Indonesian Journal of Electrical Engineering and Computer Science
Vol. 7, No. 1, July 2017, pp. 263 ~ 275
DOI: 10.11591/ijeecs.v7.i1.pp263-275

Customized Hardware Crypto Engine for Wireless Sensor Networks

Medhat Awadalla*[1], Ahmed Al Maashri[2], Lavanya Pathuri[3], Afaq Ahmad[4]
[1,2,3,4]Department of Electrical and Computer Engineering, Sultan Qaboos University, Oman
[1]Department of Communication and Computers, Helwan University, Egypt
*Corresponding author, e-mail: medhatha@squ.edu.om

Abstract

Nowadays, managing for optimal security to wireless sensor networks (WSNs) has emerged as an active research area. The challenging topics in this active research involve various issues such as energy consumption, routing algorithms, selection of sensors location according to a given premise, robustness, and efficiency. Despite the open problems in WSNs, already a high number of applications available show the activeness of emerging research in this area. Through this paper, authors propose an alternative routing algorithmic approach that accelerate the existing algorithms in sense to develop a power-efficient crypto system to provide the desired level of security on a smaller footprint, while maintaining real-time performance and mapping them to customized hardware. To achieve this goal, the algorithms have been first analyzed and then profiled to recognize their computational structure that is to be mapped into hardware accelerators in platform of reconfigurable computing devices. An intensive set of experiments have been conducted and the obtained results show that the performance of the proposed architecture based on algorithms implementation outperforms the software implementation running on contemporary CPU in terms of the power consumption and throughput.

Keywords: *Wireless sensor network, Power-efficient crypto system, Hardware accelerators, reconfigurable computing devices*

1. Introduction

Wireless Sensor Networks (WSNs) are the original order of ad-hoc networks. They comprehend a complex linkage of self-determining sensor nodes that sense, measure, and gather information from its remote environment. They send information congregated through wireless communications. WSNs are vulnerable to security attacks by their broadcast nature in transmission, tiny battery, CPU power, and constrained memory capacity. Wireless Sensor Networks (WSNs) are the special order of ad-hoc networks that comprehend a complex autonomously linked sensor nodes deployed to sense, measure, and gather information from its remote environment. Sensor nodes come along with a radio transceiver, a microcontroller, an electronic circuit to interface with sensors and energy source, usually a battery or an embedded form of energy generation. The sensor nodes are deployed in the field of observation in a manner to facilitate smooth, uninterrupted inter-nodal and gateway communication. A sensor node varies in dimensions from that of a shoebox down to the size of a grain of dust [1] although functioning "motes" of genuine microscopic dimensions have yet to be created. Nowadays, there is a drastic reduction in the size of the sensor nodes with equal or even greater sensing capability.

Technological fields adopt WSN owing to their advantageous applicability in unique critical environments. Significant ones include harsh environments (e.g. battle fields and nuclear plants), short-range (e.g. temperature and smoke detection), large area deployment (e.g. forests, agricultural land, and buildings). Equally, important applications include intrusion detection, surveillance, natural disaster relief, seismic survey, industrial, underwater and Oceanography [1]. WSNs are application-specific and dynamic resource constrained. They are deployed autonomously in a single-sink, multi-sink networks, a single hop, multi-hop networks, flat and hierarchical network topologies [2]. Notable WSN characteristics are large-scale deployment, ability to withstand harshly environmental conditions, ease of adopting, broadcast transmission, cross-layer design. Looking at the negative side, WSNs possess constraints such

Received April 10, 2017; Revised June 14, 2017; Accepted June 30, 2017

as the capacity to cope with node/communication failures, mobility, and heterogeneity of nodes. Nodal constraints include the ability to re-program, maintain with short communication range, constrained CPU power and memory, low bandwidth, limited processing and storage capability [3]. WSNs, when deployed in the field of cognizance, are vulnerable to security attacks. For this reason, providing security for these systems is a challenge issue.

While the research has made huge strides at a tremendous pace to technically secure WSN, no standardized comprehensive hardware crypto–processor exists [4-6]. Cryptographic algorithms developed to secure traditional systems, have inappropriately proved themselves to be slow and power hungry in WSNs.

The rest of this paper is organized as follows; Section 2 presents background. Section 3 describes the algorithmic profiling. Section 4 presents the developed hardware platforms. Section 5 illustrates the simulated experiments and discussions. Section 6 concludes the paper.

2. Background

In the past two decades, wireless communication networks have been one of the fastest-growing research areas. Significant progress is visible in the fields of Ad hoc network (AD-HOC) [7]. AD-HOC is a mature embedded computing wireless communication technology. More recently, Wireless sensor networks (WSN) emerged promising with improved object-to-object interfaces, advanced sensor inputs and affluent network connectivity in both physical and virtual world. WSN adopts cutting-edge micro-sensing MEMS technologies that include sensing inertial motion, environmental factor, bio-signalling, vehicle location to allow more network interactions with the physical environment. µGPS receivers introduced in WSNs collect dynamic information. Although AD-HOC and WSN are being similar in many networking aspects, they possess some major differences [8]. On the contrary, WSN is sophisticated for delivering sensor-related data. This part provides several prefaces of AD-HOC and establishes necessary improvements of WSN from AD-HOC.

Sensor networks have the potential to trigger and provide endless opportunities for the upcoming revolution in the IT sector. Recent advances in low-power VLSI embedded computing, hardware communication, nano-technology with Micro-Electro-Mechanical Systems work as a catalyst to improvise WSN technology. Terrestrial WSN finds applicability in major Environmental, Industrial, Military, Weather, Nuclear and Disaster preventive activities [9]. Underground and underwater WSN helps to observe pollution, survey undersea, prevent disaster and monitor earthquake. WSN finds applications in the industrial sector to track and monitor structures, inventory, Machine, Equipment and Chemical fields. Mobile WSN finds appropriate in habitat, military, healthcare and search and rescue operations. Their versatility draws more interest from the research and industrial community.

Though sensor networks offer limitless applications, however at the same time it poses itself as a formidable challenge. The circuit and system challenges are numerous. The urge to developing low-power microcontrollers for hardware communication, sensors, and actuators based on MEMS, energy-constrained inexpensive devices with redundant forward proper cross-layer design is vital to improving the performance of sensor networks. Data security, system integration and the ability to manage sparse deployment with the optimum quality of service is another major challenge that sensor networks offer to the circuits and systems research community. Resource constraints include limited bandwidth, network connectivity, and coverage area sensing [10].

Information leakage results in a security breach in hostile autonomous environments. Wireless communication is adverse to eavesdropping and packet injection. Security is required for wireless sensor networks to ensure operation safety and data security. Wireless sensor nodes are susceptible to various kinds of attacks [11-14] owing to it being wireless possessing sophisticated design and nodal energy constraints. A balance is essential to establish optimum decentralized security with sparse network resources available.

An exhaustive literature review [15-20] was conducted to analyze cryptographic ciphers employed in WSNs and it revealed that 33.33% of the conducted research used public keys, 10.83% assimilated elliptic curve, 6.66% block ciphers. Cryptographic ciphers mapped to Field Programmable Gate Array and adopted for WSNs has revealed that 21.80% of researchers employed a hash function, 18.75% of them incorporated Advanced Encryption Standard. XTEA (Extended Tiny Encryption Algorithm) is a block cipher notable for its simplicity of description

and implementation [21-23]. Based on the literature survey, the hash function algorithm SHA 512 has been opted, as it is highly secured and supports larger key sizes. As a symmetric key algorithm, AES has been opted, as it is faster in both hardware and software while XTEA is opted as it has the better overall performance of encryption and decryption. Therefore, SHA 512, XTEA and AES are deemed to be chosen for this paper. Profiling them by running them on the computer to address their performance metrics like speed and complexity will allow a better understanding of the computational structure and their bottlenecks. The developed hardware accelerator will positively improve the system performance.

3. Algorithmic Profiling

To speed up any algorithm, the computational structure and performance bottlenecks of the algorithm should be analyzed. Profiling defines itself as the experimental measurement of algorithmic performance. Profiling fall into two broad categories; Instruction counting that measures the number of times a particular direction(s) is executed and clock timing that measures the execution time required for individual blocks of the code. Profiling technique consists of three distinct steps; Augmentation: where a unique code is added at the required position of original program, Execution: where the augmentation code generates timing data, and finally Analysis: where a special program is run on the timing data to produce a performance report.

Instruction counting method is extremely simple to implement and does not introduce errors into the exact code and quantities. However, it fails to identify bottlenecks in the algorithms [24]. On the other hand, clock timing method plants instructions within the program. The function clock measures the time taken by allocated instructions within the program to run. A digital clock “ticks” at discrete intervals. The length of this interval is called the granularity. It helps to identify bottlenecks in the algorithms. Profiling for each algorithm is conducted for a certain number of times and averaged [25].

Algorithmic profiling requires the following hardware and software experimental setup. An integrated development environment (IDE) is software to provide extensive facilities for software development. The majority of researchers have performed profiling in Linux environment due to its open free source nature equipped with extensible plug-ins. Intel Core i7 2600 is ranked 216 out of 2128 CPUs, the first CPU being Intel Xeon E52698V3 according to the Intel Standards using Passmark software. Source code of the algorithms is taken from ARMmbed TLS (PolarSSL) in C++ format. Code cannot be altered since it represents the basic working of the cipher. The cipher is profiled with varying block and key sizes to obtain optimal code with minimal encryption/decryption time. The maximum block size is limited to 16 bytes due to software computational constraint. In C++, u128 (unsigned integer 128 bits) is not supported in the library for printing. They are not even extended integer types in the sense of the C standard [26]. Profiling is done with a combination of key sizes from 8 bytes to 128 bytes and block sizes of 16 and 8 bytes. The ciphers are profiled with 1.6 KB of message size. Each cipher is profiled with various block-key combinations at least fifteen times and results are averaged. Profiling each of the algorithms in such fashion reveals that the profiling with block size 16 bytes takes relatively less time. The reason behind this anomaly considering the above mentioned constraint is that as such algorithm encrypts/decrypts the message data in blocks; usage of larger block size reduces the number of blocks. It reduces the overall time to encrypt or decrypt the whole message.

3.1. Profiling of XTEA

Firstly, XTEA is profiled with a block size of 8-16 bytes with key sizes of 8 -128 bytes respectively. On the completion of profiling, results obtained are plotted in Figure 1. It can be analysed that as they mostly contain XOR operations and can be parallelized by pipelining or provided with a good architecture as in Figure 2. If bottlenecks are accelerated, the whole encryption can be speeded up multi-fold.

3.2. Profiling of SHA

SHA-512 is profiled with block sizes of 8-16 bytes with key size of 8-128 bytes respectively. The obtained results are shown in Figure 3. It is clear that memory transactions

are killing the processing time. Provision of dynamic cache and data reuse speeds up overall processing time multifold, the proper architecture is given in Figure 4.

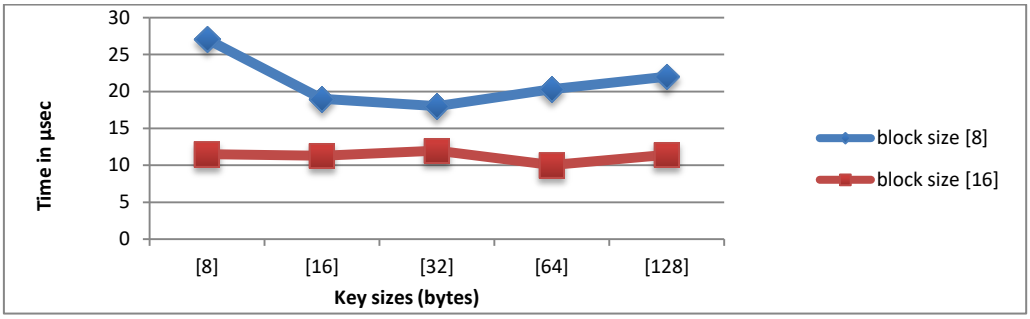

Figure 1. Total time is taken by XTEA to encrypt same message block with varying block and Key Sizes

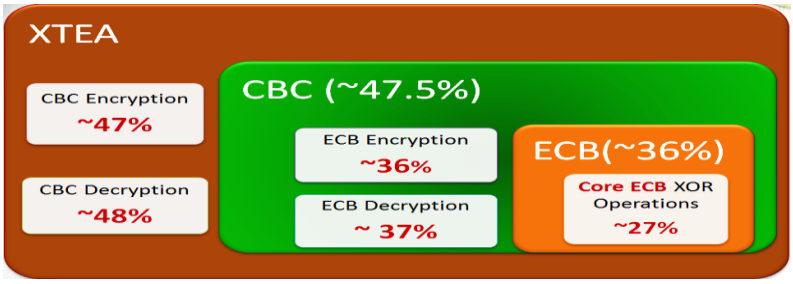

Figure 2. Profiling of XTEA algorithm

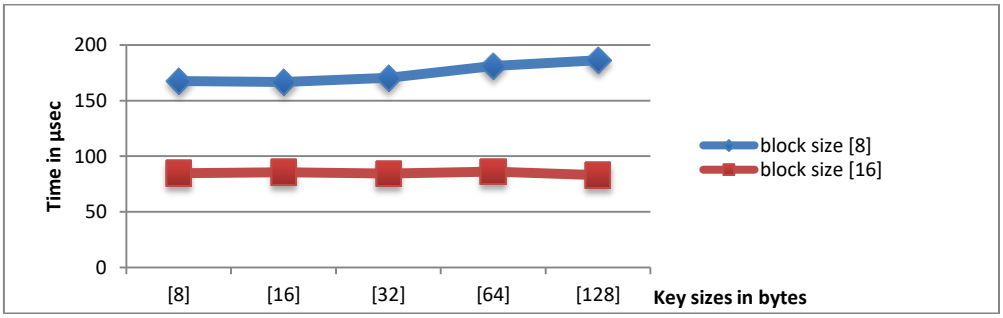

Figure 3. Total time taken by SHA to encrypt the message block with varying key size and block sizes

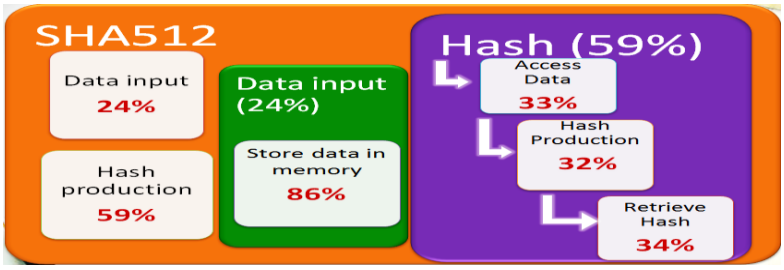

Figure 4. Profiling of SHA512 algorithm

3.3. Profiling of AES

AES is profiled with block size of 8-16 bytes with key sizes of 8-128 bytes respectively. On the completion of profiling, the achieved results are given in Figure 5. Again, the memory transactions took much processing time. Provision of dynamic cache and data reuse speeds up overall processing time multifold, again, the suggested architecture is given in Figure 6.

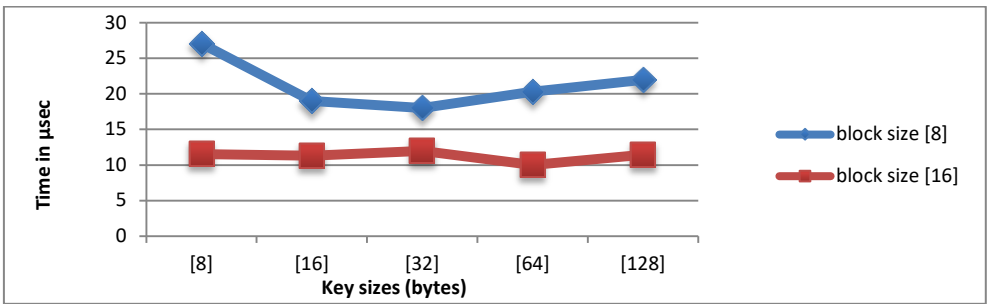

Figure 5. the total time taken by AES to encrypt the same message block with varying key and block sizes

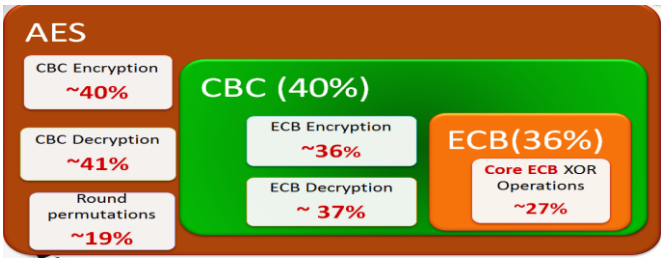

Figure 6. Profiling of AES algorithm

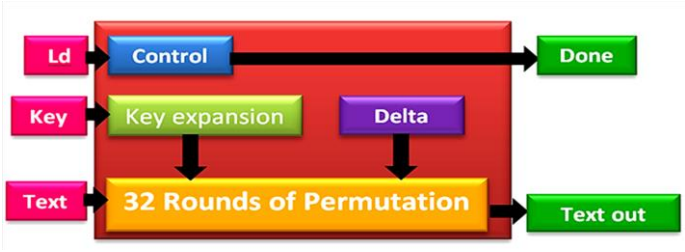

Figure 7. Block diagram of the XTEA cipher

4. The Developed Hardware Architecture of Accelerator

The basic hardware implementation of core algorithms have been considered and altered according to sensor node requirements. However, its independent implementation on FPGA with the feeder, control and output section has been greatly simplified to support smaller and faster nodal implementation in WSNs. Basic block diagrams (hardware architectures) depicting the working of ciphers are developed. Fifure 7 explains the block diagram of XTEA encipher module. Here key and the plain text are accepted in by asserting 'Ld' pin high. On completing the encryption sequence 'Done' signal is asserted for one clock cycle. On the same lines, Figure 8 explains the block diagram of XTEA decipher module the key is loaded with the assertion of 'kld'. 'kdone' marks the completion of key expansion sequence. Cipher text is loaded with the assertion of 'Ld' signal. The assertion of 'done' signal marks the end of deciphering process. XTEA is simulated in ModelSim with a test-bench.

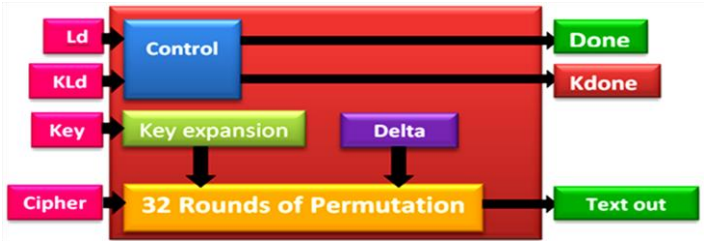

Figure 8. Block Diagram of XTEA Decryption Module

The second developed and analyzed algorithm is SHA. Figure 9 explains the basic architecture of SHA core module. Text input is fed in with the control signal Cmd_w_i (command write) enabled. The core module calculates message digest, and it is sent off with the Cmd_o (command output) is active high.

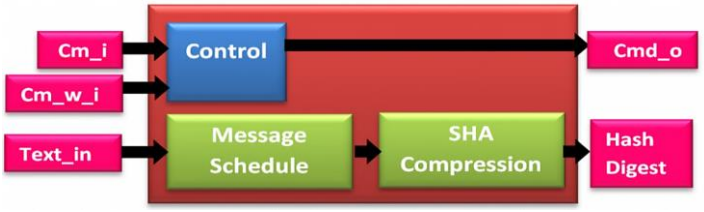

Figure 9. Hardware Structure of SHA Algorithm

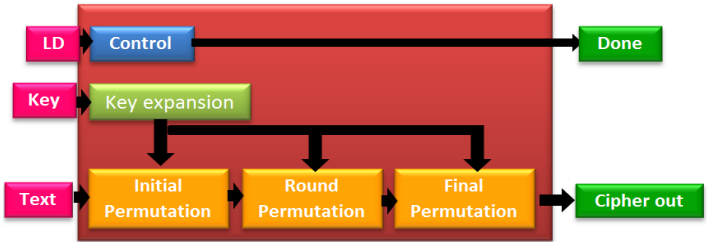

Figure 10. Architecture of AES Encipher Module

Figure 10 explains the block diagram of AES encipher module. Here, the key and the plain text are accepted by asserting 'Ld' pin high. On completing the encryption sequence, 'done' signal is asserted for one clock cycle. The instances of decryption module are inv_sbox, key_expand module, and Round counter. The inverse ciphering key is loaded first as it uses the last expanded key first, and the first expanded key last as in Figure 11.

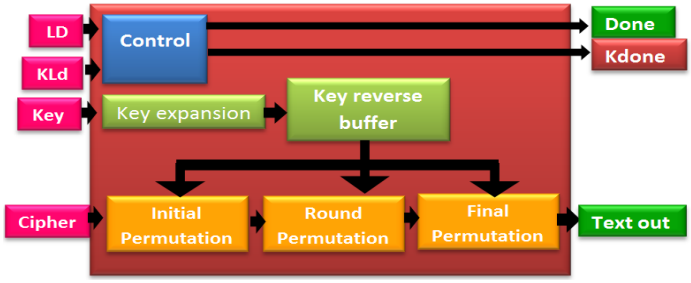

Figure 11. Architecture of AES Decipher Module

5. Experiments and Discussions

XTEA architecture is simulated in ModelSim with a testbench as in Figure 12. It is observed that from the time plain text is taken in, a couple of cycles later, the authenticated cipher text is out. This indicates the correct functioning of the hardware. The developed hardware architecture of XTEA algorithm is synthesized and analyzed.

SHA512 architecture is simulated in ModelSim with a testbench as in Figure 13. It is observed that from the time plain text is taken in, again after a couple of cycles, the authenticated digest is out. This indicates the correct functioning of the hardware. The verified hardware is synthesized and analyzed in Quartus. SHA512 is simulated in Figure 14 indicates the instance of data input, hash output and the digest time.

AES architecture is simulated in ModelSim with a testbench as in Figure 15. It is observed that from the time plain text is taken in, after a couple of cycles, the authenticated cipher text is out during the cipher time and vice versa. This indicates the correct functioning of the hardware. The developed hardware architecture of AES algorithm is synthesized and analyzed in detail. Although basic code is obtained regarding the cipher and inverse cipher module, the paper contribution is the development of the interfaces between the cipher and inverse cipher modules with control signal for perfect synchronization as in Figure 16. The cipher from the encipher module enters the decipher module to give plain text.

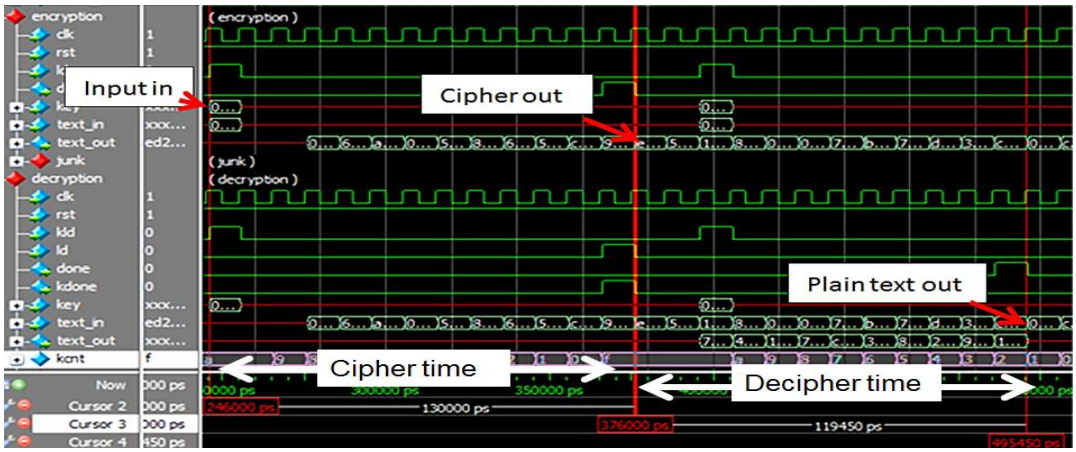

Figure 12. Simulation Result of XTEA in ModelSim

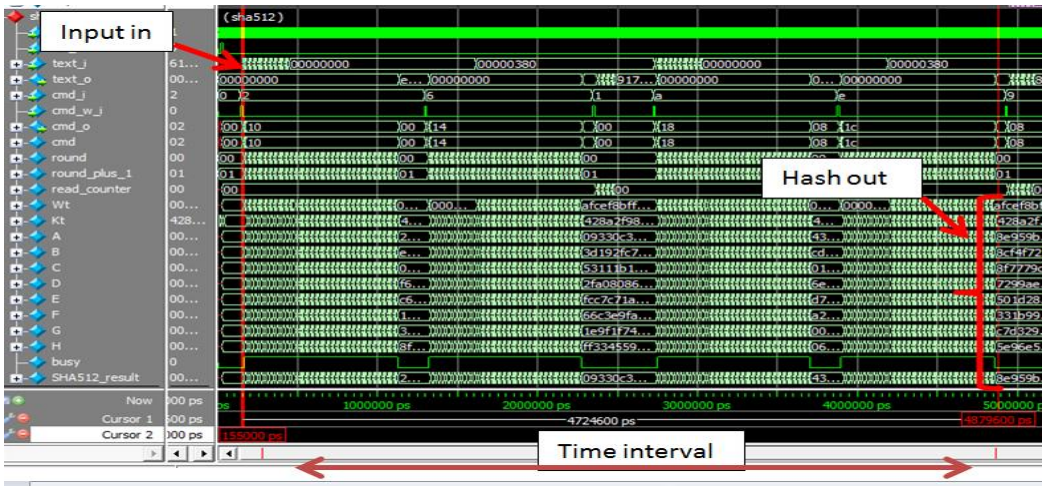

Figure 13. Simulation of SHA-512 in ModelSim

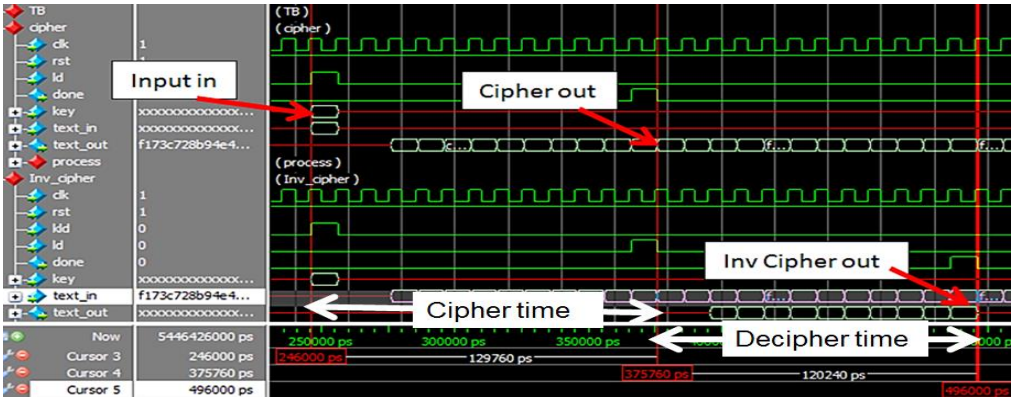

Figure 14 Simulation of AES Testbench in ModelSim.

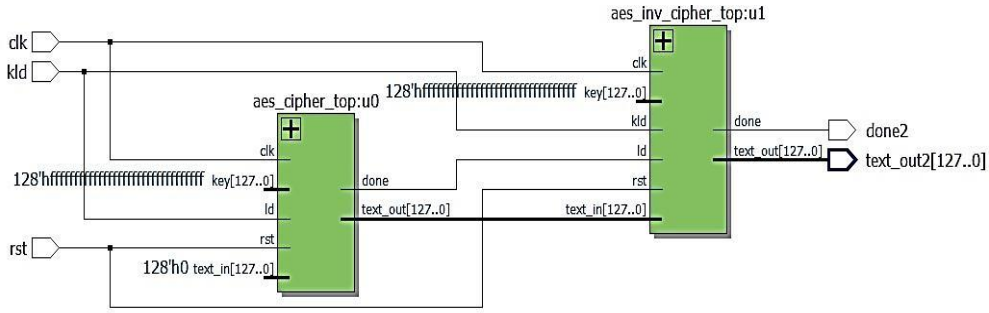

Figure 15. RTL View of AES Architecture

5.1. The SHA Hardware Prototyping on FPGA

The complete modules are implemented on TerasIC DE0 Nano Cyclone IV board. Figure 16 shows the entire SHA module. The feeder module to feed in the input with the control signal cmd_write_enable has been developed. The core module calculates message digest. It is displayed via the display module on seven segment display with the cmd_out enabled high.

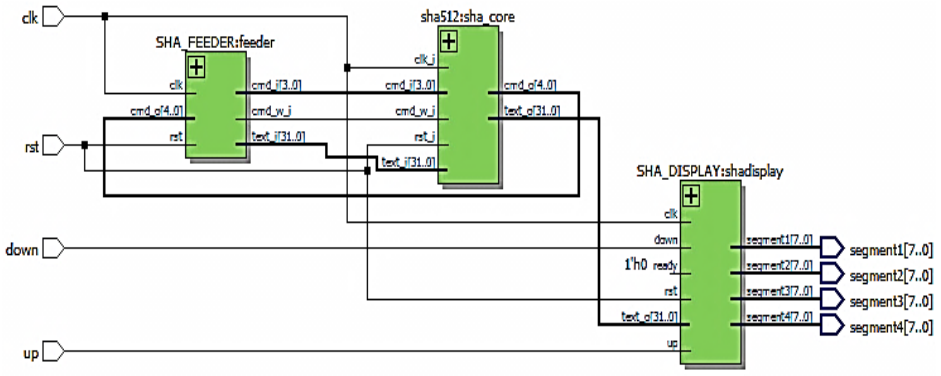

Figure 16. The RTL of complete SHA module

5.2. The AES/ XTEA Hardware Prototyping on FPGA

An independent module is developed and implemented in Altera DE0 FPGA board.

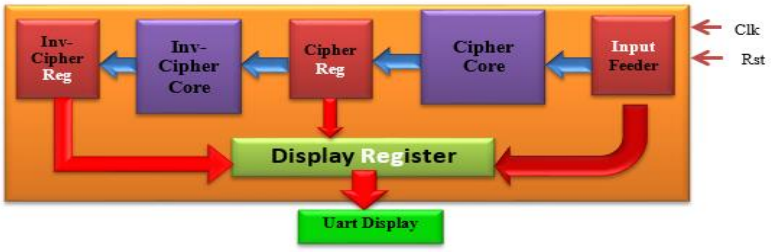

Figure 17. The RTL of complete AES/XTEA module

Figure 17 displays the basic architecure of XTEA/AES module when prototyped on FPGA after prior simulation and synthesis in order to verify the working in an virtual environment, its results and the structure. Again, the feeder module with sychronized interfaces and control signals have been developed. UART module is deployed to transmit a message from a computer to DE0 Nano board. As AES and XTEA have similar characteristics, it shares the same unique architecture as that of AES.

The achieved speed up of encryption and decryption time of data with ciphers in FPGA normalised to CPU platform. XTEA is speedened up by 16.6x, AES is speedened by 16x and SHA is speedened up by 22x as shown in Figure 18.

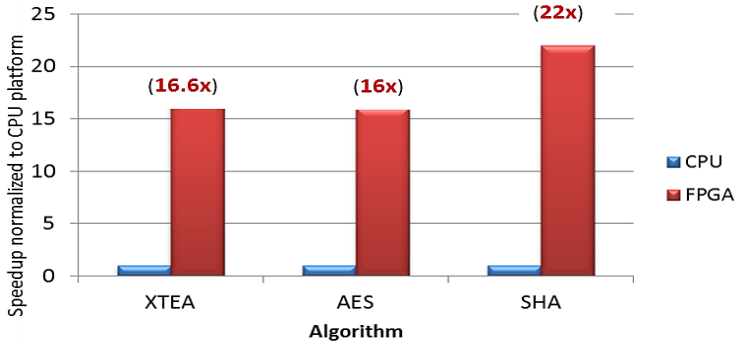

Figure 18 Speedup normalized to CPU platform

The normalized power consumption when the hardware accelerating on FPGA w.r.t. the CPU Platforms is shown in Figure 19.

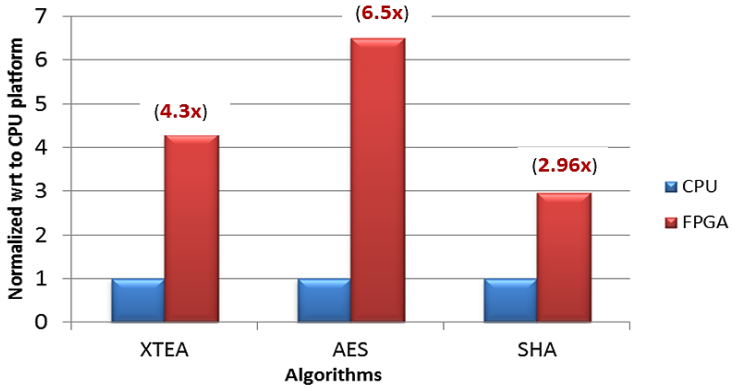

Figure 19. Power Efficiency when Securing Data Via Software Profiling and When Mapped to FPGA

There is a reduction in power consumption by 4.3x when hardware is accelerated with XTEA, 6.5x with AES, and 2.96x with SHA is achieved. Meanwhile, there was an increment in the throughput by 16x when hardware is accelerated with XTEA, 12.8x with AES, and 25x with SHA. It is shown in Figure 20.

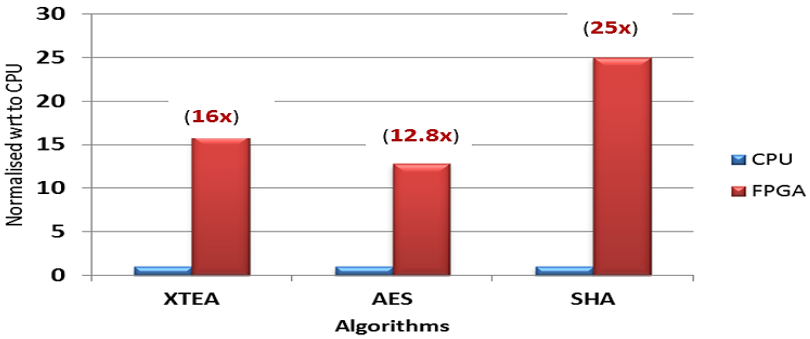

Figure 20. Throughput in Ciphers When in Software Profiling and When Mapped to FPGA

The results of XTEA, AES and SHA algorithms compared with most related work [27-35]. The proposed algorithm being iterative produces optimum throughput with optimum LUTs occupancy at only 50 MHz in all the three comparative tabulated results and shown in Table 1, Table 2 and Table 3 display.

Table 1. Comparison of Proposed Design AES with Related Algorithms

Design	Target Device	Frequency (MHz)	Throughput (MB/S)	LUTS
H/W (BRAMS)	Cyclone III	34	27	130
Folded arch.	Spartan II	60	20.75	222
Pipelined Memoryless	Virtex-E	129	16.5	11719
Pipelining	Virtex XC	120	1180	6279
Pipelining	Xilinx ISE 14.4	1048	30000	14000
Pipeline CTR	Xilinx	600	73737	576
Fully Pipelining	VirtexII-Pro	567	21500	5177
Proposed Design (Iterative) AES	Cyclone IV	50	231.8	5000

Table 2. Comparison of proposed design XTEA with related algorithms

Design	Target Device	Frequency (MHz)	Throughput (MB/S)	LUTS
Code	Cyclone III	60	100	6590
Multi-mode	Xc5vlx85-3	200	18286	17310
Hummingbird	Spartan III	65	128	1024
Proposed Design XTEA based	Cyclone IV	50	245	1678

Table 3. Comparison of Proposed Design SHA512 Algorithm with Related Algorithms

Design	Target Device	Frequency (MHz)	Throughput (MB/S)	LUTS
Permutations	Stratix EP1S40	933	653	6590
Multi-mode	Virtex V	75	467	19072
Loop unrolling-4x	Virtex 4	40	1364	11614
Loop unrolling-4x	Virtex-II	576	1616	10328
Multi-mode	Virtex 4	65	800	23000
Proposed design SHA512	Cyclone IV	50	250	5660

5.3. Heterogeneous Placement in Sensor Node

The heterogeneous architecture of the sensor node. It consists of an ARM generic processor along with various interfaces and ports to facilitate interaction with the external environment are depicted in Figure 21. Deployment of FPGA hardware accelerators as the companion processors helps in delegating the work or offloading the task of encryption and decryption while ARM controls the overall system. Thus, the overall work is distributed and the total execution time is reduced.

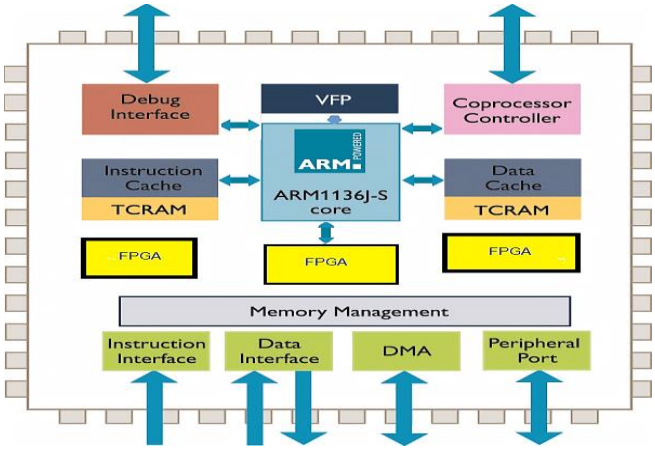

Figure 21. Proposed placement of FPGA in the sensor nodes employing ARM Processor

The work in this paper can be further extended by developing a programmable customized hardware, which hosts multiple security ciphers. They can dynamically choose the algorithm according to the desired level of security taking into considerations the tradeoffs in area that are minor. Here, as XTEA is part of AES, it can be built-in as in the Figure 22.

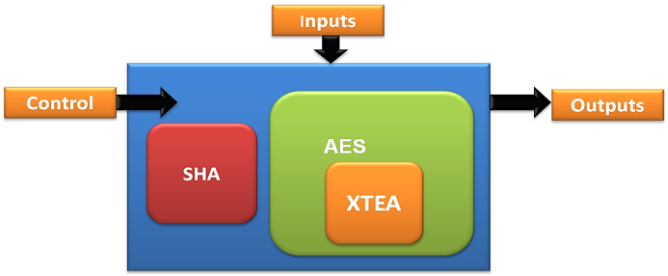

Figure 22. Architecture of a customized hardware

6. Conclusions

In this paper, profiling SHA, XTEA and AES algorithms allows a better understanding of the computational structure and bottlenecks. A suitable hardware accelerator is developed on which HDL coded algorithms are mapped. Then, based on the profiling results, the computational structure of the algorithms was analyzed to identify the performance bottlenecks. The accelerators were implemented using the iterative approach and were validated both in simulation and on platform FPGA. Based on the achieved results, XTEA is speeded up by 16.6 x, AES by 16x and SHA by 22x. Power consumption is reduced in hardware accelerator by 4.3x using XTEA, 6.5x using AES and 2.96x using SHA. Throughput is increased in hardware accelerator by 16x using XTEA, 12.8x using AES and 25x using SHA. The hardware accelerator which proved promising in accelerating the encryption and decryption operation for data security can be deployed in many other such operations.

References

[1] A. Hodjat and I. Verbauwhede, *A 21.54 Gbits/s fully pipelined AES processor on FPGA in Field-Programmable Custom Computing Machines*, 2004. FCCM 2004. 12th Annual IEEE Symposium on, 2004, pp. 308-309.

[2] Y. W. Law and J. Havinga, *How to secure a wireless sensor network*, in Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on, 2005, pp. 89-95.

[3] K. Maraiya, K. Kant, and N. Gupta, Application based study on wireless sensor network, *International Journal of Computer Applications (0975–8887)* Volume, vol. 21, pp. 9-15, 2011.

[4] L. Pathuri, A. Al Maashri, A. Ahmad, M. Ould Khaoua, M.Awadalla, *Securing Wireless Sensor Networks using Customized Hardware Crypto Engine*. National Symposium on Innovations in Information Technology, College of Applied Sciences - Sohar, Ministry of Higher Education, Sultanate of Oman; 04/2015.

[5] Ahmed Al Maashri, Lavanya Pathuri, Medhat Awadalla , Afaq Ahmed, Mohamed Ould-Khaoua Optimized Hardware Crypto Engines for XTEA and SHA-512 for Wireless Sensor Nodes *Indian Journal of Science and Technology 9(29)*, August 2016.

[6] B. Stelte, *Toward development of high secure sensor network nodes using an FPGA-based architecture*, in Proceedings of the 6th International Wireless Communications and Mobile Computing Conference, 2010, pp. 539-543.

[7] A. Nadeem and M. Y. Javed, *A performance comparison of data encryption algorithms," in Information and communication technologies*, 2005. ICICT 2005. First international conference on, 2005, pp. 84-89.

[8] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, *Analyzing and modeling encryption overhead for sensor network nodes*, in Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, 2003, pp. 151-159.

[9] M. Awadalla, Processor Speed Control for Power reduction of Real-Time Systems, *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 5, No. 4, pp. 701-713, August 2015.

[10] J. Rehana, *Security of wireless sensor network*, Helsinki University of Technology, Helsinki, Technical Report TKK-CSE-B5, 2009.

[11] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, "From wireless sensor networks towards cyber physical systems," *Pervasive and Mobile Computing*, vol. 7, pp. 397-413, 2011.

[12] K. Römer, O. Kasten, and F. Mattern, *Middleware challenges for wireless sensor networks*, ACM SIGMOBILE Mobile Computing and Communications Review, vol. 6, pp. 59-61, 2002.

[13] J. A. Stankovic, *Research challenges for wireless sensor networks,* ACM SIGBED Review, vol. 1, pp. 9-12, 2004.

[14] S. Mohammadi and H. Jadidoleslamy, *A comparison of link layer attacks on wireless sensor networks,* arXiv preprint arXiv:1103.5589, 2011.

[15] S. Mohammadi and H. Jadidoleslamy, A comparison of physical attacks on wireless sensor networks, *International Journal of Peer to Peer Networks*, vol. 2, pp. 24-42, 2011.

16] G. Xing, T. Wang, Z. Xie, and W. Jia, "Rendezvous planning in wireless sensor networks with mobile elements," *Mobile Computing, IEEE Transactions on, vol. 7*, pp. 1430-1443, 2008.

[17] Z. Gong, S. Nikova, and Y. W. Law, KLEIN: *A new family of lightweight block ciphers*: Springer, 2012.

[18] H. Ahmadi, N. Pham, R. Ganti, T. Abdelzaher, S. Nath, and J. Han, *Privacy-aware regression modeling of participatory sensing data,* in Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, 2010, pp. 99-112.

[19] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury, *Did you see bob?: human localization using mobile phones,* in Proceedings of the sixteenth annual international conference on Mobile computing and networking, 2010, pp. 149-160.

[20] C. Castelluccia, A. C. Chan, E. Mykletun, and G. Tsudik, Efficient and provably secure aggregation of encrypted data in wireless sensor networks, *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, p. 20, 2009.

[21] A. De La Piedra, A. Braeken, and A. Touhafi, "Sensor systems based on FPGAs and their applications: a survey," *Sensors*, vol. 12, pp. 12235-12264, 2012.

[22] U. Sadhvi Potluri, A. Madanayake, R. J. Cintra, F. M. Bayer, S. Kulasekera, and A. Edirisuriya, Improved 8-point approximate DCT for image and video compression requiring only 14 additions, *Circuits and Systems I: Regular Papers, IEEE Transactions* on, vol. 61, pp. 1727-1740, 2014.

[23] D. M. Pham and S. M. Aziz, Object extraction scheme and protocol for energy efficient image communication over Wireless Sensor Networks, *Computer Networks*, vol. 57, pp. 2949-2960, 2013.

[24] F. Khelil, M. Hamdi, S. Guilley, J. L. Danger, and N. Selmane, Fault analysis attack on an FPGA AES implementation, in *New Technologies, Mobility and Security, 2008*. NTMS'08., 2008, pp. 1-5.

[25] G. Murphy, A. Keeshan, R. Agarwal, and E. Popovici, *Hardware-software implementation of public-key cryptography for wireless sensor networks*. 2006.

[26] How to print uint128 in c++. Available: http://stackoverflow.com/questions/11656241/how-to-print-uint128-t-number-using-gcc

[27] R. Njuguna, *A Survey of FPGA Benchmarks*, Technical report, CSE Department, Washington University in St. Louis2008.
[28] R. Andraka, *A survey of CORDIC algorithms for FPGA based computers*, in Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, 1998, pp. 191-200.
[29] L. Shannon and P. Chow, *Using reconfigurability to achieve real-time profiling for hardware/software codesign,* in Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays, 2004, pp. 190-199.
[30] R. Dimond, O. Mencer, and W. Luk, *CUSTARD-a customisable threaded FPGA soft processor and tools,* in Field Programmable Logic and Applications, 2005. International Conference on, 2005, pp. 1-6.
[31] R. Lysecky and F. Vahid, *A study of the speedups and competitiveness of FPGA soft processor cores using dynamic hardware/software partitioning,* in Design, Automation and Test in Europe, 2005. Proceedings, 2005, pp. 18-23.
[32] J.-P. Kaps, *Chai-tea, cryptographic hardware implementations of XTEA,* in Progress in Cryptology-INDOCRYPT 2008, ed: Springer, 2008, pp. 363-375.
[J3]. M. Awadalla and H. Konsowa, "Performance Enhancement of Multicore Architecture", *International journal of Electrical and Computer Engineering, (IJECE)*, Vol. 5, No. 4, pp. 669-684, August 2015.
[34] J. M. Rabaey, J. Ammer, T. Karalar, S. Li, B. Otis, M. Sheets, et al., *PicoRadios for wireless sensor networks: the next challenge in ultra-low power design,* in Solid-State Circuits Conference, 2002. Digest of Technical Papers. ISSCC. 2002 IEEE International, 2002, pp. 200-201.
[35] S. Ghaznavi, C. Gebotys, and R. Elbaz, *Efficient technique for the FPGA implementation of the aes mixcolumns transformation,* in Reconfigurable Computing and FPGAs, 2009. ReConFig'09. International Conference on, 2009, pp. 219-224.