A CORDIC-Based Coprocessor for Accelerating Pose Estimation

Zhaoyu Lu Jiaying Yong Fengze Yu

October 8, 2025

Abstract

This project proposes a CORDIC-based trigonometric coprocessor integrated into a RISC-V System-on-Chip (SoC) to accelerate mathematical operations critical for real-time attitude estimation and control. Modern embedded systems, such as nano-satellites and robotics platforms, rely heavily on trigonometric and quaternion computations that are computationally expensive and energy-intensive when executed in software. The proposed design introduces a lightweight, memory-mapped accelerator accessed through control and status registers (CSRs), eliminating the need for custom ISA extensions while ensuring seamless integration into standard RISC-V toolchains.

The SoC architecture is built around a PicoRV32 CPU, a Pose Estimation Unit (PEU) implementing sine, cosine, and arctangent functions, a DMA controller for efficient data movement, and on-chip SRAM connected through a standard crossbar. The hardware offload of trigonometric operations provides deterministic latency and low power, improving control loop responsiveness in real-time systems.

Evaluation will focus on comparing latency and energy efficiency between software-only and hardware-accelerated implementations. Preliminary estimates indicate that the proposed PEU achieves an order-of-magnitude speedup while maintaining a compact area footprint. Overall, this work presents an energy-efficient, modular, and reusable hardware solution for high-precision embedded computation.

1 Introduction

The proliferation of autonomous systems such as drones, robotics, and augmented reality (AR) devices has led to a substantial and growing demand for efficient, real-time pose estimation. Pose estimation is the computationally intensive process of determining an object's orientation and position in 3D space, a foundational requirement for these technologies. From a societal perspective, the performance of these algorithms is critical; for instance, low-latency head tracking in AR/VR is essential for user immersion and preventing motion sickness, which directly impacts applications in remote education, surgical training, and collaborative design. Similarly, the reliability of pose estimation in robotics is fundamental to safe navigation in warehouses, hospitals, and public spaces.

The computational load of these tasks is immense. Real-time systems, such as visual-inertial odometry for AR, demand that the entire tracking pipeline execute in under 33 milliseconds to maintain a fluid 30 frames per second. The core of this pipeline involves thousands of mathematical updates per second to represent and manipulate rotations. These rotations are typically handled using either 3x3 rotation matrices or quaternions. While both achieve the same goal, their computational costs differ significantly: composing two rotations using matrix multiplication requires 27 multiplications and 18 additions, whereas using quaternion multiplication requires a more efficient 16 multiplications and 12 additions. Even with the more efficient quaternion approach, the sheer volume of these operations makes this a prime target for hardware acceleration.

Current solutions for handling this workload exhibit significant shortcomings. Software-based solutions, which utilize libraries like Eigen or ROS on general-purpose CPUs, offer maximum flexibility. However, their execution on embedded processors leads to high latency and significant power consumption, making them fundamentally unsuitable for applications with tight power and performance budgets. General-purpose hardware accelerators like GPUs, while powerful, are notoriously power-hungry. Furthermore, their programming model is often not optimal for the fine-grained, low-latency tasks required in the control loops for pose determination. Field-Programmable Gate Arrays (FPGAs) offer more customization than GPUs but come with a longer design cycle and typically consume more power than an Application-Specific Integrated Circuit (ASIC) designed for the same fixed task.

Hence, there is a growing need for a specialized yet lightweight hardware solution that can provide deterministic, energy-efficient computation for trigonometric operations. Our project aims to address this gap by developing a CORDIC-based coprocessor integrated within a RISC-V SoC, providing real-time acceleration for sine, cosine, and arctangent functions. By leveraging RISC-V's open architecture and the CORDIC algorithm's shift-add arithmetic efficiency, this design enables high-precision computation with minimal hardware cost—a crucial requirement for next-generation nanosatellite missions and other embedded control applications such as robotics or UAVs.

This work contributes not only to the advancement of satellite onboard computing but also to the broader trend of domain-specific, open-source hardware. Through an efficient CORDIC accelerator that can be easily interfaced via standard CSR access, we aim to provide a reusable hardware foundation that balances precision, real-time determinism, and low power consumption, addressing the pressing computational challenges of modern space and embedded systems.

2 Proposed Approach

To address the computational bottlenecks in high-precision satellite attitude control, our proposed solution is a CORDIC-based trigonometric coprocessor integrated into a RISC-V SoC, accessed entirely through control and status registers (CSRs) rather than custom ISA extensions. This design avoids the complexity of modifying compilers or toolchains, while maintaining a clean software interface compatible with standard RISC-V development flows. The coprocessor is responsible for accelerating the three most computation-intensive mathematical primitives in ADCS control loops — sine, cosine, and arctangent — which together form the core of attitude determination and sensor fusion algorithms.

The proposed approach introduces a lightweight, deterministic, and reusable hardware accelerator that performs trigonometric operations with predictable latency and minimal power. Instead of relying on software libraries or general-purpose accelerators, our design provides a dedicated hardware datapath that can complete each operation in a fixed number of cycles, ensuring real-time responsiveness required for satellite control loops. By integrating this coprocessor as a peripheral connected to the on-chip bus, we maintain full modularity: the unit can be reused, replaced, or scaled independently of the CPU core. This architecture also aligns with the team's broader SoC framework, which includes a RISC-V core, DMA, and memory blocks interconnected via a standard crossbar.

From a hardware implementation perspective, the use of the CORDIC algorithm eliminates the need for multipliers and floating-point units. This shift-add approach is particularly efficient in ASIC implementations, achieving high energy efficiency and compact area utilization while maintaining the precision required for 10-arcsecond-level attitude control. The deterministic and low-latency characteristics of an ASIC implementation make it ideal for ADCS, where timing uncertainty directly translates to control instability.

The target audience of this design includes developers of CubeSats, nano-satellites, and embedded robotics systems who require reliable and low-power trigonometric computation for real-time sensor fusion or control. The coprocessor's CSR-based interface provides a simple software integration path — applications can trigger operations through standard register writes and obtain results without interrupting main CPU execution. This minimizes firmware overhead and makes the accelerator readily deployable in resource-constrained embedded platforms.

The major contributions of this approach are threefold:

1. **Practical integration strategy:** It provides a plug-and-play, CSR-accessible accelerator design that simplifies hardware-software co-design and reduces bring-up time.

2. **Extended trigonometric capability:** Beyond basic sine and cosine, our design introduces a hardware-accelerated arctangent (arctan2) operation, enhancing its coverage of essential ADCS and EKF algorithms. item **Energy-efficient and deterministic ASIC design:** By leveraging CORDIC's iterative

structure, the proposed accelerator achieves a predictable execution time, meeting both performance and power constraints of on-orbit systems.

In summary, this approach proposes a domain-specific yet modular hardware coprocessor that delivers the precision, determinism, and energy efficiency demanded by next-generation space and robotics applications, without complicating the software toolchain or SoC integration flow.

3 Related Work

The demand for efficient real-time pose estimation has grown substantially with the proliferation of autonomous systems such as drones, robotics, and augmented reality devices. Pose estimation algorithms are computationally intensive, relying heavily on trigonometric and quaternion mathematics to represent and manipulate orientation in 3D space [1]. In applications like AR/VR, for instance, low-latency head tracking is critical for user immersion, and the underlying sensor fusion algorithms heavily rely on quaternion and trigonometric operations to maintain a stable virtual environment [1, 7]. Current solutions for accelerating these workloads can be broadly categorized into software-based, general-purpose hardware, and specialized hardware approaches.

To justify the need for hardware acceleration, it is crucial to quantify the computational load of these core operations. Real-time systems like visual-inertial odometry for AR/VR demand that the entire tracking pipeline, including feature extraction, matching, and pose optimization, execute in under 33 milliseconds to maintain a fluid 30 frames per second [8]. The core of the pose optimization step involves composing and transforming rotations, which are typically represented by quaternions or 3x3 rotation matrices. While both can represent orientation, their computational costs differ significantly. Composing two rotations using quaternion multiplication requires 16 floating-point multiplications and 12 additions. In contrast, composing rotations using 3x3 matrix multiplication requires 27 multiplications and 18 additions [9]. In a high-frequency tracking loop that performs thousands of these updates per second for various features and state estimations, this delta in operation count makes the underlying arithmetic a prime target for hardware acceleration.

Software-based solutions running on general-purpose CPUs (e.g., ARM, x86, or RISC-V) offer the most flexibility. Libraries like Eigen or the Robot Operating System (ROS) provide robust tools for these calculations. However, executing these operations in software on an embedded processor can lead to high latency and significant power consumption, making them unsuitable for applications with tight power and performance budgets.

General-purpose hardware accelerators such as Graphics Processing Units (GPUs) or Field Programmable Gate Arrays (FPGAs), are often used to offload these tasks. While powerful, GPUs are notoriously power-hungry and their programming model is not always optimal for the fine-grained, low-latency tasks required in control loops for pose determination. FPGAs offer more customization but involve a longer design cycle and typically consume more power than an Application-Specific Integrated Circuit (ASIC) for the same task.

Specialized hardware acceleration offers the most promising path for performance and energy efficiency. The RISC-V architecture, with its open and extensible Instruction Set Architecture (ISA), provides a natural foundation for domain-specific acceleration. Several projects have explored extending RISC-V for various applications. For instance, the PicoRV32 by Clifford Wolf is a popular size-optimized RISC-V core that serves as an excellent baseline for custom SoC development due to its simplicity [2]. Research has also been conducted on creating hardware units for specific mathematical operations. The CORDIC (COordinate Rotation DIgital Computer) algorithm, for example, is a well-established method for implementing trigonometric functions in hardware using only simple shift-and-add operations, making it highly suitable for ASIC implementation [3]. This focus on accelerating mathematical primitives extends to quaternions as well. Existing work includes a proposed dual quaternion accelerator IP for RISC-V [10], a specialized VLSI multiplier based on a 4D CORDIC-based algorithm [11], and hardware-efficient schemes designed to reduce the number of multipliers for complex operations like the discrete quaternion Fourier transform [12]. Furthermore, recent work such as DROID-SLAM has demonstrated the use of RISC-V-based processing units to accelerate real-time SLAM applications, highlighting the trend towards specialized hardware in robotics [6].

Our proposed work fills a critical gap between flexible-but-slow software libraries and highly-coupled, complex custom function units within a CPU. We differentiate our approach by creating a dedicated, memory-mapped hardware accelerator for the mathematical primitives of pose estimation (`sin`, `cos`, and quaternion operations). This architectural choice is a strategic trade-off: it provides a dedicated, energy-efficient solution that offers a significant performance uplift over pure software execution, while deliberately avoiding the high design risk and inflexibility of modifying the CPU core itself. The key advantage of this peripheral-based design is its unparalleled **modularity and extensibility**. By implementing the PEU on a standard bus, we create a reusable IP block that is not tied to a specific processor pipeline. This modularity is critical because it allows for future hardware extensions to accelerate more complex, multi-step operations. For instance, common algorithmic patterns in pose prediction involve a fusion of matrix multiplication followed by trigonometric calculations; our architecture allows the PEU to be extended to offload this entire fused operation, a task ill-suited for a simple, single-instruction CPU function unit. Therefore, our contribution is not just a single-purpose accelerator, but a flexible and powerful open-source hardware foundation capable of evolving to accelerate a wide range of current and future spatial algorithms in the robotics and embedded systems community.

4 Design Constraints

We established the system's hard constraints using estimated parameters from prior designs and observed that these constraints remain relatively lenient compared to our target objectives.

Metric	Constraint	Optimization Target
Power Consumption (at 50MHz)		
Passive Mode	<4 mW	<2 mW
Active Mode	—	<500 μW
Area		
RISC-V CPU	<0.5 mm^2	≈ 0.2 mm^2
PEU	<1.0 mm^2	≈ 0.5 mm^2
SRAM	<0.2 mm^2	≈ 0.1 mm^2
Total SoC	2 mm $\times$ 2 mm = 4 mm^2	<4 mm^2

5 System-on-Chip Architecture

Our proposed System-on-Chip (SoC) is designed around a minimal 32-bit RISC-V core, augmented with a custom-designed **Pose Estimation Unit (PEU)**. The primary design goal is to create an energy-efficient architecture that can offload the most common and intensive calculations associated with pose determination from the main processor. To further enhance performance and offload the CPU, a **Direct Memory Access (DMA)** controller is included to manage high-throughput data transfers between memory and the PEU. A key aspect of our design philosophy is to ensure the PEU is both modular and extensible, allowing it to serve as a foundation for more complex algorithms in the future.

The table below details the key components of the architecture, their purpose, origin, and source format.

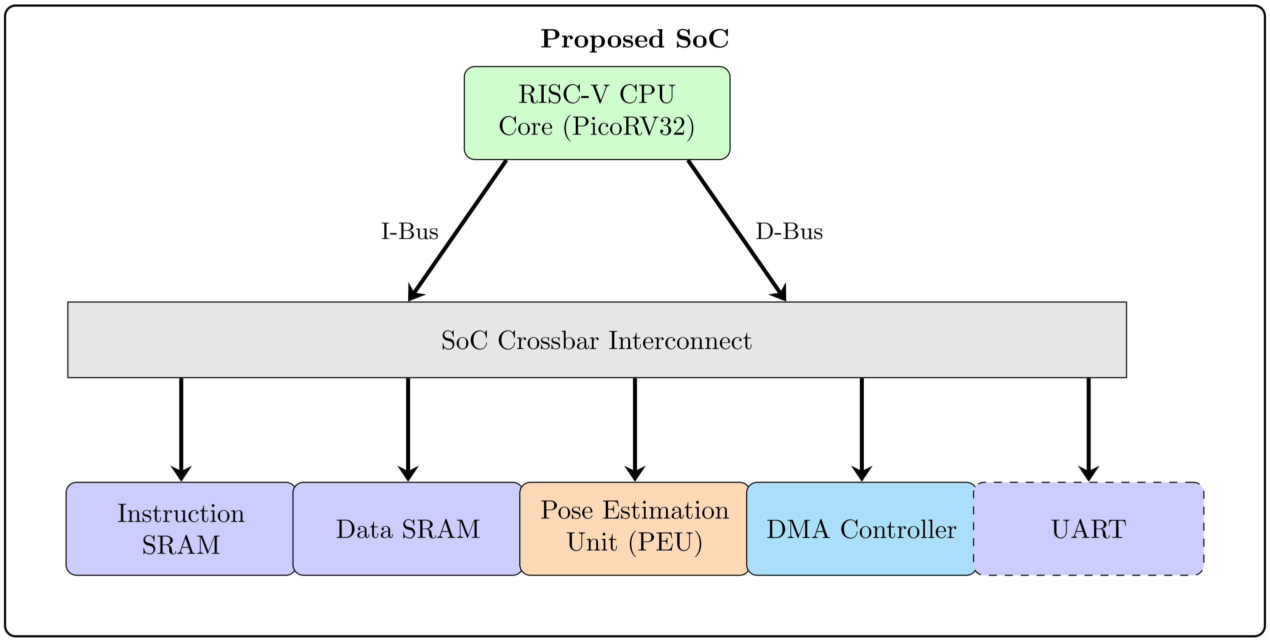
Proposed SoC
RISC-V CPU Core (PicoRV32)
I-Bus
D-Bus
SoC Crossbar Interconnect
Instruction SRAM
Data SRAM
Pose Estimation Unit (PEU)
DMA Controller
UART

Figure 1: Proposed SoC Block Diagram.

Component	**Purpose**	**Origin**	**Source Format**
RISC-V CPU Core	A single-core processor for general computation and control. Adheres to the RV32IMC ISA [4].	Open-source (PicoRV32 project [2])	Verilog RTL
Pose Estimation Unit (PEU)	The core contribution: a memory-mapped accelerator for trigonometric (`sin`, `cos`) and quaternion operations.	Developed by Team	Verilog RTL
DMA Controller	Offloads the CPU by handling bulk data transfers between SRAM and peripherals (e.g., the PEU) directly.	Developed by Team	Verilog RTL
On-Chip SRAM	Stores instructions and data for the CPU. Implemented as separate Instruction and Data SRAM blocks.	Developed by Team	Verilog RTL
SoC Crossbar Interconnect	Connects all SoC components using a standard on-chip bus protocol.	Developed by Team	Verilog RTL
UART	Standard serial interface for debugging, verification, and communication with a host computer.	Developed by Team	Verilog RTL

A significant aspect of our proposed architecture is the extensibility of the PEU. While the initial implementation will focus on the fundamental primitives of trigonometric and quaternion operations—which are broadly applicable in fields from robotics to cryptography and AI—the unit is designed to be a flexible platform. Future work could easily extend its capabilities to handle more complex, multi-step algorithms.

For instance, a common operation in AR/VR is predicting the user's next head position to reduce latency. This involves getting the current pose from an image, calculating the difference from the previous pose, and extrapolating the future pose. This entire sequence, which fuses matrix operations with trigonometric calculations, could be implemented as a single, high-throughput operation within the PEU. This capability is especially pertinent in AR/VR applications where continuous pose updates are necessary to project virtual content onto the real world from the user's perspective [7]. This modular design ensures that our SoC is not merely a single-function device, but a robust platform for accelerating a wider class of spatial and geometric algorithms.

6 Proposed Milestones

Our project is structured into two main phases. **Phase 1**, spanning from **October 8 to November 25**, focuses on developing a Minimum Viable Product (MVP). This includes designing, integrating, and verifying all core components of the SoC. **Phase 2**, which begins on **November 26**, is dedicated to **Extension Work**. If time permits, this phase will involve implementing more advanced algorithms in the accelerator, followed by final physical design and comprehensive benchmarking.

Development is organized into parallel streams to maximize productivity. Yu is responsible for the **RISC-V CPU**, Lu will develop the **Interconnect and SRAM**, and Yong will design the core **Accelerator (PEU)**. All three workstreams will commence on October 8. The entire team will collaborate on system integration and the final project deliverables.

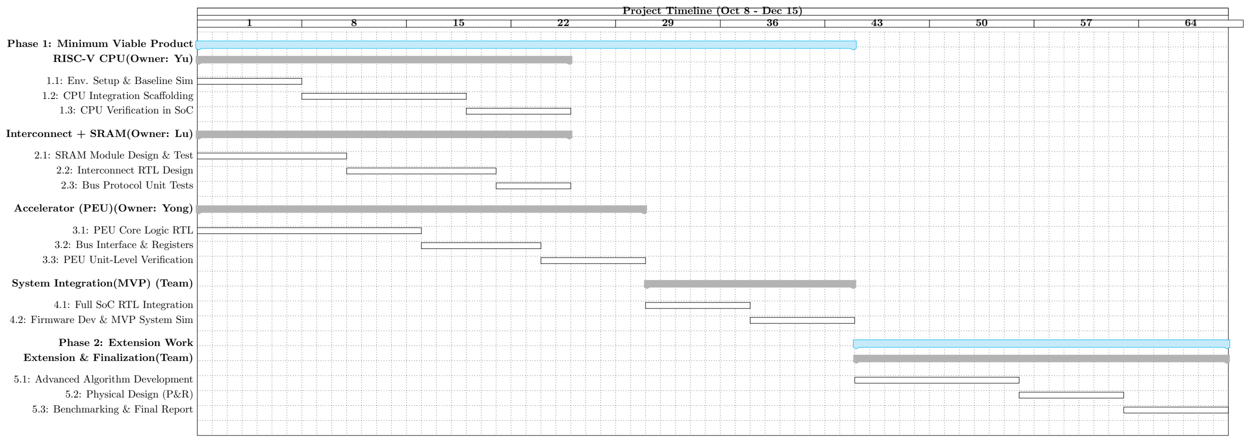

Figure 2: Detailed Gantt Chart of Proposed Milestones and Tasks.

The table below details the task assignments, deliverables, and timeline for each phase.

Major Task (Owner)	**Sub-tasks**	**Deliverable(s)**	**Timeline**
	Phase 1: Minimum Viable Product (MVP)		
RISC-V CPU (Yu)	1.1 Environment setup & baseline simulation 1.2 Create SoC top-level with CPU 1.3 Verify CPU with basic firmware	A working simulation of the standalone CPU, and synthesizable RTL for the CPU integrated in the SoC top-level.	Oct 8 - Nov 1
Interconnect + SRAM (Lu)	2.1 Design and test SRAM modules 2.2 Design crossbar interconnect RTL 2.3 Verify bus protocol and memory map	Verified Verilog modules for both SRAMs and the full interconnect subsystem.	Oct 8 - Nov 1
Accelerator (PEU) (Yong)	3.1 Design PEU core RTL for math ops 3.2 Add bus slave interface & registers 3.3 Perform unit-level verification	A verified, standalone PEU module with a complete test suite demonstrating all functions.	Oct 8 - Nov 6
System Integration (MVP) (Team: Yu, Lu, Yong)	4.1 Integrate all modules into final SoC 4.2 Develop firmware for MVP system tests	A fully integrated, verifiable SoC design demonstrating core functionality via system-level simulations.	Nov 7 - Nov 20
	Phase 2: Extension Work		
Extension & Finalization (Team)	5.1 Advanced Algorithm Development 5.2 Perform physical design (P&R) 5.3 Benchmark and write final report	(Stretch Goal) Enhanced PEU with new features. A clean GDSII layout, comprehensive performance analysis report, and final presentation.	Nov 21 - Dec 15

7 Verification

Module-level verification focuses on exhaustively testing each component of the SoC in isolation to ensure its functional correctness before system integration. For this project, a dedicated testbench will be created for the Pose Estimation Unit (PEU) to validate its trigonometric and quaternion operations against a pre-calculated model. Similarly, the DMA controller will be verified for its ability to correctly handle bulk data transfers between memory and peripherals without CPU intervention. Other modules, including the SRAM, UART, and the SoC crossbar interconnect, will also undergo standalone RTL simulations to confirm they meet their individual design specifications. This foundational step is critical for debugging issues early and simplifying the subsequent system integration process.

Following successful module-level validation, system-level verification aims to confirm that all integrated components function together correctly as a complete System-on-Chip. This phase is primarily driven by firmware, where C-language tests will be developed to simulate realistic operational scenarios. Key tests will involve the CPU configuring the PEU via its memory-mapped interface, offloading computation, and then reading back the results for validation. More complex verification scenarios will test the synergy between the DMA and PEU, ensuring the system can autonomously transfer data from memory, process it in the PEU, and write the results back. Finally, benchmarking tests will be conducted to run full algorithms and quantify the performance improvement of the PEU and DMA hardware acceleration compared to a purely software-based implementation.

Physical design verification ensures that the chip's final layout is manufacturable, electrically sound, and meets timing and performance requirements. The process begins after logic synthesis with Static Timing Analysis (STA), which checks for timing violations in the gate-level netlist without running dynamic simulations. After the layout is generated through place and route, a Design Rule Check (DRC) is performed to ensure the geometry of the layout conforms to the foundry's specific manufacturing rules. Concurrently, a Layout Versus Schematic (LVS) check compares the netlist extracted from the layout against the original synthesized schematic to guarantee electrical consistency. The final step often involves a post-layout simulation using parasitic data extracted from the layout, providing the most accurate assessment of the chip's real-world performance before tape-out.

8 Evaluation

The evaluation will be based on the metrics identified in the Design Constraints. The primary quantitative goals are to achieve an active mode power consumption under 500 μW at 50MHz and a passive mode consumption of 2 mW. Area targets are set at 0.5 mm^2 for the Pose Estimation Unit (PEU), 0.2 mm^2 for the RISC-V CPU, 0.1 mm^2 for the on-chip SRAM, and under 4 mm^2 for the total System-on-Chip (SoC) layout. A key success metric is performance uplift, where the goal is to demonstrate at least a 10x reduction in execution cycles for core pose estimation operations (sine, cosine, quaternion multiplication) when offloaded to the PEU, compared to a pure software implementation on the PicoRV32 core.

To validate the success metrics, the methodology will first involve establishing a software baseline by developing C-based firmware to perform pose estimation calculations on the CPU and measuring its performance through cycle-accurate simulation. Second, the same firmware will be modified to offload computations to the PEU to verify the hardware acceleration, measuring end-to-end execution time for the same operation sequence. Finally, a physical design analysis will be conducted after synthesis and place-and-route, using static timing analysis (STA) and power analysis tools to extract accurate area and power consumption figures from the final GDSII layout for direct comparison against the project's quantitative goals.

The primary comparison for this project will be against the status-quo baseline: a pure software implementation running on an embedded RISC-V processor. This is the most relevant comparison as the project aims to demonstrate the efficiency gains of domain-specific hardware acceleration in a specific application category.

While a direct comparison with commercial GPUs or FPGAs is not the main objective due to their different power and cost profiles, our final report will contextualize our results by referencing the computational demands of real-time systems (such as visual-inertial odometry) that require tracking pipelines to execute within 33 milliseconds. We will benchmark the number of quaternion multiplications our PEU can perform per second and compare its throughput with the operational requirements cited in existing literature. This will validate the practical applicability of our accelerator in the fields of robotics and AR/VR.

9 Workload Distribution

9.1 Backgrounds

Our team — Fengze Yu, Zhaoyu Lu, and Jiaying Yong — shares a solid foundation in System-on-Chip (SoC) design, digital VLSI implementation, and hardware/software co-design. All members have completed NYU ECE core graduate courses, including *VLSI System & Architecture Design (ECE-GY 6443)*, *Computing Systems Architecture (ECE-GY 6913)*, *Introduction to VLSI Design (ECE-GY 6473)*, and *Fundamentals of Analog Integrated Circuit Design (ECE-GY 6403)*. These courses collectively provided us with extensive experience in RTL design using Verilog/SystemVerilog, the ASIC flow from synthesis to place-and-route, and system-level verification.

This shared academic background gives the team a unified design methodology and a clear understanding of both architectural and physical design constraints, which are essential for developing a reliable RISC-V SoC with a hardware accelerator. Each member also brings distinct experience that aligns with specific technical challenges within this project, ensuring that all key aspects — computation accuracy, data communication, and system reliability — are well-covered.

Fengze Yu possesses a strong background in algorithmic control and speculative decoding techniques, enabling him to approach microarchitectural design from both a theoretical and practical perspective. He has previously explored control path optimization in pipelined architectures, which directly benefits the setup and analysis of the RISC-V CPU core. His analytical skills allow the team to achieve timing closure and performance consistency when running the full CPU tool flow from RTL simulation to synthesis and post-layout timing analysis.

Zhaoyu Lu has substantial hands-on experience with FPGA-based CNN accelerators, where he designed and optimized dataflow-oriented hardware structures. This experience gives him a deep understanding of on-chip memory interfacing, data parallelism, and latency reduction techniques, which are vital for implementing the interconnect and SRAM subsystem. His familiarity with hardware resource balancing will ensure efficient communication between the CPU, memory, and accelerator, reducing data bottlenecks and improving system throughput.

Jiaying Yong brings a system-level perspective developed through her previous work on aircraft communication link systems, where she gained experience in embedded control, real-time reliability, and hardware-software synchronization. This interdisciplinary background provides her with the ability to consider robustness and verification coverage at the SoC level. She also has experience in firmware testing and interface validation, making her well-prepared to ensure stable functionality of the accelerator and its integration with the RISC-V core.

As a team, we integrate complementary skill sets that span from low-level logic design to high-level verification. The combination of algorithmic insight, hardware efficiency, and system robustness enables us to work cohesively toward developing a functional, optimized, and verifiable SoC.

9.2 Task Distribution

The overall project is divided into three main technical focuses — the RISC-V CPU flow, the interconnect and SRAM subsystem, and the CORDIC-based accelerator logic — followed by a final integration and evaluation phase. Each team member takes lead responsibility for one focus area while maintaining active participation in all stages of the design and verification process.

Fengze Yu will primarily handle the RISC-V CPU component, setting up the design environment, verifying the baseline processor, and completing the full flow from RTL simulation to synthesis and post-layout analysis. His work ensures that the CPU operates correctly and provides a stable foundation for the integration of peripheral modules.

Zhaoyu Lu will focus on the interconnect and on-chip SRAM subsystem, building and verifying the communication paths between the CPU, accelerator, and memory. He will work on optimizing bus structures, reducing access latency, and ensuring reliable data exchange across modules to maintain system performance and consistency.

Jiaying Yong will be responsible for the CORDIC-based accelerator logic, designing the computation core for trigonometric operations such as sine, cosine, and arctangent. She will implement the control and datapath logic, verify precision and convergence, and ensure that the accelerator integrates correctly with the CPU through CSR-based interfacing.

In the final phase, all three members will collaborate equally on SoC-level integration, full-system verification, and performance evaluation. Together, we will conduct end-to-end testing, analyze timing closure, and benchmark hardware acceleration against software computation. Each team member contributes a significant and technically essential portion of the system, and all final deliverables — including the integrated RTL design, GDSII layout, and performance report — will represent collective effort and shared responsibility, as reflected in the project's Gantt chart.

10 Required Materials

All the tools required for this project are listed in Table 10. Based on the project plan, we have selected the following core Intellectual Property (IP) and Process Design Kit (PDK):

- **RISC-V CPU Core:** This project will use `PicoRV32` as an open-source RISC-V processor core.
- **PDK:** For the physical design and manufacturing process, the project will use the `TSMC N16 ADFP` Process Design Kit.

tableRequired IP for this project, including tools and designs.

Task	**Planned Solution**
RTL Simulation	VCS (Synopsys)
Logic Synthesis	Design Compiler (Synopsys)
Place and Route	IC Compiler II (Synopsys)
Timing Verification (STA)	Design Compiler (Synopsys)
Design Rule Check (DRC)	Design Compiler (Synopsys)
Layout vs. Schematic (LVS)	Design Compiler (Synopsys)
C Firmware Compiler	RISC-V GCC Toolchain
RISC-V CPU Core	PicoRV32 (temporary solution)
PDK	TSMC N16 ADFP

References

[1] J. Solà, ”Quaternion kinematics for the error-state Kalman filter,” *arXiv preprint arXiv:1711.02508*, 2017. [Online]. Available: https://arxiv.org/abs/1711.02508

[2] YosysHQ, ”PicoRV32 - A Size-Optimized RISC-V CPU,” GitHub Repository, 2024. [Online]. Available: https://github.com/YosysHQ/picorv32

[3] J. E. Volder, ”The CORDIC trigonometric computing technique,” *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330-334, 1959. [Online]. Available: https://doi.org/10.1109/TEC.1959.5222693

[4] A. Waterman, K. Asanović, and D. Patterson, ”The RISC-V Instruction Set Manual, VoP2me I: Unprivileged ISA,” RISC-V International, 2019. [Online]. Available: https://riscv.org/technical/specifications/

[5] OpenCores, ”WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, Rev. B.4,” 2010. [Online]. Available: https://cdn.opencores.org/downloads/wbspec_b4.pdf

[6] G. Gracioli, M. A. R. Martins, M. V. T. de Oliveira, L. A. Indrusiak and F. G. Moraes, ”DROID-SLAM: A RISC-V-based DPU for real-time SLAM applications,” *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1-9. [Online]. Available: https://doi.org/10.1109/ICCAD51958.2021.9_643485

[7] J. Lauri, T. O'Rourke, and S. V. G. Schiaffino, ”A review of head-pose estimation for augmented reality,” *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, 2022, pp. 581-582. [Online]. Available: https://doi.org/10.1109/VRW55335.2022.00181

[8] R. Mur-Artal and J. D. Tardós, ”ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255-1262, Oct. 2017. [Online]. Available: https://doi.org/10.1109/TRO.2017.2705103

[9] C. Li, ”From Rotation Matrices to Quaternions: Derivations, Efficient Algorithms, and Applications in Computing and Physics,” *Proceedings of the 2nd International Conference on Innovations in Applied Mathematics, Physics, and Astronomy*, 2025, pp. 342-347. [Online]. Available: https://www.scitepress.org/Papers/2025/138255/138255.pdf

[10] A. S. R. Salunke and S. S. Agrawal, "A Dual Quaternion Accelerator IP for RISC-V," *International Journal of Recent Innovations in Academic Research*, vol. 9, no. 5, pp. 77-81, 2024. [Online]. Available: https://rsisinternational.org/journals/ijrsi/digital-library/volume-9-issue-5/77-81.pdf

[11] A. V. Chernyak and A. V. Samburov, "Structure and Principles of Operation of a Quaternion VLSI Multiplier," *Applied Sciences*, vol. 14, no. 18, p. 8123, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/18/8123

[12] A. Cariow, G. Cariowa, and M. Chicheva, "Hardware-Efficient Schemes of Quaternion Multiplying Units for 2D Discrete Quaternion Fourier Transform Processors," *arXiv preprint arXiv:1703.06320*, 2017. [Online]. Available: https://arxiv.org/abs/1703.06320