HARDWARE IMPLEMENTATION OF AN ENCRYPTION FOR ENHANCEMENT DGHV

Zainab H. Mahmood [1], Mahmood K. Ibrahem [2]
[1] AL-Mamoun University College, Baghdad, Iraq
[2] College of Information Engineering, AL-Nahrain University, Baghdad, Iraq
zainabh.mahmood@gmail.com [1], mahmoodkhalel@coie-nahrain.edu.iq [2]
Received:26/8/2019, Accepted:27/9/2019

***Abstract*- A fully homomorphic encryption (FHE) scheme is considered as a major cryptographic tool in a secure and reliable cloud computing environment, as it enables arbitrary arithmetic processing of a ciphertext without revealing the plaintext. However, due to the very high computation of fully homomorphic encryption system, it stays impractical and unfit for real- time applications. One way to address this restriction is by using graphics processing units (GPUs) and field programmable gate arrays (FPGAs) to implement homomorphic encryption schemes. This paper presents the hardware implementation of enhancement van Dijk, Gentry, Halevi and Vaikuntanathan's (DGHV 10) scheme over the integer using FPGA technology for high speed computation and real time results. The proposed method was simulated via Vivado system generator tools and implemented in FPGA hardware successfully using NEXYS 4 DDR board with ARTIX 7 XC7A100T . The experimental results show that the FPGA-based fully homomorphic encryption system is 63 times faster than the simulated version of the proposed algorithm.**

keywords: **FPGA, Fully homomorphic encryption scheme, Vivado, System generator.**

I. INTRODUCTION

Fully homomorphic encryption (FHE) is an important development in the latest year's cryptographic studies [1]. An FHE scheme can be used without revealing the content of the corresponding plaintext to perform computations on a ciphertext. Therefore, a practical FHE system will open the door to many fresh safety techniques and applications that need privacy, such as cloud-based computing and privacy- preserving search. Gentry suggested the first FHE system in 2009 [1], then numerous systems were suggested based on various hardness assumptions [2], [3], [4], [5], [6], [7], [8] and certain methods were created to enhance efficiency [9], [10], [11], [12]. Four main branches of homomorphic encryption schemes have been developed since 2009: lattice- based, integer-based, learning-with-errors (LWE) or ring-learning- with- errors (RLWE)-based encryption and NTRU-like FHE schemes [13].

Homomorphic Encryption H consists of four functions. H = {Key Generation, Encryption, Decryption, Evaluation} [14], [15]:

1) Key- Gen: The client generates a secret keys sk to encrypt plaintexts.
2) Encryption: The client encrypts the plaintext PT by using a secret key sk and generates ciphertext CT that will be sent to the server , CT= (sk, PT).
3) Evaluation: Server has a function f to be implemented on ciphertext CT.
4) Decryption: New generated ciphertext Eval (f (PT)) will be decrypted by a client using its sk and it gets the original result, Eval (PT)= (sk, f (PT)) .

In any homomorphic system, it is important to consider the format of the ciphertext to be decrypted successfully after an evaluation phase. FPGAs are particularly appropriate for applications that required a high processing computation time.

They are capable of delivering high throughput even at low clock rates due to the inherent parallelism offered by internal architecture and logic resources. With the advantages of FPGA technology and the accessibility of advanced development instruments, FPGAs application regions are increasing at a very large pace and this technology is expected to even replace ASIC in the future [16].

There are few FHE schemes of hardware implementations. Cousins et al. [17], [18] suggested hardware implementation using the Matlab HDL Coder tool on an FPGA platform; and they do not deliver any outcomes of implementation or simulation. So, this work is an attempt to integrate all these issues to accelerate FHE based on FPGA.

II. Related Work

In 2016, the FPGA- based computing accelerator was used as part of a co-processor homomorphic encryption processing unit to execute important computing applications [19]. Advanced design and computation accelerators based on FPGA are introduced in this study as part of a co-processor homomorphic encryption processing unit. By decreasing the computational bottleneck of primitive lattice encryption supporting homomorphic encryption schemes, this hardware accelerator technology makes computing on encrypted information more practical.

In 2017, a software/ hardware co- designed accelerator is proposed to accelerate homomorphic computation by means of a high- level synthesis flow. This paper a big modular polynomial multiplier configurable in both degree and coefficient size and suggested a modular polynomial reducer based on polynomial with overall shape to optimize the homomorphic context [20].

In 2019, design a domain- specific architecture on a heterogeneous Arm + FPGA platform to accelerate homomorphic data computing. At 200 MHz FPGA- clock, configuration achieves more than 13x speed with regard to the extremely optimized FV homomorphic encryption system implemented on an Intel i5 processor operating at 1.8 GHz. Xilinx Zynq MPSo Ultra Scale+ [21].

The contribution of this paper is:

- Accelerate Enhancement of FHE based integer algorithm and implementing it using FPGA technology. The above special FHE algorithm is selected because the theory is extremely simpler, smaller key size, smaller ciphertext size and less execution time for comparable performance to other FHE schemes.
- New hardware architecture for Enhanced FHE over the integers is designed using the proposed multiplier and modular reduction.
- The implementation of Enhancement of FHE based integer was verified for a Xilinx Virtex- 7 FPGA, as well as the results demonstrate that the efficiency of the proposed design is significantly improved by a factor of 63 over similar software implementation. The remainder of the paper is structured as follows. Section II: Reviews the related work. In Section III: the suggested technique (enhancing the fully homomorphic encryption scheme) Section IV describes the suggested FHE encryption hardware architectures. Implementation and performance result. And finally Section V: conclusion.

III. Fully Homomorphic Encryption Proposed Algorithim

After looking in DGHV, SDC and SAM fully homomorphic encryption schemes [3], [22], [23], we take note that the message previous encrypted as one bit, as either 0 or 1, and get a huge ciphertext for one character after converting each character in plaintext to the binary format, which is the input m of the encryption equation mentioned below. Instead of converting each message character into (8- bit) binary format, our suggested encryption scheme [24] encrypts each bit to generate (8- ciphertext) for each plaintext character, the proposed encryption equation take the number or character directly and encrypt as a whole character: $CT = m + 2rp + p.q$, We modified on DGHV encryption equation where CT is the ciphertext, $m\epsilon\lfloor 0, p-1\rfloor$, r is a noise and q is a constantly large integer, produced in one ciphertext for each plaintext character. Algorithm of the proposed system (symmetric encryption) The implementation of the proposed method Enhancement of FHE over Integer scheme [24] is as follows:

- (λ): Chose a random secret key p as a big prime integer.
- Encryption phase $(sk, m\epsilon\,[0, p-1])$: encrypt a message m by private key p.

$$CT = m + 2rp + p.q \tag{1}$$

  where r is a random integer, and q is a constant large integer
- Decryption phase (sk,c): decrypt ciphertext CT

$$m = CT\, mod\, p \tag{2}$$

- Process Phase(Add Multiplication)

To demonstrate that the suggested algorithm promotes the additive and multiplicative characteristics of homomorphism
Suppose, $CT1 = m1 + 2r1p + p.q, CT2 = m2 + 2r2p + p.q$
Additive Homomorphism:
$CT3 = CT1 + CT2 = (m1 + m2) + 2(r1 + r2)p + 2p.q$
$m3 = (CT1 + CT2) mod\, p = m1 + m2$
Multiplicative Homomorphism:
$CT4 = CT1.CT2 = m1.m2 + (m1 + m2 + p.q)p.q + r1(m2 + r2 + p.q)p + r2(m1 + p.q)p.$
$m4 = (CT1.CT2) mod\, p = m1.m2.$

IV. Simulation and Hardware Results

This section includes simulation results (using Systems Generator) and hardware results (using Vivado and FPGA board).

A. Systems Generator Results:

The encryption/decryption systems and homomorphic evaluation process are designed and implemented using system generator and the following sections describe each part in detail.

1) Encryption/ Decryption designed system

The proposed method as described in section III is designed and implemented using system generator blocks as

shown in Fig. 1 The encryption and decryption algorithm is implemented as subsystems in system generator. The Four internal constant blocks (shown in Fig. 1) are used to represent the inputs to the proposed system. These inputs are m (message), p (secret key), r (small integer number), and q (big integer number). The encryption subsystem block details are shown in Fig. 2 The operation is performed using multiplier, constant-multiplier and addsub blocks. These blocks are used to perform the encryption operation described in eq. 1. In1, In2, In3 and In4 are connected to m, r, p and q respectively.

The decryption subsystem block details are shown in Fig . 3 There is no block to perform the mod operation in the system generator, so the decryption operation (mod as described in eq . 2 is built based on the following equation

$$Massege = C - P \times [C/P] \tag{3}$$

As described in the above eq. 3, the operation is performed using divider, multiplier, addsub and convert blocks. The convert blocks are used to extract the fractional parts of the division result. The output of the encryption process (C) is connected to In1, P input is connected to In2.

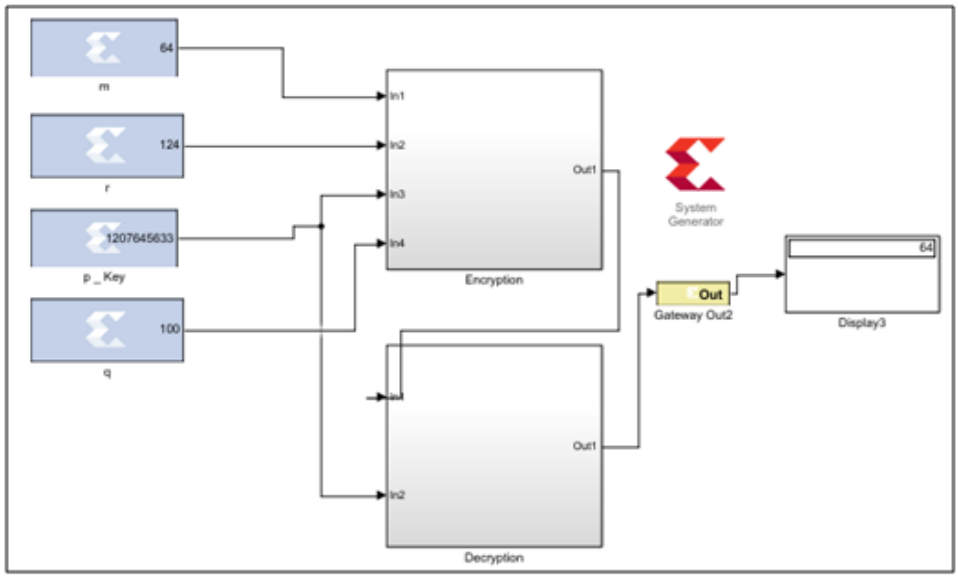


Figure 1: Encryption-decryption system using system generator

Case Study 1

Select a prime p =1207645633, and random integer q =100, r =124 and two messages $m_1 = 72 and m_2 = 65$ Now calculate CT_1: $CT_1 = m_1 + 2r.p + p.q$

$CT_1 = 72 + 2 * 124 * 1207645633 + 1207645633 * 100$

$CT_1 = 36168517777618473087272$

Then calculate CT_2 :

$CT_2 = m_2 + 2rp + p.q$

$CT_2 = 65 + 2 * 124 * 1207645633 + 1207645633 * 100$

$CT_2 = 3616851777618473087265$

Additive Homomorphic Encryption Propriety:

Let two encrypted messages $(CT_1 + CT_2)$ be added as CT_3 :

$CT_3 = CT_1 + CT_2$

$= 3616851777618473087272 + 3616851777618473087265$

$= 72337035555236946174537$

Now decrypt the result of CT_3 :

$m_3 = CT_3 mod p \rightarrow m_3 = 72337035555236946174537 mod 1207645633$

$m_3 = 137, thisisequaltom_1 + m_2 (i.e. 72 + 65 = 137)$

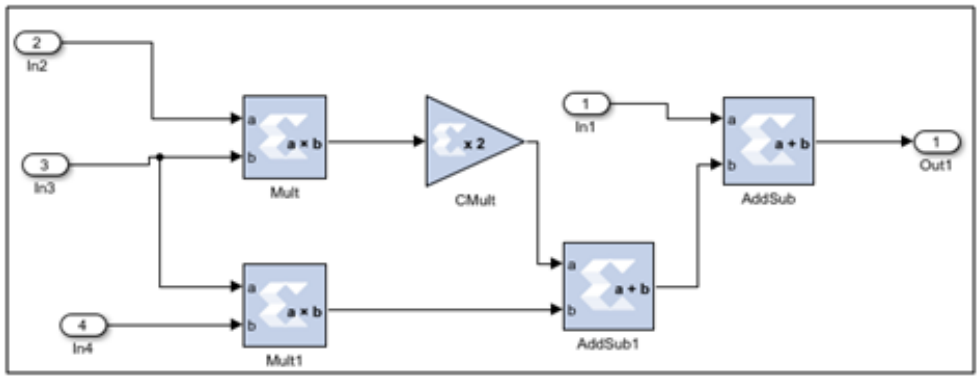


Figure 2: Encryption operation

The case study example is shown in Fig. 4 The system is implemented using the same blocks described above to clarify the results from each block.

Case Study 2

Choose a prime number p =9321, and random integer q =31, r =13 and two messages$m_1 = 60$ and $m_2 = 65$

$NowcalculateCT_1 : CT_1 = m_1 + 2r.p + p.q$

$CT_1 = 60 + 2 * 13 * 9321 + 9321 * 31$

$CT_1 = 531357$

$ThencalculateCT_2$:

$CT_2 = m_2 + 2rp + p.q$

$CT_2 = 65 + 2 * 13 * 9321 + 9321 * 31$

$CT_2 = 531297$

Multiplication Homomorphic encryption propriety:

Let two encrypted messages$(CT_1 * CT_2)$ be multiply as CT_4:

$CT_4 = CT_1.CT_2 = 531357 * 531297 = 282342918234$

Now decrypt CT_4 :

$m_4 = CT_4 mod\, p$

$m_4 = 282342918234 mod\, 9321$

$m_4 = 3900$, this is equal to$m_1 * m_2 (i.e. 60 * 65 = 3900)$

2) Homomorphic test of designed system Fig. 5 represent the Additive homomorphic property test in System Generator. Two encryption subsystem blocks are used to generate two ciphertext CT_1 and CT_2 for two input messages m_1 and m_2 respectively. Two adder blocks are added to the system, one is used to obtain the additive result of $m_1 \& m_2$ before the encryption process(display 5 block in Fig. 5), the second one is used to add $CT_1 \& CT_2$(display 4 block in Fig 5), the result from this adder is used as input to the decryption subsystem block. The results displayed in Fig. 5 (display5 & display3) proved that the Additive homomorphic process is performed successfully.

Fig. 6 represents the multiplicative homomorphic property test in System Generator. Two encryption subsystem blocks are used to generate two ciphertext $CT_1 and CT_2$ for two input messages $m_1 and m_2$ respectively. Two multiplayer blocks are added to the system, one is used to obtain the multiplication result of $m_1 \& m_2$ before the encryption process(display 5 block in Fig. 6), the second one is used to multiply $CT_1 \& CT_2$(display 4 block in Fig. 6), the output of this multiplier is used as input to the decryption subsystem block. The results displayed in Fig. 6 (display 5 & display 3) proved that the multiplicative homomorphic process is performed successfully.

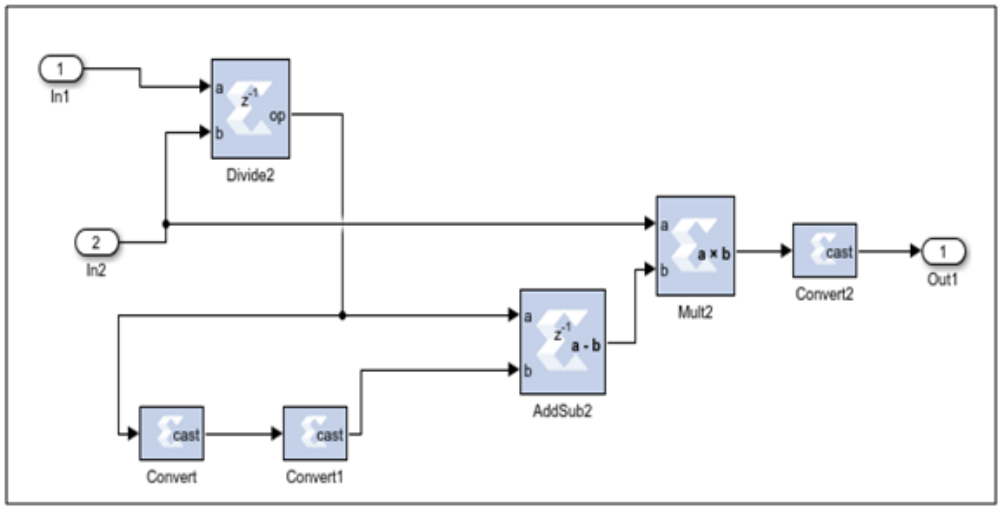


Figure 3: Decryption subsystem using system generator

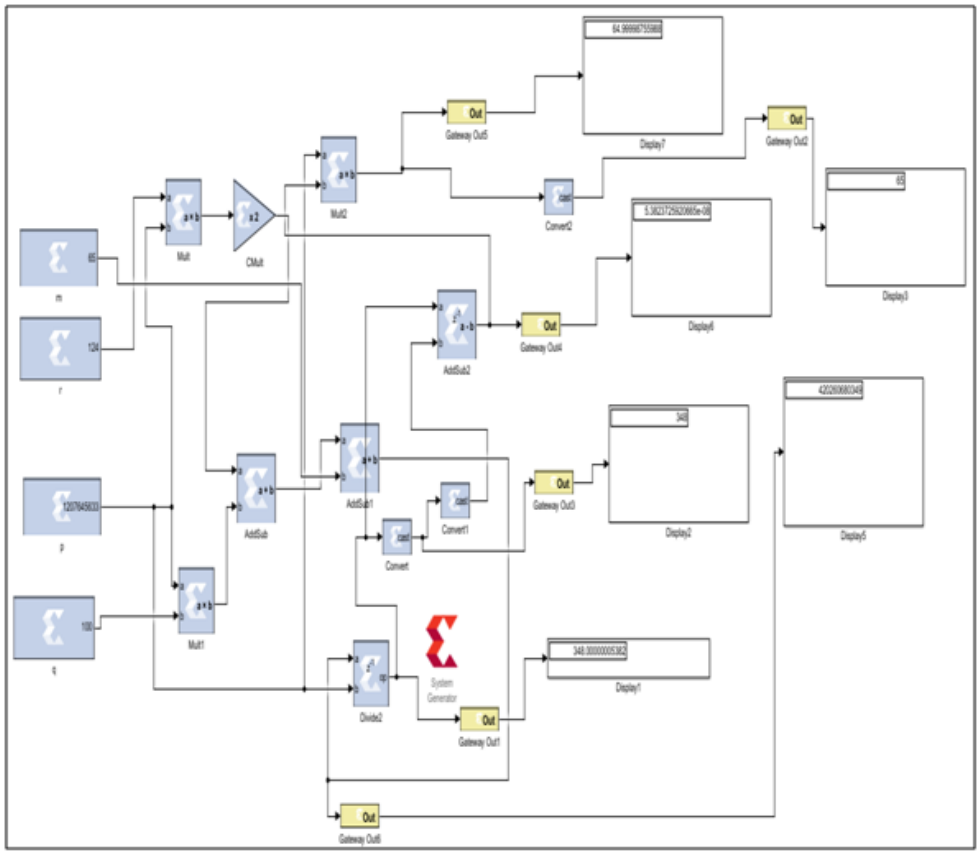


Figure 4: Implementation of case study example

B. Hardware results

All the designed systems described in section III above are converted to VHDL code successfully using HDL netlist option in system generator's settings. Vivado 2017.4 is used to synthesis, implement design and generate bitstream files for the converted VHDL code. The generated bitstream file for the encryption-decryption system is downloaded successfully to the Nexys4 DDR FPGA board as shown in Fig. 7.

Fig. 8, 9 and 10 demonstrate the RTL schematic of the encryption-decryption, encryption subsystem & decryption subsystem respectively. The results shown in Fig. 8, 9 and 10 show that the designed systems performed the encryption and decryption operations successfully.

Fig. 11 displays power analysis of (a) Encryption-Decryption system, (b) Additive homomorphic evaluation and (c) multiplicative homomorphic evolution. As a comparison between them, the power consumption of additive and multiplicative homomorphic evaluation systems are 0.249 W (73%) and 0.262 (74%) respectively, which are higher than the power consumption of encryption-decryption system 0.195 (57%) because the encryption system was duplicated in additive homomorphic evaluation and multiplicative homomorphic evolution designed systems.

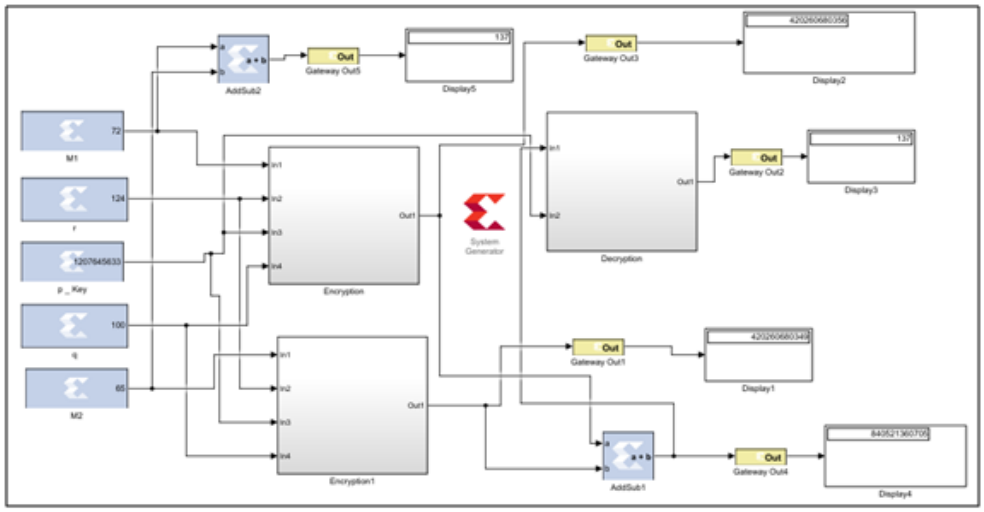


Figure 5: Additive homomorphism evaluation using system generator

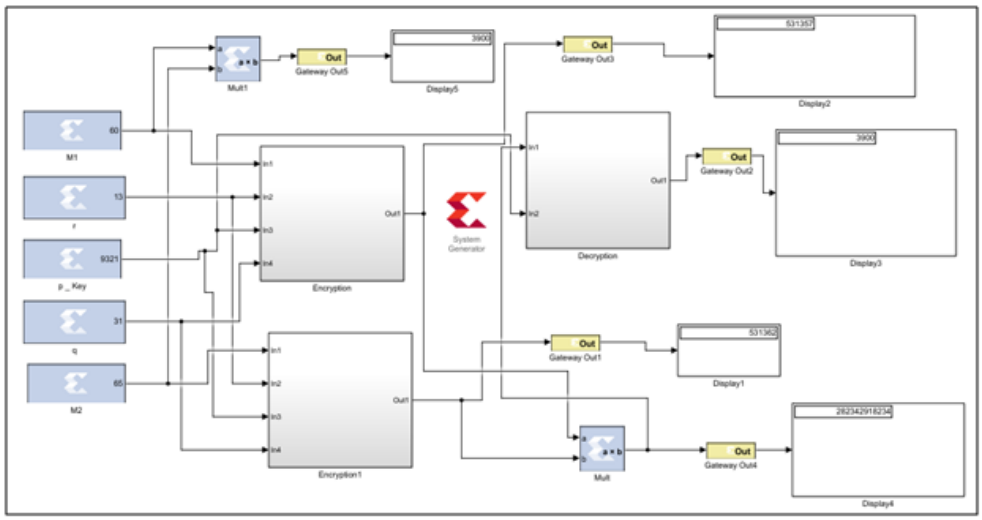


Figure 6: Multiplicative homomorphism evaluation using system generator

Table I displays the summary of the design systems utilization of encryption-decryption system, Additive homomorphic evaluation & multiplicative homomorphic evolution. Table I and Fig. 12 show that the IO blocks utilization of evaluation systems (73% , 74%) are higher than that utilized in encryption-decryption system (12%) because the resources used in evolution systems are more than double resources in an encryption-decryption system. Also resource utilization of evaluation systems take more DSP, LUT, and slices than encryption-decryption system for the same reason.

TABLE I
RESOURCE UTILIZATION OF THE TARGETED FPGA DEVICE

	IO Blocks (210)	DSP (240)	LUT (63400)	Slices (15850)
Encryption-Decryption system	26 (12%)	36 (15%)	5766 (9%)	1736 (11%)
Additive homomorphism evaluation system	154 (73%)	58 (24%)	8816 (14%)	2667 (17%)
Multiplicative homomorphism evaluation system	155 (74%)	80 (33%)	7856 (12%)	2468 (16%)

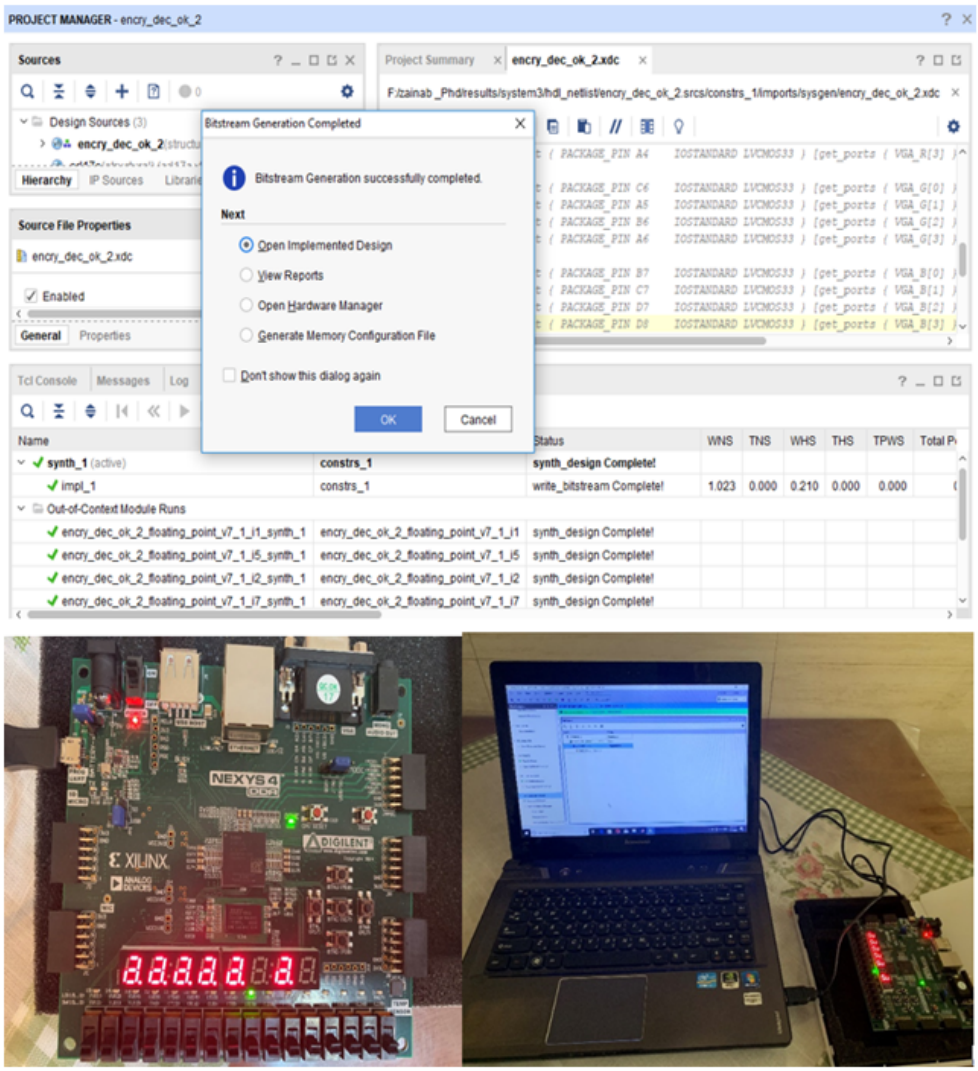


Figure 7: Programing FPGA board with homomorphic encryption bitstream

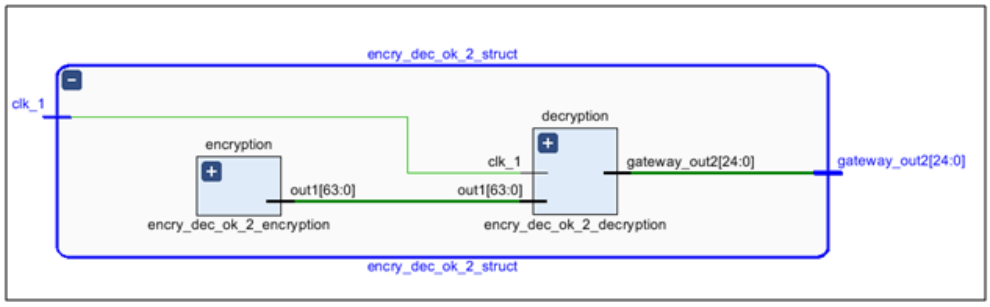


Figure 8: RTL schematic for encryption-decryption system

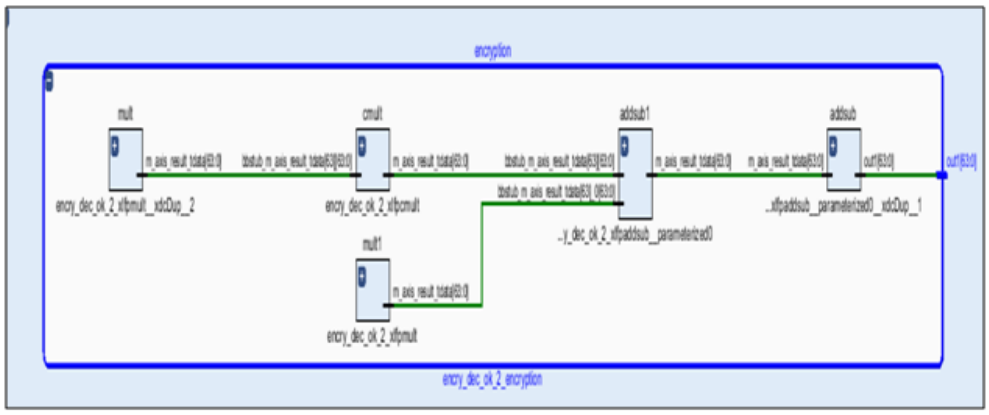


Figure 9: RTL schematic of encryption

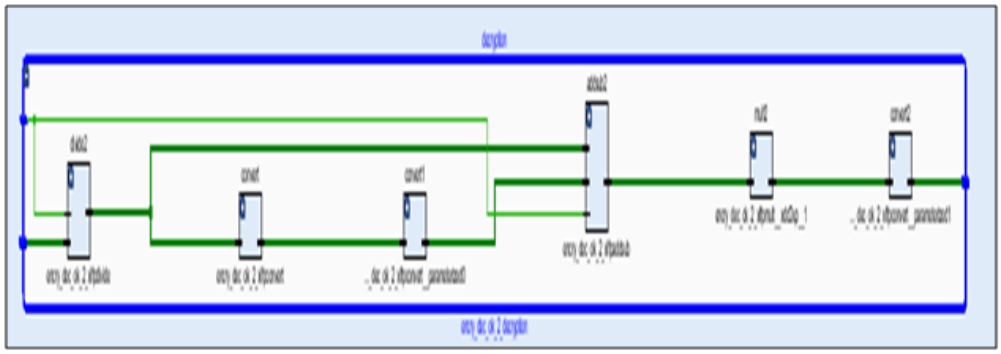
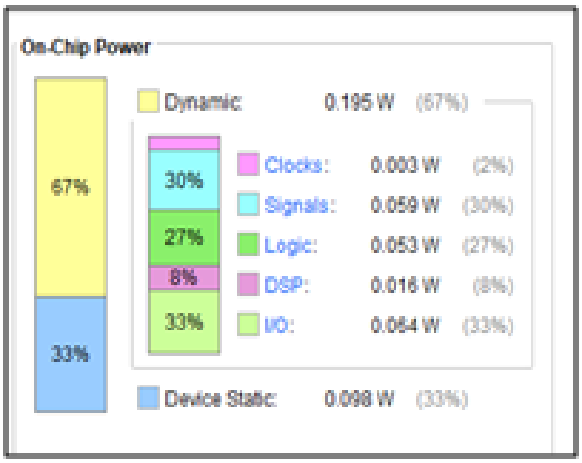
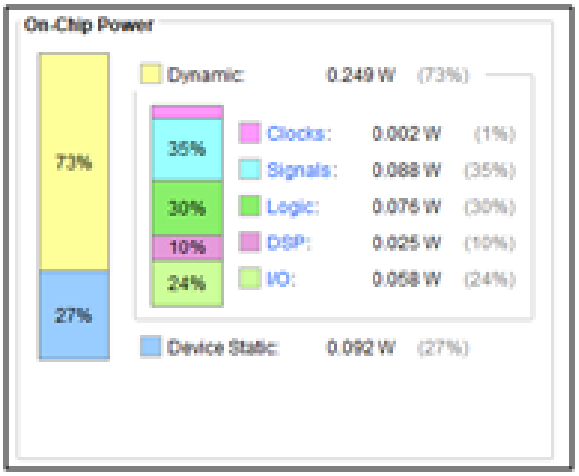


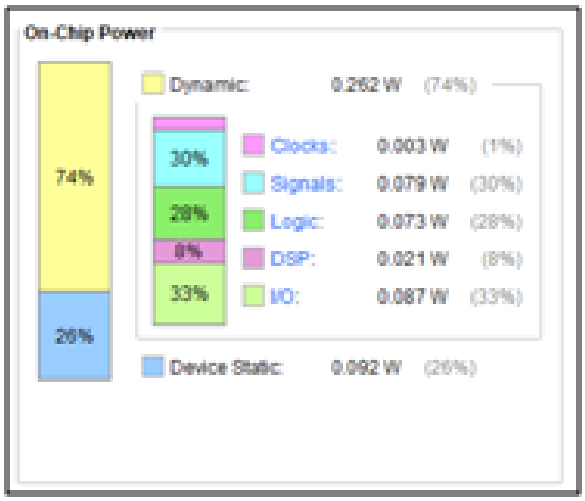
Figure 10: RTL schematic of decryption subsystem



(a) Encryption- decryption system

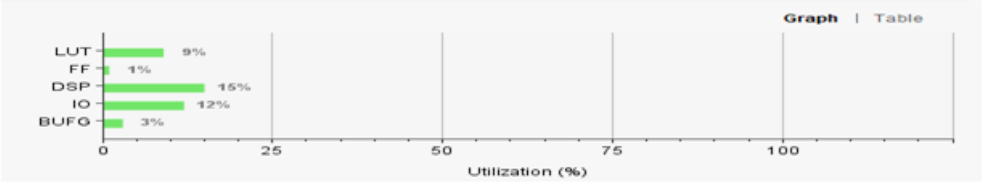


(b) Additive homomorphic evolution

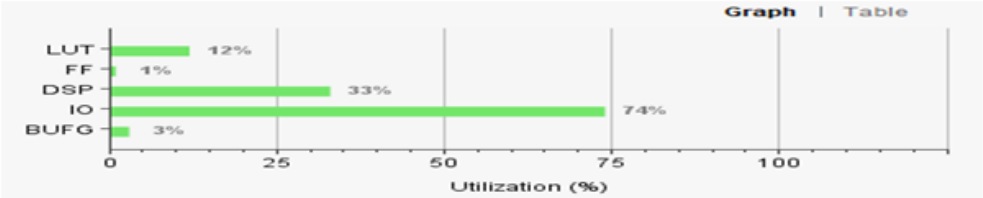


(c) multiplication homomorphic evolution

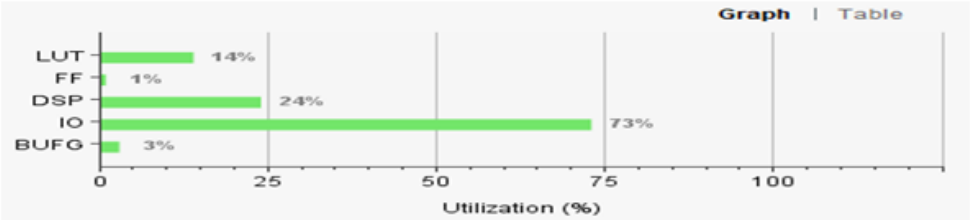
Figure 11: Power analysis of the designed systems



(a) Encryption-decryption system



(b) Additive homomorphic evaluation



(c) Multiplicative homomorphic evolution

Figure 12: Utilization summary

V. CONCLUSIONS

In this paper, the design and implementation of homomorphic system (encryption- decryption system, additive homomorphism evaluation system, and multiplicative homomorphism evaluation system) are performed in Vivado system generator tools. All the design systems are downloaded successfully to FPGA board using NEXYS 4 DDR board with ARTIX 7 XC7A100T FPGA. The targeted FPGA board (running at 100 MHz) speed up the processing time to 71 ns, which is about 63 times faster than the powerful CPU. The results show that, the IO blocks used in targeted FPGA device are 12%, 73% and 74% of encryption- decryption, additive and multiplicative homomorphic evolution systems respectively. The results proved that the FPGA implementation solved the bottleneck issue of computation time for homomorphic encryption. The FPGA-based implementation has much reduced power consumption and better results compared to the acceleration of the CPU. As a future effort towards the design of the enhancement DGHV for cloud computing, our initial development of the large-number multiplier model demonstrates a promising hardware acceleration.

REFERENCES

[1] C. Gentry, "A fully homomorphic encryption scheme" , Dissertation, no. September, p. 169, 2009.
[2] C. Gentry, "Fully homomorphic encryption using ideal lattices" , Proc. 41st Annu. ACM Symp. Symp. theory Comput. - STOC 09, p. 169, 2009.
[3] M. Van Dijk and C. Gentry, "Fully Homomorphic Encryption over the Integers" , pp. 1- 28, 2010.
[4] Z. Brakerski and V. Vaikuntanathan, "Efficient Fully Homomorphic Encryption from (Standard)LWE " SIAM J. Comput., vol. 43, no. 2, pp. 831-871, 2014.
[5] "Leveled fully homomorphic encryption without bootstrapping" , ACM Trans. Comput. Theory, pp. 309- 325, 2012.
[6] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP" , Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 7417 LNCS, pp. 868- 886, 2012.
[7] A. Lopez- Alt, E. Tromer, and V. Vaikuntanathan, "On the fly multiparty computation on the cloud via multikey fully homomorphic encryption" , p. 1219, 2012
[8] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based" , Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8042 LNCS, no. PART 1, pp. 75- 92, 2013.
[9] C. Gentry, S. Halevi, and N. P. Smart, Fully homomorphic encryption with polylog overhead, vol. 7237 LNCS. 2012.
[10] Z. Brakerski, C. Gentry, and S. Halevi, "Packed ciphertexts in LWE- based homomorphic encryption" , Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 7778 LNCS, pp. 1- 13, 2013.
[11] J. Alperin- Sheriff and C. Peikert, " Faster bootstrapping with polynomial error" , Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8616 LNCS, no. PART 1, pp. 297- 314, 2014.
[12] R. Hiromasa, M. Abe, and T. Okamoto, "Packing Messages and Optimizing Bootstrapping in GSW- FHE Fully Homomorphic Encryption FHE) ppt" , 2015.
[13] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, " A Survey on Homomorphic Encryption Schemes: Theory and Implementation" , pp. 1- 35, 2017.
[14] P. V.Parmar, S. B. Padhar, S. N. Patel, N. I. Bhatt, and R. H. Jhaveri, " Survey of Various Homomorphic Encryption algorithms and Schemes" , Int. J. Comput. Appl , vol. 91, no. 8, pp. 26- 32, 2014.
[15] G. Liu, G. Yang, H. Wang, Y. Xiang, and H. Dai, " A Novel Secure Scheme for Supporting Complex SQL Queries over Encrypted Databases in Cloud Computing" , Secur. Commun. Networks, vol. 2018, 2018.
[16] E. Ozturk, Y. Doroz, B. Sunar, and E. Savas " , Accelerating Somewhat Homomorphic Evaluation using FPGAs" , IACR Cryptol. ePrint Arch., pp. 1- 15, 2015.
[17] D. B. Cousins, K. Rohloff, C. Peikert, and R. Schantz, " An update on SIPHER (Scalable Implementation of Primitives for Homomorphic EncRyption) - FPGA implementation using Simulink Recent Developments in the SIPHER SHE Scheme A Review of Fully and Somewhat Implementing Fast Modulo Add , Subtract and Multiply " , 2012.
[18] D. Cousins, K, Rohloff, and R, S. Bbn, " Sipher: Scalable Implementation of Primitives for Homomorphic Encryption FPGA implementation using Simulink Motivation for Fully Homomorphic Encryption Our encryption scheme and primitives Examples " , nProc. 15th Annu. Work. High Perform. Embed. Comput. (HPEC 2011) , no. November, pp. 1- 16, 2011.
[19] 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.
[20] A. Mkhinini, P. Maistri, R. Leveugle, and R. Tourki, " Co-designed accelerator for homomorphic encryption applications" , Adv. Sci. Technol. Eng. Syst. , vol. 3, no. 1, pp. 426- 433, 2018.
[21] S. Sinha Roy, F. Turan, K. Jarvinen, F. Vercauteren, and I. Verbauwhede, " FPGA- Based high- performance parallel architecture for homomorphic computing on encrypted data" , Proc. - 25th IEEE Int. Symp. High Perform. Comput. Archit. HPCA 2019, pp. 387- 398, 2019.
[22] I. Jabbar, "Using Fully Homomorphic Encryption to Secure Cloud Computing" , Internet Things Cloud Comput. vol. 4, no. 2, p. 13, 2016.
[23] S. S. Hamad and A. M. Sagheer, "Design of fully homomorphic encryption by prime modular operation" , Telfor J. , vol. 10, no. 2, pp. 118- 122, 2018.
[24] Zainab H. Mahmood, Mahmood K. Ibrahem" Enhancement of Fully Homomorphic Encryption Scheme over Integer for real world application" submitted to Indonesian Journal of Electrical Engineering and Computer Science, 2019.