Dual Quaternion Hardware Accelerator for RISC-V based System

Rajeshwari B, Rithvik Kumar, Shweta P Hegde, Manav Eswar Prasad, Vandhya D M, Bajrangabali B
Department of Electronics and Communication, PES University, India

***Abstract*— In this work, a hardware accelerator has been developed for a RISC-V processor. The 'Parashu' Shakti processor is the SoC of choice for application testing and development. The IP is designed to speed up applications involving dual quaternion operations. Our module primarily aids dual quaternion multiplication which further helps with other complex operations like translation, rotation and transformation. Two solutions have been proposed for the same, i.e. either a quaternion IP if power and resource utilization is a concern, or a dual quaternion IP if performance gain is the primary objective. The latter is however at the expense of relatively more resource utilization. The former IP takes longer execution time to perform the same task but is more versatile since it can be used in applications involving both quaternion and dual quaternion operations.**

***Keywords*— RISC-V, Dual Quaternion, Shakti Processor, Hardware Accelerator, Robotics**

I.INTRODUCTION

In modern society, RISC-V is an emerging technology and is rapidly developing as it is an open ISA that offers stability, scalability and security. RISC-V is an open standard instruction set architecture (ISA) based on established reduced instruction set computer (RISC) principles. The dynamic and modular aspects make RISC-V ISA a compelling choice for IP development[1],[2]. The SHAKTI Processor Program, based on the RISC-V ISA, was started as an academic initiative back in 2014 by IIT Madras. The monopoly of a handful of companies in the processor market and the growing demand for customizable SoC's and ASIC's (Application Specific Integrated Chips) was the motivation behind the development of the open-source processor ecosystem.

Dual quaternion is a combination of Quaternions and dual number theory ,a tool for expressing and analyzing the physical properties of rigid bodies. It is considered to be better than the conventional Cartesian coordinate system since it is singularity free, avoids Gimbal lock(loss of a degree of freedom in a rotation system when two axis of rotation coincide), yields the shortest interpolation path for the system and it can formulate the problem more concisely with greater precision, since 4 states per quaternion are used to represent a 3- dimensional space.

II. RELATED WORKS

The Shakti processor ecosystem offers a plethora of processors from basic ones like E and C class to multi core one like M and S class. The E class SoC i.e. Parashu core has been selected for our application and it follows a 3 stage in –order execution pipeline [3]. There is a wide range of tools provided for application development including Shakti SDK (Software Development Kits), Platform IDE extension for Visual Studio and Arduino IDE as well. Development boards supported for these processors are the Artix 7 35T and Artix 7 100T board [4].

Dual quaternions are a singularity free and unambiguous alternative to the conventional Cartesian coordinate system which unify the translation and rotation into a single state [5] widely employed in modern day robotics [6]. This paper sheds the role of Dual quaternions in the field of kinematics for key operations like rotation, translation and transformation [7] where we have observed the prominent operation to be multiplication. We have accelerated the same by utilizing an optimized algorithm which performs 24 real multiplications and 64 real additions down from the conventional algorithm which involves 64 real multiplications and 56 real additions [8]. This multiplication IP relies on a 32 bit FPU unit that was designed as well [9]. Today, dual quaternion finds its application in the field of motion graphics as well, since it fares better both in terms of accuracy and efficiency in operations like skinning and blending than alternative algorithms like spherical blend and log-matrix respectively [10].

III. DESIGN AND IMPLEMENTATION OF HARDWARE ACCELERATOR

The hardware accelerator has been designed majorly to multiply two dual quaternions. With this motive, two types of IPs have been designed namely, Dual quaternion IP and Quaternion IP. As mentioned in [11],the dual quaternion can be represented as

$$q = q_r + q_d\epsilon \quad (1)$$

Where q_r and q_d are real and dual parts and each quaternion q can be expressed as,

$$q_r = r \qquad (2)$$

$$q_d = \frac{1}{2} tr \quad (3)$$

Where r is the quaternion representing the rotation and t is the quaternion describing the translation. Each quaternion is of the form,

$$q(w,v) = w + (xi + yj + zk) \qquad (4)$$

where w, x, y and z are the numerical values and i, j and k are the imaginary components. The designed hardware accelerators multiply the two dual quaternions by following the below-mentioned methods.

A.Dual quaternion IP

The multiplication in case of dual quaternion IP is as mentioned in the literature [11]. If Q and P are two dual quaternions of 8 elements each represented as,

$$Q = q_{r1} + q_{d1} \quad (5)$$

$$P = q_{r2} + q_{d2} \quad (6)$$

then, the multiplication of these dual quaternions can be written as,

$$T = Q.P \qquad (7)$$

Table I. Dual Quaternion Multiplication Table

Q.P	P.1	P.i	P.j	P.k	P.ϵi	P.ϵj	P.ϵk	P.ϵ
Q.1	1	i	J	K	ϵi	ϵj	ϵk	ϵ
Q.i	i	-1	K	-j	-ϵ	ϵk	-ϵj	ϵi
Q.j	j	-k	-1	I	-ϵk	-ϵ	ϵi	ϵj
Q.k	k	J	-i	-1	ϵj	-ϵi	-ϵ	ϵk
Q.ϵi	ϵi	-ϵ	ϵk	-ϵj	0	0	0	0
Q.ϵj	ϵj	-ϵk	-ϵ	ϵi	0	0	0	0
Q.ϵk	ϵk	ϵj	-ϵi	-ϵ	0	0	0	0
Q.ϵ	ϵ	ϵi	-ϵj	-ϵk	0	0	0	0

The final dual quaternion multiplication output T contains 8 elements where the, Real quaternion component of the resulting dual quaternion is obtained by:

$$T[0] = P[0]*Q[0]-P[1]*Q[1]-P[2]*Q[2]-P[3]*Q[3] \quad (8)$$

$$T[1] = P[0]*Q[1]+P[1]*Q[0]+P[2]*Q[3]-P[3]*Q[2] \quad (9)$$

$$T[2] = P[0]*Q[2]+P[2]*Q[0]-P[1]*Q[3]+P[3]*Q[1] \quad (10)$$

$$T[3] = P[0]*Q[3]+P[3]*Q[0]+P[1]*Q[2]-P[2]*Q[1] \quad (11)$$

Dual quaternion component of the resulting dual quaternion is obtained by:

$$T[4] = P[4] * Q[0] + P[0] * Q[4] + P[7] * Q[1] + P[1] * Q[7]- P[6] * Q[2] + P[2] * Q[6] + P[5] * Q[3] - P[3] * Q[5] \quad (12)$$

$$T[5] = P[5] * Q[0] + P[0] * Q[5] + P[6] * Q[1] - P[1] * Q[6]+ P[7] * Q[2] + P[2] * Q[7] - P[4] * Q[3] + P[3] * Q[4] \quad (13)$$

$$T[6] = P[6] * Q[0] + P[0] * Q[6] - P[5] * Q[1] + P[1] * Q[5]+ P[4] * Q[2] - P[2] * Q[4] + P[7] * Q[3] + P[3] * Q[7] \quad (14)$$

$$T[7] = P[7] * Q[0] + P[0] * Q[7] - P[1] * Q[4] - P[4] * Q[1]- P[2] * Q[5] - P[5] * Q[2] - P[3] * Q[6] - P[6] * Q[3] \quad (15)$$

As shown in TABLE 1, some terms with coefficient ϵ2 are zero and can be ignored. Terms with similar coefficients can be clubbed as well. Hence, it is an optimal algorithm that requires only 24 real multiplications and 64 real additions and hence reduces multiplicative complexity.

B. Quaternion IP

The multiplication of two dual quaternions in case of Quaternion IP is as follows: If Q and P are two dual quaternions of the form,

$$Q = q_{r1} + q_{d1} \qquad (16)$$

$$P = q_{r2} + q_{d2} \qquad (17)$$

then, the multiplication of these dual quaternions can be written as,

$$QP = q_{r1}\, q_{r2} + \epsilon(q_{r1}q_{d1} + q_{r2}\, q_{d1}) \qquad (18)$$

Hence instead of performing dual quaternion multiplication directly, the alternate approach is to use the logic of quaternion as mentioned in (18).

C. Vivado Simulation

These IPs are designed in Verilog HDL and have been simulated using Vivado. The IP contains its own floating-point unit to perform addition, multiplication and subtraction which is necessary for dual quaternion multiplication. The input and output operands of these IPs are fixed to single precision IEEE 754 format which are of 32-bit floating point numbers. The multiplication outputs as a result of simulation in Vivado for two specific user given inputs opa, opb which are the two dual quaternion input of 8 elements each and opc is the output result of dual quaternion multiplication have been depicted in Fig.1.

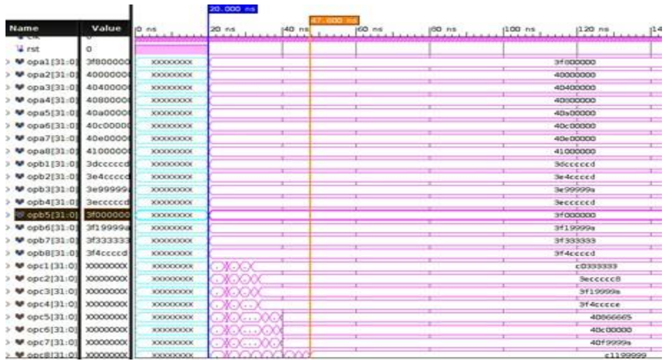

```
Name          Value
rst           0
opa1[31:0]    3f800000    3f800000
opa2[31:0]    4000000     40000000
opa3[31:0]    4040000     40400000
opa4[31:0]    4080000     40800000
opa5[31:0]    40a0000     40a00000
opa6[31:0]    40c0000     40c00000
opa7[31:0]    40e0000     40e00000
opa8[31:0]    4100000     41000000
opb1[31:0]    3dccccd     3dcccccd
opb2[31:0]    3e4cccc     3e4ccccd
opb3[31:0]    3e99999     3e99999a
opb4[31:0]    3eccccc     3ecccccd
opb5[31:0]    3f00000     3f000000
opb6[31:0]    3f19999     3f19999a
opb7[31:0]    3f33333     3f333333
opb8[31:0]    3f4cccc     3f4ccccd
opc1[31:0]    XXXXXXXX    c0333333
opc2[31:0]    XXXXXXXX    3ecccccc8
opc3[31:0]    XXXXXXXX    3f19999a
opc4[31:0]    XXXXXXXX    3f4ccccce
opc5[31:0]    XXXXXXXX    40866665
opc6[31:0]    XXXXXXXX    40c00000
opc7[31:0]    XXXXXXXX    40f9999a
opc8[31:0]    XXXXXXXX    c1199999
```

Figure 1. Vivado output for Dual Quaternion multiplication

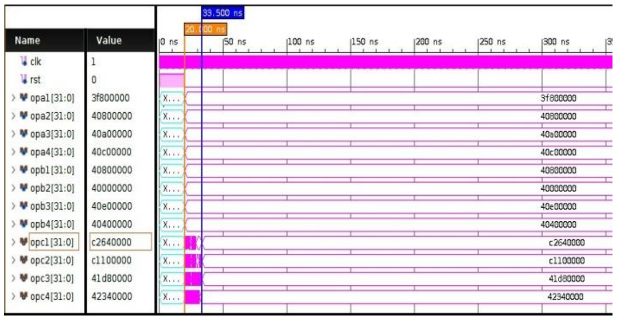

```
Name          Value
clk           1
rst           0
opa1[31:0]    3f800000    3f800000
opa2[31:0]    40800000    40800000
opa3[31:0]    40a00000    40a00000
opa4[31:0]    40c00000    40c00000
opb1[31:0]    40800000    40800000
opb2[31:0]    40000000    40000000
opb3[31:0]    40e00000    40e00000
opb4[31:0]    40400000    40400000
opc1[31:0]    c2640000    c2640000
opc2[31:0]    c1100000    c1100000
opc3[31:0]    41d80000    41d80000
opc4[31:0]    42340000    42340000
```

Figure 2. Vivado output for Quaternion multiplication

The Fig.2 depicts the quaternion multiplication, where opa and opb are two quaternion inputs of 4 elements each with opc representing the quaternion output.

Results obtained in resource utilization are for the Artix A7 FPGA board. While comparing the three cases, it can be observed that only LUT (Look Up Table), FF (Flip-flop) and DSP (Digital Signal Processors) see a significant increase whilst utilization of the other resources remains relatively unchanged. The increase, however, is more in the case of the dual quaternion IP. It is the same scenario in the case of power utilization where the dual- quaternion IP depicts a 21 % increase whilst the quaternion IP is just 4.5% more than the standalone SoC's utilization.

IV. INTEGRATION

Involves integrating the designed hardware accelerator with the processor core [12]. The IP is considered as a slave and mapped to the memory on the pre-existing SoC core which is our master. The 'Byte-RAM' is a memory space that helps to mediate the transfer of data between the Parashu Core and the IP abiding by the Axi-4 interconnect protocol. The IP is instantiated from the SoC's top module, following which the newly compiled SoC is dumped on the FPGA to further application design and testing.

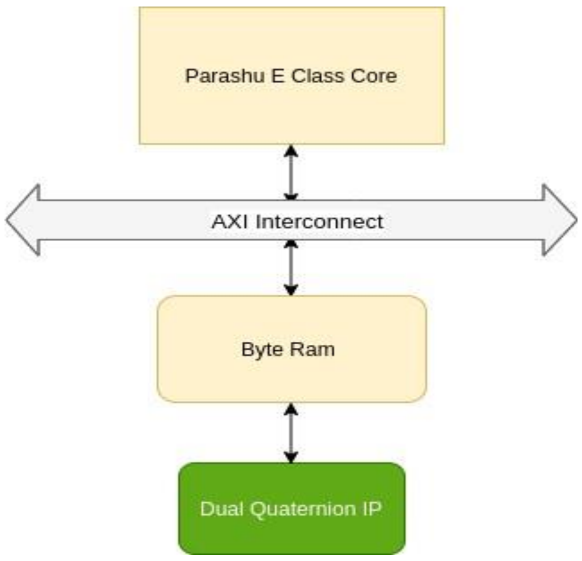


Figure 3. Integration

V. SIMULATION RESULTS

A. Quaternion vs Dual Quaternion IP

For this section comparison is made with respect to three parameters i.e. performance (in terms of execution times expressed in clock cycles), resource utilization and power utilization between the standalone SoC two proposed solutions i.e. Quaternion IP and Dual Quaternion IP outputs for which have been obtained within Vivado. For the performance analysis, a program for dual quaternion multiplication has been executed for 100 iterations and the result from an internal timer counter (that increments every 256 clock cycles) is averaged to obtain the final value of execution time.

Table II. Resource Utilization

	Parashu Core	Parashu Core + Quaternion IP	Parashu Core + Dual Quaternion IP
LUT (%)	47.92	61.97	89.99
LUTRAM (%)	5.12	5.12	5.12
FF (%)	25.99	30.18	37.54
BRAM (%)	4.44	4.44	4.44
DSP (%)	1.67	15	41.67
IO (%)	60.95	62.86	62.86
MMCM (%)	33.33	33.33	33.33
PLL (%)	16.67	16.67	16.67

Table III. Comparison Of Results

	Parashu Core	Parashu Core + Quaternion IP	Parashu Core + Dual Quaternion IP
Power Consumption (W)	0.909	0.95	1.1017
Performance (Clock Cycles)	3,05,664	39,680	10,240

Results obtained in resource utilization are for the Artix A7 100t FPGA board. While comparing the three cases, it can be observed,as per Table II that only LUT (Look Up Table), FF (Flip-flop) and DSP (Digital Signal Processors) see a significant increase whilst utilization of the other resources present on the FPGA remains relatively unchanged.The LUT,FF and DSP increase w.r.t the standalone Parashu core is observed to be just 14.05%, 4.19% and 13.33% respectively in the case of the quaternion IP whereas 42.07%, 11.55% and 40% respectively in the case of the dual quaternion IP. It is the same scenario in the case of power utilization where the dual quaternion IP depicts a 21 % increase whilst the quaternion IP is just 4.5% more than the standalone SoC's utilization as depicted above in Table III.

B. Dual Quaternion IP

If we prioritize performance gain to be the primary objective of this module, we consider our dual quaternion IP itself for the following analysis. The clock cycles taken are obtained with the help of an internal timer counter in the SoC which increments every 256 clock cycles. Hence the obtained value is multiplied by 256 to get the final clock cycle count. Here we analyze the 4 CPU intensive functions i.e. Multiplication, Translation, Rotation and Transformation all of which show a respectable decrease in execution time represented by clock cycles. As tabulated below, speedup ranging from 1.55 to 29.85 is obtained, hence justifying the case for our hardware accelerator in applications involving Dual Quaternions.

Table IV. Dual Quaternion Operations

Operations	Standalone SoC (Clock Cycles) – T1	SoC integrated with DQ IP (Clock Cycles) – T2	Speedup (T1/T2)
DQ Multiplication	3,05,664	10,240	29.85
Translation	4,82,816	37,120	13.01
Rotation	13,71,136	8,85,248	1.55
Transformation	16,61,696	9,06,752	1.83

C. FPU Argument

An additional scenario tested is to check whether the IP fares better with its inbuilt FPU unit or would its reliance on the FPU unit of the SoC be the better alternative. The test has been conducted using Verilator which is a cycle accurate RISC- V compiler since the testing required us to use the C-class ‘Parashu’ SoC which belongs to a higher tier with the support for an FPU unit is unfortunately not testable on our FPGA board.

Table V. Dual Quaternion Operations

Operation	Standalone SoC (Clock Cycles)	SoC integrated with DQ IP
DQ Multiplication	81,408	1,79,712

It is observed that the hardware accelerator performs better standalone than if it were to be relying on the SoC’s inbuilt FPU unit since it does not have to rely on the data bus for FPU operations, hence mitigating the communication delay. Additionally, this means that the IP is universally compatible with all SoCs and does not have a prerequisite of an FPU unit for attaining optimal performance.

VI. IMPLEMENTATION TRADE OFFS

As discussed earlier there is a place for both our IP’ s to accelerate dual quaternion applications. The dual quaternion IP is better when it comes to raw performance. However, if resource utilization and power are a concern then the quaternion IP is the better alternative. Not to mention the quaternion IP is applicable in both instances of quaternion and dual quaternion applications but the dual quaternion IP is restricted to the latter. The better performance and higher resource utilization of Dual quaternion compared with quaternion logic is clearly visible in Fig.4 and Fig 5.

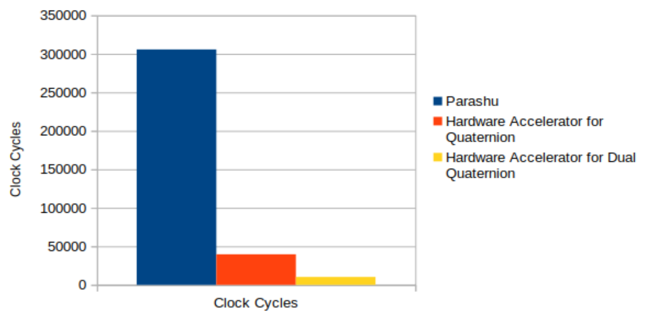


Figure 4. Comparison of Clock cycles

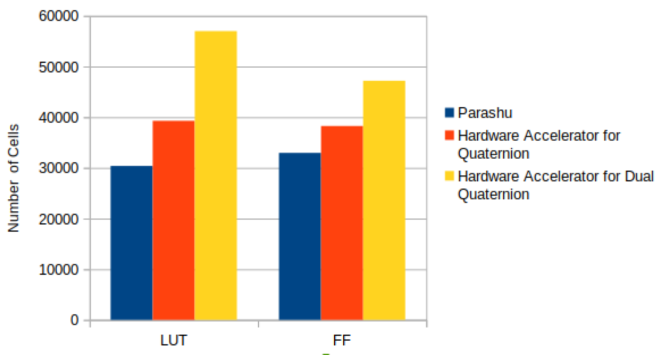
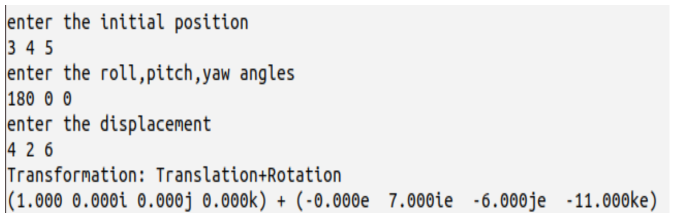


Figure 5. Comparison of Resource Utilization

VI. PRACTICAL IMPLEMENTATION

Dual Quaternion applications such as translation, rotation and transformation are visualized using rviz, which is a 3D visualization tool of ROS application [13]. The inputs are given in the form of [x, y, z, roll, pitch, yaw] where x, y, z indicates the amount of units it needs to move in x, y, z directions respectively and roll, pitch, yaw indicates the angles it needs to rotate about the x, y, z axes.



```
enter the initial position
3 4 5
enter the roll,pitch,yaw angles
180 0 0
enter the displacement
4 2 6
Transformation: Translation+Rotation
(1.000 0.000i 0.000j 0.000k) + (-0.000e  7.000ie  -6.000je  -11.000ke)
```

Figure 6. Terminal output for transformation

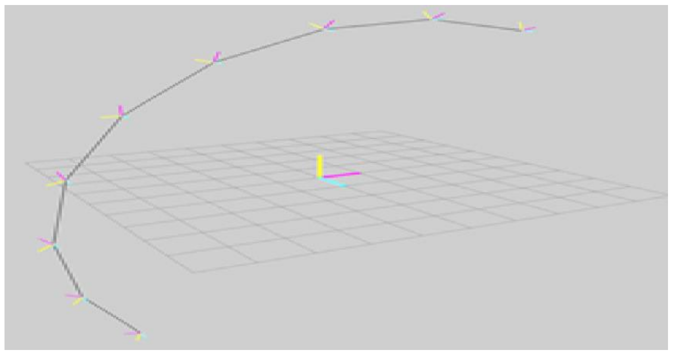


Figure 7. Visualization of transformation using rViz

The designed hardware is dumped onto the FPGA and various robotics applications have been tested [14]. The 2D bot considers its initial position as origin and as the bot moves forward, depending on the wheel movement and also the angle by which the wheel is rotated, dual quaternion is obtained, this process takes place in a loop and simultaneously updates the previous dual quaternion. In Fig.8, the left image indicates the path the bot needs to follow to reach the destination [15]. The IP does support 3D as shown in simulation and can be verified by twirling the robot manually in space and visualizing the same in ROS.

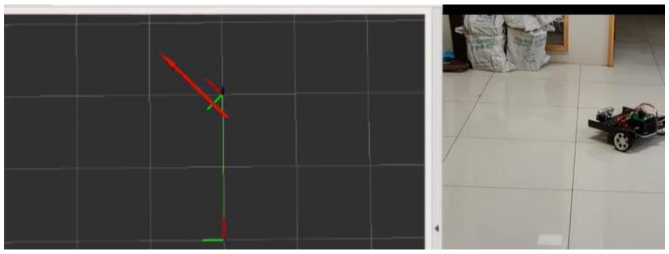


Fig 8. 2D Application for Transformation

VII.CONCLUSIONS

The primary objective of accelerating the dual quaternion applications with an IP has been achieved. The trade-off between performance and resource utilization has been discussed with the proposed solutions of our dual quaternion and quaternion IP respectively. Additionally, the proposed IP design is universally compatible with all SoC's and does not have a prerequisite of an FPU unit for attaining optimal performance. The practical implementation in the case of the robot for operations like state estimation, rotation, translation and transformation conveys that the proposed IP is industry ready for implementation as well. Although simulated in 3D, the practical implementation with the robot was restricted to work in only 2 dimensions. The future scope of the IP would be to test on a 3D application like a drone or a robotic arm.

ACKNOWLEDGMENT

We express our profound gratitude to Vishwas N.S, Research associate at PESU cRAIS (Centre for Robotics, Automation and Intelligent Systems) who has given valuable suggestions and guidance throughout the project.

REFERENCES

[1]. G. Zhang, K. Zhao, B. Wu, Y. Sun, and F. Liang, A RISC-V based hardware accelerator designed for Yolo object detection system, 2019.

[2]. W. Wang, B. Jungk, J. Walde, S. Deng, N. Gupta, J. Szefer, and R. Niederhagen, "XMSS Hardware Accelerators for RISC-V" XMSS and Embedded Systems, 2020.

[3]. N. Gala, G. Madhusudan Gs, S. Paul, M. Anmol, V. Arjun, and Kamakoti, "SHAKTI: An Open-Source Processor Ecosystem", Advanced Computing and Communications, 2018.

[4]. S. D. TEAM, SHAKTI DEVELOPMENT BOARD USER MANUAL, 2019.

[5]. G. Q. J, M. J. P. Varshney A, and C. C. F, Double quaternions for motion interpolation, American Society of Mechanical Engineers, September 1998.

[6]. A. Perez-Gracia and M. J, "Dual Quaternion Synthesis of Constrained Robotic Systems," Journal of Mechanical Design - J MECH DESIGN, 2003.

[7]. M. Gouasmi, "Robot Kinematics, using Dual Quaternions," IAES International Journal of Robotics and Automation (IJRA), 2012.

[8]. W. M. Cariow A, Cariowa G, "An FPGA-oriented fully parallel algorithm for multiplying dual quaternions," 2015.

[9]. P. A., "Fixed-Point Arithmetic Unit with a Scaling Mechanism for FPGA-Based Embedded Systems," Electronics 2021, 2021.

[10]. L. Kavan, S. Collins, C. O'Sullivan, and J. Zara, "Dual quaternions for rigid transformation blending," pp. 39–48, 2006.

[11]. K. B, A beginners guide to dual-quaternions: What they are, how they work, and how to use them for 3D character hierarchies., 2012.

[12]. B. B. Akshatha V Murthy, B Rajeshwari, SOC for image processing using SIFT accelerator, 2019.

[13]. J. Y. I. L. F. C. Figueredo, B. V. Adorno and G. A. Borges, Robust kinematic control of manipulator robots using dual quaternion representation, 2013.

[14]. S. Kirubaharan, J. P, and W. D, Low Power FPGA-based Hardware Accelerator for Autonomous Navigation of Mobile Robots.

[15]. J. Y. J. Lee, H. Chen and H. Kim, RISC-V FPGA Platform Toward ROS-Based Robotics Application, 2020.