FPGA-Based Implementation of Enhanced DGHV Homomorphic Encryption: A Power-Efficient Approach to Secure Computing

Gurdeep Singh[1], Sonam Mittal[2], Hani Moaiteq Aljahdali[3], Ahmed Hamza Osman[4], Ala Eldin A Awouda[5], Ashraf Osman Ibrahim[6*], Salil Bharany[7*]

Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, India[1, 2, 7]
Department of Information Systems-Faculty of Computing and Information Technology in Rabigh (FCITR), King Abdulaziz University, Jeddah 21911, Saudi Arabia[3, 4]
Bisha University, College of Engineering, Bisha, KSA[5]
Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Seri Iskandar, Malaysia[6]

***Abstract*—One new area of secure computing and privacy is homomorphic encryption (HE). An FPGA-based implementation of the HE algorithm, Enhanced DGHV, which helps real-time computation on encrypted text without disclosing the original data, is developed in this study. This research aims to focus on implementing the Enhanced DGHV Fully HE algorithm on FPGA hardware to achieve a more efficient scheme in terms of performance and security. The Xilinx Vivado tool implements the design on a Genesys 2 Kintex 7 FPGA board. While software simulation with 3.2% I/O usage, the simulation confirms a total power consumption of 3.12W (watts), highlighting successful synthesis with little resources. At 9.105W, the hardware implementation is comparable. Furthermore, an effective FPGA-based implementation confirms a method for achieving a balance between power consumption and performance while implementing the DGHV algorithm. The results show that the overall computational complexity can be reduced, and the hardware and software integration help to achieve an increased data security level for homomorphic encryption algorithms with improved efficiency.**

***Keywords*—Homomorphic encryption; cybersecurity; cryptography; DGHV; FPGA; Xilinx Vivado tool; Genesys Kintex**

I. INTRODUCTION

Cybersecurity is a fundamental attribute of the modern world, which protects systems, networks, and data from cyber threats. This field possesses diverse strategies, including encryption, authentication, and network security mechanisms to achieve assured secrecy, integrity, and availability of information [1]. Cryptography is the most basic foundation in cybersecurity, which conveys plaintext into unrecognizable ciphertext through mathematical transformation. Cryptography secures information from being accessed by unauthorized individuals. All cryptographic methods developed so far remain performance improvements for security, such as symmetric and asymmetric encryption techniques [2].

Advanced Encryption Standard (AES) and Data Encryption Standard (DES) are among the most popular symmetric key algorithms. Performance evaluations of these algorithms show brilliantly that AES outperforms the two in speed and security and is widely regarded as the contemporary solution for encryption [3]. Rivest Shamir Adleman (RSA) algorithms are asymmetric, and prime factorization is done to achieve acceptability; however, the cryptography process is extensive due to its nature. Unlike symmetric ones, which use a single key, RSA employs a pair of public and private keys for encryption and decryption. RSA provides security features such as secure communication or digital signatures; however, it does not perform well because of its computational complexity. Notwithstanding the praises for such cryptographic techniques, several challenges are looming. RSA becomes less efficient when large prime numbers are used for encryption and decryption. As for Fully Homomorphic Encryption (FHE), the computational overheads are too high, denying even further acceptance into mainstream applications. Hence, these challenges must be addressed for enhanced HE (Homomorphic encryption) or cryptographic solutions for real-life applications [4]. Variations have thus been suggested to improve the efficiency of RSA without losing its security features [5]. HE is a remarkable advance in cryptographic techniques that facilitates computations on encrypted data without decrypting it. FHE would allow secure data processing in cloud environments without compromising privacy during computation. Since the theoretical foundations and practical implementations of HE have received a great deal of attention, possible applications include secure multiparty computations and encrypted search queries [6].

A. Prospects of Hardware Implementation

Data security, while developing with different cryptographic algorithms, is always a main concern for researchers. These cryptographic algorithms can be attacked in various ways and leave the data in a vulnerable state. Side-channel attacks can be used against AES and DES to break them. On the other hand, their implementations lean on the information leakage from hardware implementations against which protection measures may be adopted. RSA security relies on the principle that factoring is a hard problem, but progress in quantum computing could weaken such security claims. Another issue is that, while Fully HE is theoretically proven secure, it is computationally expensive and is thus prone to resource-exhaustion attacks [7]. To some extent, more recently,

importance was placed on the hardware realization of the cryptographic algorithm. Cryptographic solutions implemented in hardware improve performance, security, and energy efficiency over software-based ones. Field Programmable Gate Arrays (FPGAs) provide a suitable platform for cryptographic implementations because of flexibility, parallelism, and embedded security features. FPGAs have indeed been used for the acceleration of cryptographic computations, as the software-based encryption methods are more vulnerable to attacks [8].

Hardware security is important for minimizing many vulnerabilities originating from software-based cryptography implementations. Hardware secure design principles, like resistance against side-channel attacks or using a secure key store, are important to improve the credibility of cryptography. Fastening hardware security features to cryptographic installations ensures that such installations are resilient against both logical and physical attacks, thereby complementing the security in digital systems [9]. However, the growing evolution of FPGA architectures makes hardware implementations of cryptographic structures much more realizable. Modern architectures of FPGAs integrate features like Physically Unclonable Functions and hardware root-of-trust mechanisms, which essentially improve the resilience of cryptographic systems against various attacks. In advance, it has opened up high-performance-low-energy-strength cryptographic systems based on real applications [10].

The FHE scheme constructed in hardware poses challenges related to its computational complexity as well as constraints faced in terms of resources. The designs of HE architecture and their optimization have been considered in previous studies, which explore further contributions of hardware-based acceleration in boosting performance and efficiency [11]. The particular study presents an implementation for a scheme, called the Dijk Gentry Halevi Vaikutanathan (DGHV) algorithm, on hardware acceleration. The DGHV scheme is the FHE scheme based on integer arithmetic and is entirely attributed to its authors. The proposed implementation makes use of FPGA-based acceleration to optimize the implementation of the DGHV scheme from the viewpoint of its computational overheads while improving practical applicability by using a shorter secret key. State-of-the-art results in FPGA-based cryptography implementations indeed revolve around the possibility of FHE algorithm acceleration via dedicated hardware. Furthermore, the use of FPGA clusters for the calculations of HE boosts efficiency, making FHE a choice to penetrate applications while preserving user privacy [12].

This study is arranged as follows: Section II discuss about the homomorphic encryption; Section III elaborates on FPGAs from an encryption standpoint; Section IV discuss about the Literature work; Section V describes the proposed approach; Section VI shows the implementation results; Section VII compares software and hardware performances; and Section VIII gives the conclusions and future work.

II. HOMOMORPHIC ENCRYPTION

HE is one of the advanced forms of cryptography that enables the computation of data in an encrypted form without ever decrypting it. This property gives HE a unique utility in real-time applications, where data privacy and security are crucial, such as in cloud computing, privacy-concerned machine learning, and secure multi-party computations. Another important theory states that traditional cryptographic encryption and security schemes always require data to be decrypted before processing. However, with HE, the actual processing is done on encrypted data, keeping the sensitive information technically safe and sound through all computations, as shown in Fig. 1 [13].

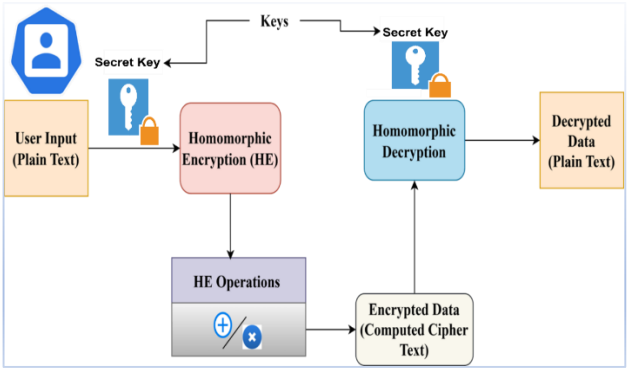

Fig. 1. Block diagram of Homomorphic encryption.

HE is categorized according to the operation types performed on encrypted data. So, the three major categories of HE are Somewhat Homomorphic Encryption (SHE), Partially Homomorphic Encryption (PHE), and Fully Homomorphic Encryption (FHE). Different types give different functionalities and complexities, and are, therefore, used for different applications, which are discussed below:

A. Somewhat Homomorphic Encryption (SHE)

Somewhat HE is an encryption scheme that permits a limited number of operations of any type on encrypted data. SHW allows only a few numbers of additions or multiplications before the ciphertext gets too noisy to decipher. The real drawback of SHE comes about from the accumulation of noise in the encrypted data, which eventually makes decryption impossible without specialized refreshing operations such as bootstrapping. However, SHE has important applications even if it does not permit many homomorphic operations; these are a few but sufficient cases, like the simple aggregation of data or secure voting mechanisms. SHE usually proves to be a better option than FHE, considering the lower computational overhead as far as speed and efficiency are concerned [14].

B. Partially Homomorphic Encryption (PHE)

Partially HE allows free operation of either addition or multiplication. However, both cannot be realized at the same time. Well-known examples of PHE include the RSA cryptosystem, which provides an example of a multiplicatively homomorphic cryptosystem, and the Paillier cryptosystem, which is a purely additive homomorphic scheme. It is applied in many areas, such as secure electronic voting and watermarking, wherein either additive or multiplicative homomorphic properties provide enough homomorphism for the application to find a solution. It is much faster and more practical with a wide range of applications [15].

C. Fully Homomorphic Encryption (FHE)

FHE is an improved version of SHE and PHE that allows any number of additions or multiplications of encrypted information. Therefore, arbitrary functions can be computed on encrypted data without decrypting it. This concept was developed for FHE by Craig Gentry in 2009, and since then, many advancements aimed at improving its efficiency and practicality have followed [16]. The scope is immense for FHE as it has wide applications in privacy concerning cloud computing, secure machine learning, and database queries. The major drawback of FHE is its immense computational expense since any execution of fully homomorphic operations on encrypted data needs lots of processing power and memory. Over the years, several optimizations have been proposed to render FHE amenable to real-world applications, such as batching techniques and improvements to bootstrapping.

III. FPGA – A Hardware Approach

The proper hardware implementation of HE is indispensable because the operations involving HE are very computationally intensive. Conventional software installations fail to meet the high processing requirements of HE, even to the extent that researchers then explore possible hardware-accelerating techniques. One of the best options for this approach is integrating FPGAs with cryptographic algorithms, thereby optimizing performance.

A. Integration of FPGA and Cryptographic Algorithms

FPGAs present a reconfigurable and parallel-processing platform, suitable for the acceleration of cryptographic computations. Unlike general-purpose CPUs that execute instructions sequentially, FPGAs can perform multiple encryption and decryption operations in parallel, thus drastically increasing speed. In implementing HE schemes on the FPGA platform, researchers can greatly improve performance, energy efficiency, and flexibility [17].

B. Why Use an FPGA?

The reasons have been that FPGAs are superior to any other hardware accelerators, such as graphics processing units (GPUs) and application-specific integrated circuits (ASICs). Some of these include: Parallel Processing Capabilities: FPGAs allow the possibility to execute cryptographic functions in parallel and are, therefore, highly useful for those computationally intensive processes. Energy Efficiency: Concerning the power needed even to achieve high throughput, FPGAs perform better than GPUs, thus making them the best choice for energy-critical applications. Convenience and Flexibility: Unlike ASICs, which are fixed-function chips, FPGAs can be reprogrammed to accommodate different encryption schemes and optimizations as required by the changing security requirements. Security Enhancements: FPGAs provide hardware-level security features that mitigate threats such as side-channel attacks to ensure secure cryptographic implementations [18].

C. Implementation of HE Using FPGA

The HE schemes on FPGA aim to multiply the encryption, decryption, and bootstrapping processes into an FPGA. The most significant challenge here is to perform those arithmetic operations in modular arithmetic without letting the overhead for computations increase. Large ciphertexts and complicated arithmetic operations that characterize DGHV require special optimizations to address performance issues.

The following several avenues have been researched for the FPGA-based implementation of the HE schemes:

Optimized Modular Arithmetic Units: The design of efficient modular addition, multiplication, and division units to efficiently compute large integers.

Pipeline Architectures: FPGA-pipelined design allows parallel processing of encryption operations, thus increasing throughput immediately.

Noise Management Mechanisms: Techniques such as ciphertext compression and optimization of bootstrapping help control the noise growth and facilitate accurate decryption.

The studies showed that the FPGA implementations for HE can generate enormous speed-ups compared to software implementations that promote their applications in privacy-preserving cloud computing, secure data analytics, and encrypted Artificial Intelligence (AI) processing [19].

IV. Literature Review

This section contains the literature survey of HE and its evaluation. Various HE schemes have different improvements over time and have also been shown above. Along with this, the integration of HE and hardware using FPGA is also studied and explained. The complete study is presented as follows:

FHE was introduced by Gentry et al. (2009) in their study [15], which would expand its bounds in terms of encrypted computations without decryption. One main issue arises when the decryption depth of the circuit extends the evaluation capacity, just explains why it is almost bootstrappable. The author gave an insight into bootstrapping, structured in part for the decryption process, reconciling circuit depths, therefore, making the scheme entirely bootstrappable. The security parameter was refined by balancing γ against (n), ensuring that breaking the encryption required exponential time complexity. An implementation of optimizations was carried out to allow for reducing the secret key size and facilitate direct processing of the ciphertext bits.

Dijk et al. (2010) proposed an FHE scheme in their study [20] based on slightly different foundations from Gentry's lattice-based approach. The problem was one of creating a HE scheme that allowed bootstrapping solely with additions and integer multiplications. The author introduced the so-called approximate GCD problem to estimate an unknown integer from the near-multiples.

Brakerski et al. (2014) proposed an FHE scheme in their study [21], which requires the ideal-lattice assumption for transactions. The critical boundary was thus that of decryption complexity concerning security. The author thus applied some re-linearization and gave a way for SHE to exist without dependencies on a ring-based hardness assumption. The dimension-modulus reduction technique allows compression of the ciphertext and improves decryption. From there, one can now design an LWE-based single-server Private Information

Retrieval (PIR) protocol with reduced communication overhead. They also gained significantly improved ciphertext efficiency under worst-case lattice hardness.

Yu et al. (2014) analyzed quantum HE and discussed its limitations in their research study [22] when it came to obtaining perfect security. The problem was to achieve perfect security in a deterministic HE that was fully homomorphic and which incurred an exponential cost. The author used an information localization argument to show that the universal quantum computation could not be done deterministically without this cost.

The work [23] of Abozaid et al. (2015) is towards embedding FHE into embedded systems, so that power and performance requirements, amidst all forms of attacks, can be circumvented. The author has proposed hardware and software co-designed with certain multiplication units for increased efficiency compared to the former, while still maintaining software flexibility. FPGA implementation demonstrated that large multiplications can be handled quite well within the given power limits.

In [24], Karabat et al. (2015) developed the THRIVE biometric system, partly because of the authentication security issue. Biometrics or standard biometric systems generally safeguard the very crucial user's data. The author proposed this threshold HE biometric system in which the user and verifier jointly provided the secure key.

Sun et al (2016) developed the leveled FHE scheme. For the Ring Learning with Error (RLWE) based FHE scheme to further enhance efficiency-based encryptions. In their study [25], the main net has to have very strong security guarantees along with practical computational efficiency. An approximate eigenvector was proposed by the author for use with a single public key, which was then extended to a multi-key setting.

Further in 2016 Fun et al. highlighted the security challenges in their study [26] that comes with outsourcing big data storage as well as computation to third-party cloud service providers, since the traditional approaches to security seem to have failed, probably due to the sheer amount of data to be modified and its diversity. Several schemes have been explored in this study for FHE, with performance ratings based on encryption-based technology.

Roy and Associates in the year 2017 introduced a recryptor box model in their study [27], which improves the depth of homomorphic evaluation and efficiency. The only limitation with SHE schemes is that they do not allow more than a limited action because at some point this starts to seriously accumulate noise. The inclusive author introduced a refreshment for ciphertext with its use to reduce the noise while avoiding very large-sized parameters.

Cousins et al. (2017) in their study [28] explain that they developed an FPGA-based HE Processing Unit (HEPU) that would accelerate encrypted computation. The main neuromuscular perturbation caused by the lack of computational efficiency was finally addressed by the primitive encryption of the lattice. Chinese Remainder Theorem (CRT) and inverse CRT were optimized in key mathematical operations. Implementation of FPGA using Xilinx Virtex-7 demonstrated the mitigation of computational bottlenecks in performing ring arithmetic operations.

The author of the study [29] Ding et al., in the year 2018 developed an attribute-based encryption scheme with ciphertexts. This addressed the privacy issues in a cloud environment. Enabling computations on the encrypted data with confidentiality maintained therein was the challenge set. They were working on the integration of the HE with attribute-based encryption so that one could perform fine-grained access control without having to repeatedly update keys.

In study [30] given by Catalano et al. in the year 2018 introduced homomorphic message authenticators (HMA) authentication methods for message verification in computing over encrypted data. In this case, the main difficulty consists of establishing the integrity of authenticated data without revealing underlying information. The author presented two types of HMA: the first one supports arbitrary composition, while the second uses short authentication tags.

Jiang et al. (2019) introduced [31] a secure comparison protocol for cloud environments using HE. The primary challenge was enabling encrypted magnitude comparisons without exposing plaintext values. The author proposed incomplete re-encryption, which preserved ordering while transforming ciphertexts.

In federated deep learning-based work [32], Liu et al. (2020) set about optimizing privacy on the grounds of security and accuracy. The challenge was that existing privacy-preserving techniques either caused a drop in model accuracy or needed excessive computational resources. The author introduced an adaptive privacy-preserving framework using layer-wise relevance propagation to optimize the trade-off concerning privacy.

Farokhi et al. (2020) proposed a scheme for privacy in encrypted transport services, trying to run encrypted queries on ride-sharing without exposing user data. The authors in their research [33] applied Paillier encryption, which supports some algebraic operations on ciphertext while remaining efficient. In this way, their users could submit queries without revealing their locations or routes.

A lightweight HE scheme was designed by Moghadam et al. (2021) to support cloud storage applications in their article [34]. This includes high computational and storage costs that make traditional encryption techniques impractical. The author proposed the secret-splitting secure method, which efficiently splits encrypted data.

There is also a systematic review [35] conducted by Mittal et al. (2021) that discussed the research concerning FHE within cloud computing. The challenge lies in understanding different trade-offs that may exist between models of encryption and their computational efficiency.

Delgado et al. (2022) designed a HE scheme in their study [36] for the secure transmission of sensor data. The challenge stemmed from the need to preserve confidentiality while enabling real-time statistical analysis of encrypted data. The author used the Paillier cryptosystem to allow for statistical computations on the encrypted data without decrypting it.

Wang et al. in the year 2022 proposed an integrated HE with identity-based signatures to secure Industrial Internet of Things (IIoT) transactions in their research work [37]. The challenge was protecting private trading data in blockchain-based energy markets. The author employed Paillier encryption for transaction confidentiality while using identity-based signatures for authentication.

Xu and colleagues (2023) proposed an NTRU-type in a research article [38]. Threshold HE schemes for securing multiparty computations. Their challenge lies in avoiding cumbersome extensions of ciphertexts during multi-key HE. Within their development of a new encryption model into a model that achieved improvements in computational efficiency without linearization, the authors introduced wide key distribution to withstand attacks from the subfield lattice and secured it under the RLWE assumption.

Pan et al. brought forward in the year 2023 their study [39], a cover for security holes that existed in the networked control systems by an FHE-encrypted controller. The main challenge was to maintain the real-time behavior while preventing the exposure of controller parameters to eavesdropping.

In the year 2024, the study [40] written by Ali and colleagues (2024) devised a dual-layer encryption method that combines HE with secret-sharing techniques. Securing cloud-based data storage while allowing computations on encrypted data poses a challenge. The author suggested distributing encrypted data among several servers to ensure no single point of failure. Performance evaluations have shown an optimal trade-off between security and computational efficiency.

A privacy-preserving network-slicing framework in the study [41] for secure communication was introduced by Wang et al. (2024). The challenge was protecting sensitive data within the network slicing while assuring efficiency in transmission. The author merged attribute-based encryption (ABE) with HE to maximize security at minimal computational cost.

Pingping et al. (2024) presented in their work [42] that gene information linkage and its accuracy binding on cloud computing are a real hustle; thus, it evolved as a privacy hazard and sluggish payback while processing gene material. To mitigate this circumstance or likeness, the author laid out a HE-based match secret protocol, which allows for the comparison of genetic sequence data without the data being decrypted. Compare gene sequence data with location.

In study [43] given by the author Ferrara et al. in the year 2025, explored Torus FHE and its use in secure computing. The author optimized bootstrapping to increase efficiencies in computing with ciphertext. Their study verified TFHE implementations for Boolean and arithmetic circuit evaluation. Their findings witness FHE for privacy-preserving computing.

Table I shows the chronological study of literature which highlights the different techniques and their results over time.

TABLE I. LITERATURE SURVEY COMPARATIVE ANALYSIS

Ref. No./Year	Technique	Summary and Result
[15]/2009	FHE	Proposed FHE with bootstrapping to balance security parameters. Optimized key size but faced practical challenges, leading to improved FHE efficiency.
[20]/2010	Approximate GCD-based FHE	Developed integer multiplication-based FHE without lattice dependence. Achieved bootstrapping but remained inefficient.
[21]/2014	LWE-based FHE	Introduced re-linearization and dimension-modulus reduction, improving ciphertext efficiency and enabling the PIR protocol.
[22]/2014	Homomorphic Encryption	Explored QHE limitations, confirming trade-offs between security and efficiency due to high computation costs.
[23]/2015	Hardware-accelerated FHE	Designed FPGA-based FHE optimizations with low-power multipliers, reducing computational overhead.
[24]/2015	Threshold HE for Biometrics	Developed an encrypted biometric authentication system, ensuring secure authentication with minimal verification time.
[25]/2016	RLWE-based Leveled FHE	Proposed an eigenvector-based multi-key FHE integrated with IBE, enhancing security and efficiency.
[26]/2016	Hybrid HE	Addressed big data encryption constraints, proposing hybrid models for improved speed, storage, and bandwidth.
[27]/2017	Recryptor Box Model for SHE	Enhanced SHE by reducing noise with ciphertext refresh methods, achieving a 20 times speed-up in FPGA tests.
[28]/2017	FPGA-based HEPU	Proposed FPGA-based unit for accelerating encrypted computations, optimizing CRT & inverse CRT.
[29]/2018	Attribute-Based HE	Enabled fine-grained access control in cloud storage, ensuring immunity to collusion attacks.
[30]/2018	Homomorphic Message Authentication	Proposed authentication methods ensure data integrity without revealing information.
[31]/2019	Secure Comparison Protocol	Developed encrypted magnitude comparison, ensuring confidentiality in cloud computations.
[32]/2020	Federated Learning with Adaptive Privacy	Introduced privacy-preserving encryption for deep learning, balancing security and model accuracy.
[33]/2020	HE for Transport Services	Applied Paillier encryption for secure ride-sharing queries, preserving user privacy.
[34]/2021	Secret-Splitting Secure Method	Designed lightweight encryption for cloud storage, improving storage efficiency and processing speed.
[35]/2021	Systematic Review of FHE	Analyzed FHE scalability and efficiency, identifying computational overhead challenges.
[36]/2022	HE for Sensor Data	Used Paillier encryption for encrypted sensor data analysis, enabling real-time anomaly detection.

[37]/2022	HE for IIoT	Integrated FHE with identity-based signatures for blockchain transactions, improving efficiency.
[38]/2023	NTRU-Type Threshold HE	Optimized multiparty encryption without linearization, securing against subfield lattice attacks.
[39]/2023	FHE Encrypted Controller	Integrated FHE in control systems, maintaining real-time performance and security.
[40]/2024	Dual-Layer HE	Combined FHE with secret sharing for cloud security, balancing computational efficiency.
[41]/2024	Privacy-Preserving Network Slicing	Merged ABE with FHE, enhancing security and efficiency in network slicing.
[42]/2024	HE for Gene Matching	Enabled secure genomic data processing, reducing encryption time and improving accuracy.
[43]/2025	Torus FHE Optimization	Optimized TFHE bootstrapping for efficient computation with ciphertext, verifying implementations for Boolean and arithmetic circuits.

HE provides privacy and security since computations are done on encrypted data, so one does not have to decrypt it. If key sizes increase, then computational complexity increases with reduced speed. Focus is on the enhancement of HE techniques through efficient hardware implementations on FPGAs for fast, low-power, and secure processing. About this, the main objective is to bridge the gap between real-time performance and security in privacy-preserving systems. This research's primary goal is to examine and evaluate the current homomorphic encryption algorithms and their hardware implementations, create and construct an improved version on an FPGA, and validate the results through in-depth analysis.

V. Proposed Methodology

The DGHV algorithm is implemented using a shorter secret key with reduced computational complexity. The Enhanced DGHV is implemented over the Genesys 2 Kintex 7 board to show the performance analysis. The proposed methodology is shown in Fig. 2.

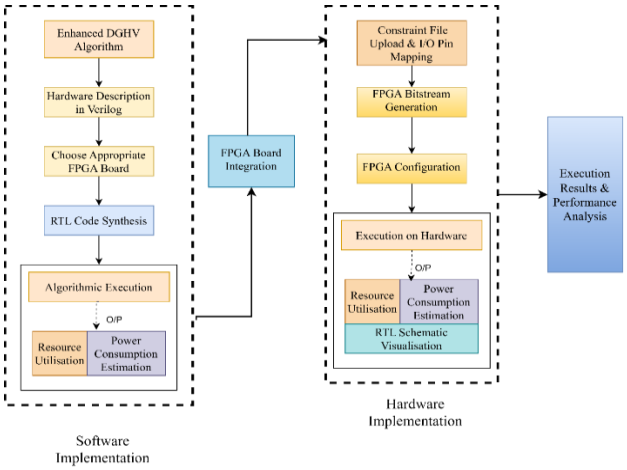

Fig. 2. Proposed methodology.

The research shows the detailed study for the implementation which is given as follows:

A. Enhanced DGHV Algorithm

Dijk Gentry, Halevi Vaikutanathan (DGHV) introduced the first FHE scheme, based on integers using only modular arithmetic. The researchers proposed a symmetric HE scheme for limited circuit depth. To design the asymmetric FHE scheme, the bootstrapping technique can be applied, which also helps to increase the circuit depth for the symmetric DGHV scheme. DGHV used Regev's scheme, the first encryption scheme, and used the same formula, $Encrypt(m, p) = m + 2r + pq$ but integers are used instead of an integral fraction of the domain size. The DGHV scheme can also be understood as a conceptually simpler version of Gentry's FHE scheme, based on integers instead of lattices, while focusing on providing conceptual simplicity by performing all operations using integers instead of ideal lattices and reducing the computational complexity. To improve the DGHV FHE scheme and create a more efficient version of the DGHV scheme, the proposed algorithm uses a shorter key size p which decreases the overall computation complexity. In addition, hardware and software integration enhances the scheme's security and efficiency because if the key size is larger, then the ciphertext also becomes larger, and it will take more computation time. To resolve this issue, a shorter key size is used.

Different parameters are used to make the scheme fully homomorphic. The scheme is based on a set of three public integers p, q, r, where p,q these are two random prime numbers because if we multiply two prime numbers, it becomes difficult to break the algorithm and r is a random error which is required to be added to the plaintext to generate the integral ciphertext, improving the security of the scheme. It is an important security key factor that helps to mask data and keep it safe from attackers. The generated ciphertext would be a list of integers, each representing an encrypted plaintext bit. These bits can be decrypted and recombined to retrieve the original plaintext message. If no error is added, then a pattern generated while performing encryption of plaintext bits may provide a clue to an adversary, and a secret key can easily be guessed. The scheme's security is based on the hardness of solving the Approximate Greatest Common Divisor Problem (AGCD). Here, the homomorphic operations (addition and multiplication) can be accomplished by homomorphic addition (XOR of bits) and multiplication (AND of bits) over the ciphertext. The size of λ directly affects the scheme's security. The larger the value of λ the more secure the scheme will be. The randomness in noise r ensures semantic security and prevents the scheme from ciphertext analysis attacks. Noise obfuscates the secret key, making the recovery more difficult. The DGHV uses the parameter, λ a security parameter for generating keys. It helps to specify the security of the key. The algorithm is as follows:

1) Key Generation ($KeyGen$). Generate a λ^2- bit, random odd integer, p, as a secret key. Select another two random numbers q and r, where r must be small such that $\mathrm{r} < \frac{\mathrm{p}}{2}$ and of λ- bits and q must be chosen randomly in the specified range $\left[-\frac{\mathrm{p}}{2}, \frac{\mathrm{p}}{2}\right]$ and of λ^5 bits.

2) Encryption (Encrypt) . Encrypt each bit of plaintext $m \in \{0, 1\}$. The algorithm encrypts the message m to obtain the ciphertext, c. in the following way.

$$Encrypt(p, q, m) = m + 2 * r + p * q \quad (1)$$

Encrypted ciphertext, c must be an integer whose residue mod p has the same parity as the plaintext.

3) Decryption ($Decrypt$): The formula for decryption is as follows:

$$Decrypt\ (p, c): m' \leftarrow (c\ mod\ p) mod\ 2 \quad (2)$$

The DGHV scheme is somewhat homomorphic, and it can perform computations up to a limited depth. So, the author introduced a new bootstrapping technique to obtain the FHE scheme and introduced an asymmetric fully homomorphic version of the DGHV scheme.This algorithm limitation is that, because of the multiplication property, the ciphertext's size will likewise increase if the plaintext or security parameters are increased. Then, because of hardware constraints, it became a limitation for hardware implementations, and the complexity of operations also increases. When all of these factors are combined, the algorithm may fail.

B. Software Implementation

The Enhanced DGHV algorithm software implementation was translated into hardware description using Verilog. The next step involves the selection of a specific FPGA board to satisfy the requirements of computation and resources of the design. Upon selection, synthesis of Register Transfer Level (RTL) code is achieved from the high-level hardware description into a gate-level representation for execution in an FPGA. Following this is the algorithmic execution testing in the simulation environment to ensure correctness before actual deployment. Performance metrics such as utilization of resources and power consumption estimates are analyzed to achieve optimal efficiency before the hardware stage.

C. Hardware Implementation

After verification of the software implementation, the design is taken to the stage of board integration, which entails uploading constraint files and mapping I/O pins to specify how signals will interact with the physical FPGA. The bitstream file is generated, which takes the synthesized design into a format that would be understood by the FPGA hardware. This bitstream is then used for FPGA configuration before running it on the hardware board. During the observation of real-time resource utilization and power consumption, the RTL schematic is also observed as a visualization to evaluate whether the implemented design is correct or not. Finally, this whole process ends with the execution results being followed up with the performance analysis, thereby validating that the FPGA implementation is adequate in terms of function and efficiency.

1) System specification. The Dell EMC PowerEdge R640 is powered by an Intel Xeon 6246R 3.4GHz 16-core processor that improves the performance and speed of the system. The rest of the specifications of the systems are mentioned in Table II.

2) Vivado specification. The Xilinx simulation tool Vivado 2019.1 (64-bit) is used to implement the HE on the hardware board. The Xilinx Vivado Design Suite 2019.1 has made itself a complete FPGA and System on Chip (SoC) development environment with a strong complement of tools that facilitate the designing, synthesizing, implementation, and debugging of complex digital systems. Other specifications are given in Table III.

3) FPGA Board specification. For the hardware implementation, the Genesys 2 Kintex 7 FPGA board (XC7K325T-2FFG900C) is used. Genesys 2 is an FPGA development kit that lends its high-performance nature mainly to data and video applications shown in Table IV. The board features a rather rich set of peripheral resources alongside potent data-processing capabilities, making it a splendid choice for many applications.

TABLE II. SYSTEM SPECIFICATION

Feature Category	Details
Processor	Intel Xeon Gold 6246R
Number of Cores	16
Number of Threads	32
Cache	35.75MB
Base Clock Speed	3.4GHz
Enhanced SpeedStep Technology	Balances performance with power environments
Thermal Management	Yes, with temperature & thermal monitoring for protection
Memory Capacity	16GB
Storage Configuration	3.6TB

TABLE III. VIVADO TOOL SPECIFICATION

Feature Category	Description
Version Used	Vivado 2019.1 (64-bit) for homomorphic encryption implementation
FPGA & SoC Development	Provides a complete environment for designing, synthesizing, implementing, and debugging complex digital systems
Supported FPGA Devices	Kintex-7, Virtex-7, Zynq-7000, UltraScale+ series
High-Level Synthesis (HLS)	Converts C, C++, and SystemC-based code into hardware description language (HDL)
Simulation & Debugging	Vivado Logic Analyzer & Hardware Debugger provide hardware-level debugging and simulation
Floor Planning & Optimization	Tools for floor planning, power analysis, and timing optimization to maximize resource efficiency

TABLE IV. FPGA BOARD SPECIFICATION

Feature Category	Details
FPGA Chip	Xilinx Kintex-7™ (XC7K325T-2FFG900C)
Logic Resources	50,950 logic slices, each with four 6-input LUTs & 8 flip-flops
Memory	1 GB DDR3 RAM (1800 MT/s, 32-bit data width)
Block RAM	16 Mbit fast block RAM
Clocking	Internal clock speeds exceeding 450 MHz, 10 clock management tiles with PLLs
DSP Processing	840 DSP slices for high-performance signal processing
Analog-to-Digital Conversion	On-chip XADC (Analog-to-Digital Converter)

VI. Implementation Results

In this section, a description of the results of the implementation and analysis of software and hardware deployment of the DGHV HE schemes onto a Genesys 2 Kintex 7 FPGA board. Evaluation will include and put into perspective performance and resource utilization, compared through FPGA-based execution against software and hardware implementations. In terms of parameters such as power consumption, hardware utilization, and temperature, the analysis assesses the viability of FPGA acceleration for DGHV. Understandable trade-offs between hardware and software implementations will open up the findings for optimization, enhancing practicality in real-world applications of homomorphic technology.

A. Software Simulation Analysis

After the implementation of DGHV homomorphic encryption using the Xilinx Vivado 2019.1 suite, an analysis of the resource utilization on the Genesys 2 Kintex-7 was performed, and the results are as follows:

The important metrics are Look Up Tables (LUTs). They are the basic logic blocks inside an FPGA that implement combinational circuits, whereas I/O utilization measures the number of I/O pins that are in use for external communication which as shown in Fig. 3. From the results, it consumed 4 LUTs out of 203,800, which is extremely low in logic resources consumed, thereby the design is lightweight and does not impose significant computational overhead on the FPGA. The I/O utilization of 16 out of 500 (3.2%) is fairly high.

Resource	Utilization	Available	Utilization %
LUT	4	203800	0.00
IO	16	500	3.20

Fig. 3. Hardware utilization readings during software simulation.

The output graph in Fig. 4 has three parts; in the first column, which is the title (Name), the port names are listed; the second column (Value) shows the input values, while the third column holds the output results in hexadecimal format; the inputs are called m1 and m2. In this graph of this block, the plaintext values in the state values $m1 = 0$ and $m2 = 1$. This law of encryption provides different cryptographic keys p, $q, r1, r2$ upon which homomorphic encryption computations are performed. The module input operates on plain-text values represented by $m1, m2$, in this case, 0 and 1, respectively. These inputs have been used with encryption keys $p, q, r1, r2$ for HE computation. The resulting outputs from the encryption module are presented here as hexadecimal values: $c1, c2$. The encryption process results are: $c1 = 9$ and $c2 = A$.

The total power consumption report on power in Fig. 5 shows a total on-chip power of 3.12W (Watts): dynamic power accounts for 2.943W (94%) while device static power is given at 0.177W (6%). This means that most of the power will be consumed by active switching operations within FPGAs, as only a small portion is used to keep the device in an idle state. Power distribution states that I/O operations account for 2.857W (97%), meaning that a big portion of power is utilized for external data communications. Signals consume 0.066W (2%) while logic elements require just 0.020W, thus indicating that the computational load within the FPGA is minimal. The junction temperature is located at 30.5°C, which is a safe operational limit.

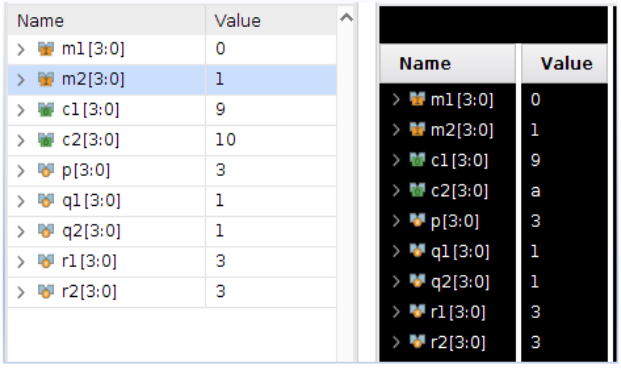

Fig. 4. Output in software simulation.

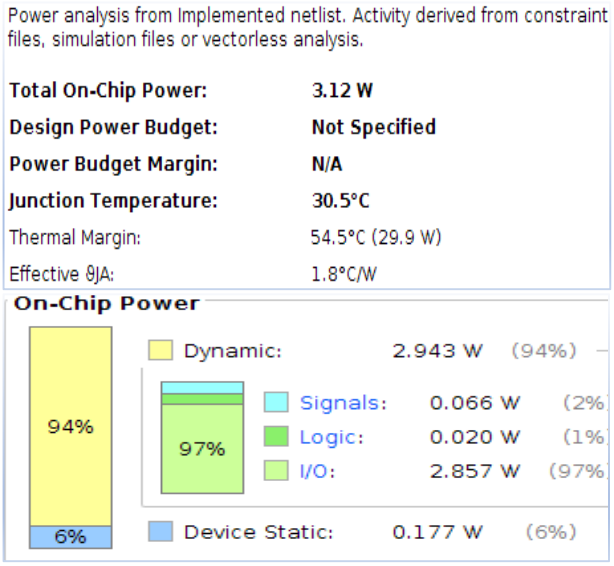

Fig. 5. Simulation power readings.

1) Secret key vs power consumption analysis. The changes showed an overall power consumption across the Genesys 2 Kintex-7 FPGA as employed among secret key values of homomorphic encryption in Table V. While the consumption power varies between each secret selected key, it shows a very slight difference and therefore stands to say something including that the more the complexity of the key, the greater the inefficiency in power use. For secret keys of sizes 5 and 13, the consumed power is 6.739 W and 6.836 W, respectively. With the increasing size of keys, variations in power are seen, the highest being recorded at 6.980W (key 47) and the lowest at 6.687W (key 181). The variations can be interpreted to point out that some key values yielded power savings while others incurred slightly higher computation overhead on the FPGA, and the graph in Fig. 6.

TABLE V. SECRET KEY VS. POWER CONSUMPTION COMPARISON

Secret Key Size	Power Consumption
5	6.739
13	6.836
29	6.718
47	6.980
83	6.818
101	6.904
149	6.873
181	6.687
199	6.919
211	6.818

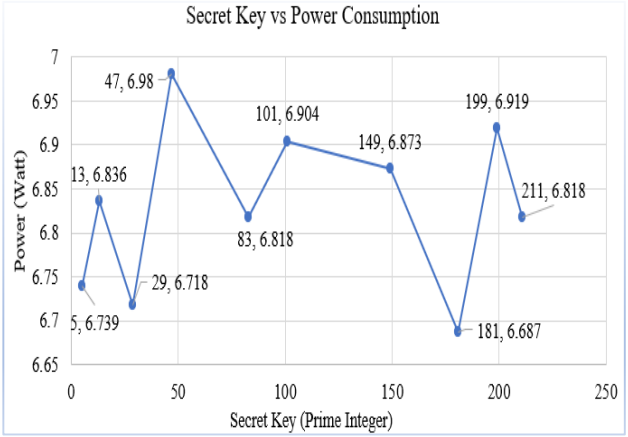

Fig. 6. Secret key vs. power consumption comparison graph during software simulation.

B. Hardware Analysis

The hardware analysis is the actual consumption of resources that occurs on the Genesys 2 Kintex-7 FPGA board after the successful implementation and burning of the DGHV HE algorithm onto the device.

The total on-chip power consumed in Fig. 7 is 9.105 watts, with dynamic power being 8.872 watts (97%), and therefore, the high active power consumption. The device's static power for the FPGA was 0.232 watts (3%), acting as the FPGA's baseline power consumption. I/O dominates the power at 8.677 watts (97%), and it seems data transfer happens quite frequently when processing ciphertext and sending encryption keys. Only a minimal amount of power, <1%, was consumed through logic, thanks to an efficiently deployed resource. The die junction temperature was measured at 41.2 degrees Celsius, thus leaving a thermal margin of 43.8 degrees Celsius, which should make operations smooth. In conclusion, the entire encryption process is power-consuming, but should safely remain below the thermal threshold.

RTL layout was identified to map the hardware structure of the DGHV homomorphic encryption on the Genesys 2 FPGA, depicting how input signals are processed in Fig. 8.

Inputs "$m1[3:0]$" *and* "$m2[3:0]$" are passed through input buffers and processed within the LUTs-Unit that applies the encryption transformations. The encrypted outputs "$c1[3:0]$" *and* "$c2[3:0]$" are then passed through output buffers to external interfaces. This well-designed approach caters to minimum resource utilization while securely maintaining an encrypted data primitive through the processing stream.

Upon confirming the generation of the bitstream, the success of which implies the FPGA design as a correctly synthesized, implemented unit, ready to be programmed onto the Genesys 2 Kintex-7 FPGA shown in Fig. 9. The Hardware Manager verifies and recognizes the FPGA device so that it can be directly programmed.

It proposes various options in a pop-up dialog for viewing data reports or generating memory configuration, while the latter is meant to specify a bit sequence for programming into the device. Having the bitstream is the final step of the build process that needs to be completed before programming an FPGA to implement the DGHV homomorphic encryption algorithm into real hardware.

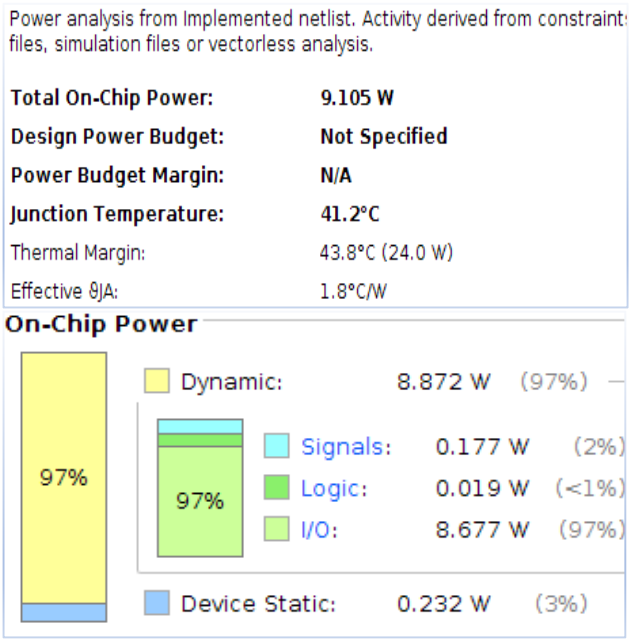

Fig. 7. Hardware power readings.

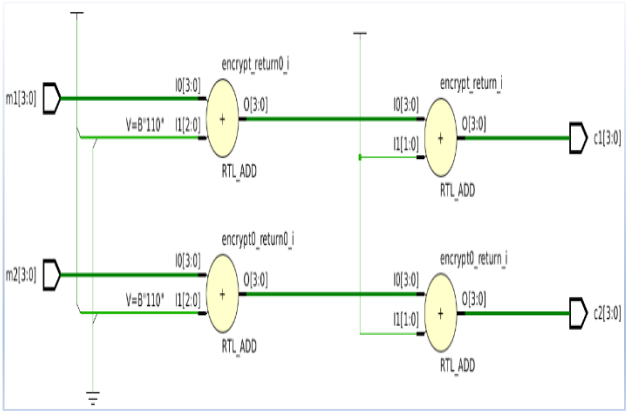

Fig. 8. Internal RTL layout.

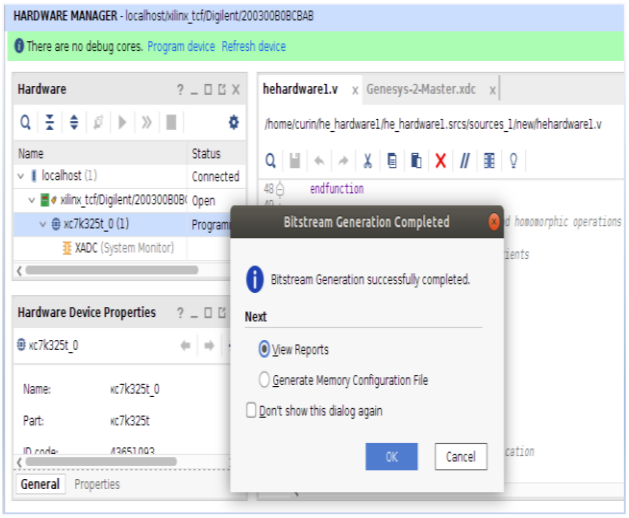

Fig. 9. Successfully bit stream generation.

The FPGA board is programmed as shown in Fig. 10. The board is now up and running. The LEDs represent the binary digits to be encrypted in the ciphertext; a further demonstration of this test is seen following. While the glow LEDs signify a "1" for the digit "1", an OFF LED points to "0" as the encrypted information is displayed in real-time. This test indicates that the FPGA has received the expected input values and computed the expected encrypted operation to produce the output. The correctness of the RTL implementation, when the program runs correctly on hardware, is validated by ensuring that encrypted data is processed securely within the FPGA environment.

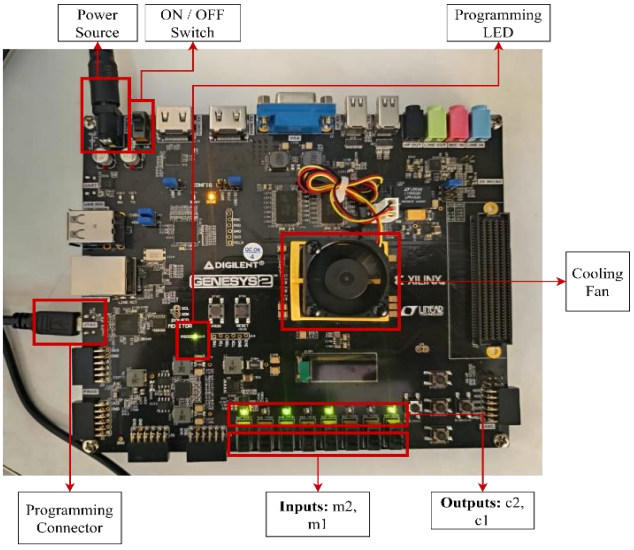

Fig. 10. Code burned on the FPGA kit.

VII. Software versus Hardware Analysis

This section compares various parameters for software and hardware implementation of the DGHV HE algorithm with the FPGA kit.

Table VI represents the total power consumption reading between the FPGA board in software simulation and hardware implementation. The total power consumed in software simulation using Genesys 2 Kintex-7 FPGA amounts to 3.12W, while when implemented in hardware, the power consumption amounts to 9.105W on account of real-world execution overhead.

The static power, which accounts for leakage and idle power consumption, shows a comparatively lower value in both cases; that is, 0.177W in software and 0.232W in hardware. Mainly accounts for the increase of power in hardware due to dynamic switching activity and actual FPGA resource usage during the execution, and the graph is shown in Fig. 11.

TABLE VI. Total Power Consumption

Results	Board	Static Power (W) (1)	Dynamic Power (W) (2)	Total Power (W) (1+2)
Software	Genesys 2 Kintex 7	0.177	2.943	3.12
Hardware		0.232	8.872	9.105

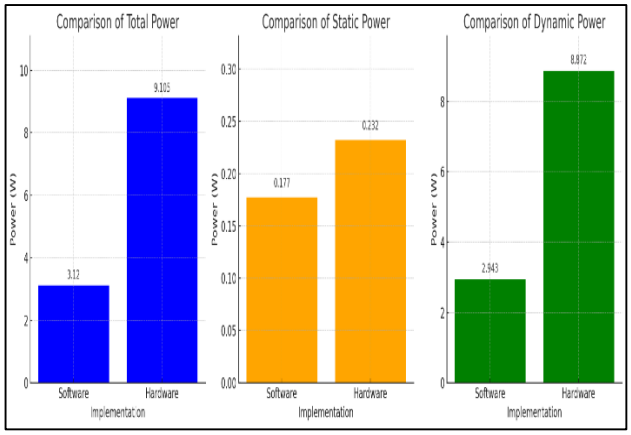

Fig. 11. Comparison of the total power usage graph.

The dynamic power distribution of the Genesys 2 Kintex-7 FPGA in software simulation and hardware implementation is specified in Table VII. The I/O power consumption in simulation is 2.857W, while the consumption in hardware increases in magnitude to 8.677W, signifying that real-world transfer of data and communication exerts higher power demands. On the other hand, logic essentially stays the same, with powers of 0.02W in software and 0.019W in hardware, thereby indicating an approximately similar logic resource utilization in both software and hardware.

TABLE VII. Dynamic Power Consumption

Results	Board	Dynamic Power (W)		
		IO Power	Logic Power	Signal Power
Software	Genesys 2 Kintex 7	2.857	0.02	0.066
Hardware		8.677	0.019	0.177

The power consumed due to signal switching across the FPGA tracks is much higher in hardware than in software, from 0.066W in software to 0.177W, confirming increased switching activity with real-time processing overheads. The results, therefore, reveal that actual hardware implementations lead to dynamic power dissipation on an entirely different scale,

especially in I/O, whereas logic power remains fairly flat and signal power progresses upwards moderately. These implications greatly affirm the weight of power optimization strategies as designs are transferred from the simulation to real working FPGA systems, and the graph is shown in Fig. 12.

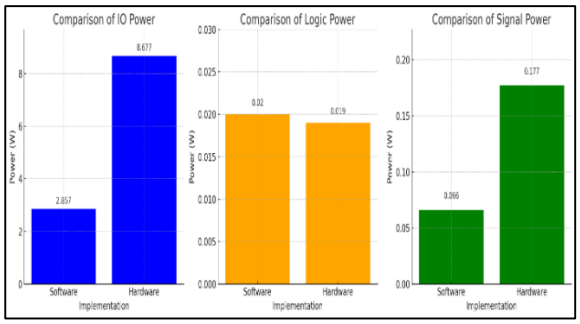

Fig. 12. Comparative analysis of dynamic power comparison.

The thermal performance analysis of the Genesys 2 Kintex-7 FPGA is shown in Table VIII, encompassing software simulation and hardware implementation. With software simulations, the junction temperature is 30.5°C, which means there is minimal heat generated since no actual hardware is working. During hardware implementations, however, the junction temperature reaches 41.2°C, indicating an increase in power dissipation due to actual processing loads that contribute to heat dissipation.

The thermal margin temperature difference between the maximum operating temperature of the FPGA and the current temperature is 54.5°C in the software and drops to 43.8°C in the hardware. This suggests that the FPGA runs much closer to its thermal limits in real-world execution, hence the necessity for adequate cooling and thermal management. From these observations, it is concluded that with hardware implementations, more thermal stress is introduced than in simulations; this calls for effective heat dissipation techniques to guarantee stable FPGA performance, and the graph is shown in Fig. 13. Several recent studies emphasize the need for secure and energy-efficient computing. Bharany and Sharma [49] explore blockchain and machine learning integration in IoT, aligning with secure hardware design goals [50-51]. Talwar et al. [50] and Badotra et al. [52] focus on fault tolerance and network vulnerabilities, highlighting the relevance of resilient architectures like FPGA-based encryption. Shamshad et al. [51] and Kumar et al. [53] stress model efficiency and privacy, supporting homomorphic encryption for secure data processing.

A comparison table of FPGA-based homomorphic encryption implementations is shown in Table IX. Different FPGA platforms that were used for the implementation of homomorphic encryption techniques are shown in the table.

TABLE VIII. TEMPERATURE ANALYSIS

Results	Board	Junction Temperature (°C)	Thermal Margin (°C)
Software	Genesys 2 Kintex 7	30.5	54.5
Hardware		41.2	43.8

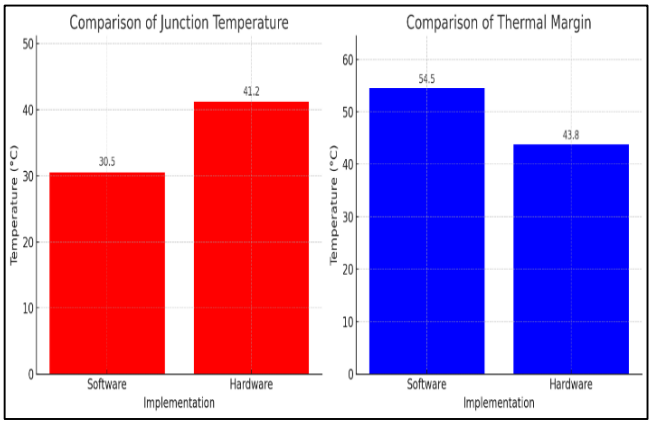

Fig. 13. Comparison of temperature readings.

TABLE IX. COMPARATIVE ANALYSIS OF PROPOSED ALGORITHM WITH EXISTING FPGA-BASED APPROACHES

Ref. No / Year	Technique Used	Board Used	LUT	FF	DSP
[27]/ 2017	Fan-Vercauteren SHE	Xilinx Virtex 6	3379	-	4
[44]/ 2019	Integer based	Nexys 4 DDR	5766	-	36
[45]/2021	Fast FHE over Torus FHE	Zynq-7000 ARM	36373	-	-
[46]/2022	Brakerski,Vaikuntanathan FHE	Intel Agilex FPGA	720	-	3
[47]/2022	Torus FHE	Virtex UltraScale+ VU13P	925 K	729 K	6240 K
[48]/2024	Cheon Kim Kim Song HE	Intel Agilex 7	87912	-	960
Proposed work	Integer-based DGHV	Genesys 2 K7	4	-	-

VIII. CONCLUSION

This research demonstrated that it is practicable to run the homomorphic encryption algorithm on FPGA platforms for facilitating privacy-preserving computations with enhanced performance. The integration of the Enhanced DGHV homomorphic encryption algorithm into the Genesys 2 Kintex-7 FPGA illustrates the possibility of real-time encrypted computation with reasonable on-chip resource requirements. The simulation results show a total power consumption of 3.12W and a negligible utilization of the resource I/O: 3.2% compared to the results of implementation, which shows a much higher power consumption of 9.105W, and slightly less resource utilization I/O: 3.2%. Finally, the junction temperature from this thermal margin analysis rises from a software 30.5°C to 41.2°C with hardware, and hence the importance of thermal management is thrown to the fore in FPGA computations. The study here discovers the possibilities in FPGA technology for privacy-preserving applications and calls for more optimization towards the significant reduction in power consumption with optimized performance. In this quest, attention has to be put on using sophisticated FPGA architectures, combined with innovative energy efficiency methodologies, and this will certainly enhance the computational effectiveness of huge-scale encrypted operations. Future work will involve working on other HE schemes and integrating them with hardware in a way to improve performance, efficiency, and power consumption.

FUNDING

This project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, Saudi Arabia, under grant no. (GPIP: 1913- 830-2024).

ACKNOWLEDGMENT

The authors, acknowledge with thanks DSR for technical and financial support.

REFERENCES

[1] K. Kim, "Cryptography: A new open access journal," Cryptography, vol. 1, no. 1, pp. 1–4, 2017, doi: 10.3390/cryptography1010001.

[2] K. Cabaj, Z. Kotulski, B. Księżopolski, and W. Mazurczyk, "Cybersecurity: trends, issues, and challenges," Eurasip J. Inf. Secur., vol. 2018, no. 1, pp. 10–12, 2018, doi: 10.1186/s13635-018-0080-0.

[3] A. K. Mandal, C. Parakash, and A. Tiwari, "Performance evaluation of cryptographic algorithms: Des and AES," 2012 IEEE Students' Conf. Electr. Electron. Comput. Sci. Innov. Humanit. SCEECS 2012, pp. 1–5, 2012, doi: 10.1109/SCEECS.2012.6184991.

[4] S. M. Khatarkar Tech Scholar and R. Kamble Asst Professor, "A Survey and Performance Analysis of Various RSA based Encryption Techniques," Int. J. Comput. Appl., vol. 114, no. 7, pp. 975–8887, 2015.

[5] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," ACM Comput. Surv., vol. 51, no. 4, 2018, doi: 10.1145/3214303.

[6] Martins, L. Sousa, and A. Mariano, "A survey on fully homomorphic encryption: An engineering perspective," ACM Comput. Surv., vol. 50, no. 6, 2017, doi: 10.1145/3124441.

[7] Q.-Y. Zhang and Y.-G. Jia, "A Speech Fully Homomorphic Encryption Scheme for DGHV Based on Multithreading in Cloud Storage," Int. J. Netw. Secur., vol. 24, no. 6, pp. 1042–1055, 2022, doi: 10.6633/IJNS.202211.

[8] S. M. Trimberger and J. J. Moore, "FPGA security: Motivations, features, and applications," Proc. IEEE, vol. 102, no. 8, pp. 1248–1265, 2014, doi: 10.1109/JPROC.2014.2331672.

[9] Y. Jin, "Introduction to hardware security," Electron. , vol. 4, no. 4, pp. 763–784, 2015, doi: 10.3390/electronics4040763.

[10] A. Boutros and V. Betz, "FPGA Architecture: Principles and Progression," IEEE Circuits Syst. Mag., vol. 21, no. 2, pp. 4–29, 2021, doi: 10.1109/MCAS.2021.3071607.

[11] A. C. Mert, E. Ozturk, and E. Savas, "Design and Implementation of Encryption/Decryption Architectures for BFV Homomorphic Encryption Scheme," IEEE Trans. Very Large Scale Integr. Syst., vol. 28, no. 2, pp. 353–362, 2020, doi: 10.1109/TVLSI.2019.2943127.

[12] H. Liao et al., "TurboHE: Accelerating Fully Homomorphic Encryption Using FPGA Clusters," Proc. - 2023 IEEE Int. Parallel Distrib. Process. Symp. IPDPS 2023, pp. 788–797, 2023, doi: 10.1109/IPDPS54959.2023.00084.

[13] M. Ogburn, C. Turner, and P. Dahal, "Homomorphic encryption," Procedia Comput. Sci., vol. 20, pp. 502–509, 2013, doi: 10.1016/j.procs.2013.09.310.

[14] W. Yang, S. Wang, H. Cui, Z. Tang, and Y. Li, "A Review of Homomorphic Encryption for Privacy-Preserving Biometrics," Sensors, vol. 23, no. 7, pp. 1–23, 2023, doi: 10.3390/s23073566.

[15] C. Gentry, "Fully homomorphic encryption using ideal lattice," in In Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 169–178. doi: 10.1109/TIFS.2013.2287732.

[16] J. H. Cheon, "Batch Fully Homomorphic Encryption," pp. 315–335.

[17] W. Hu, C. H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An Overview of Hardware Security and Trust: Threats, Countermeasures, and Design Tools," IEEE Trans. Comput. Des. Integr. Circuits Syst., vol. 40, no. 6, pp. 1010–1038, 2021, doi: 10.1109/TCAD.2020.3047976.

[18] J. Serrano, "Introduction to FPGA design," CAS 2007 - Cern Accel. Sch. Digit. Signal Process. Proc., pp. 231–247, 2008.

[19] M. Vanitha and R. Mangayarkarasi, "Comparative study of different cryptographic algorithms," Int. J. Pharm. Technol., vol. 8, no. 4, pp. 26433–26438, 2016, doi: 10.4236/jis.2020.113009.

[20] J. H. Cheon et al., "Batch fully homomorphic encryption over the integers," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 7881 LNCS, pp. 315–335, 2010, doi: 10.1007/978-3-642-38348-9_20.

[21] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," ACM Trans. Comput. Theory, vol. 6, no. 3, 2014, doi: 10.1145/2633600.

[22] L. Yu, C. A. Pérez-Delgado, and J. F. Fitzsimons, "Limitations on information-theoretically-secure quantum homomorphic encryption," Phys. Rev. A - At. Mol. Opt. Phys., vol. 90, no. 5, pp. 1–5, 2014, doi: 10.1103/PhysRevA.90.050303.

[23] G. Abozaid, A. Tisserand, A. El-Mahdy, and Y. Wada, "Towards FHE in Embedded Systems: A Preliminary Codesign Space Exploration of a HW/SW Very Large Multiplier," IEEE Embed. Syst. Lett., vol. 7, no. 3, pp. 77–80, 2015, doi: 10.1109/LES.2015.2436372.

[24] C. Karabat, M. S. Kiraz, H. Erdogan, and E. Savas, "THRIVE: threshold homomorphic encryption based secure and privacy preserving biometric verification system," EURASIP J. Adv. Signal Process., vol. 2015, no. 1, pp. 1–18, 2015, doi: 10.1186/s13634-015-0255-5.

[25] X. Sun, J. Yu, T. Wang, Z. Sun, and P. Zhang, "Efficient identity-based leveled fully homomorphic encryption from RLWE," Secur. Commun. Networks, vol. 9, no. 18, pp. 5155–5165, 2016, doi: 10.1002/sec.1685.

[26] T. S. Fun and A. Samsudin, "A survey of homomorphic encryption for outsourced big data computation," KSII Trans. Internet Inf. Syst., vol. 10, no. 8, pp. 3826–3851, 2016, doi: 10.3837/tiis.2016.08.022.

[27] S. S. Roy, F. Vercauteren, J. Vliegen, and I. Verbauwhede, "Hardware Assisted Fully Homomorphic Function Evaluation and Encrypted Search," IEEE Trans. Comput., vol. 66, no. 9, pp. 1562–1572, 2017, doi: 10.1109/TC.2017.2686385.

[28] D. B. Cousins, K. Rohloff, and D. Sumorok, "Designing an FPGA-accelerated homomorphic encryption co-processor," IEEE Trans. Emerg. Top. Comput., vol. 5, no. 2, pp. 193–206, 2017, doi: 10.1109/TETC.2016.2619669.

[29] Y. Ding, B. Han, H. Wang, and X. Li, "Ciphertext retrieval via attribute-based FHE in cloud computing," Soft Comput., vol. 22, no. 23, pp. 7753–7761, 2018, doi: 10.1007/s00500-018-3404-6.

[30] D. Catalano and D. Fiore, "Practical Homomorphic Message Authenticators for Arithmetic Circuits," J. Cryptol., vol. 31, no. 1, pp. 23–59, 2018, doi: 10.1007/s00145-016-9249-1.

[31] L. Jiang, Y. Cao, C. Yuan, X. Sun, and X. Zhu, "An effective comparison protocol over encrypted data in cloud computing," J. Inf. Secur. Appl., vol. 48, 2019, doi: 10.1016/j.jisa.2019.102367.

[32] X. Liu, H. Li, G. Xu, R. Lu, and M. He, "Adaptive privacy-preserving federated learning," Peer-to-Peer Netw. Appl., vol. 13, no. 6, pp. 2356–2366, 2020, doi: 10.1007/s12083-019-00869-2.

[33] F. Farokhi, I. Shames, and K. H. Johansson, "Private routing and ride-sharing using homomorphic encryption," IET Cyber-Physical Syst. Theory Appl., vol. 5, no. 4, pp. 311–320, 2020, doi: 10.1049/iet-cps.2019.0042.

[34] S. Sobati-Moghadam, "Efficient information-theoretically secure schemes for cloud data outsourcing," Cluster Comput., vol. 24, no. 4, pp. 3591–3606, 2021, doi: 10.1007/s10586-021-03344-x.

[35] S. Mittal and K. R. Ramkumar, "Research perspectives on fully homomorphic encryption models for cloud sector," J. Comput. Secur., vol. 29, no. 2, pp. 135–160, 2021, doi: 10.3233/JCS-200071.

[36] J. L. López Delgado, J. A. Álvarez Bermejo, and J. A. López Ramos, "Homomorphic Asymmetric Encryption Applied to the Analysis of IoT Communications," Sensors, vol. 22, no. 20, 2022, doi: 10.3390/s22208022.

[37] H. Wang, Y. Xiao, Y. Feng, Q. Qian, Y. Li, and X. Fu, "Cloud-Assisted Privacy Protection Energy Trading Based on IBS and Homomorphic Encryption in IIoT," Appl. Sci., vol. 12, no. 19, 2022, doi: 10.3390/app12199509.

[38] K. Xu, B. Hong Meng Tan, L. P. Wang, K. Mi Mi Aung, and H. Wang, "Threshold Homomorphic Encryption From Provably Secure NTRU,"

Comput. J., vol. 66, no. 12, pp. 2861–2873, 2023, doi: 10.1093/comjnl/bxac126.

[39] J. Pan et al., “Secure Control of Linear Controllers Using Fully Homomorphic Encryption,” Appl. Sci., vol. 13, no. 24, 2023, doi: 10.3390/app132413071.

[40] S. Ali, S. A. Wadho, A. Yichiet, M. L. Gan, and C. K. Lee, “Advancing cloud security: Unveiling the protective potential of homomorphic secret sharing in secure cloud computing,” Egypt. Informatics J., vol. 27, no. July, p. 100519, 2024, doi: 10.1016/j.eij.2024.100519.

[41] W. Wang, R. Liu, and S. Cheng, “Privacy protection of communication networks using fully homomorphic encryption based on network slicing and attributes,” Sci. Rep., vol. 14, no. 1, pp. 1–18, 2024, doi: 10.1038/s41598-024-69501-5.

[42] P. Li and F. Zhang, “Cloud-based Full Homomorphic Encryption Algorithm by Gene Matching,” J. Inf. Process. Syst., vol. 20, no. 4, pp. 432–441, 2024, doi: 10.3745/JIPS.03.0199.

[43] M. Ferrara, A. Tortora, and M. Tota, “an Overview of Torus Fully Homomorphic Encryption,” Int. J. Gr. Theory, vol. 14, no. 2, pp. 59–73, 2025, doi: 10.22108/ijgt.2023.139030.1869.

[44] Z. H. Mahmood and M. K. Ibrahem, “HARDWARE IMPLEMENTATION OF AN ENCRYPTION FOR ENHANCEMENT DGHV,” Iraqi Journal of Information & Communications Technology, vol. 2, no. 2, pp. 44–57, Nov. 2019, doi: https://doi.org/10.31987/ijict.2.2.69.

[45] S. Gener, P. Newton, D. Tan, S. Richelson, G. Lemieux, and P. Brisk, “An FPGA-based Programmable Vector Engine for Fast Fully Homomorphic Encryption over the Torus,” in SPSL: Secure and Private Systems for Machine Learning (ISCA Workshop), [Online]. Available: https://par.nsf.gov/biblio/10282639. Accessed: May 19, 2025

[46] S. Behera and J. R. Prathuri, "Design of Novel Hardware Architecture for Fully Homomorphic Encryption Algorithms in FPGA for Real-Time Data in Cloud Computing," in IEEE Access, vol. 10, pp. 131406-131418, 2022, doi: 10.1109/ACCESS.2022.3229892

[47] T. Ye, R. Kannan and V. K. Prasanna, "FPGA Acceleration of Fully Homomorphic Encryption over the Torus," 2022 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 2022, pp. 1-7, doi: 10.1109/HPEC55821.2022.9926381.

[48] S. Behera and J. R. Prathuri, “FPGA-Based Acceleration of K-Nearest Neighbor Algorithm on Fully Homomorphic Encrypted Data,” Cryptography, vol. 8, no. 1, p. 8, Mar. 2024, doi: https://doi.org/10.3390/cryptography8010008.

[49] S. Bharany and S. Sharma, “Intelligent green internet of things: An investigation,” in Machine Learning, Blockchain, and Cyber Security in Smart Environments. Chapman and Hall/CRC, 2022, pp. 1–15.

[50] B. Talwar, A. Arora and S. Bharany, "An Energy Efficient Agent Aware Proactive Fault Tolerance for Preventing Deterioration of Virtual Machines Within Cloud Environment," 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2021, pp. 1-7, doi: 10.1109/ICRITO51393.2021.9596453.

[51] N. Shamshad et al., "Enhancing Brain Tumor Classification by a Comprehensive Study on Transfer Learning Techniques and Model Efficiency Using MRI Datasets," in IEEE Access, vol. 12, pp. 100407-100418, 2024, doi: 10.1109/ACCESS.2024.3430109.

[52] S. Badotra et al., “A DDoS Vulnerability Analysis System against Distributed SDN Controllers in a Cloud Computing Environment,” Electronics, vol. 11, no. 19, p. 3120, Sep. 2022, doi: 10.3390/electronics11193120.

[53] S. Kumar et al., “Exploitation of Machine Learning Algorithms for Detecting Financial Crimes Based on Customers’ Behavior,” Sustainability, vol. 14, no. 21, p. 13875, Oct. 2022, doi: 10.3390/su142113875.