IAES International Journal of Robotics and Automation (IJRA)
Vol. 15, No. 2, June 2026, pp. 377~387
ISSN: 2722-2586, DOI: 10.11591/ijra.v15i2.pp377-387

Hybrid LUT–CORDIC architecture on FPGA for efficient and accurate trigonometric computation in robot manipulators

Nia Gella Augoestien, Jazi Eko Istiyanto, Ahmad Ashari, Andi Dharmawan
Department of Computer Science and Electronics, Universitas Gadjah Mada, Yogyakarta, Indonesia

Article Info

Article history:

Received Dec 11, 2025
Revised Apr 24, 2026
Accepted May 13, 2026

Keywords:

Coordinate rotation digital computer
FPGA
Lookup table
Robot manipulator
Trigonometry function

ABSTRACT

Although computational resources on robots are often limited, real-time, accurate computation of trigonometric functions is essential in robot manipulators, particularly for forward and inverse kinematics, dynamic analysis, trajectory planning, and motion control. The LUT method requires a large number of LUTs to improve accuracy. The accuracy of the CORDIC method is highly dependent on the number of computational latencies, which affects the computation speed. This paper combines two general approaches for computing trigonometric functions on robot manipulators that improve accuracy without increasing resource utilization and computational latencies. The design uses a 10-bit format (0.125° input resolution and 2^{-10} output precision) and is implemented in VHDL on a Xilinx Artix-7 XC7A100T-CSG324 FPGA. Compared with a CORDIC-only baseline, the maximum absolute error is reduced from 0.083007812 to 0.009801151 for sine and from 0.079101563 to 0.008901377 for cosine, while MSE drops from 2.4031×10^{-4} and 2.32974×10^{-4} to 5.87754×10^{-6} and 5.87862×10^{-6}, respectively. The hybrid core also reduces slice usage from 81 to 69 and shortens computation time from 35.271 ns to 30.627 ns, making it suitable for resource-constrained real-time robotic control.

Corresponding Author:

Jazi Eko Istiyanto
Department of Computer Science and Electronics, Universitas Gadjah Mada
Bulaksumur 21, Yogyakarta 55281, Indonesia
Email: jazi@ugm.ac.id

1. INTRODUCTION

Real-time computation plays a crucial role in modern robotic systems, especially in manipulators that operate under continuous-path control. Trigonometric functions are utilized in robot manipulators for forward and inverse kinematics computation, dynamic analysis, trajectory planning, and motion control. In their implementation, trigonometric functions must be computed in real time to meet the robot's computational needs. Software-based trigonometric computation introduces significant delays, making it unsuitable for real-time robotic systems [1]. Field-programmable gate arrays (FPGA) offers flexibility for real-time implementation, thereby enhancing processing speed [2]. It is a reconfigurable hardware platform capable of performing naturally parallel computations, making it suitable for executing intensive computational algorithms [3]. The precise parallel computation capability of the FPGA ensures that system design requirements are met [4]. Moreover, FPGA-based acceleration in robotics offers a competitive alternative in power-limited and latency-critical scenarios, as it can outperform CPUs and GPUs in both energy efficiency and performance [5].

A major challenge in robotic computation is minimizing energy consumption while maintaining real-time performance [6]. Consequently, other essential aspects of trigonometric computation include accuracy and computational efficiency [7]. This is because trigonometric functions are part of a larger

computation, and their computational units are often integrated with other modules that require greater computational resources and handle more complex tasks. Thus, it is important to implement trigonometric computations that are both resource-efficient and fast, while remaining accurate.

General approaches for implementing trigonometric functions on FPGAs include the lookup table (LUT) and coordinate rotation digital computer (CORDIC) methods [8]. The LUT method is considered simple, effective, and fast, but requires a large number of LUTs to achieve high accuracy [9], [10]. The mapping of sine and cosine functions can be performed out using an array map in the LUT with a specific angular resolution [11]. Although the mechanism is simple, the required number of LUTs increases significantly as the resolution becomes finer to improve accuracy. The application of the twiddle factor can reduce the quadrant coverage to accommodate a larger input range for trigonometric functions [12]. Meanwhile, accuracy can be enhanced by combining LUTs with Taylor series expansion [13]. Its implementation adopted a finite state machine (FSM), requiring 12 steps and 53,248 bits of memory storage for sine and cosine calculations. Consequently, this method is unsuitable for applications requiring high accuracy [14].

The implementation of the CORDIC algorithm for trigonometric function computation on an FPGA is very convenient because it occupies minimal area [15]. This is because CORDIC only requires simple adders and shift registers [16]. On the other hand, the accuracy of the CORDIC algorithm depends on the number of iterations; higher accuracy results in greater latency [17]. Increased latency affects computational speed, making it difficult to meet real-time requirements. Several studies have been conducted to improve the speed of CORDIC operations, including Radix-4, Higher Radix, Double-Step Branching, Hybrid CORDIC, and Binary CORDIC techniques. These methods effectively reduce latency but consume significant hardware resources [14]. The precision of the CORDIC algorithm for computing Arcsine and Arccosine functions can be improved by applying appropriate gain compensation at each iteration [15]. Modifications to CORDIC can reduce latency and improve accuracy in sine and cosine computations by utilizing first-order Taylor series approximations [18]. The combination of these two approaches can enhance accuracy and reduce latency, but at the cost of higher computational resource usage. The state of the art in CORDIC research continues to seek an optimal trade-off among area, delay, power consumption, and precision; however, achieving simultaneous optimization remains challenging [19].

Another method for implementing trigonometric functions involves series expansion, such as the McLaurin series, to enhance computational accuracy and precision [20]. However, such series expansions, often used in commercial software, are not suitable for integer arithmetic operations [21]. A combination of LUTs and interpolation between adjacent points can also be used to approximate nonlinear functions such as the Arctangent [21]. Furthermore, several hybrid LUT-CORDIC methods have been proposed [22]–[25]. In these approaches, LUTs are commonly used to store the initial CORDIC stages (or pre-rotation) in order to reduce the number of iterations and thus the latency [22], [23]. Another variation uses a small LUT to store the least significant bit (LSB) correction values of the input angle, which are then accumulated with the CORDIC computation result [24]. More recently, LUT-based quadrant conversion has also been used to expand the angular range by mapping the input angle into other quadrants [25]. Overall, these hybrid designs primarily emphasize latency reduction or range handling; accuracy improvements are typically limited or indirect. Therefore, these characteristics create a practical FPGA design tension among area, speed, and accuracy. Despite extensive FPGA implementations of trigonometric computation, there is a lack of balanced architecture that simultaneously optimizes speed, accuracy, and hardware cost.

In this work, we propose an error-aware hybrid LUT–CORDIC architecture for FPGA-based trigonometric computation, tailored to the requirements of real-time robotic manipulator control. Instead of using LUTs for broad-stage replacement, we first map the absolute-error distribution of a baseline fixed-point CORDIC across the input-angle domain, then select rotation constants (α_n) to concentrate the significant errors into narrow boundary intervals. A very small ROM/LUT is then used only within these intervals to apply LSB-level correction, while the CORDIC path is retained for the rest of the input range. Using a 10-bit data format (0.125° angle resolution and 10-bit sine/cosine output precision) to match the manipulator joint resolution, the proposed design achieves a substantial reduction in computation error and improved timing performance with modest resource changes. To summarize, the main contributions of this paper are:

- An error-aware hybrid LUT–CORDIC architecture that improves the accuracy of sine/cosine computation while keeping the memory requirement minimal (LUT used only in narrow error-dominant intervals).
- A low-latency fixed-point implementation using a 10-bit input/output data format aligned with robotic manipulator joint resolution.
- Determining the rotation constants (α_n) by analyzing the CORDIC absolute-error distribution, thereby localizing dominant errors and directly determining the required LUT/ROM capacity.

This paper is organized as follows: Section 2 method, section 3 results and discussion, and rection 4 concludes the paper.

2. METHOD

In this study, a hardware architecture for trigonometric function computation will be designed by combining CORDIC and LUT methods. Accordingly, the functionality of the developed device is to compute sine and cosine values based on a given angle input. In general, the interface of the trigonometric computation unit to be developed is shown in Figure 1. As illustrated in Figure 1, there is one input representing the angle (θ) and two outputs representing the computed sine and cosine values.

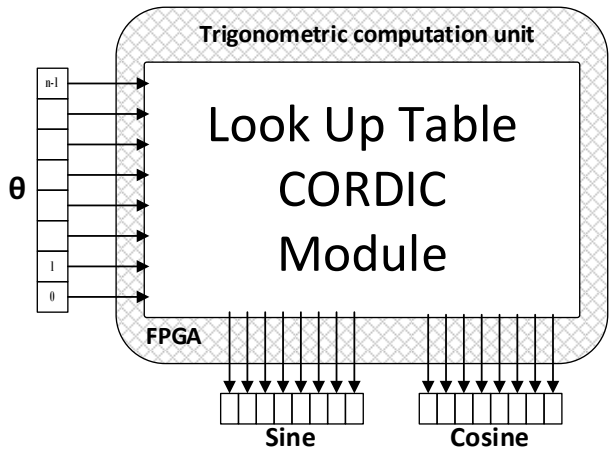

Figure 1. Trigonometry computing unit interface

The data format uses 10 bits per input and output, as shown in Table 1. The 10-bit fixed-point format was chosen based on the servo motors resolution to be used in the joint robot manipulator. Based on calculations of the sine and cosine functions for the resolution angle and its multiples, it was found that the 10-bit data format is optimal. On the other hand, there are considerations of accuracy and hardware cost for embedded robotics applications. Wider precision increases the data bandwidth and usually requires additional iterations (or larger correction tables) to maintain proportional error reduction, which increases area, latency, and power. The chosen 10-bit setting provides adequate resolution while maintaining a compact implementation.

Table 1. 10-bit data format

Data	Parameter	Bit position										Decimal Values
		9	8	7	6	5	4	3	2	1	0	
Angle(θ)	Binary Value	1	0	1	0	1	0	1	1	0	1	85,625
	Weight	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	
Sine or Cosine	Binary Value	1	0	1	0	1	0	1	1	0	1	0,6689453125
	Weight	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	

With a 10-bit input representation, the angle data can be expressed with a resolution of 0.125° over the interval 0.0–90.0°. For the 10-bit output data, the sine and cosine values can be represented with a precision of 2^{-10} over the interval 0.0–0.9990234375. This data format is adjusted to match the joint resolution of the robotic manipulator to be used. To support wider angle ranges, a simple quadrant-mapping range-reduction stage can be placed ahead of the core to map any input angle (e.g., 0°-360°) into the first-quadrant domain while applying sign and swap rules at the outputs. Likewise, higher bit-width precision can be obtained by widening the fixed-point format and increasing the CORDIC iterations, with optional LUT refinement to store additional LSBs for error-critical intervals.

The design procedure begins by modeling the trigonometric computation unit using the CORDIC method, which performs a sequence of shift-ad micro-rotations governed by the standard rotation constants can be written as (1).

$$\alpha_n = tan^{-1}(2^{-n}) \quad (1)$$

These micro-rotations are implemented as an iterative process using the CORDIC update in (2).

$$X_{n+1} = X_n - Y_n d_n 2^{-n}$$
$$Y_{n+1} = Y_n + X_n d_n 2^{-n} \quad (2)$$
$$Z_{n+1} = Z_n - d_n\, \alpha_n$$

The variables x_{n+1}, y_{n+1}, and $z_{n+1}$are updated at each iteration, to obtain the values corresponding to the cosine() function, the sine() function, and the angular change, respectively, where

$$d_n = \begin{matrix} 1\ for\ Z_n < 0 \\ -1\ for\ Z_n <= 0 \end{matrix} \quad (3)$$

To maintain consistency with the proposed 10-bit fixed-point angle representation, each α_n is quantized to the same internal angular format as θ, using the angular resolution of the data format (Table 1), ensuring that the angle accumulator and the input angle operate in same fixed-point domain. The implementation results are then analyzed to assess the method's performance. From this analysis, it was found that the computational accuracy of the trigonometric functions using the CORDIC method yielded a mean squared error (MSE) of 2.403×10^{-4}. When the absolute error in the CORDIC computation results were mapped across possible variations in input angles, it was observed that significant absolute errors occurred within specific angle intervals, depending on the selected rotation angle constants α_n and the fixed-point rounding effects. Based on this error-distribution study, the α_n set and iteration depth were selected as listed in Table 2 to localize the dominant error into narrow angular ranges, enabling a targeted correction mechanism in later stages while avoiding unnecessary increases in datapath width or iteration count that would raise area and latency with diminishing accuracy returns under 10-bit precision. The binary representation of the angle θ in Table 2 follows the 10-bit angle data format described in Table 1. For example, the angle 71.625° is represented as the 10-bit binary value 1000111101, where the 7 MSBs encode the integer part (71°), and the 3 LSBs encode the fractional part (0.625°).

Table 2. The rotation constant values

Iteration (n)	α_n	
	θ	Binary
0	71,625º; θ > 45	1000111101
	-18,5º ; θ <45	1101101100
1	14º	0001110000
2	7,125º	0000111001
3	3,625 º	0000011101
4	1,75º	0000001110
5	0,75º	0000000110
6	0,5º	0000000100
7	0,25º	0000000010
8	0,125º	0000000001

Based on the α_n constant values in Table 2, the significant absolute computation errors can be concentrated within the angle interval of 85.5°–90°. The angles that produce computation results with significant absolute errors are precomputed to obtain their sine values. The interval 85.5°–90° denotes the error-dominant subrange identified from the fixed-point error curve in the first quadrant. In the proposed hybrid scheme, the LUT correction is enabled only when $\theta \in [85.5°, 90°]$for sine and $\theta \in [0°, 4.5°]$for cosine using comparator-based threshold switching. These thresholds were chosen empirically to target the region where quantization and rounding cause the largest deviation in the CORDIC-only output, while keeping the LUT extremely small. These sine values on this interval are then represented in 10-bit format according to the output data specification. When comparing the precomputed data, four distinct groups of angles are identified, as shown in Table 3. From the table, it can be observed that the sine values vary only in the two least significant bits (LSBs). Therefore, these 2 LSBs are modeled and stored in a 4×2-bit ROM memory. The same procedure is applied for the cosine function within the interval 0°–4.5°.

Table 3. Precomputation of sin(θ) and cos(90° – θ) functions in binary format

θ	90 – θ	Sin (θ)/ Cos(90- θ)
85,5	4,735	11 1111 1100
85,625 – 86,375	3,5-4,25	11 1111 1101
86,5 – 87,375	2,375-3,375	11 1111 1110
85,375 – 90	0-2,25	11 1111 1111

The hybrid LUT-CORDIC methods proposed for computing trigonometric functions is illustrated in Figure 2. The hardware architecture shown in Figure 2 demonstrates that the input angle is compared against a threshold value of 85.5 degrees. If the input angle is greater than the threshold value, the sine function is computed using the LUT method according to the sine values listed in Table 3. Conversely, if the input angle is smaller than the threshold value, the sine function is calculated using the CORDIC method. Similarly, for cosine computation, the input angle is compared with the value (90° – threshold value). If the input angle is greater than this value, the cosine function is computed using the CORDIC approach, whereas if the input angle is smaller, the value is obtained using the LUT approach.

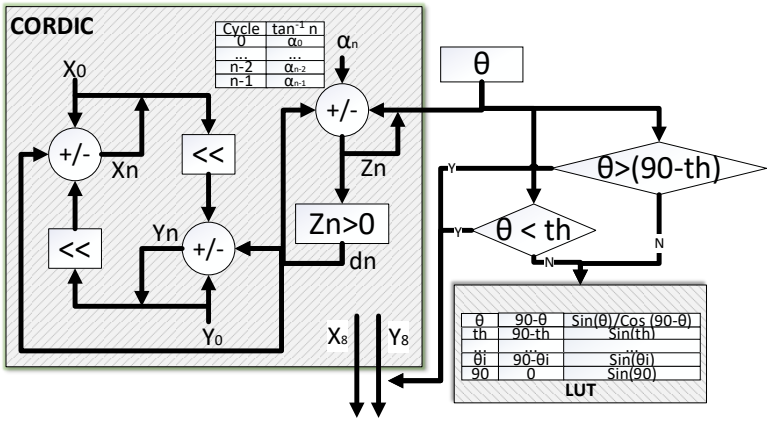

Figure 2. Proposed architecture

3. RESULTS AND DISCUSSION

In this study, a hardware architecture for computing trigonometric functions on an FPGA-based robotic manipulator is proposed. The proposed hardware architecture Hybrid LUT-CORDIC methods, modeled using VHDL in ISE 14.7. The resulting architecture model is then implemented on a Xilinx Artix-7 series XC7A100T-CSG324 FPGA. Figure 3 presents the functional simulation results of sine and cosine computation using the Hybrid LUT-CORDIC methods. From Figure 3, it can be observed that when the input angle is 85.375°, which corresponds to the hexadecimal value 2ABH, the computed sine and cosine values at the 9th clock cycle are 3FAH and 055H, respectively—equivalent to 0.994140625 and 0.0830078125 in decimal. When compared to the actual sine and cosine values at 85.375°, the absolute error for sine computation is 0.002603165, and for cosine computation is 0.002373968.

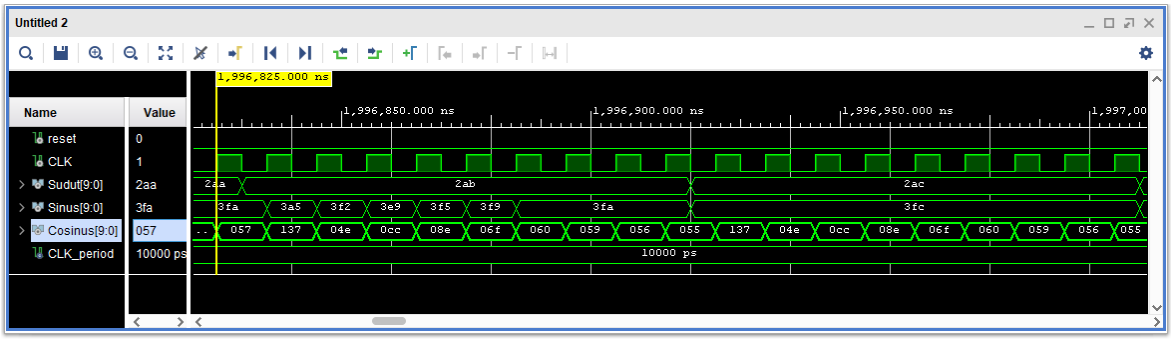

Figure 3. Computational simulation of trigonometric functions

Figure 3 also shows that to obtain sine and cosine computation results with sufficient accuracy using the CORDIC algorithm, nine iterations are required. This differs from trigonometric computation using the LUT method, which can produce results within the same clock cycle as the input angle. This can be observed when the input angle is 85.5° (hexadecimal: 3ACH), which equals the threshold value. In this case, the sine computation is performed using the LUT approach, yielding a value of 0.99609375 (in hexadecimal: 3FCH).

This computation result has an absolute error of 0.000823584 and is obtained in the same clock cycle as the input angle. This demonstrates that trigonometric computation using a LUT can be implemented using combinational circuitry, allowing the output to be produced in the same cycle as the input.

On the other hand, the cosine computation for the same input angle must be realized using the CORDIC approach. Therefore, the computation result is obtained at the 9th clock cycle, yielding 0.0830078125 (055H in hexadecimal format). This behavior aligns with the proposed hardware architecture. The cosine computation using the LUT method applies when the input angle is less than 4.5°, as shown in the simulation results in Figure 4. In Figure 4, when the input angle is 4.375° (in hexadecimal: 023H), the cosine computation result is obtained within the same clock cycle—0.99609375 (in hexadecimal: 3FCH)—using the LUT approach. However, the sine computation at this angle yields 0.067382812 (045H) using the CORDIC approach.

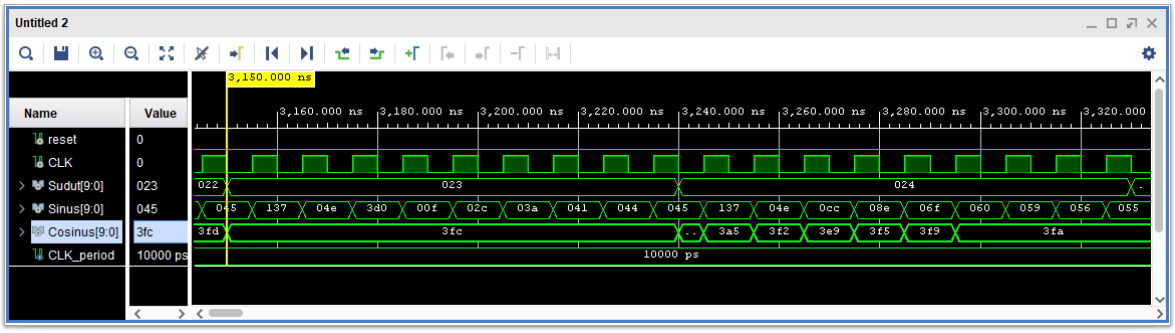

Figure 4. Computational simulation of trigonometric functions 2

When comparing the computation results obtained using both methods, it is evident that the absolute error from Hybrid LUT-CORDIC methods is significantly lower than that of the CORDIC method for both trigonometric functions. The absolute error values of trigonometric function computations using the CORDIC and Hybrid LUT-CORDIC methods are shown in Figure 5.

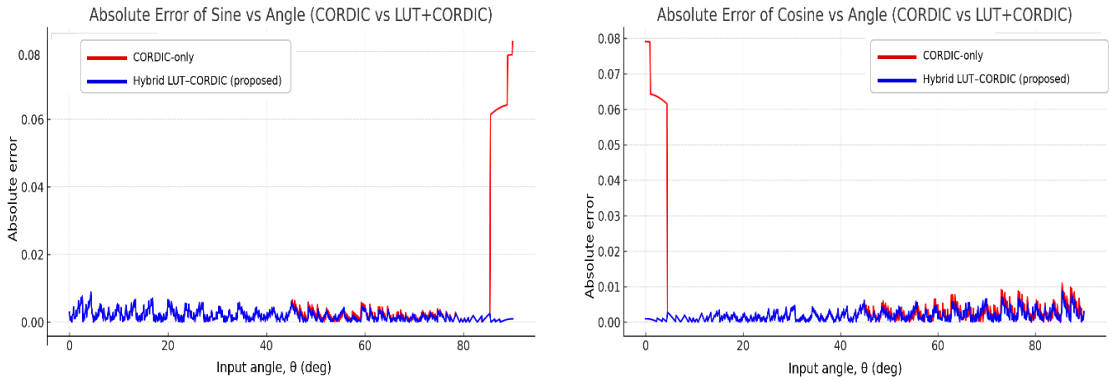

Figure 5. Absolute error graph of (a) sine and (b) cosine computation

From the graph in Figure 5, it can be observed that the absolute error in sine and cosine computations using the CORDIC method at certain angles can be reduced by applying the Hybrid LUT-CORDIC method. For the sine function in Figure 5(a), the maximum absolute error using the CORDIC method is 0.083007812, whereas when using the Hybrid LUT-CORDIC, it decreases to 0.009801151. A similar result is observed for the cosine function in Figure 5(b), where the maximum absolute error decreases from 0.079101563 (using the CORDIC method) to 0.008901377 when the Hybrid LUT-CORDIC method is applied.

The mean squared error (MSE) results show that the computation of the sine function using the Hybrid LUT-CORDIC method decreases significantly from 0.00024031 to 0.00000587754. This aligns with a similarly significant reduction in the cosine function computation, which decreases from 0.000232974 to 5.87862×10^{-6}. Thus, the sine function computation error is reduced by more than 97.5% (from 0.00024031 to 0.00000587754), consistent with the improvement achieved in the cosine function. More detailed data comparing the hardware performance is presented in Table 4. Since MSE reflects the average squared error over the evaluated range, the proposed design reduces the average squared deviation by approximately 97.5%, indicating that the computed trigonometric values are consistently closer to the reference across angles, not only at isolated points. In addition, the maximum absolute error is reduced by about 8×, which is important for worst-case safety thresholds in the overall kinematics computation of robot manipulators.

In robotic manipulators, trigonometric functions are the core building blocks for rotation matrices and homogeneous transformations used in forward/inverse kinematics and trajectory tracking. If an element of a rotation matrix is calculated as $\sin(\theta)$ or $\cos(\theta)$, then the trigonometric error directly affects the matrix computation results. As a result, the orientation and position computations of the end-effector will potentially experience increasingly large, accumulated errors. This large accumulated computational error impacts the smoothness of the robot's movement and the end-effector's ability to reach its target.

As shown in Table 4, the significant reduction in MSE obtained using the Hybrid LUT-CORDIC method also contributes to a decrease in the number of slices, from 81 slices using the CORDIC method to 69 slices when using the Hybrid LUT-CORDIC method. In addition, the proposed hybrid LUT–CORDIC implementation reduces FPGA resource usage, which is beneficial for manipulators that must share logic resources with other real-time modules (PWM generation, sensor interfacing, safety monitoring, and communication). Lower slice/LUT consumption allows additional control functionality to be integrated on the same device or enables deployment on smaller-cost FPGAs, while shorter computing time increases the overall computing speed.

Table 4. Hardware performance result

Method	CORDIC	Combined CORDIC-LUT	LUT only
Slice	81	69	197
Slice Register	62	60	0
Slice LUT	156	180	445
I/O	31	31	30
F Maks (MHz)	255,161	293,841	
Min Period (ns)	3,919	3,403	6.579ns
Clock Cycle	9	9	1
MSE Sine	2,403E-4	5,87754E-06	3,33280758997621E-07
MSE Cosine	2,32974E-4	5,87862E-06	

From a performance perspective, computing the sine function using the CORDIC method requires 9 clock cycles, yielding a computation time of 35.271 ns. In contrast, the computation using the hybrid LUT-CORDIC method takes 30.627 ns. Here, the computational time is defined as $T_{\text{eval}} = N_{\text{cycles}} \times T_{\text{clk}}$. Although both architectures require 9 clock cycles, the hybrid LUT–CORDIC implementation achieves a shorter clock period (higher $F_{\max}$) due to a reduced critical path, resulting in a lower total computational time of 30.627 ns versus 35.271 ns for the CORDIC-only design. This shows that the hybrid LUT–CORDIC method is faster than that of the CORDIC method alone. This improvement is due to the smaller minimum period in trigonometric computation when using the hybrid LUT-CORDIC compared to the standalone CORDIC method. A larger minimum period corresponds inversely to the maximum operating frequency — the higher the maximum operating frequency, the faster the computation can be performed. In practical applications, this trigonometric computation module will be integrated with other computation modules to form a unified real-time robotic computation system. Therefore, it is essential to ensure that each module operates at a high maximum frequency to ensure the entire hardware system functions at optimal speed. This combination of improved numerical fidelity and more efficient hardware execution is particularly advantageous for real-time robotic control, where both accuracy and deterministic timing are critical.

Unlike previous hybrid CORDIC–LUT methods, which mainly focused on latency reduction [22]–[24], this work simultaneously improves accuracy and computational efficiency for FPGA-based trigonometric computations. Meanwhile, [25] proposed a combination of LUT and CORDIC for trigonometric computations to expand the range of computed input angular intervals. This advantage stems from a different error-handling strategy than that used in many earlier hybrids: instead of allocating a larger LUT for pre-rotation or increasing CORDIC refinement depth to suppress residual error, our design applies range-selective correction. The measured fixed-point error peaks are confined to narrow angular intervals; therefore, we use a tiny LUT that stores only the necessary LSB correction patterns within these intervals,

while leaving the remaining angles to the CORDIC core. As a result, the architecture attains substantial worst-case and average error reduction with negligible memory overhead and without increasing the cycle count.

From a hardware perspective, the proposed hybrid design shifts the accuracy–cost trade-off: instead of increasing CORDIC iterations (which increases latency and datapath activity) or using a large LUT (which increases memory), we apply a minimal correction LUT only in error-dominant angle ranges. This yields a substantial accuracy gain (≈8× reduction in maximum error and ≈97.5% reduction in MSE) while adding negligible ROM overhead. Resource utilization decreases (81→69 slices) because the hybrid switching simplifies the logic in the critical region, thus improving routing and shortening the critical path. As a result, although both designs execute in 9 cycles, the hybrid core achieves a smaller clock period (higher F_{max}), leading to a lower total evaluation time (35.271 ns→30.627 ns). For embedded manipulators, these trade-offs translate into more timing margin for high-rate control loops and more FPGA capacity for co-located modules such as PWM generation, sensor interfaces, and safety monitoring.

Table 5 compares the implementation results of prior studies with those of this work across several FPGA platforms. Based on the comparison in Table 5, the proposed Hybrid LUT–CORDIC design consistently demonstrates a favorable resource–latency trade-off for FPGA-based robotic control. On Xilinx Artix-7, it reduces Slice LUT from 8865 (LUT-only [9]) to 180, achieving an ≈49.3× LUT reduction (≈98% savings) while maintaining a practical evaluation time of 30.627 ns; although LUT-only is faster (3.158 ns), it incurs a very large logic cost that is typically impractical when the FPGA must also accommodate PWM/servo generation, sensor interfacing, communication, and safety logic. On Xilinx Spartan-6, compared with a 16-bit CORDIC design [8], the proposed core lowers Slice LUT from 951 to 249 (≈3.82× reduction) and Slice FF from 95 to 76, with only a modest T_{eval} increase from 55.488 ns to 60.795 ns (≈9.6%), enabled by a shorter minimum clock period (6.755 ns vs 9.248 ns) despite using 9 cycles. On Xilinx Kintex-7, relative to a 16-bit CORDIC sin/cos reference [26], the proposed method achieves a substantially lower T_{eval} (26.37 ns vs 55.556 ns, ≈2.11× faster) while also reducing Slice LUT (396→243) and Slice FF (287→74), indicating improved timing efficiency and a smaller hardware footprint. Overall, while cross-study comparisons are influenced by differing FPGA families, constraints, and precision targets, these results highlight that the proposed architecture achieves meaningful performance with markedly reduced logic usage, making it well-suited for real-time embedded manipulator systems where deterministic timing and resource availability are both critical.

Table 5. Comparison of hardware performance result

FPGA	Design	Target Function	Method	Precision	Latency	Min Period (ns)	T_{eval} (ns)	Slice LUT	Slice FF
Xilinx Artix-7	Satti *et al*. [9]	Sin and Cos	LUT	13 bit	-	-	3,158	8865	64
	Our proposed	Sin and Cos	Hybrid LUT-CORDIC	10 bit	9	3,403	30,627	180	60
Xilinx Spartan-6	Salehi *et al.* [8]	Sin and Cos	CORDIC	16 bit	6	9,248	55,488	951	95
	Our proposed	Sin and Cos	Hybrid LUT-CORDIC	10 bit	9	6,755	60,795	249	76
Xilinx Kintex-7	Xilinx [26]	Sin and cos	CORDIC	16 bit	16	3,472	55,556	396	287
	Our proposed	Sin and Cos	Hybrid LUT-CORDIC	10 bit	9	2,93	26,37	243	74

Although the current implementation targets a first-quadrant angle range (0.0°-90.0°) with 10-bit fixed-point input/output formats, the proposed LUT-CORDIC core can be extended to wider angle ranges by adding a lightweight quadrant-mapping stage. This preprocessing map any input angle (e.g., 0°-360°) into an equivalent first-quadrant angle and applies quadrant-dependent sign and swap rules, so the internal data path and LUT contents remain unchanged. In addition, higher bit-width precision can be supported by parameterizing the fixed-point word length, increasing the number of CORDIC iterations in line with the added fractional bits, and expanding the LUT refinement (e.g., storing additional LSBs or using more breakpoint intervals).

Power consumption is also important for embedded and battery-powered deployments, while this study emphasizes accuracy, latency, and resource utilization. In FPGA implementations, dynamic power is largely driven by switching activity in the iterative add-shift data path and LUT access logic. Therefore, increasing the fixed-point bit-width or the number of CORDIC iterations typically increases switching and power. As future work, we plan to report post-place-and-route static and dynamic power using vendor-driven power analysis tools.

4. CONCLUSION

This research successfully combines the LUT and CORDIC approaches to perform trigonometric computations on an FPGA. Hardware performance measurements demonstrate that the proposed hardware

architecture significantly improves the accuracy (about an 8× drop in maximum error and roughly a 97.5% reduction in MSE) of trigonometric function computation while simultaneously reducing resource utilization (81→69) and slightly increasing computational speed (35.271 ns→30.627 ns), making it more suitable for real-time embedded manipulator applications. Future work will include post-layout power and energy characterization, as well as low-power optimizations such as clock gating and operand isolation.

FUNDING INFORMATION

This research was supported by internal funding from the Department of Computer Science and Electronics, Universitas Gadjah Mada, and also by the Faculty of Mathematics and Natural Sciences research funding with the contract number 3905/UN1/FMIPA.1.3/KP/PT.01.03/2025 The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Nia Gella Augoestien	✓	✓	✓	✓	✓	✓		✓	✓	✓				✓
Jazi Eko Istiyanto	✓				✓		✓			✓		✓	✓	
Ahmad Ashari				✓		✓				✓		✓		
Andi Dharmawan		✓				✓				✓	✓	✓	✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no conflicts of interest relevant to this study

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] A. M. Dalloo, A. J. Humaidi, A. K. A. Mhdawi, and H. Al-Raweshidy, "Low-power and low-latency hardware implementation of approximate hyperbolic and exponential functions for embedded system applications," *IEEE Access*, vol. 12, no. February, pp. 24151–24163, 2024, doi: 10.1109/ACCESS.2024.3364361.

[2] V. D. Cong, "Industrial robot arm controller based on programmable system-on-chip device," *FME Transactions*, vol. 49, no. 4, pp. 1025–1034, 2021, doi: 10.5937/FME2104025C.

[3] O. Tkachenko and K. T. Song, "Hardware accelerated inverse kinematics for low power surgical manipulators," *International Conference on Advanced Robotics and Intelligent Systems, ARIS*, vol. 2020, 2020, doi: 10.1109/ARIS50834.2020.9205769.

[4] Z. Jiang, Y. Dai, J. Zhang, and S. He, "Kinematics calculation of minimally invasive surgical robot based on FPGA," *2017 IEEE International Conference on Robotics and Biomimetics, ROBIO 2017*, vol. 2018-Janua, pp. 1726–1730, 2017, doi: 10.1109/ROBIO.2017.8324667.

[5] Z. Wan *et al.*, "A survey of FPGA-based robotic computing," *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 48–74, 2021, doi: 10.1109/MCAS.2021.3071609.

[6] H. Xu *et al.*, "Toward efficient computing for robotics: from a circuit and system view," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 7, pp. 3051–3056, 2022, doi: 10.1109/TCSII.2022.3179871.

[7] W. Tang and F. Xu, "A noniterative radix-8 CORDIC algorithm with low latency and high efficiency," *Electronics (Switzerland)*, vol. 9, no. 9, pp. 1–17, 2020, doi: 10.3390/electronics9091521.

[8] F. Salehi, E. Farshidi, and H. Kaabi, "Novel design for a low-latency CORDIC algorithm for sine-cosine computation and its Implementation on FPGA," *Microprocessors and Microsystems*, vol. 77, p. 103197, 2020, doi: 10.1016/j.micpro.2020.103197.

[9] P. Satti, N. Sharma, G. Singh, and B. Garg, "A flexible dynamic master–user look-up table approach for evaluation of trigonometric values," *Wireless Personal Communications*, vol. 127, no. 4, pp. 3425–3434, 2022, doi: 10.1007/s11277-022-09924-3.

[10] Y. S. Gener, S. Goren, and H. F. Ugurdag, "Lossless look-up table compression for hardware implementation of transcendental

functions," *IEEE/IFIP International Conference on VLSI and System-on-Chip, VLSI-SoC*, vol. 2019-Octob, pp. 52–57, 2019, doi: 10.1109/VLSI-SoC.2019.8920330.
[11] S. Mahmood, P. Shydlouski, and M. Hubner, "An application specific framework for HLS-based FPGA design of articulated robot inverse kinematics," *2018 International Conference on Reconfigurable Computing and FPGAs, ReConFig 2018*, pp. 1–6, 2018, doi: 10.1109/RECONFIG.2018.8641691.
[12] D. Guerrero Martos *et al.*, "Using the complement of the cosine to compute trigonometric functions," *Eurasip Journal on Advances in Signal Processing*, vol. 2020, no. 1, 2020, doi: 10.1186/s13634-020-00692-5.
[13] W. C. Chen, C. S. Chen, F. C. Lee, and Y. S. Kung, "FPGA-realization of the kinematics IP for SCARA robot," *Microsystem Technologies*, vol. 27, no. 4, pp. 1075–1090, 2021, doi: 10.1007/s00542-018-4061-5.
[14] K. Li, H. Fang, Z. Ma, F. Yu, B. Zhang, and Q. Xing, "A low-latency CORDIC algorithm based on pre-rotation and its application on computation of arctangent function," *Electronics (Switzerland)*, vol. 13, no. 12, 2024, doi: 10.3390/electronics13122338.
[15] P. Paz and M. Garrido, "CORDIC-based computation of arcsine and arccosine functions on FPGA," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 9, pp. 3684–3688, 2023, doi: 10.1109/TCSII.2023.3262353.
[16] F. Lyu, C. Wu, Y. Wang, H. Pan, Y. Wang, and Y. Luo, "An optimized hardware implementation of the CORDIC algorithm," *IEICE Electronics Express*, vol. 19, no. 21, pp. 1–6, 2022, doi: 10.1587/elex.19.20220362.
[17] Y. Zhao, H. Lv, J. Li, and L. Zhu, "High performance and resource efficient FFT processor based on CORDIC algorithm," *Eurasip Journal on Advances in Signal Processing*, vol. 2022, no. 1, 2022, doi: 10.1186/s13634-022-00855-6.
[18] N. Bai, R. Qu, Y. Xu, Y. Wang, X. Chen, and L. Li, "Low-iteration hybrid computing CORDIC architecture," *Microelectronics Journal*, vol. 156, no. November 2024, 2025, doi: 10.1016/j.mejo.2024.106481.
[19] P. Kumar Meher and S. Aggarwal, "Efficient design and implementation of scale-free CORDIC with mutually exclusive micro-rotations," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 72, no. 5, pp. 2243–2251, 2025, doi: 10.1109/TCSI.2025.3549974.
[20] M. Mustapha and N. A. Zulkarnain, "Full cycle trigonometric function on Intel Quartus II Verilog," *AIP Conference Proceedings*, vol. 1930, no. February 2018, 2018, doi: 10.1063/1.5022938.
[21] M. K. Wu, Y. S. Kung, Y. H. Huang, and T. H. Jung, "Fixed-point computation of robot kinematics in FPGA," *2014 International Conference on Advanced Robotics and Intelligent Systems, ARIS 2014*, pp. 35–40, 2014, doi: 10.1109/ARIS.2014.6871523.
[22] X. Fei, "Efficient arithmetic functional modules for FPGA-based VLSI design," Doctoral dissertation, 2005.
[23] Y. S. Juang, L. T. Ko, J. E. Chen, T. Y. Sung, and H. C. Hsin, "Optimization and implementation of scaling-free CORDIC-based direct digital frequency synthesizer for body care area network systems," *Computational and Mathematical Methods in Medicine*, vol. 2012, 2012, doi: 10.1155/2012/651564.
[24] L. Moroz, S. Nagayama, T. Mykytiv, I. Kirenko, and T. Boretskyy, "Simple hybrid scaling-free CORDIC solution for FPGAs," *International Journal of Reconfigurable Computing*, vol. 2014, pp. 12–15, 2014, doi: 10.1155/2014/615472.
[25] Y. Chen and Y. Liu, "Research and implementation of a numerical control oscillator with improved pipelined CORDIC algorithm," *Academic Journal of Science and Technology*, vol. 5, no. 1, pp. 32–37, 2023, doi: 10.54097/ajst.v5i1.5301.
[26] Xilinx, "LogiCORE IP CORDIC v4.0, DS249," 2011.

BIOGRAPHIES OF AUTHORS

Nia Gella Augoestien is currently pursuing her Ph.D. at Department of Computer Science and Electronics, Universitas Gadjah Mada, focusing on embedded processing system for robotics. She earned her bachelor's degree in Electronics and Instrumentation in 2011 and her master's degree in computer science in 2015, both from the Department of Computer Science and Electronics, Universitas Gadjah Mada, Indonesia. Her research interests include embedded system and hardware architecture for cryptography and robotics. She may be contacted at nia.gella@mail.ugm.ac.id.

Jazi Eko Istiyanto is currently Full Professor in Electronics and Instrumentation at Universitas Gadjah Mada, Faculty of Mathematics and Natural Sciences, a position he held since 2010 before took office at BAPETEN (Indonesia Nuclear Energy Regulatory Agency) as the Chairman from February 2014 until October 2021. Before serving the Government of Indonesia, he held academic managerial positions as Head of the Computer Science and Electronics Department (2011-2014), and Head of the Physics Department (2007-2011) Universitas Gadjah Mada. Jazi Eko Istiyanto holds a Ph.D. (1995) in Electronic Systems Engineering, an M.Sc. (1988) in Computer Science, and a Postgraduate Diploma (1987) in Computer Programming and Microprocessors' Applications from University of Essex, Colchester, United Kingdom, and a B.Sc. (1986) in Nuclear Physics from Universitas Gadjah Mada, Yogyakarta, Indonesia. His research interests cover embedded systems and cyber-physical systems security. He is also a registered engineer (electronic engineering) in Indonesia and ASEAN countries. He may be contacted at jazi@ugm.ac.id.

Ahmad Ashari is currently Full Professor in Department of Computer Sciences and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada. His work has been published in several international journals for Computer and Network area. The research focuses are computer network, network security, high performance computing, IoT and Cloud. He may be contacted at ashari@ugm.ac.id.

Andi Dharmawan was born in Surakarta, Indonesia, in 1984. He received his B.Sc. in Electronics and Instrumentation in 2006, M.Sc. in Computer Science in 2009, and Ph.D. in Computer Science in 2017, all from Universitas Gadjah Mada, Indonesia. From 2007 to 2009, he worked as a Research Assistant at the Department of Computer Science and Electronics, Universitas Gadjah Mada. Since 2009, he has been a faculty member at the same department, where he is also a part of the Embedded Systems and Robotics Laboratory. His research interests include control systems for robotics, autonomous unmanned systems, and advanced control system development. He may be contacted at andi_dharmawan@ugm.ac.id or andi_dharmawan@mail.ugm.ac.id.