Comparison of Hand-Written RTL code against High-Level Synthesis for Blowfish and Tiny Encrpytion Algorithm (TEA)

Muhammad Awais Hussain
Electrical Engineering Department,
National Central University,
Zhongli. Taiwan
105521601@cc.ncu.edu.tw

Rabiah Badar
Electrical Engineering Department,
COMSATS Institute of Information
Technology, Islamabad. Pakistan
rabiah.badar@comsats.edu.pk

Syed Waqar Nabi
School of Computing Science
University of Glasgow,
UK
syed.nabi@glasgow.ac.uk

***Abstract*—Cryptography is the backbone of a secure and reliable communication system. Data security while transmission depends upon the strength of cryptographic algorithm. In this work, Tiny Encryption Algorithms (TEA) and Blowfish algorithms has been implemented using the High Level Synthesis (HLS) and hand-written Register Transfer Level (RTL) approaches in Xilinx Vivado HLS and Xilinx ISE. Comparative evaluation for both implementation approaches has shown that RTL approach is outperforming HLS approach in both algorithms for different parameters like throughput, frequency etc., due to flexibility of designing modules in RTL as compared to HLS approach.**

Keywords—cryptograpphy; blowfish; TEA; FPGA; High Level Synthesis; RTL

I. INTRODUCTION

Data communication is an essential part of many man made systems to ensure their reliable operation. Data communication is not only important for normal daily life of common man but also for systems like power system, control systems, military applications, etc. However, the security of data being stored or transmitted in a communication system is of primary importance to ensure user privacy and secure operation of a system. However, threat of breach of data privacy known as *eavesdropping* from a simple mobile communication of common person to high level communication of armed forces and agencies, is always there.

Cryptography is a process which ensures the data security while transmission over insecure channel by converting data into some uninterpretable format using keys. These keys change the form of original data into random unreadable data such that the data could be received and decrypted only by the intended receiver. The receiver already knows the shared keys to retrieve the original data.

With the passage of time, there has been evolution in encryption algorithms used in cryptography. Tiny Encryption Algorithm (TEA), Blowfish Encryption Algorithm, Advanced Encryption Standard and International Data Encryption Algorithm (IDEA) are some examples of encryption algorithms which have been playing an important role in data security. For the current case study, TEA and Blowfish Encryption algorithm have been selected because both algorithms have feistel network structure, 64-bit processing of data and both are symmetrical key algorithm. Both algorithms are competing algorithms and can be used for replacing DES and IDEA [1], [2].

FPGAs have been a strong choice for energy efficient accelerated computing. For this purpose, Hardware Descriptive Languages (HDLs) have remained primary choice for hardware designers. Verilog and VHDL played an important role for a long time in digital chip industry. In 2005, System Verilog joined HDLs which was also a hardware verification language. Increase in chip complexity with passage of time demanded the design automation at higher functional level which eventually led to the emergence of High Level Synthesis (HLS) tool.

Due to advancement in hardware platforms and development tools, System-on-Chip (SoC) has drawn attention of researchers for implementation of cryptographic algorithms to take benefits of hardware acceleration. Huge literature can be found for implementation of these cryptographic algorithms considering different performance dimensions and applications like changing key structure, pipelining, Internet of Things (IOT), etc. [3]–[7]. Modern HLS tools can facilitate to reduce the design time for applications with critical time to market.

Early development of HLS mainly targeted ASIC designs, however, with improved capacity, speed and programmability of FPGAs many HLS tools have been developed specifically for FPGAs. Advanced Encryption Standard (AES) implementation using HLS has been discussed showing the tradeoffs for use of both tools in [8]–[10] and they show that HLS is outperforming software implementations. In [11] a detailed survey of HLS development for different applications and input programming languages has been presented. Previous studies mostly focused on comparison of software and HLS implementation. Although [12] and [13] have done comparison of performance of HLS and handwritten RTL code but their study focus is on filters and image processing rather than cryptography. [7] shows comparison of RTL and high level synthesis approach in domain of cryptography but for AES only.

In 2004, commercial HLS products were introduced to synthesize circuits specified at C level to Register Transfer

Level (RTL) [11]. Currently, HLS tools can generate RTL descriptions from some popular high level languages like CUDA, MATLAB, C, C++ and OpenCL etc. [14].

The aim of this work is to;

- Check whether handwritten RTL code outperforms HLS generated codes for cryptographic algorithms.
- Compare handwritten RTL implementation of TEA and Blowfish algorithm in case of selection of specific algorithm for application on the basis of speed, resources usage etc.

The rest of the paper is organized as follows; section II presents the structural details and operation of cryptographic algorithms. Section III presents the simulation results and their analysis, whereas, section IV finally concludes the findings of this research.

II. Cryptographic Algorithms

Based on the nature of key, encryption algorithms have two types; symmetric and asymmetric key algorithm. Asymmetric key algorithm uses different while symmetric key algorithms use the same key for encryption and decryption. The encryption strength of symmetric key algorithm depends upon the strength of keys. Symmetric key algorithms are almost 1000 times faster than asymmetric key algorithms and thus are more suitable to encrypt large amount of data. Also, asymmetric algorithm must use strong key as compared to symmetric to achieve the same security level.

Symmetric key algorithms can further be divided into block and stream ciphers. Stream cipher processes the digital data stream bit by bit or byte by byte, whereas, block cipher encrypts data in the form of blocks of plain text. In this work, two symmetric key block algorithms, Tiny Encryption Algorithm (TEA) and Blowfish Encryption Algorithm, are analyzed to evaluate their hardware implementation using HLS and RTL approaches because software implementation of both algorithms is slow although it is easier than hardware implementation. So, one of our aims is to attain speed up along with comparison of RTL and HLS implementation.

A. Blow Fish Algorithm

Blowfish is a Feistel network which iterates encryption/decryption function 16 times. It was first proposed by Brute Schneider in 1993. It is still unpatented and has high security feature. Blowfish runs faster than DES or AES because it does not require changing of keys frequently [15]. This property makes it best for applications where small number of data packages are sent e.g. emergency control signals. Blowfish is a symmetric key algorithm. The key length may vary from 42 to 448 bits. Fig. 1 shows the complete encryption process for Blowfish.

The algorithm has 4 sub-key arrays named S1, S2, S3, S4 and P. S1, S2, S3 and S4 has 256 entries while P array has 18 entries of 32 bits. There are 16 rounds in the process. Each round r has 4 actions as described in following relations;

$$y_{k+1} = F\left(y_k \oplus P_k\right) \oplus z_k \tag{1}$$

$$z_{k+1} = y_k \oplus P_k \tag{2}$$

Where, F is Feistel function.

In last round, following operations are carried out without swapping the data from preceding round;

$$y_{18} = y_{17} \oplus P_{18} \tag{3}$$

$$z_{18} = \left(F \oplus z_{16}\right) \oplus P_{17} \tag{4}$$

Where, y_k and z_k are left and right halves of input data, respectively.

Decryption algorithm for Blowfish follows the same process with P blocks used in reverse order.

The hex digits of pi are used to initialize the values of S-boxes and P-boxes. These values are *XOR*ed with variable length user input key. Then a block of zeros is encrypted and results of this encryption are used for P_1 and P_2 entries. The resulting cipher-text from zero block is encrypted again and results are used for P_3 and P_4 box. This whole process is repeated again and again until all entries of P-box and S-box have been replaced completely. This complex key scheduling makes Blowfish an effective and high security cryptographic algorithm.

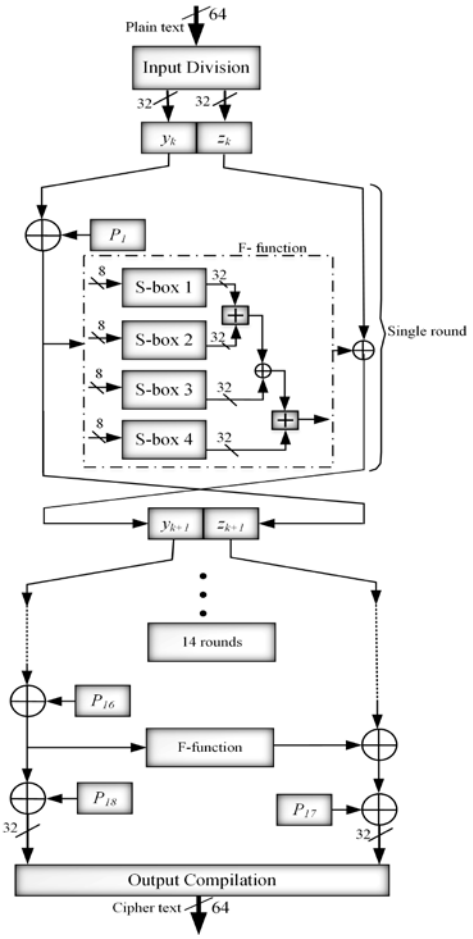

Fig. 1. Generalized depiction of encryption for Blowfish algorithm

B. Tiny Encryption Algorithm (TEA)

Tiny Encryption Algorithm (TEA) was introduced by David Wheeler and Roger Needham in 1994 at Cambridge University [1]. TEA uses generic algebraic operations like addition, XOR and shift operations. It is known as block cipher because it encrypts and decrypts plain text in blocks of specific size [16].

It uses a large number of rounds for encryption rather than a big and complicated program. Nonlinear iterations of different number of rounds make it secure [17]. The memory requirements of TEA is one fourth the requirements of AES [18]. TEA is suitable for embedded implementation due to simple arithmetic operations. TEA has weakness for small number of rounds and it can be seen in Avalanche effect which was found for 6 rounds [1]. Fig. 2 shows the block diagram of complete encryption and decryption round for TEA.

The algorithm processes data in the form of two blocks of 32 bits by splitting the input data of 64 bits into left and right halves. The keys are generated using simple key scheduling algorithm by splitting the 128 bit key into four groups of 32 bits each i.e. k_0, k_1, k_2, k_3. The key is known to both encryption and decryption processes. In order to improve the reliability, the algorithm is applied for more than one round by repeating the basic module. It could normally be repeated for 16 or 32 cycles to get sufficiently secured results.

The addition and subtraction operation would be performed depending upon the encryption or decryption module, respectively. The 64-bit input is divided into two halves of 32 bits, *y* and *z*. The values of *y* and *z* for a single round are updated using following relation;

$$y_{k+1} = y_k \pm \left\{\left(\left(z_k \square\ 4\right)+k_0\right) \wedge \left(z_k \pm Delta_m\right) \wedge \left(\left(z_k \square\ 5\right)+k_1\right)\right\} \quad (5)$$

$$z_{k+1} = z_k \pm G(y_k) \quad (6)$$

Where, $G(y_k)$ is a functional of y_k and is given as;

$$G\left(y_k\right) = \left(\left(y_{k+1} \square\ 4\right)+k_2\right) \wedge \left(y_{k+1} \pm Delta_n\right) \wedge \left(\left(y_{k+1} \square\ 5\right)+k_3\right) \quad (7)$$

Where, '<<', '>>' and '^' are left, right shift and XOR operators respectively.

A constant delta is used to ensure the distinct sub-keys generation and is derived from golden number ratio using the following relation;

$$Delta_{in} = \left(\sqrt{5}-1\right)2^{31} \quad (8)$$

In order to ensure the convergence, the value of delta is updated at the end of every round using the following relation;

$$Delta_{k+1} = Delta_{in} + Delta_k \quad (9)$$

A dummy variable *sum* has been introduced to update the value of delta.

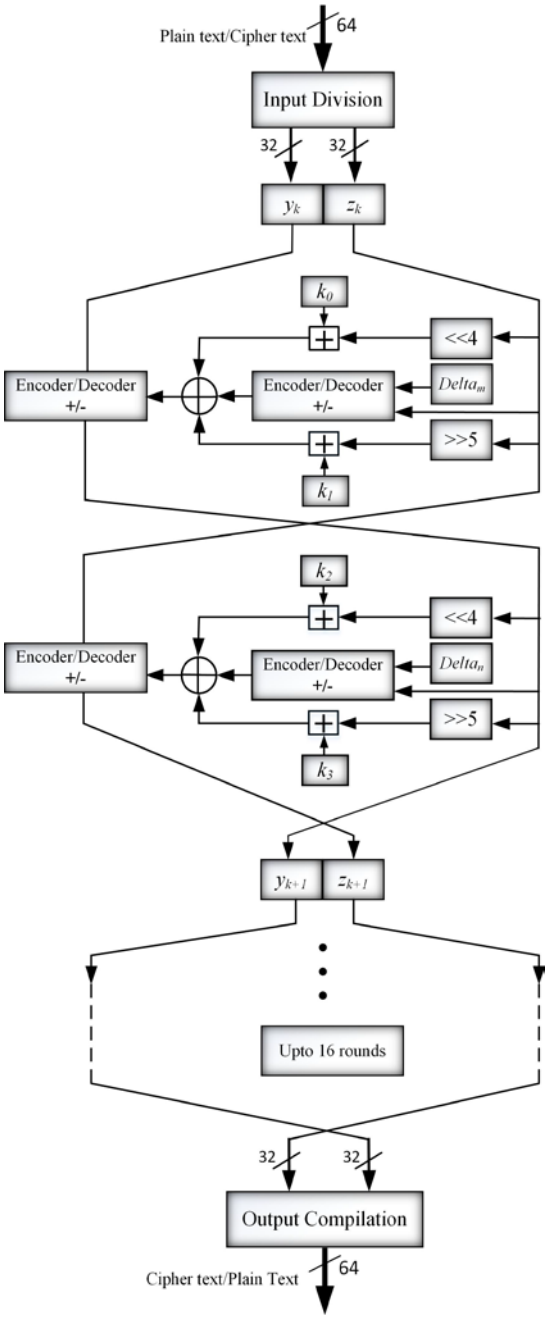

Fig. 2. Generalized depiction of encryption and decryption for TEA

III. METHODOLOGY

For Verilog implementation of both algorithms, whole codes were broken down into different modules which made it easy to access specific module in case of any errors.

In Blowfish implementation, array data was converted into linear register before passing it into modules, wherever it was necessary, due to the limitations of IEEE standards of Verilog [19]. In start integer manipulation was done, whose output was used to calculate sub-keys. Selected sub-keys were used in calculation of encrypted and decrypted results. After one round of encryption and decryption was performed, a specific

signal was triggered to reset values of specific registers and to load new inputs for next round of encryption and decryption.
In calculation of sub-keys and performing operations of encryption and decryption, sequential and combinational design techniques were used at different parts to speed up operations. Sub-keys are calculated using combinational technique and these selected keys are used in main operations of encryption and decryption, which are being performed under sequential technique.
In implementation of TEA, 'if' statements are being checked on a single clock edge. Depending upon count of specific counter, multiple operations are performed under single clock edge. This brings parallelism in design. Specific counter is increased at every clock cycle and after completion of single round of encryption or decryption, its value is reset to perform operations on new input.

IV. Simulation Results and Discussion

Blowfish and TEA are implemented for comparison in HLS tool and handwritten HDL code. Details of reference software implementation of C code of TEA and Blowfish can be found in [1], [17], [20]. Functional correctness was ensured by creating test benches for C code in HLS tool. Vivado HLS 2014.2 is used for synthesis of HDL code for HLS-based approach.

In order to validate the RTL Verilog implementation of TEA and Blowfish algorithms the behavioral simulation results were achieved using ISIM simulator included in Xilinx ISE Design Suite 14.5.

After ensuring the validity of design using behavior simulation the place and route simulation results were synthesized for the device Spartan 6-xc6slx45 with package CSG324C and speed grade of -3. XPower Analyzer, a tool in Xilinx ISE 14.5 package has been used for power utilization results.

The results shown in Fig. 3 give the details of hardware resource utilization for 16 rounds of encryption and decryption modules for TEA block cipher using HLS and RTL approaches. Throughput and throughput in terms of area has been calculated using the following formulae;

$$\text{Throughput} = \frac{\text{no. of bits processed}}{\text{no. of clock cycles} \times \text{clock period}} (Mb/s) \quad (10)$$

$$\text{Area Throughput} = \frac{\text{Throughput}}{\text{Area slices}} \left(Mb/slice\right) \quad (11)$$

RTL design has much higher working clock frequency as compared to HLS approach with significantly low latency which in turns results in high throughput for RTL based implementation. HLS approach is taking almost 8 times more clock cycles than RTL code. RTL implementation has less utilization of Look Up Tables (LUTs) as compared to HLS based design. However, HLS implementation has efficient use of slice registers as compared to RTL based approach. Due to high throughput of RTL based approach, throughput/area is also higher in RTL implementation as compared to HLS based implementation.

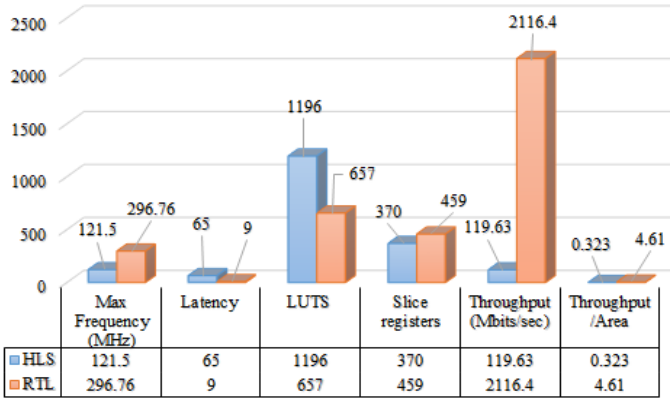

Fig. 3. RTL and HLS results for TEA

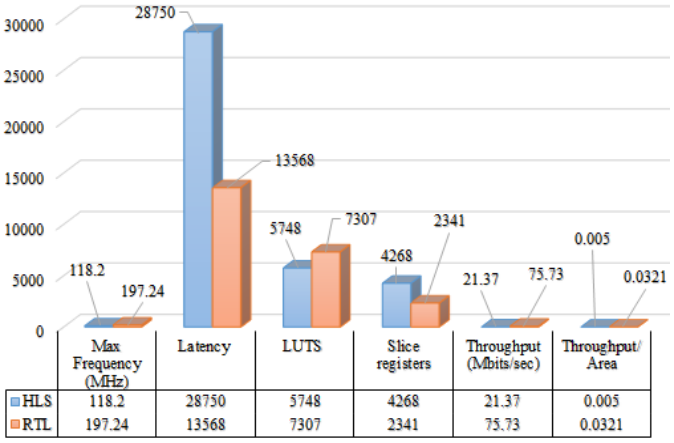

Fig. 4. RTL and HLS results for Blowfish

Fig. 4 summarizes the results of hardware resource utilization for Blowfish in RTL Verilog and HLS. Results for clock frequency, latency, throughput and throughput/area follow the same trend as in case of TEA algorithm. However, LUTs and slice register utilization is opposite to that of TEA for HLS and RTL. Latency for HLS based implementation is almost double as compared to RTL based implementation.

V. Conclusion

This article presents FPGA implementation of two cryptographic algorithms i.e. Blowfish and TEA, using HLS and RTL based approaches. Firstly, the comparative evaluation of both cryptographic algorithms has been done based on resource utilization and different performance evaluation parameters. Then the comparison has been made in context of HLS or handwritten RTL code. Both algorithms were implemented using Verilog HDL. The accuracy results for both algorithms are same, however, TEA having relatively simple computational structure utilizes less resources as compared to Blow fish algorithm. Furthermore, it has been found that RTL based approach is outperforming HLS for different performance measures. HLS based approach has efficient usage of LUTs and Slice registers. The results show that TEA is a good choice for delay sensitive applications, however, the strong key scheduling and complex encryption scheme makes Blowfish a suitable choice for high level of data protection. In

Verilog RTL code, speed of both algorithms can further be improved by introducing pipeline approach. Creating RTL code in Verilog is more time consuming as compared to HLS code generation. A hardware designer can exploit the time saving feature in HLS based designs to optimize the code for efficient utilization of FPGA area and better clock frequency.

REFERENCES

[1] D. J. Wheeler and R. M. Needham, "TEA, a tiny encryption algorithm," *Fast Softw. Encryption*, vol. 1008, no. 3, pp. 363–366, 1995.

[2] D. Salomon, *Data Privacy and Security: Encryption and Information Hiding*. Springer Science & Business Media, 2003.

[3] B. Cody, J. Madigan, S. Macdonald, K. W. Hsu, and S. Industries, "High Speed SOC Design for Blowfish Cryptographic Algorithm," pp. 284–287, 2007.

[4] S. R. Chatterjee, S. Majumder, B. Pramanik, and M. Chakraborty, "FPGA Implementation of Pipelined Blowfish Algorithm," in *2014 Fifth International Symposium on Electronic System Design*, 2014, pp. 208–209.

[5] P. Israsena, "Design and Implementation of Low Power Hardware Encryption for Low Cost Secure RFID Using TEA," pp. 1402–1406, 2005.

[6] P. Israsena, "On XTEA-based Encryption/Authentication Core for Wireless Pervasive Communication," *Int. Symp. Commun. Inf. Technol. 2006. Isc. '06*, pp. 59–62, 2006.

[7] E. Homsirikamol and K. Gaj, "Can High-Level Synthesis Compete Against a Hand-Written Code in the Cryptographic Domain ? A Case Study ∗," *ReConFig*, pp. 1–8, 2014.

[8] K. Rupnow, Y. Liang, Y. Li, and D. Chen, "A study of high-level synthesis: Promises and challenges," *Proc. Int. Conf. ASIC*, pp. 1102–1105, 2011.

[9] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-level synthesis for FPGAs: From prototyping to deployment," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 30, no. 4, pp. 473–491, 2011.

[10] M. Watanabe, K. Iwai, H. Tanaka, and T. Kurokawa, "High-Speed Implementation of Encryption Circuit using a High-Level Synthesis Tool," vol. 3, no. 1, pp. 63–66, 2014.

[11] G. Martin and G. Smith, "High-Level Synthesis: Past, Present, and Future," *Ieee Dtc*, vol. 26, no. 4, pp. 18–25, 2009.

[12] T. Damak, L. A. Ayadi, N. Masmoudi, and S. Bilavarn, "HLS and manual Design methodology for H.264/AVC deblocking filter," *2015 World Congr. Inf. Technol. Comput. Appl. WCITCA 2015*, 2015.

[13] M. D. Zwagerman, "High Level Synthesis , a Use Case Comparison with Hardware Description Language," 2015.

[14] Y. Liang, K. Rupnow, Y. Li, D. Min, M. N. Do, and D. Chen, "High-Level Synthesis: Productivity, Performance, and Software Constraints," *J. Electr. Comput. Eng.*, vol. 2012, pp. 1–14, 2012.

[15] S. R. Chatterjee, S. Majumder, and B. Pramanik, "2014 Fifth International Symposium on Electronic System Design FPGA Implementation of Pipelined Blowfish Algorithm," pp. 208–209, 2014.

[16] S. Gaba, I. Aggarwal, and S. Pandey, "Design of Efficient XTEA Using Verilog," vol. 2, no. 6, pp. 1–5, 2012.

[17] Rajashekarappa, S. K M, and Devi Sumithra, "Study on Cryptanalysis of the Tiny Encryption Algorithm," *Int. J. Innov. Technol. Explor. Eng.*, vol. 2, no. 3, pp. 88–91, 2013.

[18] V. Venugopal and D. M. Shila, "High throughput implementations of cryptography algorithms on GPU and FPGA," *Conf. Rec. - IEEE Instrum. Meas. Technol. Conf.*, pp. 723–727, 2013.

[19] "IEEE Standard Verilog Hardware Description Language," p. 0_1-856, 2001.

[20] Y. Hara, H. Tomiyama, S. Honda, and H. Takada, "Proposal and Quantitative Analysis of the CHStone Benchmark Program Suite for Practical C-based High-level Synthesis," *J. Inf. Process.*, vol. 17, pp. 242–254, 2009.