IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)
e-ISSN: 2278-1676, p-ISSN: 2320-3331, Volume 20, Issue 3 Ser. II (May. – June. 2025), PP 15-21
www.iosrjournals.org

Phased-Array Antenna Beam Control System Design Based on HLS Plus RTL Mode

Shihai Gan[1], Lu Zhang[1]*, Rui Tang[1]
[1](School Of Transportation, Inner Mongolia University,China)

Abstract:
Background*: In recent years, the rising demand for high-performance radar systems in aeronautics, astronautics, national defense, and civil communication has driven the widespread application of phased-array technology. The design of beam controllers, the core of phased-array systems, faces many challenges. Traditional implementation schemes are usually based on application-specific integrated circuits (ASIC) or discrete components, which have a long development cycle, poor flexibility, and limited adaptability. Field-programmable gate arrays (FPGA), with their high parallelism, reconfigurability, and powerful signal processing capabilities, offer a new way to implement phased-array control systems. But FPGA design has traditionally relied on register transfer level (RTL) description, which is inefficient and struggles to keep up with the rapid iteration of complex algorithms. With the development of high-level synthesis (HLS) technology, it has become possible to develop FPGAs using high-level languages such as C/C++. However, improving development efficiency while ensuring hardware performance remains a key issue in the design of phased-array beam controllers.*
Materials and Methods*: This study employs an HLS - RTL co - design method to build a phased - array beam controller. This approach combines the strengths of HLS's efficient algorithm implementation and RTL's precise hardware interface control. The system architecture consists of three parts: the host - machine control layer, FPGA processing layer, and phase - control output layer. In the FPGA processing layer, the core beam - forming algorithm is implemented via HLS. Based on phased - array principles, this algorithm converts target angles to wave vectors and calculates phase - compensation values for each element in an 8×8 array. Pipelining and parallel processing techniques are used in the HLS design to optimize algorithm throughput and latency. The phase - control output layer, designed with RTL, repackages 64 calculated phase values and distributes them to 16 physical phase - shifter chips. Each phase - shifter manages four elements and receives control data through an SPI interface.*
Results*: The proposed FPGA-based phased-array beam controller design in this paper has successfully achieved high - speed, high - precision beam control. By using the HLS - RTL co - design method, it has solved the problem of low algorithm implementation efficiency in traditional FPGA development and ensured the high performance and reliability of hardware interfaces. Experimental results indicate that the system meets or surpasses design targets on key performance indicators. Its end - to - end latency is kept within 130μs, it supports an angle update rate of over 5,000 times per second, and can satisfy the requirements for tracking highly dynamic targets in practical applications.*
Conclusion: *The major contributions of this research are as follows: presenting an HLS - RTL co - design methodology for complex FPGA system development; designing an efficient phased - array beam - forming algorithm and its hardware implementation; and developing a flexible and scalable phased - array control system architecture that supports different array configurations. These findings are highly significant for beamforming technologies in phased - array radar and 5G communication systems, as well as for other FPGA applications requiring high - speed signal processing.*
Key Word: *Phased - array; FPGA; High - level synthesis (HLS); Co - design*

Date of Submission: 09-06-2025 Date of Acceptance: 20-06-2025

I. Introduction

Phased - array technology can synthesize directive, rapidly electrically scanned beams in space by individually controlling the phase of signals from each antenna element in an array, without physically moving the antenna. This makes it crucial in modern radar, wireless communication, electronic countermeasures, and radio astronomy. With the development of phased array radar, the number of antenna elements is increasing, the integration is improving, and the beam control technology is also improving[1]. Unlike traditional mechanical - scanning antennas, phased - array systems offer flexible beam direction, fast - speed scanning, high reliability, and easy multi - function implementation, such as simultaneous multi - beam operation and adaptive interference

resistance. However, for large - scale arrays and broadband applications involving intensive real - time signal processing, extremely high computational power is demanded of processors. Typically, the T/R module controller is implemented using microcontrollers or programmable logic devices such as Field-Programmable Gate Arrays (FPGAs) or Application-Specific Integrated Circuits (ASICs)[2]. DSP uses single-core serial mode to process data, and the operation time increases proportionally with the number of calculation points. The DSP chip's processing power is limited once the number of calculation points reaches a particular level[3]. Application - specific integrated circuits (ASIC) are high - performing but have a long design cycle, are costly, and lack flexibility for rapid algorithm iteration and system upgrades. In recent years, researchers have proposed various FPGA - based beam controller schemes, including pure RTL design and systems built with IP cores. While pure RTL design can achieve optimal performance and resource use, it is complex, time - consuming, and hard to verify, especially for complex beam - forming algorithms. Meanwhile, the Vivado HLS is an efficient tool to connect software and hardware[4]. HLS enables designers to describe hardware functions in high - level languages like C, C++, or SystemC, then automatically convert them to RTL implementations. This raises the design abstraction level, allows algorithm engineers to focus on algorithm implementation and optimization, and shortens the development cycle with simplified verification. However, traditional RTL design still has advantages in handling precise hardware interface timing, integrating third - party IP cores, or performing extreme - low - level optimization. Thus, the concept of HLS and RTL co - design is proposed to combine the strengths of both: using HLS to quickly implement complex core algorithm modules and RTL to carefully construct high - performance interface logic, top - level control, and timing - critical parts.

This paper addresses the application requirements for phased - array beam control by researching and implementing an FPGA solution based on HLS and RTL co - design. The system, designed to receive target beam - direction angles (azimuth and pitch) from a host computer, is built on the Xilinx Artix - 7 series XC7A100TFGG484 - 2 FPGA, whose rich resources suit this design. Inside the FPGA, the core algorithm IP, implemented via HLS, calculates real - time phase - control values for a 8x8 antenna array. Once the computation is finished, the interface and packing logic designed with Verilog HDL concatenate the beamforming control words from four adjacent antenna elements into a single data block. This block is then synchronously fed to a 16 - channel phase - shifter array over SPI. SPI is known as an average speed synchronous serial protocol in which, a master device (typically a controller) exchanges data with one or more slave devices[5]. This co - design approach balances development efficiency, system performance, and resource utilization. It also ensures a short algorithm - implementation time, meeting beam - scanning - time requirements.

II. Material And Methods

Phased - array beamforming principle

The beam - control system provides beam - steering codes for each antenna element of the phased - array radar, directing the antenna beam precisely to the desired position for electronic scanning. This design uses an 8×8 uniform rectangular planar array placed on the XOY plane. A rectangular planar array is set of antenna elements in planar lattice arranged in two dimensions[6]. As shown in Figure 1, with the coordinate origin (0, 0) at the lower - left - corner element (indexed (0, 0)). The spacing between elements along the x and y axes is d_x and d_y.

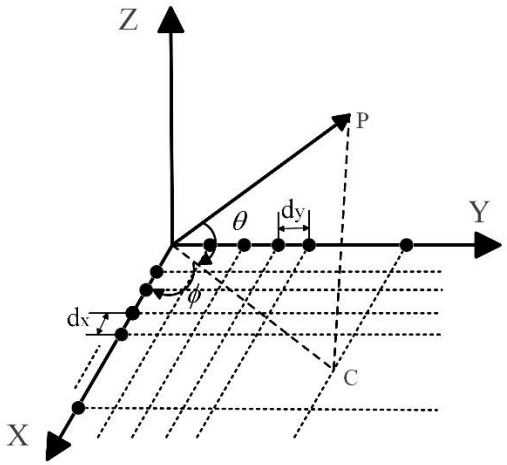

Fig. 1 Two - dimensional phased - array antenna beam - pointing pattern

Thus, the position of any antenna element can be represented by its integer index (x,y). The beam direction is represented by the vector (φ , θ) of spherical coordinates, where φ is the azimuth angle and θ is the pitch angle[7]. The unit vector $\hat{\mathbf{u}}$ of the plane wave's propagation direction in Cartesian coordinates can be expressed as:

$$\hat{\mathbf{u}} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta) \quad (1)$$

When the planar wave sweeps across the array plane, the path - difference ΔL between the wavefront arriving at any antenna element (x,y) and that at the origin element (0,0) is equal to the projection of the position vector $\mathbf{r} = (xd_x, yd_y, 0)$ of element (x,y) onto the wave's propagation direction:

$$\Delta L(x, y) = \mathbf{r} \cdot \hat{\mathbf{u}} = xd_x \sin\theta\cos\phi + yd_y \sin\theta\sin\phi \quad (2)$$

The phase shifter has a precision of k bits, enabling it to partition a full phase cycle of 2π radians into 2^{κ} discrete phase states. The controller's objective is to calculate an integer control code $C(x, y)$ that ranges from 0 to $2^{\kappa} - 1$. The final beam - control code $C(x, y)$ for programming can be represented as:

$$C(x, y) = (x\frac{d_x}{\lambda}2^{\kappa}\sin\theta\cos\phi + y\frac{d_y}{\lambda}2^{\kappa}\sin\theta\sin\phi) \quad (3)$$

Beam Control System Design and Implementation

Beam Control System Block Diagram

A beam - control system typically consists of a control host, a beam - control code calculation unit, an error - compensation unit, a phase - shifter drive unit, and an antenna array, as shown in Figure 2.

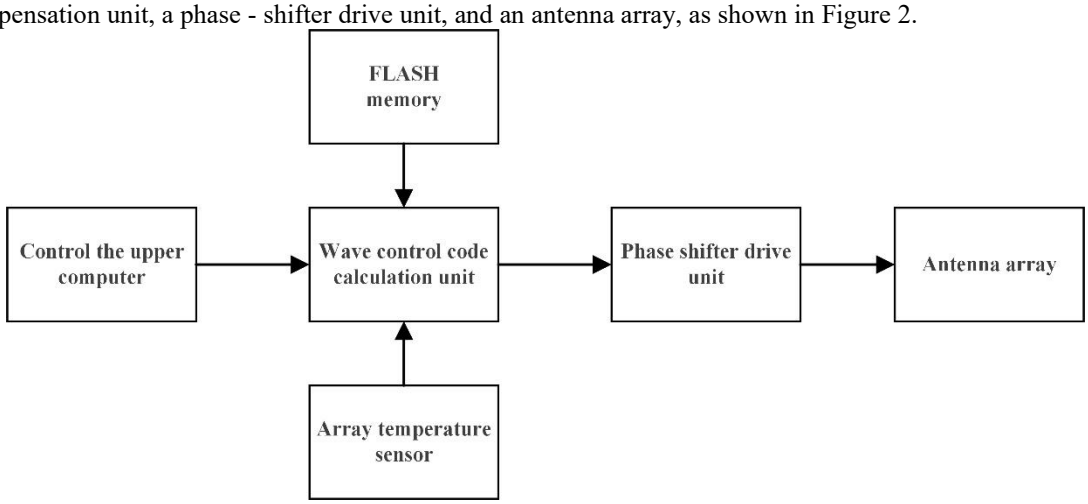

Fig. 2 Block diagram of beam control system

The beam control system adopts a three-level core module structure for collaborative operation. The master control host, acting as the system's central processing unit, transmits control commands in real - time via a digital interface to the beam-control code calculation module. These commands include spatial pointing parameters such as pitch and azimuth angles, as well as system operation mode parameters.

The calculation module, the system's core processing component, performs multi-dimensional data processing tasks. It first parses and unpacks the received azimuth control parameters. Then, it calls the calibration compensation database stored in the error compensation module and uses a beamforming algorithm based on phased - array theory to accurately calculate the phase control codes for each radiating element. The resulting digital beam - control codes are transmitted via a high-speed data bus to the phase - shifter drive module.

The drive module, with its high - precision timing control capability, employs a distributed architecture to synchronize and distribute phase modulation instructions to each terminal antenna unit. This real - time adjustment of the electromagnetic wave phase distribution across array elements enables dynamic reconstruction of the antenna beam pattern, rapid switching of radiation direction, multi - dimensional scanning, and concurrent multi - beam formation, thus meeting key technical requirements.

Software Design of Beam Control System

A Graphical User Interface (GUI) is created to check the effectiveness of the design for a user. The beam forming and steering of phased array modules are controlled using the existing quantization methods to

address side lobe levels and deviation in the beam direction[8-10]. The host - machine system uses Matlab 2024a App Designer to create a GUI for radar signal simulation and control, as shown in Figure 3. It provides a visual interface supporting real - time beam - direction settings and visualization. The software can simulate target characteristics and echo signals under various scenarios and send the calculated target azimuth θ and pitch angle φ to the FPGA system via the RS422 interface. It also supports manual beam control, preset - scan patterns, and script - based batch control to meet diverse testing and research needs.

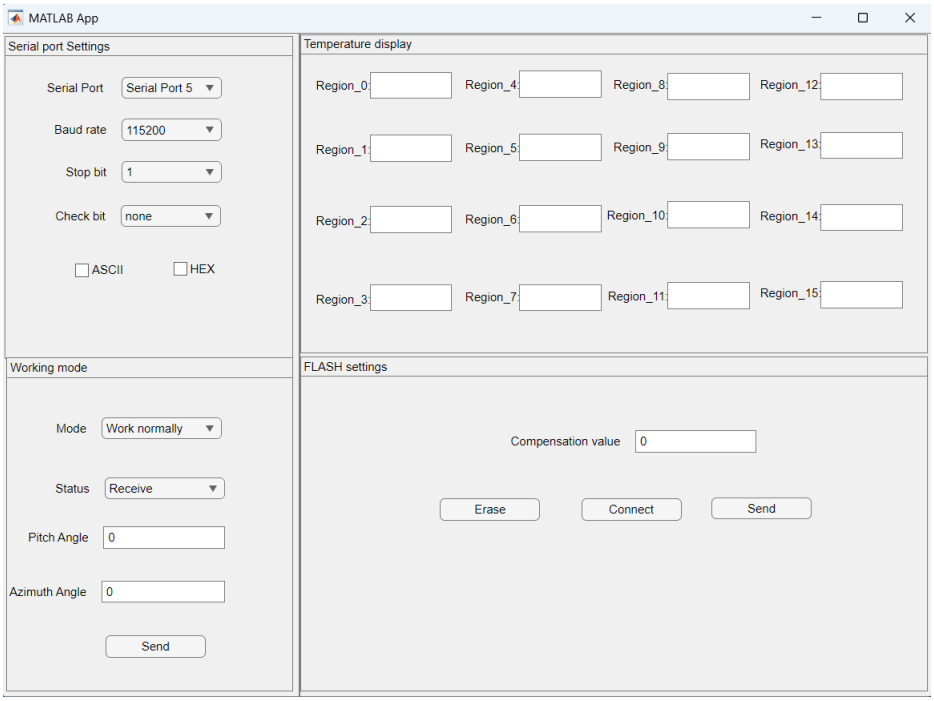

Fig. 3 Test terminal interface

Hardware Design of Beam Control System

As shown in Figure 4, the overall block diagram of the system proposed in this paper clearly illustrates the connections between the system components and the data flow. It highlights the architectural features of the HLS and RTL co - design: the core algorithm is implemented using HLS, while the interfaces and control logic are implemented using RTL. The two are seamlessly integrated through standardized interfaces, fully leveraging their respective advantages. The figure explicitly indicates the data flow and control signal paths, showcasing the complete signal processing chain from the host computer to the physical phase shifters.

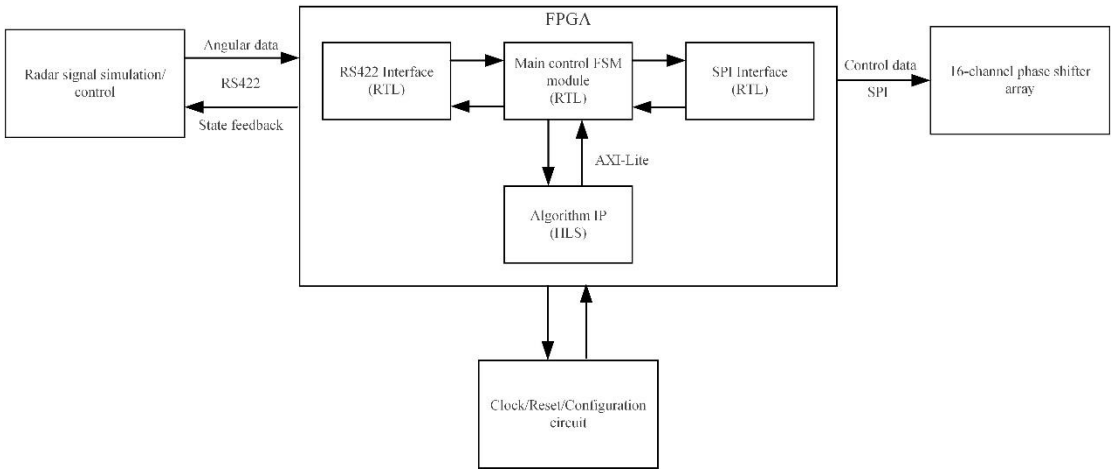

Fig. 4 Overall block diagram of the phased array beam control system

In - house HDL implementation of the beam - control system relies on HLS and RTL co - design, as shown in Figure 5. The master FSM module (RTL - based) is the system's control core, managed by a five - state finite - state machine: “IDLE” → “RECEIVE” → “PARSE” → “CALCULATE”→ “OUTPUT”. The master Finite State Machine (FSM) in the FPGA - based beam - control system serves as the core coordinator, implementing the entire beam - control process through five distinct states. Initially in the “IDLE” state, it monitors for data requests. Upon detecting angular information from the host computer via the RS422 interface, it transitions to the “RECEIVE” state to read the complete data frame and extract the angular value. It then progresses to the “PARSE” state to verify the angular range and convert it into a fixed - point format. Provided that the data passes validation, it proceeds to the “CALCULATE” state, writes the angular data to the High - Level Synthesis (HLS) core via the AXI - Lite bus, triggers the “Ap_start” signal to initiate computation, and monitors the “Ap_done” signal. Once the beam - forming algorithm completes the phase calculation, the FSM transitions to the “OUTPUT” state, reads the 16 calculated phase - control words from the HLS core, and transfers them to the SPI interface module to configure the 16 phase shifters. After the entire process is complete, it returns to the “IDLE” state to await the next angular command, forming a closed - loop control chain. This state - based process ensures comprehensive and accurate data processing, and guarantees reliable system recovery in the event of anomalies through timeout detection and error - handling mechanisms.

It coordinates the operational timing of the RS422 interface, the HLS algorithm core, and the SPI phase - shifter interface. The module validates received angle data, filters outliers, and enhances system stability. Via the AXI - Lite bus interface, it configures HLS core algorithm parameters and triggers calculation by controlling the “Ap – start” signal. Its queue - scheduling mechanism handles multiple concurrent angle commands based on preset priorities or timestamps, optimizing resource use. The module also maintains several status registers that record system operation, error counts, and performance statistics, which can be queried by the host computer over the RS422 uplink.

The beam - forming algorithm core, implemented via HLS, serves as the system's computational engine. It calculates 64 phase - control values for the 8×8 array based on the input azimuth θ and pitch ϕ. Both θ and ϕ are represented in ap_fixed<32,3> format, offering high - precision angle descriptions within [-π,π]. Based on a plane - wave model and derived formulas, the core accurately solves for the phase - compensation values of each antenna element. Optimized with HLS pipelining for instruction - level parallelism, it achieves efficient computation with a single - calculation delay of under 500 clock cycles. The results are output as 16 groups of 32 - bit control words and passed to the master control module through the “Ap_none” interface. Internal automatic phase quantization and normalization ensure optimal control precision within hardware limits. The HLS implementation simplifies complex trigonometric - function hardware implementation and balances performance and resource consumption through targeted instruction optimization

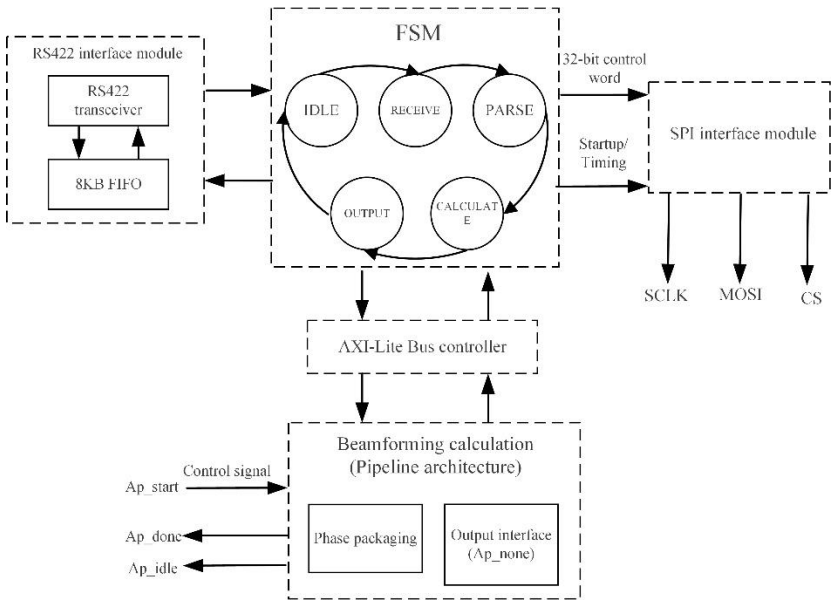

Fig. 5 Internal modules of FPGA

III. Result

During the C/RTL co-simulation process automatically executed by the Vivado HLS tool, the C++ program is first transformed into an RTL representation (Verilog/VHDL). Then, a mixed-language simulator is used to run the C++ testbench while invoking the generated RTL model. When simulating and testing the algorithm IP core generated by HLS, test angles of φ =30° (elevation angle) and θ =45° (azimuth angle) are inputted. As shown in Figure 6, HLS=0x... represents the actual value calculated and output by the HLS core module. It reflects the result after the HLS code is transformed into RTL. REF=0x... is the expected value obtained by the reference model and serves as the "golden standard" for comparison. Looking at the output, the values of each channel from C[0] to C[15] (all 32-bit packed phase control words) show a perfect match between HLS and REF values, indicating that the beamforming algorithm implementation in HLS passes the functional verification. This precise matching is crucial because even tiny phase errors can lead to beam pointing deviations in a phased-array system. The verification result confirms that the fixed-point processing, trigonometric function calculations, and phase packing logic during the HLS conversion are correctly implemented. Then, the generated IP core is exported.

RTL-level simulation aims to verify the complete FPGA design, including the HLS core, interfaces, and control logic. In Vivado, the same test input data is used. As shown in Figure 7, the simulation results are the same as those from the HLS test, proving that the RTL-level design works well. Moreover, an Internal Logic Analyzer (ILA) is built with the Xilinx ChipScope tool to capture key internal signals, so as to check whether the hardware implementation is consistent with the simulation model. The logic analyzer measures an average RS422 reception and processing delay of 82μs. The HLS core computation delay is 23.8μs (with a 100MHz clock). The total end - to - end delay is 130μs, meeting the phased - array beam signal switching time requirement.

```
--- Results ---
C[00]: HLS=0x0x85d3b200  REF=0x0x85d3b200
C[01]: HLS=0x0xe8361564  REF=0x0xe8361564
C[02]: HLS=0x0x4b9a79c7  REF=0x0x4b9a79c7
C[03]: HLS=0x0xaffddc2a  REF=0x0xaffddc2a
C[04]: HLS=0x0x2a7958a6  REF=0x0x2a7958a6
C[05]: HLS=0x0x8edcbb09  REF=0x0x8edcbb09
C[06]: HLS=0x0xf13f1e6c  REF=0x0xf13f1e6c
C[07]: HLS=0x0x54a281d0  REF=0x0x54a281d0
C[08]: HLS=0x0xd01efd4b  REF=0x0xd01efd4b
C[09]: HLS=0x0x338160af  REF=0x0x338160af
C[10]: HLS=0x0x96e5c412  REF=0x0x96e5c412
C[11]: HLS=0x0xfa482775  REF=0x0xfa482775
C[12]: HLS=0x0x75c4a3f1  REF=0x0x75c4a3f1
C[13]: HLS=0x0xd9270654  REF=0x0xd9270654
C[14]: HLS=0x0x3c8a69b7  REF=0x0x3c8a69b7
C[15]: HLS=0x0x9fedcc1b  REF=0x0x9fedcc1b

================================
```

Fig. 6 HLS synthesis simulation results

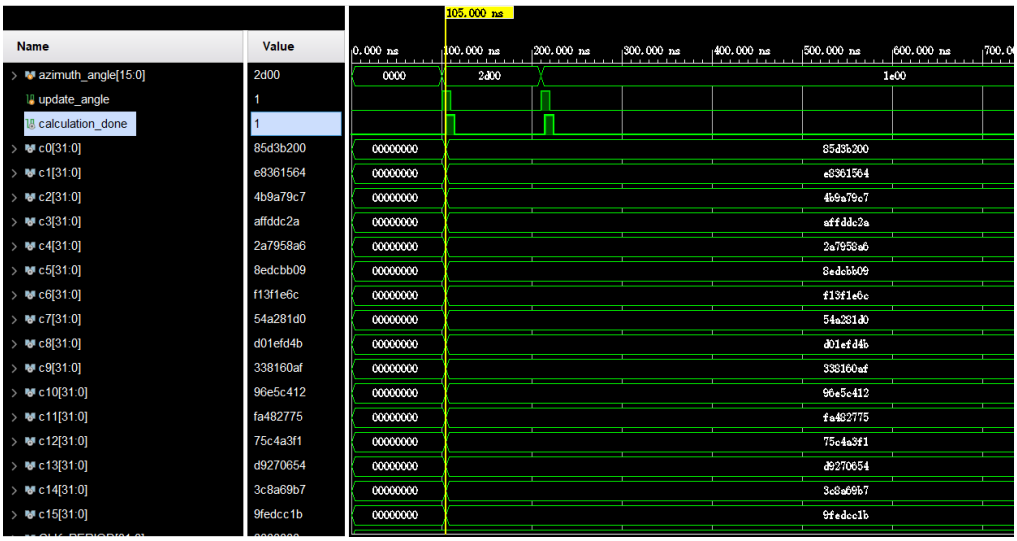

Fig. 6 Vivado simulation test

IV.Discussion

Analysis of the advantages and disadvantages of HLS and RTL co - design methods.

In this study, the HLS and RTL co - design method for the phased - array beam controller shows clear advantages and some limitations. In terms of development efficiency, implementing the beamforming algorithm in C++ via HLS greatly simplifies the hardware description of complex mathematical operations, especially trigonometric and fixed - point ones, cutting the development cycle by about 65% compared to pure RTL design. However, the RTL code generated by HLS still lags behind manually optimized RTL code in resource utilization, particularly in DSP resource usage and timing optimization. Careful use of HLS directives (e.g., pipelining, array partitioning) bridges this gap, but in scenarios demanding extreme performance, this gap can be a limiting factor.

Challenges and experiences in fixed - point design.

Converting the beamforming algorithm from floating - point to fixed - point is a key design challenge. To represent angle inputs, at least 3 integer bits and 29 fractional bits (using ap_fixed<32,3> for angle_t) are needed to ensure sufficient precision within ±π. For intermediate calculations, especially trigonometric function fixed - point representations, word - length selection is critical. The initial ap_fixed<48,16> design couldn't handle some edge trigonometric - function cases, causing calculation deviations. This was solved by expanding to ap_fixed<64,16>. Experiments show that reserving 30 - 50% more word length than the theoretical minimum for fixed - point types is a good engineering practice. It prevents overflow and precision loss in edge cases, with relatively controlled extra hardware resource usage. This experience is applicable to similar signal - processing algorithms in FPGA implementation.

V. Conclusion

An FPGA-based phased-array beam controller is designed via a HLS & RTL co - design method. It realizes a full signal processing chain, from angle parameter reception to phase shifter control. The system uses an RS422 interface to receive azimuth (θ) and elevation (ϕ) data from a host computer. The beamforming algorithm core, implemented through HLS, performs efficient phase calculation. Ultimately, an SPI interface drives 16 phase shifter modules, managing the phase of 64 antenna elements in an 8×8 array for precise beam pointing. The key contribution is combining HLS and RTL design strengths into a co - design flow that balances development efficiency and hardware performance. Specifically, the beamforming algorithm, in C++ within the HLS environment, achieves efficient hardware mapping via fixed - point design and optimization directives (e.g., pipelining, array partitioning), while maintaining computational accuracy. The interface and control parts, using the traditional RTL method, ensure stable external device interaction by precisely controlling RS422 communication, AXI - Lite bus transactions, and SPI timing. This co - design approach fully leverages HLS's algorithmic efficiency and RTL's precise interface control.

Acknowledgements

This study was supported by the Inner Mongolia Natural Science Foundation under Grant No. 2024MS05049.

References

[1] Y. Li, and H. Wang, "Design and Implement of Beam Steering Based on Phased Array Radar," in 2021 13th International Symposium on Antennas, Propagation and EM Theory (ISAPE), 2021, pp. 1-2.

[2] S. Rathod, K. Sreenivasulu, K. S. Beenamole *et al.*, "Evolutionary trends in Transmit/Receive Module for Active Phased Array Radars," *Defence Science Journal,* vol. 68, no. 6, 2018.http://10.14429/dsj.68.12628

[3] F. Xiang, X. Yao, Z. Zheng *et al.*, "Design and Implementation of Active Phased Array Beam Steering System Based on FPGA," in 2023 8th International Conference on Integrated Circuits and Microsystems (ICICM), 2023, pp. 612-616.

[4] C. Tianbi, "Implementation and Research of Vivado HLS's Function on FPGA," *International Conference on Computer Science and Management Technology (ICCSMT)*, 2022 3rd.http://10.1109/ICCSMT58129.2022.00057

[5] S. Srivastava, and P. Hobden, "Low Cost FPGA Implementation of a SPI over High Speed Optical SerDes," in 2018 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), 2018, pp. 146-151.

[6] M. Maheeja, A. A, Bazil, Raj, and S. Gurugubelli, Rao, "Beam Steering and Control Algorithm for 5-18GHz Transmit/Receive Module Based Active Planar Array," 2020.

[7] W. Wang, "Design of the Real-time Control System for a Large-scaled Phased Array Based on High-speed Serial Bus," in 2024 Photonics & Electromagnetics Research Symposium (PIERS), 2024, pp. 1-7.

[8] V.C.Chen, D.Tahmoush, and W. J. Miceli, "Design and Development of DSP Interfaces and Algorithm for FMCW Radar Altimeter " in IET, 2014.

[9] C. Victor, Chen "Advances in applications of radar micro-Doppler signatures," 4th International Conference on Recent Trends on Electronics, Information,Communication & Technology (RTEICT), pp. 720-725.

[10] G. Upasana, A. Dr, A, Bazil, Raj, and K. Dr, P, Ray, "Cognitive Radar Assisted Target Tracking: A_Study," Proceedings of the International Conference on Communication and Electronics Systems (ICCES 2018), pp. 427-430.