Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 2 (2025)

FPGA Implementation of Compact Architecture for Lightweight Hash Algorithm for Resource Constrained Devices

Preeti R. Lawhale[1*], Dr. Sujata N Kale[2], Dr. Hemant Kasturiwale [3], Dr. Yogesh N. Thakare[4]

1PhD Scholar, Department of Applied Electronics, Sant Gadge Baba Amravati University Amravati, India
2 Professor ,Department of Applied Electronics, Sant Gadge Baba Amravati University Amravati, India
3Professor, Electronics and Computer Science department ,Thakur college of engineering and Technology Mumbai
4Assistant Professor,School of Computer Science and Engineering, Ramdeobaba University, Nagpur,India
1*preeti.lawhale88@gmail.com, 2sujatakale@sgbau.ac.in, 3yash4767@gmail.com, 4hemant.kasturiwale@tcetmumbai.in

Article History:

Received: *16-07-2024*

Revised: *06-09-2024*

Accepted: *28-09-2024*

Abstract:

The security implications of the emerging world of smart devices brought about by the Internet of things' recent rise are enormous. IoT devices typically have restricted resources, such as minimal memory, low processing power, and a short battery life, in addition to their stated security level. In the context of IoT, lightweight cryptographic primitives are presented taking into account the trade-off between speed and security assurance. In this research, we proposed lightweight cryptographic hash methods and optimized hardware for devices with limitations. The size, speed, efficiency, and power consumption of our suggested solutions are examined on FPGA systems. Bit permutation, linear transformation, and S-Box functionality are used in the suggested design. To analyze large data sets and Internet of Things applications, conventional hash algorithms need memory and time. Thus, the demand arises for a lightweight cryptographic protocol that is both rapid and safe. This technique offers the necessary security criteria of conventional hash functions and the parameters for lightweight cryptographic protocols. The proposed architecture is tested and validated based on three key criteria: memory, speed, and power consumption, which are important factors in determining its lightweight nature. Based on the findings, the proposed architecture outperforms other lightweight protocols regarding memory usage, performance, and power consumption. For an appropriate FPGA-based application deployment, a thorough comparison of our suggested designs is detailed on various FPGA families.

Keywords: Lightweight Cryptography, FPGA, IoT, Hash Function, High performance, High throughput.

1. Introduction

Over the past two decades, a wide range of networked devices have proliferated in an ecosystem known as the Wireless Sensor Networks (WSN) / Internet of Things (IoT), leading to the emergence of Lightweight Cryptography (LWC). Large amounts of data are produced by these devices. The majority of IoT devices are vulnerable to hacking, nevertheless, as they lack strong security and privacy measures. Since these linked, low-cost, ubiquitous devices only do a few calculations, it is crucial to satisfy their security and privacy requirements. The computational complexity of the

standard cryptographic primitives meant that they could not meet the security and privacy requirements of the Internet of Things/WSN paradigm. While it was designed to fulfill certain use cases, it sacrifices versatility and has a narrower scope of use, lightweight cryptography is not predestined to replace traditional encryption while it remains effective. Creating a broad variety of interconnected devices with diverse applications—like health monitoring systems, automated supply chain management, public transit, phone cards, etc.—is the aim of lightweight cryptography.

Various tagging technologies enable the identification of interconnected edge devices in WSN/IoT, enabling communication and information sharing among all end nodes and physical objects. Authentication, confidentiality, and integrity services are required for the safe exchange of data between end devices in IoT/WSN. Limited computing power, smaller RAM and ROM space, and reduced physical footprint are some of the new constraints for working in these contexts. However, these limited Internet of Things devices/end nodes rely on conventional security methods [1], which need greater processing power and RAM. Because of this, the discipline of lightweight cryptography (LWC) emerged through the use of IoT-enabled devices. As a result, several hardware and software versions of efficient encryption methods have been developed specifically for IoT applications. Hashing functions, stream ciphers, and block ciphers are the general categories for these. The cost of software deployments is reduced, and production and maintenance flexibility is increased. The literature does, however, suggest two basic categories of lightweight cryptography based on several application types that have properties related to both software and hardware (SW) and HW. Focused on IoT/WSN end nodes, ubiquitous LWC addresses their diverse end applications, necessitating distinct security protocols. Generally speaking, ubiquitous lightweight cryptography serves as the foundation for most research. While using standard resources already in place, ultra-lightweight cryptography may nevertheless be implemented. To make these lightweight cryptographic primitives more efficient and have a smaller footprint, a customized or optimized solution is always required.

The two types of lightweight cryptography are symmetric and asymmetric, much like in conventional cryptography. A cryptography system that has to analyze data quickly and transmit valid data must include a secure hash algorithm. IoT applications that entail frequent and sensitive data transfers need a lightweight cryptographic hash function since these devices must first authenticate themselves in order to connect. With an emphasis on IoT/WSN applications, this study focuses on lightweight hash primitives. Lightweight cryptographic primitives may be used in a variety of contexts, including IoT-based healthcare. This is a place where a lot of personal information is gathered, and the safety of the medical care a patient receives can have an effect on their ultimate outcome [3]. Since these devices connect medical databases and cloud computing servers to the edge/fog computing layers, implementing various cryptographic primitives such authentication, encryption, and authenticated encryption is difficult due to resource limits. Among the many problems, difficulties, and benefits that have resulted from the deployment of blockchain technology in the IoT architecture is their own growing field [4]. To integrate blockchain into the Internet of Things, linked nodes must authenticate. The development of a strong Internet of Things architecture is made easier by the discovery that stump-based hash primitives are more appropriate for blockchain-based applications in terms of both energy and performance consumption [5].

In this study, a method that prioritizes data reuse, minimizes memory access, and maximizes performance per clock cycle is presented. It does this by analyzing each individual operation that constitutes the lightweight hash algorithm as well as the data dependencies associated with it. This evaluation allowed us to reduce the amount of basic operations and build the arithmetic and logic units, as well as the data pipeline, that were required. An iterative design method was applied in order to reduce hardware needs and shorten the critical route.

The work's primary contributions are:

- Development of an efficient lightweight cryptographic hash algorithm that offers both strong security and high-speed hashing capabilities.
- Develop a lightweight framework capable of handling big data, as well as supporting applications and devices related to IoT.

The rest of the paper content are structured as given. In section 2, the related work on other conventional and lightweight hash-based cryptographic algorithms is discussed, and the proposed lightweight processor's hardware design is outlined in section 3. In section 4, the findings and parallels to related research are discussed. This work is brought to a conclusion in section 5.

2. Literature Review

These techniques are shown to be successful on two FPGA boards [11]. It is detailed how the ECIES protocol may be realized on hardware using a secure SoC [12]. A practical two-stage hardware-accelerated attack against SHA-3-based MAC was shown [13]. Variational irreducible polynomials may be implemented on FPGA with a novel method based on a hashing algorithm [14]. It is possible to implement a hash unit on an FPGA [15]. A SHA-256 FPGA hardware module was used in the development of the IEEE 1609.2 vehicle communications (VC) security protocol [16]. OpenCL compatibility is enhanced in the Bob Jenkins lookup3 hash function's open-source implementation [17]. An IP that interacts with Cortex-M0 is a round-iterative 1600-bit data-path SHA3 design that was described [18]. An effective processor built on an FPGA is detailed, together with Keyed-Hash Message Authentication Codes (HMAC) using SHA3-224 [19]. An ideal implementation of this function, which seeks to decrease circuit complexity while boosting speed, is shown [20], which might be useful for hardware implementations of the cryptographic hash function.

A methodology for automating the creation of hash algorithms suitable for network flow hashing is presented [21]. Using hardware HDL and FPGA synthesis on a Xilinx Virtex-4, the pipelined architecture of the SHA-256 hash function has been optimized [22]. A hardware framework for quickly developing algorithmic hardware implementations to accelerate cryptographic hash functions on the Xilinx ZYNQ SoC was presented [23]. This innovative hybrid architecture, based on a three-stage pipeline structure, may process many blocks in parallel to improve performance [24]. This article compares three potential approaches for SHA-3 hash algorithm implementation on FPGAs [25]. A generic approach for parameterizing cellular automata rules is presented [26], In addition, the text refers to applying one-dimensional cellular automata using parameters. The SHA256 algorithm implementation alternatives for hardware and software are recommended [27]. A large collection P of l-byte patterns is recorded in advance in this efficient FPGA implementation of the Bloom filter

[28]. The Troika hash function's hardware designs provided reconfigurable devices with nearly zero circuitry use and enough speed [29]. The FPGA implementations of the Secure Hash Algorithms were compared and shown by the three distinct hashing standards [30]. The newly introduced hash algorithms based on irregular programs are responsible for unpredictable code paths and input data [31].

3. Proposed work

We provide a lightweight hash algorithm that works with IoT-sensitive gadgets. The main layout of the suggested design is seen in Figure 2. The padding approach is used to increase the size of an input message (M) that has a variable length such that it is more than 512. Next, 512-bit blocks of identical size, *B1, B2,..., Bn*, are formed from the padded message *(M)*. The compression function F applies a bijective operation on the split blocks, performing XOR operations with a piece of the Initial Hash Value (IHV) both before and after the processing of the compression function. The Initial Hash Value (IHV) has a capacity of 1024 bytes and is initialized according to the desired output hash size. For instance, when a 256-bit output hash size is needed, the initial two bytes of the Initial Hash Value (IHV) will be 0x0100, while the rest of the bytes will be filled with zeros. The padding, parsing, IHV setup, compression function computations, and final hash creation steps make up the five stages of the suggested design. In the subsequent paragraphs, each stage are discussed in detail.

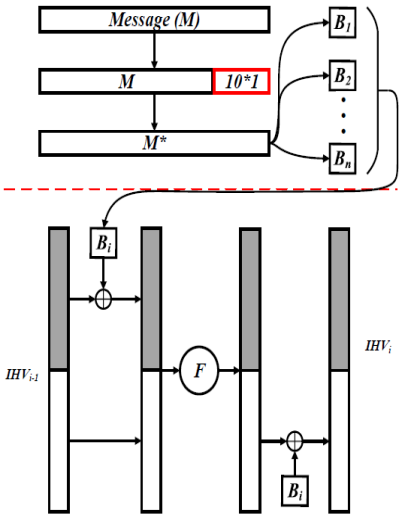

Figure.1 The architecture of lightweight hash processor

3.1 Message Bit Padding

The padding strategy was employed in the proposed design. The following algorithm, Algorithm 1, illustrates the primary framework of this strategy. The objective of padding an input message (M) with variable length is to augment its size to exceed the block size (B). The remaining bits are calculated by dividing the message's length by the block size, as the procedure illustrates, and this quantity is then needed for padding. Next, two 1s are added to the input message to indicate the

padding boundaries, together with the necessary amount of bits (P). After padding, the input message is represented by M, which is created.

3.2 Parsing of the Padded Message

The padded message is split up into 512-bit equal-size blocks *(B1, B2,..., Bn*). Eight 64-bit words make up a 512-bit block; for example, the message block Bi is represented by the eight words *B0i, B1i, B2i,..., B7i*. The computation of the compression function receives each message block.

3.3 Initialization of Hash Values

During this phase, the approach employed by the JH-hash submission [18]. The Initial Hash Value (IHV) is determined by the intended length of the output hash. The size of the output hash is stored in the first two bytes of the IHV, while the remaining bytes of the IHV are set to zero. The suggested design accommodates many hash lengths, such as 160, 224, 256, 384, and 512.

3.4 Compression Function Calculation

The equation demonstrates the sequential handling of message blocks with the compression function F. The compression function consists of five steps: bit grouping, S-box, linear transformation, permutation, and degrouping. During the grouping stage, the input message block Bi and the Initial Hash Value (IHV) are combined in such a way that each input (message block and IHV) contributes two bits to each S-Box. Using the round constant (Rc), the S-Box step determines the S-Box to utilize. The bits are shuffled before being permuted to the next step by the linear transformation layer. This procedure keeps on for every block until the last of the eighteen rounds. After the current block is processed, the output is sent as IHV to the computation of the next block. Each block in the suggested design goes through eighteen cycles, the number of which is determined.

After the input is split into two words in the grouping stage, it is sent to the S-box selection layer. The selection of an S-Box from the *4 × 4-bit* S-Boxes is determined by the round constant (Rc). The input received by the S-Box includes an equivalent number of bits from both the message block and the Initial Hash Value (IHV). The linear transformation layer (L) employs an XOR operation to sequentially transfer two words to the received words. When the Pd permutation is applied, the words are rearranged and placed next to different words in the subsequent round of block computing. To create the final hash, all bits are moved back to their original positions during the degrouping step, which follows the processing of the 18 rounds. After the final block is processed in our design, the result is 1024 bits. In order to generate the final hash, the LSB of needed bits are taken in the hash generating process. It displays the inside round function's overall structure. It displays the permutations and passes through the linear transformation and S-Box layers for the input data *(x0, x1,..., x15).* The all equations mentioned below are used to calculate all the functions in the SHA-256 algorithms.

S0 = (a rotateright 2) xor (a rotateright 13) xor (a rotateright 22);

S1 = (e rotateright 6) xor (e rotateright 11) xor (e rotateright 25);

Ch (e,f,g) = (e and f) xor ((not e) and g);

Maj (a,b,c) = (a and b) xor (a and c) xor (b and c);

temp1 = h + S1 + ch + k[i] + w[i];

temp2 = S0 + maj;

3.5 Generation of the Final Hash

Taking the least significant number of bits from the final output generates the final hash. For example, if 256-bit hash required, the final hash would be the least significant 256 bits.

4. Experimental Results and Discussion

4.1 Implementation flowchart

Figure 2 depicts the implementation of proposed framework based on FPGA design. Initially, design specification decided with parameters and implementation platform. After developing the specified RTL design, functional simulation is performed to check the behavior of proposed architecture as per the designed specification. If the function verification is correct then FPGA synthesis and implementation on it is performed with timing simulation on RTL design. If synthesis is performed successfully then functional FPGA implementation of proposed architecture is possible with successful design. Altera Quartus II IDE was utilized for the design analysis, synthesis, placement, and routing of the suggested lightweight hash design, which was developed in VHDL. Modelsim simulator was utilized for functional simulation of proposed architecture.

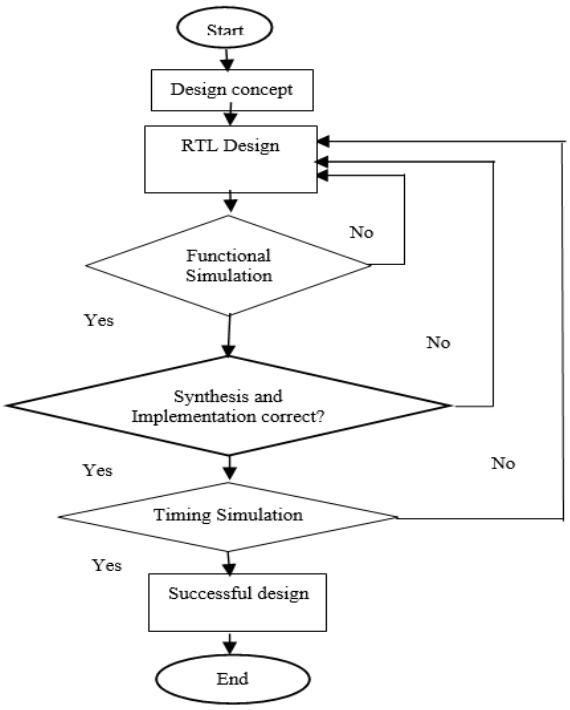

Figure.2 Implementation flowchart design

4.2 Experimental Setup

Altera Quartus II was utilized for the design analysis, synthesis, placement, and routing of the suggested lightweight hash design, which was developed in VHDL. To transfer the recommended VHDL design to an Altera FPGA board, Quartus II is utilized. ISE tools are used to simulate the VHDL design of the proposed architecture and examine its data flow. We compare every variable to a MATLAB-emulated version of the standard lightweight hash method at the conclusion of each loop. Tables provide the synthesized device utilization summary for the suggested architecture and a performance comparison between the recommended approach and other designs.

4.3 Evaluation Parameters

The FPGA implementation of SHA standards necessitates memory and speed due to the large number of computations kept during hash synthesis. In order to get the most out of an FPGA, it's essential to optimize the architecture in terms of speed and memory. The following metrics must be considered while evaluating FPGA performance:

1. *Logic Resource Utilization:* Indicative of the overall amount of lookup tables or configurable logic blocks employed in a given design. However, because to variations in placements methods between FPGA manufacturers, there may be situations in which a direct comparison of area is misleading. Therefore, using an FPGA from the same vendor is advised for this purpose.
2. *Minimum Propagation Delay:* A signal's latency is the time it takes to go from its source to its intended destination signal.
3. *Maximum Frequency:* The maximum possible operating clock rate for a certain FPGA.
4. *Power Consumption:* It is indicative of the overall hardware power consumption when a certain design is implemented. This characteristic is typically characterized by its frequency.
5. *Throughput:* Hardware implementation speed is measured in terms of the number of bits processed in a particular amount of time. The formula for throughput:

$$Throughput = (Block\_Size) / (T * Nclk)$$

6. *Efficiency:* Ration of throughput to number of slices. It is defined by below equation.

$$Efficiency = Throughput / Number\ of\ Slices$$

4.4 Result Analysis

Using Modelsim simulator, the lightweight hash's functionality is constructed and confirmed. Figures 3 and 4 present the input data and the hashed output message as a consequence of the time simulation. Initially, input data is applied at every positive edge of the clock with an active low reset while enable is high, and an output hash value is created after a series of cycles. To produce the output hash values, the various test data are tested. The produced hash values are used to confirm the output hash values once again.

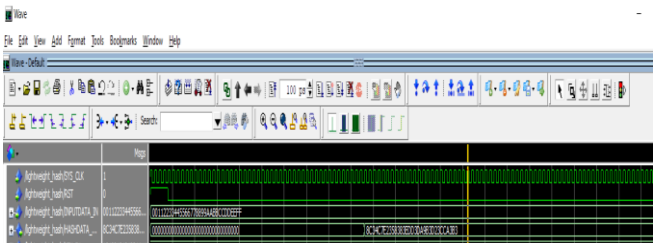

Figure.3 Simulation result of lightweight hash with input1

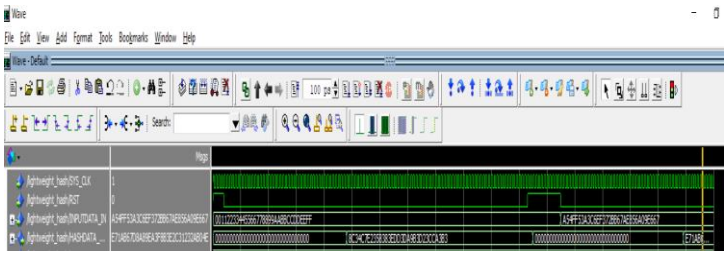

Figure.4 Simulation result of lightweight hash with input2

Tables 1 show the full implementation of SHA256 on a range of ALTERA FPGA devices. The throughput increases approximately linearly with the quantity of high-performing FPGAs in each of the assessed FPGA families. Since the throughput is mostly determined by the number of clock cycles, this makes sense. The design on Stratix III reaches a maximum throughput of 1090.512 Mbps with an efficiency of 0.327 Mbps per logical resource (lr).

Table. 1 Performance Evaluation of SHA256 on various FPGA device platforms

Evaluation Parameters	**FPGA Devices**			
	CycloneII-EP2C35F672C6	**CycloneIV-EP4CE75F29C7**	**StratixII-EP2S15F672I4**	**StratixIII-EP3SE50F780I4L**
Logic Resources (LR)	2835	2880	2236	2378
Min Propagation Delay (ns)	5.42	2.12	4.78	2.06
Max Frequency (MHz)	98.55	150.45	100.80	143.85
Power Consumption (mW)	50.74	38.64	25.44	30.91
Throughput (Mbps)	300.54	840.425	280.478	700.546

The performance of several FPGA devices is visualized in figure 4-9 with respect to logic resources, minimal propagation delay, maximum frequency, power consumption, throughput, and efficiency. The Cyclone and Stratix FPGA families, which are often low power and high-performance FPGA, are the two primary FPGA kinds on which the evaluation performance is concentrated. A Stratix II FPGA device with a minimal logic resource usage of 3321 results in a minimal power demand of 31.64 mW. I/O thermal power consumption accounts for the power usage. The StratixIII device, which has strong FPGA performance, has a minimum propagation delay of 3.668 ns, also referred to as clock to output latency. Due to its increased throughput, the StratixIII device has an efficient efficiency performance (0.327) in terms of the ratio of throughput to logic resources used.

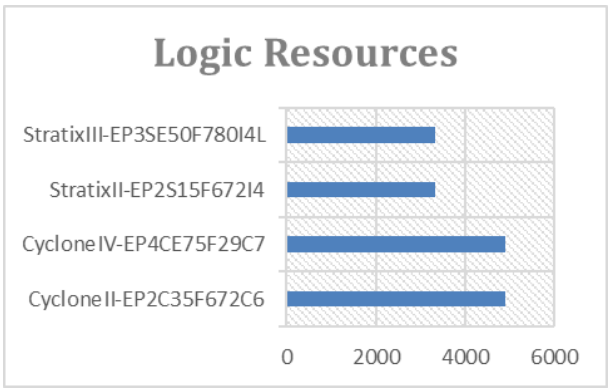

Figure.5 Logic Resource Utilization Performance

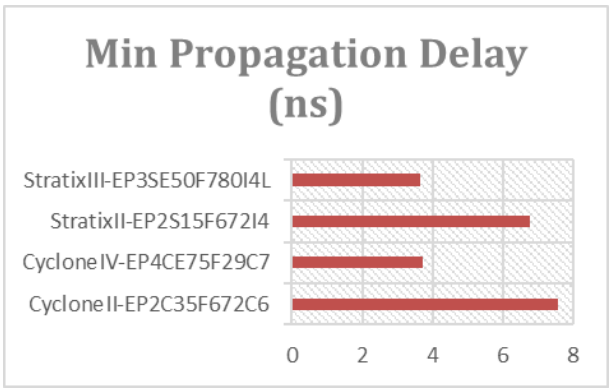

Figure.6 Minimum Propagation Delay Performance

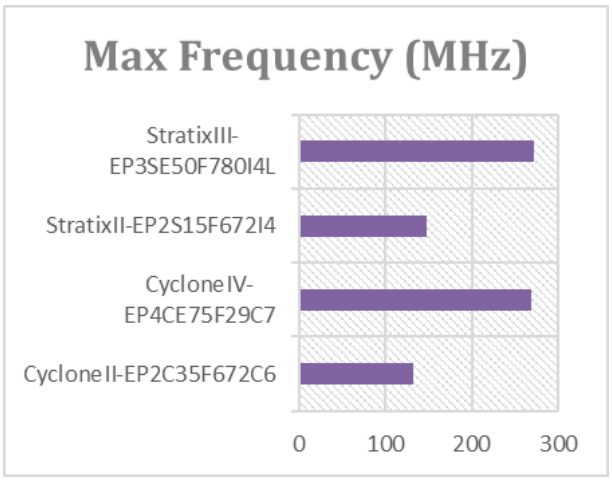

Figure.7 Maximum Frequency Performance

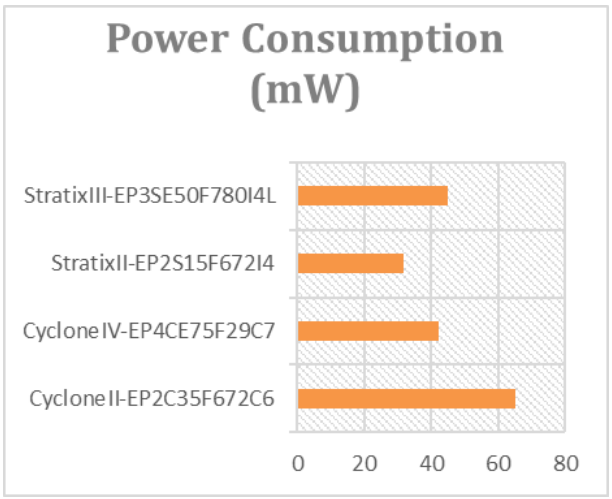

Figure.8 Power Consumption Performance

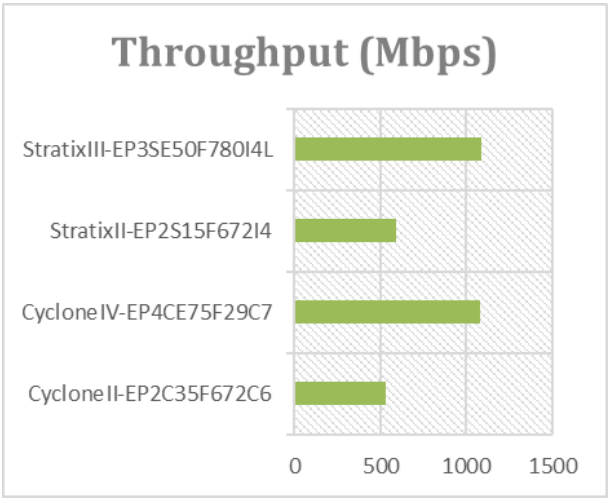

Figure.9 Throughput Performance

Although comparing different FPGA implementations is challenging due to the variety of technologies used, Table 2 below makes an effort to provide a fair comparison of common FPGA implementations of the lightweight method under the same circumstances. Table 5 illustrates the savings in logic resources by contrasting many designs using the ALTERA FPGA platform [7,10]. Compared to the state of the art, our solution uses a lot less hardware logic area resources. Our approach, which clocks in at 272.628MHz and 1090.512Mbps, respectively, is superior when maximum frequency and throughput performance are also considered.

Table 2. Comparative Analysis

Ref Work	**Platform**	**Logic Resources**	**Max Freq. (MHz)**	**Throughput (Mbps)**
[11]	Cyclone-IV	104760	74	595
[16]	Spartan 6	2150	143.164	909.816
Proposed Work	StratixIII-EP3SE50F780I4L	2378	143.85	700.546

5. CONCLUSION

The Internet of Things has spread widely due to resource constraints, the widespread deployment of small devices, and their wireless communication with the Internet and one another. Lightweight cryptography is used to address this issue in light of resource constraints and security requirements. In this article, a lightweight cryptographic hash function supporting IoT and big data applications was introduced. The bit permutation, linear transformation, and S-Box paradigms are used in the proposed study. The outcomes demonstrate notable gains in terms of performance, memory, and power use. We have designed our system to meet the broad needs of lightweight cryptography protocols while adhering to traditional hash standard security specifications. In order to accommodate a range of IoT applications, the suggested architecture will thereafter be evaluated on diverse hardware and platforms.

Conflicts of Interest : The authors declare no conflicts of interest.

References

[1] S. binti Suhaili and T. Watanabe, "Design of high-throughput SHA-256 hash function based on FPGA," 2017 *6th International Conference on Electrical Engineering and Informatics (ICEEI)*, Langkawi, Malaysia, 2017, pp. 1-6, doi: 10.1109/ICEEI.2017.8312449.

[2] F. G. Bîrleanu and N. Bizon, "A Review and Application of Integrity Techniques for Securing FPGA-based Designs," *2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Iasi, Romania, 2018, pp. 1-8, doi: 10.1109/ECAI.2018.8678994.

[3] J. -B. Choi, D. -S. Kim, J. -Y. Choe and K. -W. Shin, "Hardware Implementation of ECIES Protocol on Security SoC," 2020 International Conference on Electronics, Information, and Communication (ICEIC), Barcelona, Spain, 2020, pp. 1-4, doi: 10.1109/ICEIC49074.2020.9051263.

[4] C. Y. Chu and M. Lukowiak, "Two Step Power Attack on SHA-3 Based MAC," *2018 25th International Conference "Mixed Design of Integrated Circuits and System" (MIXDES)*, Gdynia, Poland, 2018, pp. 209-214, doi: 10.23919/MIXDES.2018.8436910.

[5] Huang, Si-Cheng, Shan Huang, Hua-Lei Yin, Qing-Li Ma, and Ze-Jie Yin. 2023. "High-Speed Variable Polynomial Toeplitz Hash Algorithm Based on FPGA" *Entropy 25*, no. 4: 642. https://doi.org/10.3390/e25040642

[6] A. Fairouz and S. P. Khatri, "An FPGA-Based Coprocessor for Hash Unit Acceleration," 2017 IEEE International Conference on Computer Design (ICCD), Boston, MA, USA, 2017, pp. 301-304, doi: 10.1109/ICCD.2017.53.

[7] C. Jeong and Y. Kim, "Implementation of efficient SHA-256 hash algorithm for secure vehicle communication using FPGA," *2014 International SoC Design Conference (ISOCC)*, Jeju, Korea (South), 2014, pp. 224-225, doi: 10.1109/ISOCC.2014.7087617.

[8] Z. Jin and H. Finkel, "Bob Jenkins Lookup3 Hash Function on OpenCL FPGA Platform," *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 4736-4741, doi: 10.1109/BigData.2018.8621960.

[9] D. S. Kim, S. H. Lee and K. W. Shin, "A Hardware Implementation of SHA3 Hash Processor using Cortex-M0," *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, Auckland, New Zealand, 2019, pp. 1-4, doi: 10.23919/ELINFOCOM.2019.8706419.

[10] J. Li, L. Wu and X. Zhang, "An efficient HMAC processor based on the SHA-3 HASH function," *2017 IEEE 12th International Conference on ASIC (ASICON)*, Guiyang, China, 2017, pp. 252-255, doi: 10.1109/ASICON.2017.8252460.

[11] F. Assad, F. Elotmani, M. Fettach and A. Tragha, "An optimal hardware implementation of the KECCAK hash function on virtex-5 FPGA," *2019 International Conference on Systems of Collaboration Big Data, Internet of Things & Security (SysCoBIoTS)*, Casablanca, Morocco, 2019, pp. 1-5, doi: 10.1109/SysCoBIoTS48768.2019.9028020.

[12] D. Grochol and L. Sekanina, "Fast Reconfigurable Hash Functions for Network Flow Hashing in FPGAs," *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Edinburgh, UK, 2018, pp. 257-263, doi: 10.1109/AHS.2018.8541401.

[13] M. Padhi and R. Chaudhari, "An optimized pipelined architecture of SHA-256 hash function," *2017 7th International Symposium on Embedded Computing and System Design (ISED)*, Durgapur, India, 2017, pp. 1-4, doi: 10.1109/ISED.2017.8303943.

[14] O. Panait, L. Dumitriu and I. Susnea, "Hardware and Software Architecture for Accelerating Hash Functions Based on SoC," *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, Bucharest, Romania, 2019, pp. 136-139, doi: 10.1109/CSCS.2019.00031.

[15] Algredo-Badillo, Ignacio, Miguel Morales-Sandoval, Alejandro Medina-Santiago, Carlos Arturo Hernández-Gracidas, Mariana Lobato-Baez, and Luis Alberto Morales-Rosales. 2022. "A SHA-256 Hybrid-Redundancy Hardware Architecture for Detecting and Correcting Errors" Sensors 22, no. 13: 5028. https://doi.org/10.3390/s22135028

[16] M. Sundal and R. Chaves, "Efficient FPGA Implementation of the SHA-3 Hash Function," *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Bochum, Germany, 2017, pp. 86-91, doi: 10.1109/ISVLSI.2017.24.

[17] Y. Tanasyuk, A. Perepelitsyn and S. Ostapov, "Parameterized FPGA-based implementation of cryptographic hash functions using cellular automata," *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Kyiv, UKraine, 2018, pp. 225-228, doi: 10.1109/DESSERT.2018.8409133.

[18] Kammoun, Manel et al. "HW/SW Architecture Exploration for an Efficient Implementation of the Secure Hash Algorithm SHA-256." *Journal of Communications Software and Systems* (2021), pp. 87-96

[19] T. Wada, N. Matsumura, K. Nakano and Y. Ito, "Efficient Byte Stream Pattern Test using Bloom Filter with Rolling Hash Functions on the FPGA," *2018 Sixth International Symposium on Computing and Networking (CANDAR),* Takayama, Japan, 2018, pp. 66-75, doi: 10.1109/CANDAR.2018.00016.

[20] T. Yalçin and E. B. Kavun, "Almost-Zero Logic Implementation of Troika Hash Function on Reconfigurable Devices," *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig),* Cancun, Mexico, 2019, pp. 1-6, doi: 10.1109/ReConFig48160.2019.8994780.

[21] Zeyad A. Al-Odat, Mazhar Ali, Assad Abbas, and Samee U. Khan. "Secure Hash Algorithms and the Corresponding FPGA Optimization Techniques" 2020, *ACM Comput. Surv. 53, 5, Article 97* (September 2021), 36 pages. https://doi.org/10.1145/3311724

[22] Q. Zhou, "Irregular-Program-Based Hash Algorithms," *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON),* Newark, CA, USA, 2019, pp. 125-128, doi: 10.1109/DAPPCON.2019.00024.

[23] P.Pavithara, R.Renuka, P.Sabena Yasmin, K.Naresh, "Design and FPGA implementation of folded SHA-256 using 4-2 adder compressor", *Nat. Volatiles & Essent. Oils, 2021*; 8(5): pp. 112 – 121.

[24] K. Kumar, K. R. Ramkumar, A. Kaur and S. Choudhary, "A Survey on Hardware Implementation of Cryptographic Algorithms Using Field Programmable Gate Array," 2020 *IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT),* Gwalior, India, 2020, pp. 189-194, doi: 10.1109/CSNT48778.2020.9115742.

[25] L. Pyrgas and P. Kitsos, "An 8-bit Compact Architecture of Lesamnta-LW Hash Function for Constrained Devices," *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS),* Genoa, Italy, 2019, pp. 743-746, doi: 10.1109/ICECS46596.2019.8965025.

[26] X. Cui, H. Shi, J. Zhao, Y. Ge, Y. Yin and K. Zhao, "High Accuracy Short Reads Alignment Using Multiple Hash Index Tables on FPGA Platform," *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC),* Chongqing, China, 2020, pp. 567-573, doi: 10.1109/ITOEC49072.2020.9141738.

[27] S. Al-Shara'a, R. K. Ibraheem and O. Bayat, "Implementation of cryptanalysis based on FPGA hardware using AES with SHA-1," *2019 International Conference on Smart Applications, Communications and Networking (SmartNets),* Sharm El Sheikh, Egypt, 2019, pp. 1-7, doi: 10.1109/SmartNets48225.2019.9069786.

[28] D. E. S. Kundi, A. Khalid, A. Aziz, C. Wang, M. O'Neill and W. Liu, "Resource-Shared Crypto-Coprocessor of AES Enc/Dec With SHA-3," in *IEEE Transactions on Circuits and Systems* I: Regular Papers, vol. 67, no. 12, pp. 4869-4882, Dec. 2020, doi: 10.1109/TCSI.2020.2997916.

[29] Rommel García, Ignacio Algredo-Badillo, Miguel Morales-Sandoval, Claudia Feregrino-Uribe, René Cumplido, A compact FPGA-based processor for the Secure Hash Algorithm SHA-256, *Computers & Electrical Engineering*, Volume 40, Issue 1, 2014, Pages 194-202, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2013.11.014.

[30] Shamsiah Suhaili and Norhuzaimin Julai, "FPGA-based Implementation of SHA-256 with Improvement of Throughput using Unfolding Transformation", Pertanika *Journal of Science & Technology*, Volume 30, Issue 1, January 2022 DOI: https://doi.org/10.47836/pjst.30.1.32

[31] Fatimazahraa Assad, Mohamed Fettach, Fadwa El Otmani, Abderrahim Tragha, "High-performance FPGA implementation of the secure hash algorithm 3 for single and multi-message processing", *International Journal of Electrical and Computer Engineering (IJECE)* Vol. 12, No. 2, April 2022, pp. 1324~1333 ISSN: 2088-8708, DOI: 10.11591/ijece.v12i2.pp1324-1333.

[32] Wajih El Hadj Youssef, Ali Abdelli, Fethi Dridi, Mohsen Machhout, "Hardware Implementation of Secure Lightweight Cryptographic Designs for IoT Applications", *Security and Communication Networks*, vol. 2020, Article ID 8860598, 13 pages, 2020. https://doi.org/10.1155/2020/8860598

[33] Z. A. Al-Odat, E. M. Al-Qtiemat and S. U. Khan, "An Efficient Lightweight Cryptography Hash Function for Big Data and IoT Applications," *2020 IEEE Cloud Summit*, Harrisburg, PA, USA, 2020, pp. 66-71, doi: 10.1109/IEEECloudSummit48914.2020.00016.

[34] Wali, Heera & Iyer, Nalini. (2023). Power, Performance and Area Analysis of Sponge Based Lightweight HASH Function. International *Journal of Electrical and Electronic Engineering & Telecommunications.* 245-256. 10.18178/ijeetc.12.4.245-256.

[35] Preeti Lawhale, Dr.Sujata.N.Kale," A Survey On Secure Architectures Using Hash Function Based On Fpga For Block Chain Enabled Iot Devices", Dr.Sujata.N.Kale *2023 IEEE 11th International Conference on Emerging Trends in Engineering & Technology - Signal and Information Processing (ICETET - SIP),* DOI**:** 10.1109/ICETET-SIP58143.2023.10151459

[36] Thakare, Yogesh, et al. "Development and design approach of an sEMG-based Eye movement control system for paralyzed individuals." *Journal of Integrated Science and Technology* 12.5 (2024): 811-811.

[37] Thakare, Yogesh N.,et al. "IoT-Enabled Environmental Intelligence: A Smart Monitoring System.", *2023 3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pp. 37-42. IEEE, 2023.