# Harnessing Programmable Logic for Quaternion Multiplication

Yassen Gorbounov[1](✉) and Hao Chen[2]

[1] New Bulgarian University, Sofia, Bulgaria
ygorbounov@nbu.bg
[2] China University of Mining and Technology, 221000 Xuzhou, People's Republic of China
hchen@cumt.edu.cn

**Abstract.** Quaternions are four-dimensional hyper-complex numbers discovered by Sir William Hamilton in the 19th century. Compared to Euler angles, quaternions allow combining rotations in three-dimensional space and help overcome the problem of not being able to rotate about an axis regardless of rotation about other axes. They find numerous applications in various fields such as mechatronics, computer graphics and signal processing. However, the computational complexity of quaternion multiplication limits its effectiveness in real-time applications. To solve this problem, the paper proposes a hardware-oriented solution based on a programmable logic device (PLD). The design structure is based on the parallel processing and reconfiguration capabilities of the PLD, which can significantly improve the quaternion multiplication performance compared to traditional implementations based on a purely software approach. The paper covers the elaborated multiplier architecture and discusses its advantages in terms of performance and resources used. The experimental work performed highlights the flexibility of the approach used and demonstrates its effectiveness in accelerating quaternion multiplication, making it suitable for real-time applications.

**Keywords:** Quaternions · Matrix Multiplication · Hardware Acceleration · FPGA · Optimization

## 1 Introduction

Quaternions are a mathematical structure that is an extension of complex numbers obtained by adding three imaginary parts to the real part. Expressed as a four-dimensional vector, they represent a system of hyper-complex numbers [13, 34] that allows a convenient representation of rotations and changes in orientation in three-dimensional (3D) space. In recent years, they have attracted the attention of more and more researchers in many and varied scientific fields. For example, in the field of quantum mechanics, the elliptic partial differential equation (Poisson's equation), and the wave equations [4] are considered as a direct outcome of the representation of the Cauchy model of the elastic continuum, i.e. it is practical to express the wave equations using quaternions. In the electrodynamics, the quaternions naturally supports the existence of magnetic


T. Zlateva and G. Tuparov (Eds.): CSECS 2024, LNICST 609, pp. 247–260, 2025.
https://doi.org/10.1007/978-3-031-84312-9_17

monopoles, and the algebra of quaternions transfers its properties of symmetry to the classic electrodynamic theory [7]. In the area of geodesics, the representation of rotations by quaternions [23] allows an easier proof of the equivalence between descriptions of geodesic lines on SO(3) (special orthogonal group for 3D rotation) as motions with uniform angular speed and great circles on a three-dimensional sphere. In [22, 24] the emphasis is put on manipulating multidimensional signals such as color image and video processing, rotation and orientation handling, with the aid of adaptive filtering employing discrete hyper-complex circuit theory. In a 1988 paper [27] quaternions solve orientation and rotation computer graphics and animation problems, such as for splining quaternions for keyframe animation. In [34] it is discussed how to use quaternions to perform rotations in 3-dimensional space with applications in computer graphics. The authors of [20] suggest an intelligent quaternion Orthogonal Frequency-Division Multiplexing (OFDM) quaternion transform (MPFT) to improve the wireless transmission security in telecommunication systems. There can be pointed myriad of examples in the field of robotics which develops at an extremely fast pace. Many specialists in the field of robotics or computer graphics use quaternions to evaluate the interpolation of 3D rotations [6], because formulating homogeneous transformations for complex geometric objects using a point-based method is too challenging. For instance [1] introduces a novel Ju-Gibbs [19] quaternion method for kinematic modeling of robot arms, which simplifies the kinematic analysis and reduces computational complexity in modeling multiaxial rotation systems. The authors of [29] discuss the use of quaternions as a tool for path planning, modeling, and controlling robotic manipulators. In [33] it is discussed the application of dual quaternions for pose representation in spacecraft control for complex proximity operations and rendezvous tasks involving spacecraft-mounted robotic manipulators. In [2] a model-free controller for enabling the simultaneous control of both the position and orientation of a robot arm's end effector uses a new method that employs locally weighted bi-quaternions. It allows the manipulation of robotic arms without prior kinematic knowledge or joint angle information. In [5] a derivation of bi-quaternion exponential and logarithm aids the removal of zero-angle singularity, and demonstrates the efficiency of an implicit form of dual quaternions. This approach is advantageous for addressing forward and inverse position kinematics issues. Quaternions apply also in motor diagnostics and control. In [3] Quaternion Signal Analysis (QSA) employs quaternion arithmetic to rotate and describe signal dynamics, enabling accurate classification with fewer samples on an FPGA device for real-time validation. Of course, quaternions have many other applications, providing an efficient way to represent and manipulate different transforms in 3D space. For this reason, the listed applications represent only a short and non-exhaustive list of examples, which consideration falls beyond the scope of the present study.

For long time, mathematicians around the world have been working on extending the representation of numbers and the concept of algebraic structures beyond the complex numbers by trying to find their three-dimensional or four-dimensional counterparts [36]. Major research topics included algebraic systems that went beyond traditional algebra, including matrix and multidimensional structures. It was necessary to find an efficient way to represent rotations and orientations in the three-dimensional space. Irish mathematician Sir William Hamilton also joined the ranks of researchers in the search for

3-dimensional numbers. At first, he tries 3D space, but fails, proving later that it is impossible. Then he tries a four-dimensional space. In a letter to Reverend Archibald H. Hamilton, dated August 5, 1865 Sir W. Hamilton wrote [31]:

> *Every morning in the early part of the above-cited month /October, 1843/, on my coming down to breakfast, your (then) little brother William Edwin, and yourself, used to ask me, "Well, Papa, can you multiply triplets"? Whereto I was always obliged to reply, with a sad shake of the head: "No, I can only add and subtract them."*

After 15 years of effort, he succeeded in introducing ordered arrays of four real numbers where multiplication is not commutative and defined the first hyper-complex number, calling it a quaternion [9, 10]. He identified the potential of these mathematical structures for use in geometry and algebra to describe position or orientation in space. According to his own account, on October 16, 1843, during a walk along the Royal Canal near Dublin with his wife, Hamilton was suddenly inspired with the idea of introducing a fourth dimension to multiply triplets. Legend has it that while crossing Broome Bridge on the Royal Canal, Hamilton etched the newly discovered quaternion Eq. (1)

$$q = w + x.i + y.j + z.k, \textit{where } i^2 = j^2 = k^2 = ijk = -1 \tag{1}$$

into the bridge's stone worrying that he might forget it. Today, a plaque at the exact location commemorates this event (Fig. 1) [32].

*Here as he walked by on the 16th of October 1843 Sir William Rowan Hamilton in a flash of genius discovered the fundamental formula for quaternion multiplication*

$i^2 = j^2 = k^2 = i\,j\,k = -1$

*& cut it on a stone of this bridge.*


**Fig. 1.** Quaternion inscription on Brougham (Broom) Bridge, Dublin

Hamilton devoted the remainder of his life to the study of quaternions, marking the first exploration into non-commutative algebra. Following the discovery of quaternions, subsequent developments included the concepts of double complex systems, dual- or bi-quaternions, and octonions.

In the first decades after their discovery, quaternions were investigated for their potential applications in geometry, physics and astronomy. The discovery of vector calculus by Josiah Willard Gibbs and Oliver Heaviside [30] in the late 19th century slightly slowed interest in quaternions, as it was more intuitive and easy to understand, and became more popular for describing physical systems. However, the rapid development

of computer graphics, robotics, automation, aviation, scientific simulations and modeling, combined with the increasing computational capabilities of modern technology, lead to a resurgence of interest in quaternions. A key problem in their use however turns out to be multiplication, which is non-commutative. Harnessing Field Programmable Gate Arrays (FPGAs) for quaternion multiplication demonstrates significant potential because of their capability to deliver higher speed, precision, and energy efficiency. FPGAs naturally support parallel operations, enabling the concurrent multiplication of the quaternion components. This characteristic highlights the focus of the research discussed in this article.

The remainder of this work is organized as follows: Sect. 2 discusses in brief the relation between the Euler angles and quaternions. Section 3 describes the problem of quaternion multiplication and why is needed to multiply quaternions. In Sect. 4 the proposed algorithm, implementation is presented together with some experimental results. Section 5 gives directions for future improvement. The last Sect. 6 summarizes the results of the research.

## 2 Euler Angles and Quaternions

In the beginning of the 18th century, the Swiss mathematician Leonhard Euler introduced three angles of inclination to define a rigid body orientation and position in a fixed coordinate system. Euler's rotation theorem states that any rotation of a rigid body in the three-dimensional space can be achieved with a single rotation around a fixed axis. The angles associated with this rotation are the Euler angles. They include the roll angle ($\alpha$, rotation about the x-axis), the pitch angle ($\beta$, rotation about the y-axis), and the yaw angle ($\gamma$, rotation around the z-axis), as shown in Fig. 2.

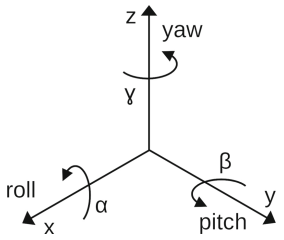



**Fig. 2.** Yaw, pitch and roll (Euler Angles)

Several problems may emerge with Euler angles representation for rotations that can affect their reliability and usability. Probably the most serious one is gimbal lock, which happens when two out of the three rotation axes coincide. In this case, a degree of freedom is lost, possibly leading to singularities. Small changes in one angle can cause large changes in another, leading to numerical instability. Multiple sets of Euler angles can produce the same orientation because the angles are periodic—that is, they suffer from non-uniqueness. All of the above said increases the overall complexity of the calculations involving Euler angles and makes their usage prone to unexpected or

non-smooth rotations in some particular cases. It is, however, possible to convert Euler angles into other representations of rotations, such as quaternions and rotation matrices, and thus avoid some limitations, like the gimbal lock. Euler angles convert to quaternions by using (2) [16]. The yaw, pitch, and roll are counterclockwise rotations around the corresponding axis in radians.

$$\begin{aligned} qw &= \cos\tfrac{roll}{2}.\cos\tfrac{pitch}{2}.\cos\tfrac{yaw}{2} \pm \sin\tfrac{roll}{2}.\sin\tfrac{pitch}{2}.\sin\tfrac{yaw}{2} \\ qx &= -\sin\tfrac{roll}{2}.\cos\tfrac{pitch}{2}.\cos\tfrac{yaw}{2} \pm \cos\tfrac{roll}{2}.\sin\tfrac{pitch}{2}.\sin\tfrac{yaw}{2} \\ qy &= -\cos\tfrac{roll}{2}.\sin\tfrac{pitch}{2}.\cos\tfrac{yaw}{2} \pm \sin\tfrac{roll}{2}.\cos\tfrac{pitch}{2}.\sin\tfrac{yaw}{2} \\ qz &= -\cos\tfrac{roll}{2}.\cos\tfrac{pitch}{2}.\sin\tfrac{yaw}{2} \pm \sin\tfrac{roll}{2}.\sin\tfrac{pitch}{2}.\cos\tfrac{yaw}{2} \end{aligned} \tag{2}$$

The various possible sequences and their reflection on the sign in the trigonometric representations [16] are illustrated in Fig. 3.

| | *roll pitch yaw* | *roll yaw pitch* | *pitch yaw roll* | *pitch roll yaw* | *yaw roll pitch* | *yaw pitch roll* |
|---|---|---|---|---|---|---|
| *qw* | - | + | - | + | - | + |
| *qx* | - | + | - | - | + | + |
| *qy* | + | + | - | + | - | - |
| *qz* | - | - | + | + | - | + |

**Fig. 3.** Euler to quaternion transformation matrix depending on the sequence of rotation

Quaternions convert to Euler angles for the sequence *Pitch-Yaw-Roll* by using (3).

$$\begin{aligned} pitch &= \tan^{-1}\left[\tfrac{-2.(qx.qz+qw.qy)}{qw^2+qx^2-qy^2-qz^2}\right] \\ yaw &= \sin^{-1}\left[2.(qx.qy - qw.qz)\right] \\ roll &= \tan^{-1}\left[\tfrac{-2.(qw.qx+qy.qz)}{qw^2-qx^2+qy^2-qz^2}\right] \end{aligned} \tag{3}$$

Other useful conversions exist, such as the Euler to matrix, matrix to Euler, quaternion to matrix, and matrix to quaternion, but they fall beyond the scope of this paper.

## 3 Quaternion Multiplication

The quaternion multiplication is a powerful tool for representing and manipulating rotations in 3D space. The fundamental rules introduced by Hamilton [15] are (4).

$$\begin{aligned} ij &= k = -ji \\ ik &= i = -kj \\ ki &= j = -ik \end{aligned} \tag{4}$$

Because these units depend on each other quaternion multiplication is noncommutative (5), unless one of its operands is a real number or both operands are complex numbers [25].

$$q.p \neq p.q \tag{5}$$

The quaternion multiplication is associative. For two quarternions $p$ and $q$, the Hamilton product is determined by the products of their basis elements. Since quaternion multiplication is distributive over addition, the product expand into a sum of the products of these basis elements. Based on the above rules the basis elements multiply as per (6) [12]:

$$\begin{aligned} q.p = (w_0 + x_0.i + y_0.j + z_0.k).(w_1 + x_1.i + y_1.j + z_1.k) = \\ = (w_0.w_1 - x_0.x_1 - y_0.y_1 - z_0.z_1).1 + \\ (w_0.x_1 + x_0.w_0 + y_0.z_1 - z_0.y_1).i + \\ (w_0.y_1 - x_0.z_1 + y_0.w_1 + z_0.x_1).j + \\ (w_0.z_1 + x_0.y_1 - y_0.x_1 + z_0.w_1).k \end{aligned} \tag{6}$$

Computing quaternion products requires performing 16 multiplications and 12 additions of real numbers in total. This task is highly compute-intensive and its performance depends on the capabilities of the hardware for vector processing [21, 26]. There exist however electronic circuits such as the Application Specific Integrated Circuits (ASIC), and the FPGAs which are inherently capable of parallel processing. The conceptual FPGA device design is the subject of the next chapter, which aims at the implementation of a digital controller for quaternion multiplication.

## 4 Algorithm Implementation

In essence, quaternion multiplication requires signed floating-point arithmetic. Programmable logic devices lack embedded floating-point units by default so they do not support the floating-point (FP) standard IEEE-754 [14]. It is not very hard to design custom floating-point unit but it may unnecessarily increase the overall complexity offering unsatisfactory throughput, and negatively affecting the design size and the execution speed. In fact there is no real reason of using FP arithmetic if preparing the numerical data in advance and downscaling the FP numbers to integer ones [8, 17, 18] as in Fig. 4.

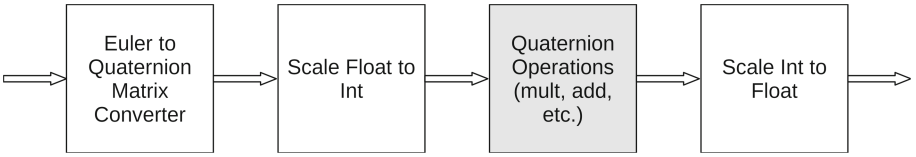



**Fig. 4.** Numbers preprocessing flow

Having (6) as the basic representation it can be seen that the quaternion multiplication requires in fact four multiply-accumulate (MAC) blocks. The DSP48E slice contained in a Virtex-5 FPGA [35] is such a digital signal processing (DSP) module that supports mathematical functions including the multiply, MAC, multiply-add, barrel shift, magnitude comparator, and others. Multiple DSP48E slices can be daisy chained to extend to wider mathematical functions. Having the above said in mind Eq. (6) can be rewritten as in Fig. 5.

| | | address 0 | address 1 | address 2 | address 3 |
|---|---|---|---|---|---|
| MAC 1 | | $w_0.w_1$ | $-x_0.x_1$ | $-y_0.y_1$ | $-z_0.z_1$ |
| MAC 2 | + | $w_0.x_1$ | $x_0.w_0$ | $y_0.z_1$ | $-z_0.y_1$ |
| MAC 3 | + | $w_0.y_1$ | $-x_0.z_1$ | $y_0.w_1$ | $z_0.x_1$ |
| MAC 4 | + | $w_0.z_1$ | $x_0.y_1$ | $-y_0.x_1$ | $z_0.w_1$ |

**Fig. 5.** The process of multiplication using four MAC blocks

The setup in Fig. 6 supplements the above figure.

| *w0* | *x0* | *y0* | *z0* |
|---|---|---|---|
| *w1* | *-x1* | *-y1* | *-z1* |
| *x1* | *w1* | *z1* | *-y1* |
| *y1* | *-z1* | *w1* | *x1* |
| *z1* | *y1* | *-x1* | *w1* |

(a)

→

| *0* | *1* | *2* | *3* |
|---|---|---|---|
| *0* | *-1* | *-2* | *-3* |
| *1* | *0* | *3* | *-2* |
| *2* | *-3* | *0* | *1* |
| *3* | *2* | *-1* | *0* |

(b)

*ROM w (real)*
*ROM x (imaginary **i**)*
*ROM y (imaginary **j**)*
*ROM z (imaginary **k**)*

(c)

**Fig. 6.** The proper coefficients can be addressed with the aid of a LUT: (a) Eq. (6) as a matrix, (b) coefficient positioning (addressed) and operation sign, (c) reference to memory elements

The above figures show that the multiplication process can be optimized if using four pipelined multiply-accumulate blocks working in parallel, and two multiplexers for supplying the $w, x, y$, and $z$ coefficients. The inputs of the first multiplexer connect with the coefficients of the first quaternion (multiplicand), and the second one with the coefficients of the second quaternion (multiplier). The use of addition and subtraction operations and the inconsistent arrangement of the multiplier coefficients, show it is convenient to drive the selector bits of the multiplexers with the aid of a Read-Only Memory (ROM)

Look-Up Table (LUT). The block diagram that implements the quaternion multiplication algorithm is shown in Fig. 7.

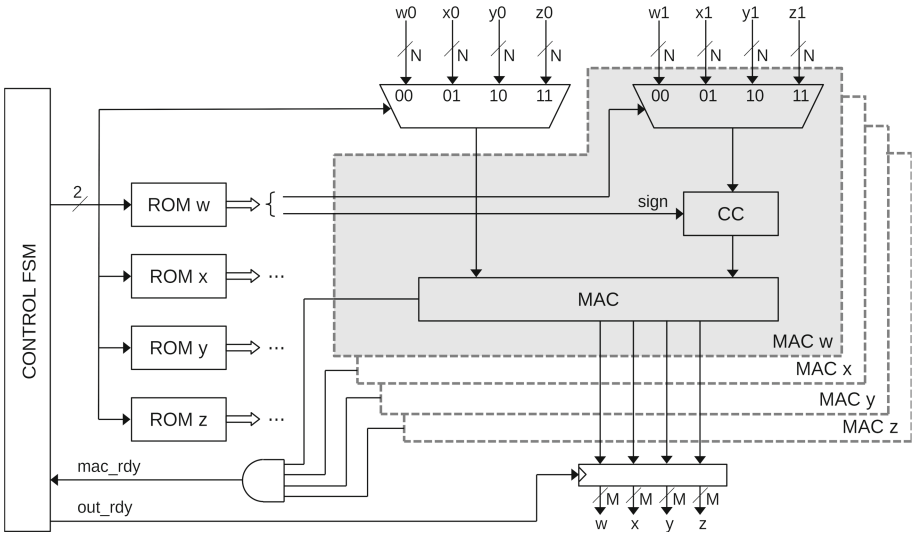
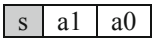



**Fig. 7.** Structure of the quaternion multiplication algorithm

Following the principles of modularity and regularity [11] the designed MAC block together with the multiplier multiplexer and the code converter (CC) is instantiated four times (one per row, see Fig. 5, 6). The control finite state machine (FSM) synchronizes the multiplication process by providing the next address that corresponds to the next quaternion coefficient once the MAC unit completes its current operation. The latter is indicated with the *mac_rdy* signal, which is generated only if all individual (per column) parallel MAC operations are completed. ROM memories store the inconsistent addresses of the multiplier coefficients so that driving the selector inputs of the multiplier multiplexer happens in an uneven order. The sign takes part of the address as shown in Fig. 8.




*Sign-magnitude representation:*
*s – sign (0 = positive, 1 = negative); {a1, a0} – address*

**Fig. 8.** Address format for the multiplexer of the multiplier

Instead of subtracting numbers the subtract operation is performed by simply inverting the sign which is done by the CC block. The functioning of the CC module is thoroughly discussed in [8]. Since the sequential multiplication may lead to intermediate erroneous results, the actual output of the digital quaternion multiplication controller have to be registered. This task is achieved by the generation of the *out_rdy* signal by the control FSM.

The sequential address change mechanism allows for one multiplication per unit time (four simultaneous multiplications per column), which ensures efficient pipeline processing, which leads to a reduction of the used hardware resources. Moreover, using ROM memories instead of hard wiring the addresses provides very high flexibility by allowing easy change of the rotation sequence as shown in Fig. 3.

The working of the MAC unit is depicted in Fig. 9.

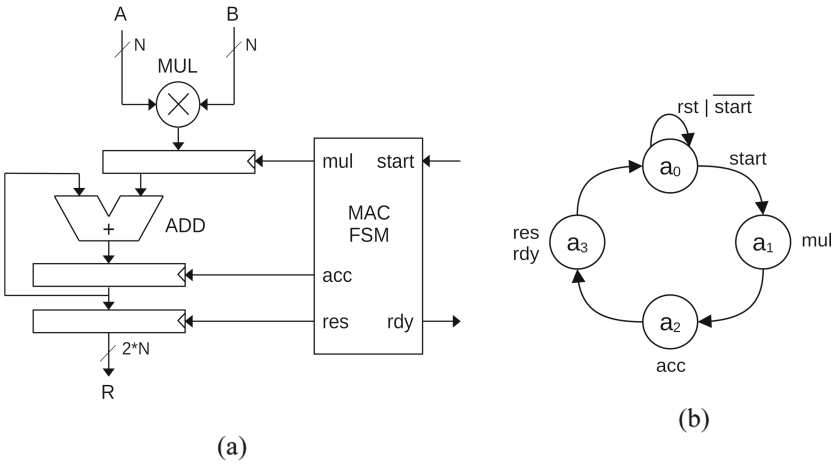



**Fig. 9.** The multiply accumulate module: (a) block diagram, (b) directed graph of the MAC FSM

The multiply accumulate circuit consists of a signed multiplier and a signed adder which outputs are buffered in two registers (Fig. 9a). The adder register allows the implementation of the integration property of the circuit (the accumulator). The actual output needs also to be registered. The controlling automaton is a Moore finite state machine (Fig. 9b), which governs the entire process. The timing diagram from the simulation of the MAC module is given in Fig. 10.

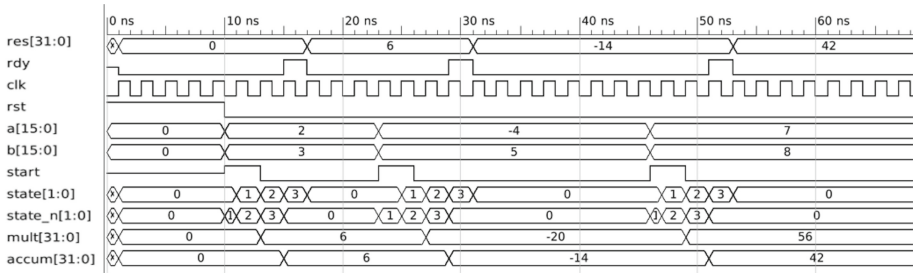



**Fig. 10.** Timing diagram of the multiply accumulate module

The simulation shows one full MAC cycle in 16-bits input mode. Some internal signals such as the intermediate *mult[31:0]* and *accum[31:0]* are provided for clarity. All

the modules in the overall quaternion multiplier controller are implemented as parameterized Verilog modules, which provide means for easily switching between 8-bits, 16-bits, 32-bits, and virtually any-bits designs where the bit width is limited by the resources of the FPGA family.

The Moore finite state machine of the control automaton for the MAC unit is shown in Fig. 11 (left). The FSM connects with the MAC and a 2-bit counter whose clock is driven by the *mac_rdy* signal (Fig. 11 right).

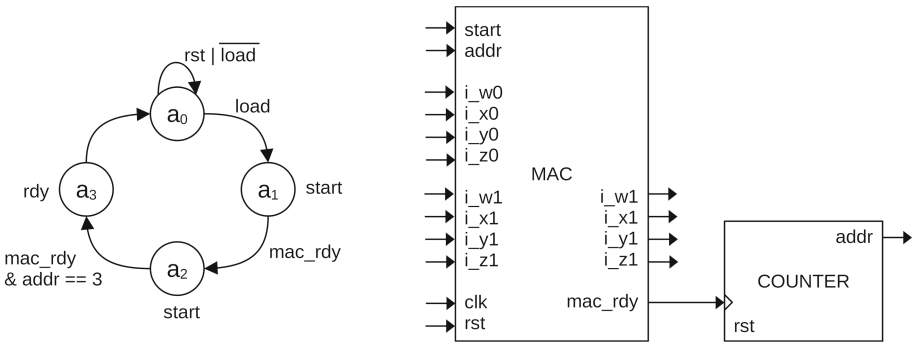



**Fig. 11.** The control algorithm of the multiply accumulate module

The simulation timing diagram of the entire quaternion multiplier module is given in Fig. 12 where the intermediate results from the multiply accumulate process flow are added for clarity.

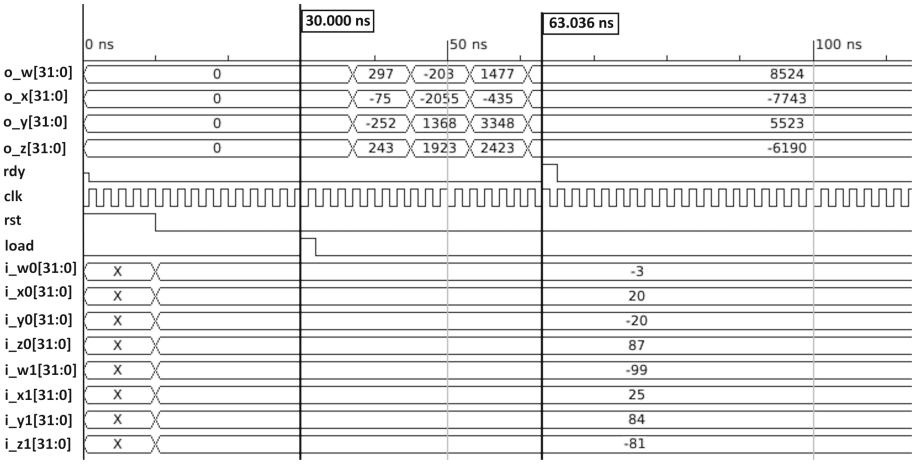



**Fig. 12.** Timing diagram of the full quaternion multiplier module

The entire design has been implemented in the Verilog hardware description language and tested in simulation on a Xilinx/AMD XC5VLX50-3-FF1153 FPGA programmable

logic device which contains a total of 48 DSP48E digital signal processing slices (an extension over the DSP48 Virtex-4 slice). The total conversion time for multiplying two quaternions takes 33.036 ns at 504 MHz on the chosen device. The resource usage statistics is summarized in Fig. 13.

| | 8-bits | | 16-bits | | 32-bits | |
|---|---|---|---|---|---|---|
| | Utilized | Usage [%] | Utilized | Usage [%] | Utilized | Usage [%] |
| Number of Slice Registers | 142 | 0% | 270 | 0% | 526 | 1% |
| Number of Slice LUTs | 156 | 0% | 315 | 1% | 611 | 2% |
| Number of used LUT-FF pairs | 77 | 34% | 141 | 31% | 269 | 30% |
| Number of bonded IOBs | 132 | 23% | 260 | 46% | 516 | 92% |
| Number of BUFGCTRLs | 1 | 3% | 13 | 40% | 13 | 40% |
| Number of DSP48Es | 4 | 8% | 4 | 8% | 16 | 33% |

**Fig. 13.** Hardware resource usage for XC5VLX50-3-FF1153 FPGA

Enhancing the calculation efficiency of the multiply-accumulate (MAC) unit can significantly boost clock speed, reduce instruction time, and improve overall operation performance, thereby accelerating hardware compute intensive tasks such as the quaternion multiplication.

## 5 Future Work

While the current implementation of the hardware-oriented solution has proven effective in accelerating quaternion multiplication and enhancing performance for real-time applications, several directions for future work can be identified, that can further advance the research field. One such goal is performing a comprehensive power consumption analysis to identify potential areas for energy optimization. Another future goal is to substitute the current quaternion multiplier as a submodule in a more generalized quaternion processor (QP) as illustrated in Fig. 14.

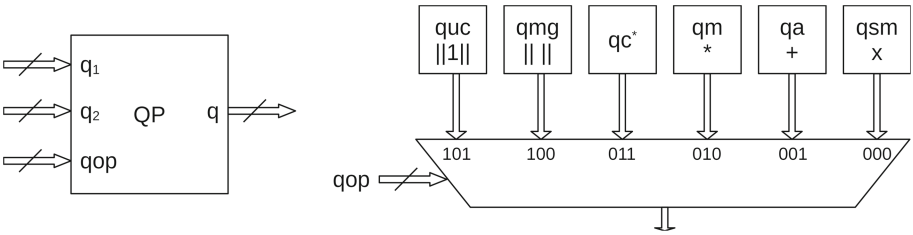



**Fig. 14.** Conceptual overview of the quaternion processor

In the figure the QP is controlled by the quaternion operation (*qop*) which selects between quaternion scalar multiplication (*qsm*), quaternion addition (*qa*), quaternion multiplication (*qm*), quaternion conjugate (*qc*), quaternion magnitude (*qm*), and quaternion unit condition (*quc*) operations.

Finally, the complexity of algebraic calculations can be cut down by employing the concept of dual quaternions. They offer singularity-free alternative to conventional Cartesian coordinate system by effectively transforming the algebraic rotors (quaternions) to motors (bi-quaternions) thus unifying the translation and rotation into a single state. This task can be accomplished by extending the current algorithm in accordance with Fig. 15 [28].

| $Q_0.Q_1$ | $Q_1$.real | $Q_1$.i | $Q_1$.j | $Q_1$.k | $Q_1$.εi | $Q_1$.εj | $Q_1$.εk | $Q_1$.ε |
|---|---|---|---|---|---|---|---|---|
| $Q_0$.real | 1 | i | J | K | *εi* | *εj* | *εk* | *ε* |
| $Q_0$.i | i | -1 | K | -j | -ε | *εk* | *-εj* | *εi* |
| $Q_0$.j | j | -k | -1 | I | *-εk* | *-ε* | *εi* | *εj* |
| $Q_0$.k | k | J | -i | -1 | *εj* | *-εi* | *-ε* | *εk* |
| $Q_0$.εi | *εi* | *-ε* | *εk* | *-εj* | 0 | 0 | 0 | 0 |
| $Q_0$.εj | *εj* | *-εk* | *-ε* | *εi* | 0 | 0 | 0 | 0 |
| $Q_0$.εk | *εk* | *εj* | *-εi* | *-ε* | 0 | 0 | 0 | 0 |
| $Q_0$.ε | *ε* | *εi* | *-εj* | *-εk* | 0 | 0 | 0 | 0 |

**Fig. 15.** Multiplication of dual quaternions

In this figure, ε is the dual unit (7):

$$\varepsilon^2 = 0 \, and \, \varepsilon \neq 0 \tag{7}$$

This property allows the dual number to represent infinitesimal quantities. Combining dual number theory with quaternions, producing dual quaternions, provides an efficient mathematical framework for representing rigid transformations in the 3Dspace.

## 6 Conclusions

This paper presents a novel method for implementing a hardware-oriented mathematical computing device for quaternion multiplication. The proposed solution is evaluated and validated in simulation. Experimental results show promising performance and accuracy. The design method offers a parameterizable configuration, which allows the bit width to be scaled according to specific needs. The approach is modular, and opens doors for future work towards implementing a more complex generalized quaternion processor that is capable of performing simultaneous rotation and translation via dual quaternion calculations. The designed module can find a broad range of applications in mechatronics, computer graphics and signal processing. The authors hope that the proposed work may provide new ideas for committed researchers in the field of digital computing and robotics.

**Acknowledgments.** This research is supported by a 2024 grant issued by The Central Fund for Strategic Development of the New Bulgarian University.

## References


1. Ahmed, A., Ju, H., Yang, Y., et al.: An improved unit quaternion for attitude alignment and inverse kinematic solution of the robot arm wrist. Machines **11**(669) (2023). https://doi.org/10.3390/machines11070669
2. AlAttar, A., Kormushev, P.: Kinematic-model-free orientation control for robot manipulation using locally weighted dual quaternions. Robotics **9**(76) (2020). https://doi.org/10.3390/robotics9040076
3. Contreras-Hernandez, J., Almanza-Ojeda, D., Ledesma, S., et al.: Motor fault detection using quaternion signal analysis on FPGA. Measurement **138**, 416–424 (2019). https://doi.org/10.1016/j.measurement.2019.01.088
4. Danielewski, M., Sapa, L.: Foundations of the quaternion quantum mechanics. Entropy (12), 1424 (2020). https://doi.org/10.3390/e22121424
5. Dantam, N.: Robust and efficient forward, differential, and inverse kinematics using dual quaternions. Int. J. Robot. Res. **40**(10–11), 1087–1105 (2021). https://doi.org/10.1177/0278364920931948
6. Bayro-Corrochano, E.: A survey on quaternion algebra and geometric algebra applications in engineering and computer science 1995–2020. IEEE Access **9**, 104326–104355 (2021). https://doi.org/10.1109/ACCESS.2021.3097756
7. Giardino, S.: Quaternionic electrodynamics. Modern Phys. Lett. **35**(39) (2020). https://doi.org/10.1142/S0217732320503277
8. Gorbounov, Y., Chen, H.: Context-switching neural node for constrained-space hardware. In: Zlateva, T., Goleva, R. (eds.) Computer Science and Education in Computer Science, CSECS 2022. LNICST, vol. 450, pp. 45–59. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17292-2_4
9. Hamilton, W.: On quaternions. Proc. R. Irish Acad. **3**, 1–16 (1847)
10. Hamilton, W.: Theory of quaternions. Proc. R. Irish Acad. 1836–1869 **3**, 1–6 (1844)
11. Harris, D., Harris, S.: Digital Design and Computer Architecture, 2nd edn. Morgan Kaufmann Elsevier (2013). ISBN 978-0-12-394424-5
12. Hazewinkel, M., Gubareni, N., et al.: Algebras, Rings and Modules. Mathematics and Its Applications, vol. 575, no. 1 (2006). ISBN 978-1-4020-2690-4
13. Ibrayev, A.: Method for constructing a commutative algebra of hypercomplex numbers. Symmetry **15**(1652) (2023). https://doi.org/10.3390/sym15091652
14. IEEE Std 754-2019, IEEE Computer Society. 2019. IEEE Standard for Floating-Point Arithmetic IEEE STD 754-2019, pp. 1–84, ISBN 978-1-5044-5924-2
15. Jia, Y.: Quaternions and Rotations. Com S **477**(577) (2015). https://api.semanticscholar.org/CorpusID:7075201
16. JSBSim Open Source Flight Dynamics Software Library. https://jsbsim.sourceforge.net/
17. Karatsuba, A.: The complexity of computations. Proc. Steklov Inst. Math. **211**, 169–183 (1995). Translation from Trudy Mat. Inst. Steklova, 186202, 169–183 (1995)
18. Kastner, R., Matai, J., Neuendorffer, S.: Parallel Programming for FPGAs, The HLS Book. ArXiv e-prints (2022). https://doi.org/10.48550/arXiv.1805.03648
19. Kruglov, S., Barzda, V.: Modified Gibbs's representation of rotation matrix. Annales de la Fondation Louis de Broglie **42**(2), 1–15 (2017). https://doi.org/10.48550/arXiv.1703.00300
20. Labunets, V.G., Ostheimer, E.: Intelligent OFDM telecommunication systems based on many-parameter complex or quaternion Fourier transforms. In: Hu, Z., Petoukhov, S., He, M. (eds.) Advances in Intelligent Systems, Computer Science and Digital Economics, CSDEIS 2019. AISC, vol. 1127, pp. 129–144. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39216-1_13

21. Leiterman, J.: Vector Games Math Processors (Wordware Game Math Library). Wordware Publishing (2005). ISBN 978-1556229213
22. Miron, S., Flamant, J., Bihan, N., et al.: Quaternions in signal and image processing: a comprehensive and objective overview. IEEE Signal Process. Mag. **40**(6), 26–40 (2023). https://doi.org/10.1109/MSP.2023.3278071
23. Novelia, A., O'Reilly, O.: On geodesics of the rotation group SO(3). Regul. Chaot. Dyn. (2015). https://doi.org/10.1134/S1560354715060088
24. Ortolani, F., Uncini, A.: Quaternion digital signal processing: a hypercomplex approach to information processing. In: International Siberian Conference on Control and Communications (SIBCON), pp. 1–7 (2016). https://doi.org/10.1109/SIBCON.2016.7491831
25. Parfieniuk, M., Park, S.Y.: Versatile quaternion multipliers based on distributed arithmetic. Circuits Syst. Signal Process. **37**, 4880–4906 (2018). https://doi.org/10.1007/s00034-018-0789-5
26. Parfieniuk, M., Petrovsky, N., Petrovsky, A.: Rapid prototyping of quaternion multiplier: from matrix notation to FPGA-based circuits. In: Rapid Prototyping Technology – Principles and Functional Requirements. InTech (2011). https://doi.org/10.5772/20939
27. Pletincks, D.: The use of quaternions for animation, modelling and rendering. In: Magnenat-Thalmann, N., Thalmann, D. (eds.) New Trends in Computer Graphics, pp. 44–53. Springer, Heidelberg (1988). https://doi.org/10.1007/978-3-642-83492-9_4
28. Rajeshwari, B., Rithvik, K., Shweta P., et al.: Dual quaternion hardware accelerator for RISC-V based system. Int. J. Res. Sci. Innov. **9**(5), 77–81 (2022). ISSN 2321-2705
29. Ricardo, C., Camarillo-Gómez, K.A.: Unit quaternions: a mathematical tool for modeling, path planning and control of robot manipulators (2008). https://doi.org/10.5772/6197
30. Rocci, A.: Back to the roots of vector and tensor calculus. Heaviside versus Gibbs. History and Philosophy of Physics. arXiv:2010.09679 (2020). https://doi.org/10.48550/arXiv.2010.09679
31. Sachdev, S., Krishnaswami, G.: Algebra and geometry of Hamilton's quaternions. Reson. J. Sci. Educ. **21**(6), 529–544 (2016). https://doi.org/10.48550/arXiv.1606.03315
32. Tevian, D., Corinne, M.: The Geometry of the Octonions. World Scientific, pp. 1–228 (2015). ISBN 978–9814401814, https://doi.org/10.1142/8456
33. Valverde, A., Tsiotras, P.: Spacecraft robot kinematics using dual quaternions. Robotics **7**(64) (2018). https://doi.org/10.3390/robotics7040064
34. Vince, J.: Quaternions for Computer Graphics. Springer, London (2021). https://doi.org/10.1007/978-1-4471-7509-4. ISBN 978-1-4471-7508-7
35. Virtex-5 FPGA XtremeDSP Design Considerations User Guide. https://docs.amd.com/v/u/en-US/ug193, UG193 (v3.6), 27 July 2017. Accessed 01 March 2024
36. Ying-Qiu, G.: Miraculous hypercomplex numbers. Math. Syst. Sci. **1**(1) (2023). https://doi.org/10.54517/mss.v1i1.2258