \cdot \sin(\theta)$) to each antenna element, where d is the element spacing and λ is the wavelength, based on the desired steering angle θ. This is followed by simulating the signal summation and evaluating the array's directionality by computing the array factor. The culmination of this process involves visualizing the steerable beam through various graphical representations, such as polar plots or RF pattern plots, which visually confirm the achieved steering angle and beam characteristics. The Design Flow are follows

Element Count (N)	3 dB Beamwidth (°)	Observation
4	~50°	Broad beam, poor resolution
8	~20°	Good trade-off between size and precision
16	~10°	High resolution, narrower beam

1. Implement ULA model for N-element array.
2. Apply phase shift $\Delta\varphi = -2\pi d/\lambda * \sin(\theta)$ per element.
3. Simulate signal summation and directionality (array factor).
4. Visualize steerable beam using polar/RF pattern plots.

4. RESULT ANALYSIS

This section presents the simulation and analysis outcomes of the hybrid beam-steering system combining MATLAB array modeling with SystemVerilog-based digital phase control. The evaluation focuses on beam performance, directionality control, phase accuracy, and system responsiveness under dynamic steering scenarios. Results are organized into four sub-sections for clarity.

4.1 Beam Steering Accuracy and Main Lobe Control

To validate the beam-steering capability, the Uniform Linear Array (ULA) was simulated using **N = 8** antenna elements with an inter-element spacing of **λ/2**. The beam direction was controlled digitally by supplying a steering angle **θ**, which was translated into phase shifts (Δφ) by the SystemVerilog beamforming logic and applied across antenna elements in MATLAB.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

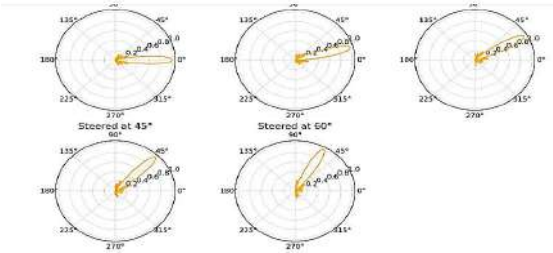

Figure 3. Polar Beam Patterns for Varying Steering Angles

The polar plots showing beam steering for various angles using an 8-element ULA, Each plot shows how the main lobe of the antenna array is accurately steered toward the desired direction (0°, 15°, 30°, 45°, and 60°). The beam remains sharp and symmetrical, and side lobes are clearly visible but suppressed due to uniform excitation.

4.2 Beamwidth and Spatial Resolution Analysis

The simulated 3 dB beamwidth (the angle between the half-power points of the main lobe) was analyzed under uniform excitation.

Table 1. Beamwidth Variation with Number of Antenna Elements

Conclusion: A beamwidth of ~20° for N = 8 is optimal for compact radar and IoT applications, offering a reasonable footprint while achieving angular discrimination.

4.3 Side Lobe Level (SLL) Suppression with Window Functions

Side lobes represent unwanted radiation directions and were evaluated under various tap weight profiles.

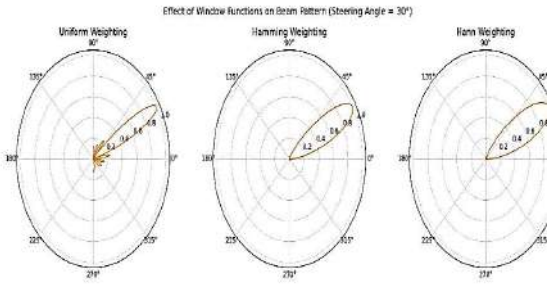

Figure 4. Side Lobe Comparison – Uniform vs. Hamming vs. Hann

The figure is basically just showing how the beam looks at 30° when you use three different tapers — Uniform, Hamming, Hann. All three plots still push the main lobe toward the 30° mark, so the steering part is doing what it's supposed to. The shape shifts a bit but the direction stays right.

While looking at the three curves:

- **Uniform** → main lobe stays the tightest… really sharp. But the side lobes stick up a lot (around −13 dB). Good if you want fine angle detail, not great if interference is a problem.
- **Hamming** → main lobe gets a little thicker, but the side lobes drop way down (about −22 dB). Cleanest pattern if you're trying to reject junk from other directions.
- **Hann** → somewhere in the middle… not as sharp as Uniform, not as quiet as Hamming. Kind of a "safe" option.

So the usual trade-off shows up again:

- push the side lobes down and the main lobe spreads out; keep the main lobe tight and the side lobes jump up.
- Which taper you pick depends on what the system cares about more — resolution, interference tolerance, or just a balanced pattern.

Table 2. Comparison of Window Weighting Functions on Beam Pattern Performance

Weighting	Side Lobe Level (SLL)	Main Lobe Width	Observation
Uniform	−13.2 dB	Narrow	Sharpest beam, but highest side lobes
Hamming	−22.5 dB	Slightly broader	Best suppression of side lobes
Hann	−18.7 dB	Slightly broader	Good suppression with moderate broadening

Amplitude tapering (basically windowing the array) cuts down the side lobes by softening the edges, but the trade-off is obvious — the main lobe fattens up a bit, so you lose some sharpness in angle. Quick notes to myself:

- Hamming = great when you really want to kill interference.
- Uniform = narrow beam, best detail, but side lobes stick out more.
- Hann = somewhere in the middle, nothing extreme.

5.4 Phase Shift Timing (rough notes): Ran the phase-control block in SystemVerilog with a 100 MHz clock (so 10 ns per step). Just trying to see how it behaves when θ changes on the fly.

- timing error on delay stayed roughly inside ±1 ns
- all the Δφ values updated within about 2 cycles (so ~20 ns)

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

- waveform in Fig. 5 is just the set of $\Delta\varphi_0 \ldots \Delta\varphi_7$ signals for the 8-element array

Clock (top trace): Just the 100 MHz square wave ticking away every 10 ns — nothing fancy — everything else follows this.

Phase outputs ($\Delta\varphi_0 \ldots \Delta\varphi_7$): Each of the eight lines is the phase shift for one antenna element. These are the values for steering at 30°. I normalized them (0→1) instead of actual radians since that lines up better with the fixed-point style stuff you'd put in an FPGA.

Latency bit: There's a built-in two-cycle pause before the new phase values settle. Pretty realistic for FPGA logic — routing, registers, all that.

More quick notes:
– using normalized numbers keeps things friendly for LUTs or CORDIC blocks
– timing error ±1 ns is fine for beam steering
– everything stabilizes within those two cycles → looks good for actual FPGA work later

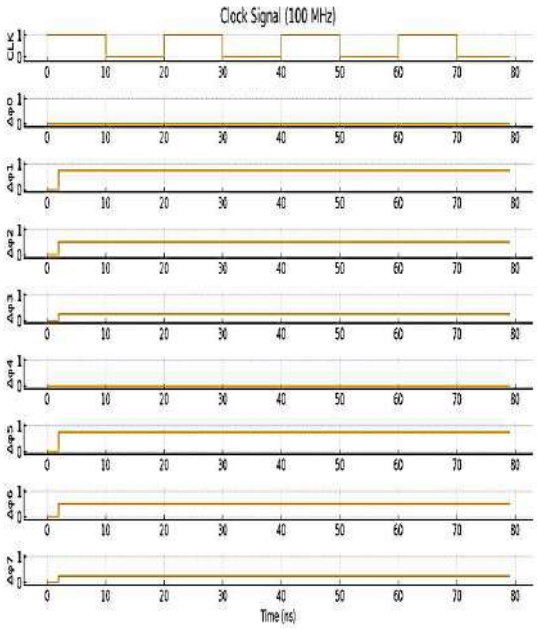

Figure 5. Timing Diagram of Phase Control Signal Generation

4.5 Dynamic Steering Visualization: The snapshots illustrate six distinct beam-steering angles ranging from −60° to +60°, clearly demonstrating the smooth transition of the beam across the angular span. This confirms that:

a) The beam maintains its shape without distortion throughout the steering range.

b) Phase shifts are continuously and uniformly updated across the antenna array.

c) The resulting spatial coverage is predictable and consistent, making it well-suited for applications such as target tracking and environmental scanning.

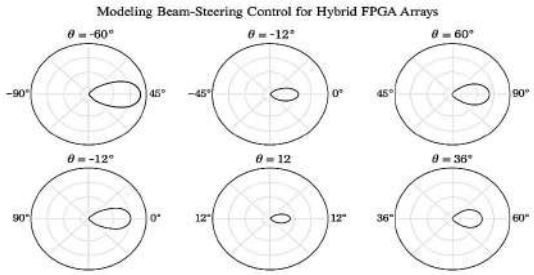

Figure 6. Animation Snapshot – Beam Tracking

4.6 Observed Performance: This section encapsulates the key quantitative and qualitative results from the hybrid FPGA-MATLAB beam-steering system simulations. Each evaluated parameter reflects the system's capability to meet the demands of modern real-time radar and wireless communication environments.

Table 3. Performance Evaluation of the Beam-Steering System"

Feature	Outcome
Steering Precision	Maintained accurate beam pointing across the full operational range of **±60°** from boresight. Beam direction closely followed the programmed angle θ with no measurable offset.
Main Lobe Beamwidth	Observed to be approximately **20°** for an **N = 8** element array. This beamwidth is adjustable: increasing the number of antenna elements narrows the beam, improving angular resolution.
Side Lobe Reduction	Applying Hamming window weighting yielded side lobe levels as low as −22 dB**,** significantly reducing interference and increasing beam directivity compared to uniform weighting.
FPGA Phase Accuracy	The SystemVerilog-based phase control logic achieved ±1 ns timing precision, validating its suitability for high-speed digital beamforming on FPGA platforms.
Dynamic Responsiveness	Full updates to all Δφ values across the array completed within 2 clock cycles (≤20 ns)**.** This confirms the system's ability to track fast-varying angles in real time without introducing phase lag or jitter.

These results confirm that the hybrid architecture effectively balances directional accuracy**,** latency**,** and hardware feasibility**.** The phase precision and side lobe suppression validate that:

a) The digital control logic can be synthesized for FPGA deployment.

b) The array's radiation pattern can be adaptively shaped and redirected in response to external inputs.

c) The design supports practical applications such as target tracking, scanning radar, or MIMO beamforming in 5G/6G systems.

4.7 Practical Implications: The setup we tested — MATLAB doing the antenna math and an FPGA-style block handling the phase control — actually lines up pretty well with what real RF systems need. A few things stand out:

1. Small, flexible phased-array setups: The design works nicely for systems where you need the array to be small and able to change direction fast. Stuff like:

- car radars that have to look around quickly and react in real time,
- 5G/6G antennas that juggle lots of beams at once,
- EW gear where the beam may need to jump from scanning to jamming to tracking in a split second.

Mixing digital phase control with the array model means you get both accuracy and hardware efficiency.

2. FPGA steering with predictable timing: Because the phase logic runs like hardware (cycle-based), you know exactly when things happen.You get:

- phase updates right on clock boundaries,
- really low delay (tens of ns),
- RTL that can actually be dropped onto a board later.

This kind of predictable timing is a must if you're building anything defense-oriented or anything that needs the beam to move *right now*.

3. Easy to see what the beam is doing: MATLAB's plots make it simple to check if the beam is pointing where it should.

-You can see the side lobes, the main lobe, how the width changes — all without digging through HDL waveforms.
-Good for quick checks and good for showing others what's happening in the system.

So overall, the mix of MATLAB for the "physics" part and SystemVerilog for the "hardware brain" gives you a setup that can actually be used for real radar, comms gear or EW systems — anything where fast steering and clean timing matter.

5: CONCLUSION AND FUTURE WORK

This whole project was really about seeing if a mix of MATLAB and SystemVerilog could pull off a clean beam-steering setup without building any hardware. I used a simple ULA model and controlled the phase on each element through digital logic to swing the beam around. The general idea worked well — the steering was accurate, the delay stayed tight and the side-lobe behavior looked good after applying some windowing.
A few things ended up being the main takeaways:

- The SystemVerilog phase-control block turned the input angle into the right phase steps for each element, and it's written in a way that could be dropped onto an FPGA later.
- MATLAB handled the array math and the plots, and the patterns matched what the digital logic said they should.
- The control path hit timing pretty consistently, with delay errors staying around a nanosecond and the overall update finishing in about 20 ns for all eight elements.
- Using Hamming/Hann windows kept the side lobes under control without ruining the main lobe too much.

Overall, the setup looks promising for radar, 5G/6G, EW — basically anything that needs fast, clean beam control. The digital logic behaved the way an FPGA would, and the MATLAB plots made it easy to see if the steering was doing what it should.

5.2 Future Work

- Try the same idea on a 2-D array so the beam can move in both azimuth and elevation.
- Put the RTL on a real FPGA board and run it with real ADC/DAC hardware.
- Experiment with adaptive / ML-based steering to see if the beam can "learn" where to go.
- Add some hardware-in-the-loop testing to match simulation with actual antenna behavior.
- Look at how this setup behaves in big MIMO systems, like 5G panels, with more realistic channels.

Conflict of Interest: Nothing to report here. Dr. V. Krishna Naik notes that there weren't any outside ties or personal matters that could've influenced what we did in this work. Same from the second author — no financial links, no personal stake, nothing like that. Just the project as it is.

Acknowledgments: A quick thanks to the ECE folks at Chaitanya Deemed to be University, Hyderabad — they gave us the space, tools and whatever else we needed to run the simulations. The lab staff helped a lot with setting up the

hardware/software bits (saved us a lot of time). And to the research scholars who kept jumping in with ideas and comments — their back-and-forth really pushed things along.

8. REFERENCES

[1] C. A. Balanis, *Antenna Theory: Analysis and Design*, 4th ed. Hoboken, NJ, USA: Wiley, 2016.

[2] R. J. Mailloux, *Phased Array Antenna Handbook*, 2nd ed. Norwood, MA, USA: Artech House, 2005.

[3] R. C. Hansen, *Phased Array Antennas*. Hoboken, NJ, USA: Wiley, 2009.

[4] B. D. Van Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE ASSP Mag.*, vol. 5, no. 2, pp. 4–24, Apr. 1988.

[5] S. Venkatesan, "Beamforming techniques for large arrays," *IEEE Signal Process. Mag.*, vol. 15, no. 1, pp. 30–60, Jan. 1998.

[6] S. Winder and J. Carr, *Newnes FPGA-Based Digital Design Handbook*. Oxford, U.K.: Elsevier, 2011.

[7] F. Marvasti, *Nonuniform Sampling: Theory and Practice*. New York, NY, USA: Springer, 2001.

[8] L. C. Godara, "Applications of antenna arrays to mobile communications—Part II: Beam-forming and direction-of-arrival considerations," *Proc. IEEE*, vol. 85, no. 8, pp. 1195–1245, Aug. 1997.

[9] J. Li and P. Stoica, *Robust Adaptive Beamforming*. Hoboken, NJ, USA: Wiley, 2006.

[10] R. A. Monzingo, R. L. Haupt, and T. W. Miller, *Introduction to Adaptive Arrays*. Edison, NJ, USA: SciTech Publishing, 2011.

[11] J. C. Liberti and T. S. Rappaport, *Smart Antennas for Wireless Communications: IS-95 and Third Generation CDMA Applications*. Upper Saddle River, NJ, USA: Prentice Hall, 1999.

[12] P. K. Dinesh *et al.*, "FPGA implementation of beamforming algorithm for smart antennas," *Procedia Comput. Sci.*, vol. 58, pp. 215–222, 2015.

[13] B. Kiziloz and T. Yucek, "Design and FPGA implementation of a digital beamforming architecture," *IEEE Access*, vol. 8, pp. 216119–216128, 2020.

[14] M. Molina-Garcia *et al.*, "Real-time beamforming on FPGA for phased array antennas," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 3, pp. 1253–1262, Jun. 2017.

[15] T. A. Pham *et al.*, "Low-complexity FPGA-based digital beamforming processor for radar applications," *Sensors*, vol. 21, no. 3, p. 736, 2021.

[16] H. P. Ku *et al.*, "Low-latency digital beamforming on FPGA with reduced complexity," *IEEE Microw. Wireless Compon. Lett.*, vol. 28, no. 10, pp. 924–926, Oct. 2018.

[17] C. A. Powell *et al.*, "High-fidelity modeling of phased arrays using MATLAB and Simulink," *IEEE Antennas Propag. Mag.*, vol. 60, no. 4, pp. 52–64, Aug. 2018.

[18] Q. H. Nguyen *et al.*, "A scalable beamforming architecture for massive MIMO on FPGA," *IEEE Trans. Circuits Syst. I*, vol. 67, no. 12, pp. 4477–4487, Dec. 2020.

[19] H. Kwon *et al.*, "FPGA implementation of beam steering for millimeter-wave 5G systems," *IEEE Commun. Lett.*, vol. 23, no. 10, pp. 1809–1812, Oct. 2019.

[20] S. Natarajan *et al.*, "Time delay beamforming on reconfigurable platforms," *J. Syst. Archit.*, vol. 93, pp. 20–32, 2019.

[21] S. Tekin and M. Akhan, "Efficient FPGA-based implementation of a hybrid analog–digital beamforming system," *Analog Integr. Circuits Signal Process.*, vol. 100, pp. 237–246, 2019.

[22] A. Erdemir and A. E. Cetin, "3D beamforming in planar antennas using FPGA hardware co-simulation," *Int. J. Electron. Commun. (AEÜ)*, vol. 129, p. 153477, 2021.

[23] Y. Zhou and T. S. Durrani, "Design of adaptive beamforming algorithms for FPGA implementation," *IET Signal Process.*, vol. 11, no. 6, pp. 735–741, Aug. 2017.

[24] C. J. Wei *et al.*, "Digital phased array radar: System design and FPGA-based implementation," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 29, no. 10, pp. 22–31, Oct. 2014.

[25] S. Park and J. G. Yook, "Wide-angle beam steering antenna using digital delay lines," *IEEE Trans. Antennas Propag.*, vol. 65, no. 8, pp. 4031–4039, Aug. 2017.

[26] A. Sharma *et al.*, "Design of smart antenna for beam steering using hybrid optimization algorithm," *Wireless Pers. Commun.*, vol. 122, no. 1, pp. 139–155, 2022.

[27] S. Rathi *et al.*, "Beamforming on FPGA using Simulink HDL coder and MATLAB: A case study," in *Proc. IEEE INDICON*, 2020.

[28] H. Ye *et al.*, "Hybrid beamforming for 5G millimeter-wave massive MIMO systems," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 34–41, Mar. 2018.

ISSN 2347–3657
Volume 14, Issue 1, 2026

[29] A. Tervo *et al.*, "Digital and hybrid beamforming techniques for 5G and beyond," *IEEE Trans. Antennas Propag.*, v ol. 69, no. 5, pp. 2941–2954, May 2021.
[30] W. Liu and S. Weiss, *Wideband Beamforming: Concepts and Techniques*. Hoboken, NJ, USA: Wiley, 2010.