FPGA Implementation of Low Power SLAM Accelerated Core

Mohammad Nazma Sultana
M.Tech (VLSID & ES)
Seshadri Rao Gudlavalleru Engineering College
Gudlavalleru, Krishna District, Andhra Pradesh, India – 521356

S. Ravi, PhD
Associate Professor
Seshadri Rao Gudlavalleru Engineering College
Gudlavalleru, Krishna District, Andhra Pradesh, India – 521356

ABSTRACT
Simultaneous Localization and Mapping (SLAM) is critical for autonomous systems because it enables real-time environmental mapping and navigation. Implementing SLAM algorithms in hardware, particularly on low-resource platforms, poses challenges owing to the computational complexity of operations such as matrix multiplications and quaternion transformations. This study introduces a novel accelerated core for SLAM algorithms that is optimized for hardware resource efficiency and high computational performance. By leveraging dedicated instruction set and memory reuse strategies, this core supports various SLAM approaches. The experimental results demonstrate the coprocessor's high precision, low resource consumption, and adaptability to multiple SLAM algorithms.

General Terms
Processor, Real-Time Systems, FPGA, Robotics

Keywords
Simultaneous Localization and Mapping (SLAM), Quaternion, Matrix multiplications, Rotation matrix.

1. INTRODUCTION
Simultaneous Localization and Mapping (SLAM) is a critical technology in robotics, autonomous vehicles, and augmented reality, enabling systems to construct a map of an unknown environment while tracking their position within it. Traditional SLAM algorithms rely on computationally intensive tasks such as matrix operations, quaternion transformations, and Extended Kalman Filters (EKF), making real-time processing on conventional processors challenging. Field Programmable Gate Arrays (FPGAs) offer a promising solution due to their parallel processing capabilities, reconfigurability, and energy efficiency. Unlike general-purpose processors, FPGAs can be customized to execute specific SLAM operations efficiently, reducing latency and power consumption. Recent research has focused on designing reconfigurable coprocessors tailored for SLAM, incorporating dedicated instruction sets, memory optimization techniques, and specialized hardware accelerators. This paper explores various accelerated core architectures developed for SLAM on FPGA platforms. It examines key computational techniques, including matrix multiplication accelerators, CORDIC-based trigonometric units, and hybrid fixed-point and floating-point arithmetic designs. By leveraging reconfigurable FPGA architectures, these cores enhance the flexibility and scalability of SLAM implementations, supporting both feature-based and learning-based SLAM algorithms.

2. LITERATURE REVIEW
- **Liu et al.** developed a runtime-reconfigurable FPGA accelerator optimized for SLAM-specific data locality, sparsity, and parallelism. The design enables efficient robotic localization by dynamically adapting computational resources based on algorithmic demands, leading to improved processing efficiency [1].
- **Wang et al.** designed a reconfigurable matrix multiplication coprocessor to accelerate matrix operations in vision-based navigation algorithms. The coprocessor significantly improves area and energy efficiency while maintaining high computational precision, making it suitable for autonomous robotic applications [3,4]
- **Gautschi et al.** introduced a specialized logarithmic unit in hardware to accelerate nonlinear function kernels, reducing computational overhead in SLAM-related mathematical operations. This unit enhances the energy efficiency of SLAM processors while maintaining high accuracy in pose estimation calculations [10,11].
- **Tertei et al.** proposed an FPGA-SoC-based hardware accelerator for matrix multiplication using systolic arrays to enhance the performance of EKF-SLAM algorithms. The architecture reduces computational latency by optimizing matrix processing operations, making it more efficient for real-time SLAM implementations [12].

3. ALGORITHM
3.1 Quaternion and Rotation Matrix
SLAM algorithms require accurate and efficient methods for representing 3D rotations and transformations. Quaternions and rotation matrices are the two most widely used mathematical representations for handling rotational motion in SLAM. Each method has distinct advantages in terms of computational efficiency, numerical stability, and hardware implementation. FPGA based reconfigurable cores integrate quaternion and rotation matrix computations to optimize SLAM performance for real-time applications [14].

3.1.1 Quaternion Representation in SLAM
Quaternion provides a compact and numerically stable representation of 3D rotations. Unlike Euler angles, which suffer from gimbal lock, quaternions [12] allow for smooth and continuous rotations. A quaternion "q" is expressed as:

$$q = w + a \cdot i + b \cdot j + c \cdot k$$

where,

- w is the real (scalar) component, representing the magnitude of rotation.
- a, b and c are the imaginary (vector) components defining the axis of rotation.

Quaternions are particularly useful in SLAM because they efficiently compute rotations through quaternion multiplication, avoiding the need for complex trigonometric

functions. Given two quaternions, q and p, their product is computed as shown in equation 1

$$q \cdot p = \begin{bmatrix} w_p w - a_p a - b_p b - c_p c \\ a_p w + w_p a + b_p c - c_p b \\ b_p w + w_p b + c_p a - a_p c \\ c_p w + w_p c + a_p b - b_p a \end{bmatrix} \quad (1)$$

SLAM core leverages FPGA-based parallel computing to accelerate quaternion operations. Dedicated hardware units handle

- Quaternion multiplication for pose updates.
- Quaternion-to-matrix conversion for coordinate transformations.

3.1.2 Rotation Matrix Representation in SLAM

A rotation matrix is a 3×3 matrix that represents a rotation in three-dimensional space [13]. It is commonly used in SLAM for transforming coordinates and sensor measurements. A rotation matrix R(q) corresponding to a quaternion q is shown in equation 2

$$R(q) = \begin{bmatrix} 1-2b^2-2c^2 & 2ab-2cw & 2ac+2bw \\ 2ab+2cw & 1-2a^2-2c^2 & 2bc-2aw \\ 2ac-2bw & 2bc+2aw & 1-2a^2-2b^2 \end{bmatrix} \quad (2)$$

This matrix is then used to transform a 3D vector $d' = R \cdot d$

Where,

- d is the original 3D position vector.
- d′ is the transformed 3D position vector after applying the rotation.

Quaternion and rotation matrices are essential mathematical tools in SLAM implementations. While quaternion offers a compact and efficient way to represent rotations, rotation matrices provide direct transformation capabilities useful in many SLAM algorithms [7]. FPGA-based reconfigurable cores integrate both methods to achieve high performance, low latency, and real-time SLAM processing.

3.2 CORDIC Trigonometric Functions

Trigonometric functions such as sine, cosine, tangent, and their inverses play a critical role in SLAM (Simultaneous Localization and Mapping) algorithms. These functions are used for pose estimation, sensor fusion, and coordinate transformations. However, traditional implementations using floating-point arithmetic or lookup tables can be computationally expensive and memory-intensive [3]. The CORDIC (COordinate Rotation DIgital Computer) [8] algorithm provides an efficient, hardware-friendly solution for computing trigonometric functions using only shift and add operations, making it ideal for FPGA-based reconfigurable core. A LiDAR sensor returns a distance (r) and an angle (θ). To convert to x, y coordinates, we use

$x = r \times \cos(\theta)$

$y = r \times \sin(\theta)$

Using CORDIC, we compute cos(θ) and sin(θ) without expensive floating-point operations [9].

4. HARDWARE ARCHITECTURE

The efficient execution of SLAM algorithms on FPGA-based platforms requires well-optimized hardware architecture. An accelerated core tailored for SLAM must handle pose estimation, matrix operations, quaternion transformations, and trigonometric functions while optimizing power consumption and real-time performance [11]. This section discusses the key components of a SLAM-accelerated core and how FPGA-based acceleration [5] enhances performance. Figure 1 shows a SLAM-accelerated core, which is designed with multiple specialized processing cores, optimized memory management, and dedicated instruction sets to maximize computational efficiency. The general architecture includes:

Key Components

- **Matrix Computation Core (MC)**: Handles matrix operations such as multiplication.
- **Special Computation Core (SP)**: Performs quaternion transformations, rotation matrix operations, and trigonometric computations.
- **Program Controller (PC)**: Manages instruction execution and synchronizes data flow between computation cores.
- **Memory (MEM)**: Stores vectors, scalars, and temporary data efficiently.
- **Floating arithmetic:** Performs addition, multiplication, and reciprocal operations.

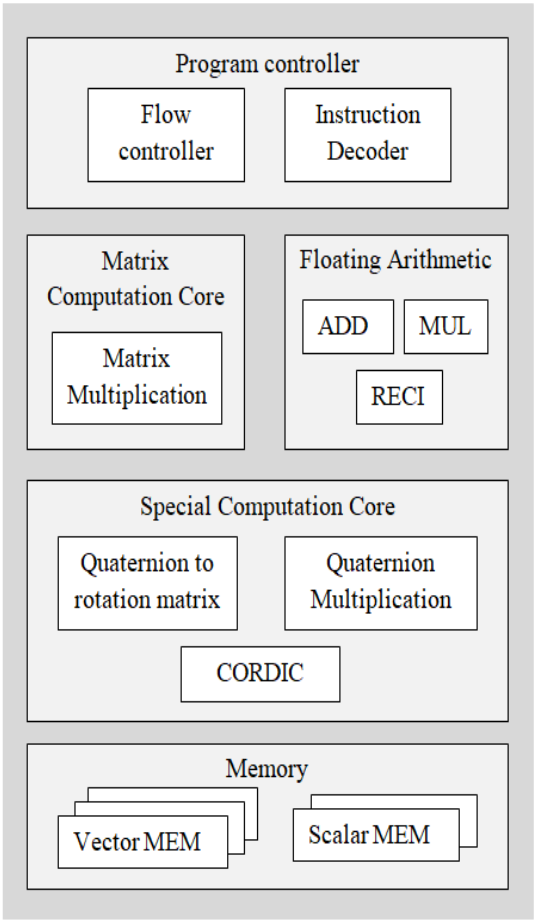

Figure 1: Hardware architecture of SLAM accelerated core

4.1 Matrix Computation Core (MC):

The Matrix Multiplication Core (MC) is designed to efficiently perform matrix-matrix multiplications. Matrix multiplication is

a fundamental operation in SLAM algorithms, as it is extensively used in pose estimation, sensor fusion, Extended Kalman Filters (EKF), and optimization-based SLAM techniques. Efficient hardware implementation of matrix multiplication is crucial for real-time SLAM performance, making it a primary component of FPGA-based reconfigurable cores [2]. The system operates in three stages: data fetch, computation, and write-back. In the Data Fetch stage, matrix data is retrieved from memory (MEM) using memory banking and caching to optimize access speed. The computation stage utilizes a pipelined systolic array, where multiple processing elements perform matrix multiplication by computing partial sums, which are stored in accumulators before final storage. In the write-back stage, the computed matrix is written back to memory, either triggering the next computation or forwarding data to the Special Computation Core (SP) for further processing.

4.2 Program Controller (PC):

The Program Controller (PC) is a critical component in an FPGA-based SLAM core, responsible for instruction execution, data management, synchronization, and control of parallel computation cores as shown in Figure 2. In real-time SLAM applications, efficient instruction flow management is essential to ensure low-latency execution of matrix computations, quaternion transformations, and sensor fusion tasks [12].

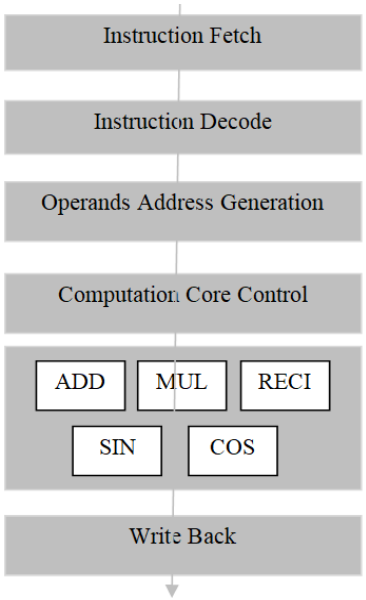

Figure 2: Flow control of SLAM accelerated core

4.2.1 Functions of the PC

- **Instruction Fetch & Decode**: Reads instructions from memory and directs them to the appropriate core.
- **Operand Addressing**: Manages memory access for matrix and vector operations.
- **Synchronization Control**: Ensures parallel computations complete in the correct order.

4.3 Instruction Set Architecture (ISA):

Table 1 show a dedicated ISA is implemented to optimize SLAM-specific computations.

Types of Instructions

- **Matrix Operations** (Multiplication).
- **Quaternion Operations** (Multiplication, Rotation Conversion).
- **Trigonometric Computations** (CORDIC-based sine/cosine calculations).
- **Floating arithmetic** (addition, multiplication, and reciprocal)

Table 1: ISA of SLAM accelerated core

Opcode	Operation	Operands	Description
0001	ADD	R1, R2, R3	Adds R1 and R2, stores result in R3.
0010	MATRIX_MUL	R1, R2, R3	Multiplies matrices R1 and R2, result in R3.
0100	QUATERNION_MUL	Q1, Q2, Q3	Multiplies quaternions Q1 and Q2, result in Q3.
0111	SIN_COS	ANGLE, SIN, COS	Computes sine and cosine of the given angle.

4.4 Special Computation Core (SP)

The Special Computation Core (SP) in an FPGA-based SLAM core is responsible for quaternion operations, trigonometric computations (CORDIC), and rotation matrix computations. It plays a crucial role in pose estimation and motion tracking by handling non-matrix computations that are fundamental to SLAM algorithms.

- **Quaternion Multiplication**: Used for pose estimation and sensor fusion.
- **Rotation Matrix Computation**: Converts between quaternion and matrix representations.
- **CORDIC Trigonometric Computations**: Calculates sine and cosine functions efficiently.

The data flow in the SP core is designed for efficient computation while minimizing memory bottlenecks. It begins with the Program Controller (PC) fetching and decoding an instruction, such as quaternion multiplication, determining operand addresses in vector and scalar memory (MEM), and activating the SP core. Next, input operands like quaternions and rotation matrices are retrieved from MEM. These operands are loaded into input registers for processing. The computation stage then executes operations such as quaternion multiplication using floating-point MACs, rotation matrix computation via matrix-vector multiplication, and CORDIC-based trigonometric functions with shift-and-add, with results stored in temporary registers. Finally, the computed results are written back to MEM for further processing, while the PC marks the operation as complete and fetches the next instruction.

4.5 Floating Point Arithmetic

Floating-point arithmetic is essential for high-precision SLAM computations, particularly in matrix operations and quaternion transformations. Unlike fixed-point arithmetic, floating-point operations maintain numerical accuracy, making them ideal for pose estimation, trajectory mapping, and error minimization in SLAM.

4.6 Memory (MEM)

Vector and Scalar Memory

- Stores matrix and vector data for SLAM computations.
- Organized into multiple banks for parallel access by the Matrix Computation Core (MC) and Special Computation Core (SP).
- Dual-port RAM allows simultaneous read/write operations, reducing data transfer latency.

5. EXPERIMENTAL RESULTS

The FPGA-SLAM core has better speed, power efficiency, and accuracy. Figure 3 shows the RTL schematic of the SLAM-accelerated core, which shows all the sub blocks of a core, and Figure 4 shows the Technology schematic of the SLAM-accelerated core, which shows all the internal components used in the core. The quaternion w, quaternion x, quaternion y, and quaternion z from the waveform in Figure 5 represent the result of quaternion multiplication, which is used for pose updating, and sine and cosine are used for angle updating

Table 2: Comparison with prior works

	This work	CICC 2022 [1]	JSSC 2019 [8]	TC 2020 [6]	RSS 2017 [10]
Platform	FPGA	FPGA	ASIC	ASIC	FPGA
Technology	28nm	28nm	28nm	65nm	28nm
Type	SLAM	SLAM	SLAM	SLAM	SLAM
Power	3.291 W	3.45 W	243.6 mW	5.50 W	1.46 W
Frequency	145 MHz	143 MHz	240 MHz	83.3 MHz	143 MHz

Table 2 shows the comparison of this work with the recent prior works, and Table 3 shows the comparison of functionalities supported by existing work and proposed work.

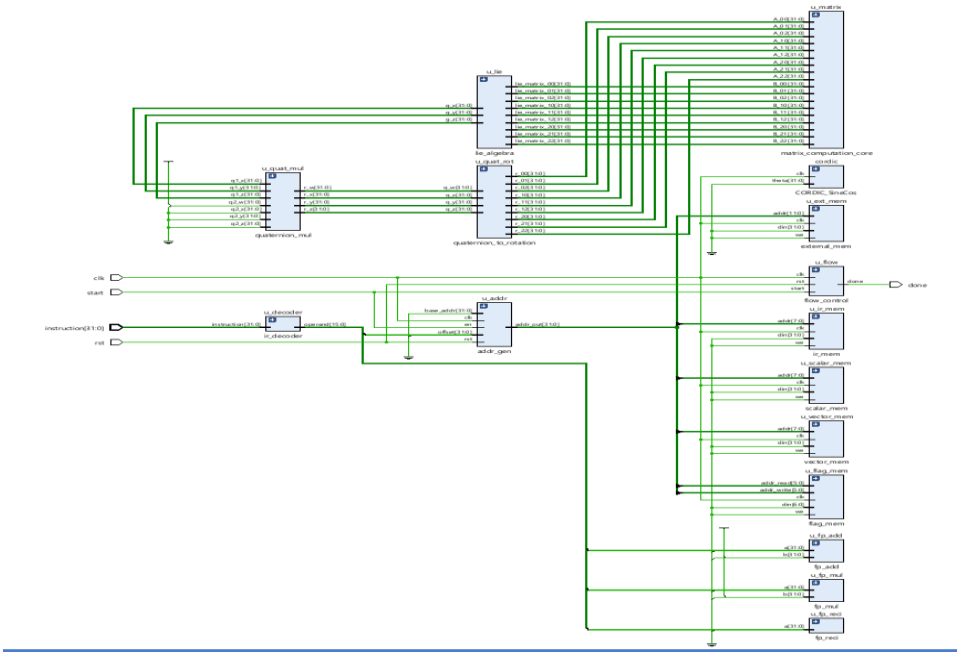

Figure 3: RTL schematic of a SLAM accelerated core

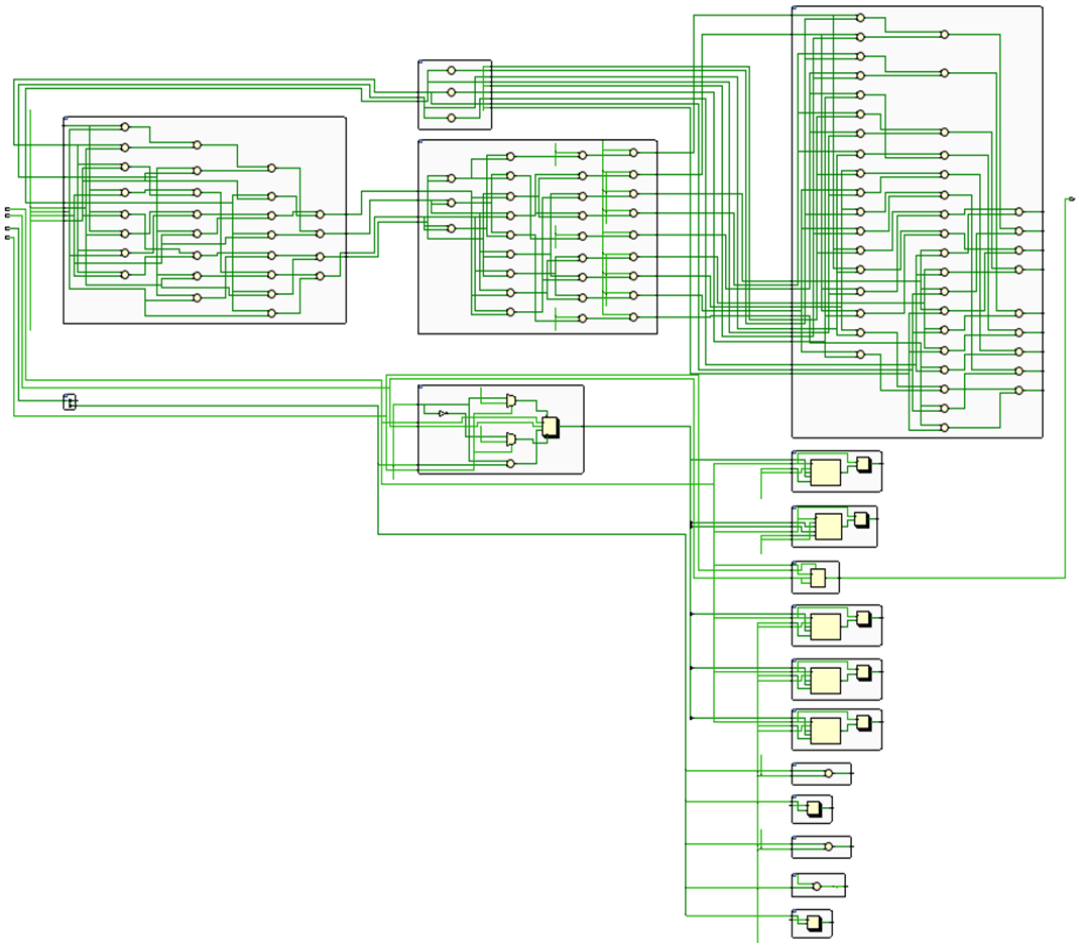

Figure 4: Technology schematic of a SLAM-accelerated core

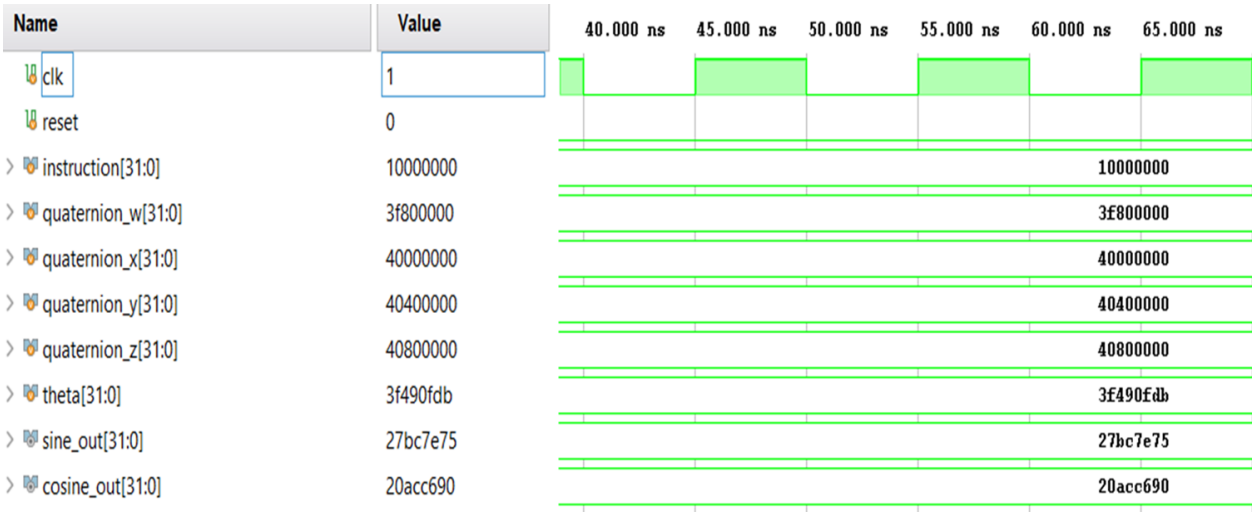

Figure 5: Waveforms of a SLAM-accelerated core

6. CONCLUSION

The paper presents a reconfigurable core designed for implementing various SLAM algorithms on FPGA, addressing the computational challenges of pose estimation in feature-based and learning-based methods. The core utilizes a dedicated instruction set architecture, a memory-reuse strategy to optimize storage requirements, and two parallel computing cores for floating-point and fixed-point matrix operations. By leveraging quaternion mathematics and CORDIC-based trigonometric functions, the design achieves high accuracy with minimal hardware resource consumption. Experimental results demonstrate superior efficiency in processing SLAM algorithms, making it a viable solution for real-time applications in robotics and autonomous systems.

7. REFERENCES

[1] Q. Liu, Z. Wan, B. Yu, W. Liu, S. Liu, and A. Raychowdhury, “An energy-efficient and runtime-reconfigurable FPGA-based accelerator for robotic localization systems,” in Proc. IEEE Custom Integr. Circuits Conf. (CICC), 2022, pp. 1–2.

[2] Y. Gan et al., "Eudoxus: Characterizing and Accelerating Localization in Autonomous Machines," HPCA, Mar. 2021.

[3] J. Wang et al., “A Reconfigurable matrix multiplication coprocessor with high area and energy efficiency for visual intelligent and autonomous mobile robots,” in Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC), 2021, pp. 1–3, doi: 10.1109/A-SSCC53895.2021.9634793.

[4] N. Cao, M. Chang, and A. Raychowdhury, “A 65-nm 8-to-3-b 1.0–0.36-V 9.1–1.1-TOPS/W hybrid-digital-mixed-signal computing platform for accelerating swarm robotics,” IEEE J. Solid-State Circuits, vol. 55, no. 1, pp. 49–59, Jan. 2020, doi: 10.1109/JSSC.2019.2935533.

[5] Q. Liu et al., "π-BA: Bundle Adjustment Hardware Accelerator Based on Distribution of 3D-Point Observations," TC, July 2020.

[6] Z. Li et al., "An 879GOPS 243mw 80fps VGA Fully Visual CNNSLAM Processor for Wide-Range Autonomous Exploration," ISSCC, Feb. 2019.

[7] A. Suleiman et al., "Navion: A 2-mw Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones," JSSC, Apr. 2019.

[8] Y. Kim, D. Shin, J. Lee, Y. Lee, and H.-J. Yoo, “A 0.55 V 1.1 mW artificial intelligence processor with on-chip PVT compensation for autonomous mobile robots,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 65, no. 2, pp. 567–580, Feb. 2018.

[9] J. E. Volder, “The CORDIC trigonometric computing technique,” IRE Trans. Electron. Comput., vol. EC-8, no. 3, pp. 330–334, Sep. 1959, doi: 10.1109/TEC.1959.5222693.

[10] M. Gautschi, M. Schaffner, F. K. Gürkaynak, and L. Benini, “An extended shared logarithmic unit for nonlinear function Kernel acceleration in a 65-nm CMOS multicore cluster,” IEEE J. Solid-State Circuits, vol. 52, no. 1, pp. 98–112, Jan. 2017, doi: 10.1109/JSSC.2016.2626272.

[11] D. T. Tertei, J. Piat, and M. Devy, “FPGA design of EKF block accelerator for 3D visual SLAM,” Comput. Elect. Eng., vol. 55, pp. 123–137, Oct. 2016.

[12] Shivaprasad B K, K. D. Shinde, and V. Muddi, “Design and implementation of parallel floating point matrix multiplier for quaternion computation,” in Proc. Int. Conf. Control Instrument. Commun. Comput. Technol. (ICCICCT), 2015, pp. 289–293, doi: 10.1109/ICCICCT.2015.7475292.

[13] E. Doukhnitch and E. Ozen, “Hardware-oriented algorithm for quaternion-valued matrix decomposition,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 58, no. 4, pp. 225–229, Apr. 2011, doi: 10.1109/TCSII.2011.2111590.

[14] G. Rubin, M. Omieljanowicz, and A. Petrovsky, “Reconfigurable FPGA-based hardware accelerator for embedded DSP,” in Proc. 14th Int. Conf. Mixed Design Integer. Circuits Syst., 2007, pp. 147–151, doi: 10.1109/MIXDES.2007.4286138.