Low power and area SHA-256 hardware accelerator on Virtex-7 FPGA

Ali H. Gad[1,*], Seif Eldeen E. Abdalazeem[1,*], Omar A. Abdelmegid[1,*], and Hassan Mostafa[1,2,†]

[1]*University of Science and Technology, Nanotechnology and Nanoelectronics Engineering Program, Zewail City of Science and Technology, October Gardens, 6th of October, Giza 12578, Egypt*

[2]*Electronics and Communication Department, Faculty of Engineering, Cairo University, Giza 12613, Egypt*

* These authors have contributed equally to this work.

† Author to whom correspondence should be addressed. Email: hmostafa@uwaterloo.ca

{s-ali.h.gad@zewailcity.edu.eg, s-seif.emad@zewailcity.edu.eg, s-omar.ashraf@zewailcity.edu.eg}

***Abstract*— Lately, there have been many technological developments in communication especially in online transactions, so the demand for highly secure systems and cryptographic algorithms has increased. Cryptographic hash functions are used to protect and authenticate information and transactions. SHA-256 (Secure Hash Algorithm-256) is a one-way hash function characterized by being highly secure and fast while having a high collision resistance. This paper presents a new hardware architecture of SHA-256 with low power consumption and area based on a sequential computation of the message scheduler and the working variables of SHA-256. The hardware was described in HDL and implemented on Virtex-7 FPGA which offers high efficiency and speed. Different optimization techniques were used to further reduce the power and area such as gated clock conversion, arithmetic resource sharing, and structural modeling of small building blocks. The proposed design ran with a maximum frequency of 83.33 MHz. The implementation reports indicated a dynamic power consumption of 13 mW and area utilization of 275 slices while maintaining a good throughput of 0.637 Gbits/s and a relatively high efficiency of 2.32 Mbits/s per slice. Such design with low power and area can be used to hash messages on a portable device opening a whole new area for different applications and opportunities.**

Keywords—SHA-256, low power, low area, hardware acceleration, FPGA, hash, blockchain.

I. INTRODUCTION

Due to the rapid development and huge growth of the digital world of technology and the increase in internet speed, information security has become a necessity. these algorithms are used widely in different applications banking, shopping, and different money related activities and recently they are utilized in blockchain technologies and internet of things applications due to the extensive research and projects focused on these two fields. especially hash functions, have become an essential part of securing data communication [1], [2].

Hash functions are used to protect data, verify a digital signature, or authenticate a wide range of online processes and transactions, using complex logical operations. The hash functions have unique properties making them highly secure. They are one-way functions, meaning for a given hash output h, it is computationally infeasible to retrieve the input message $x, where\ H(x) = h$. They also have a very high collision resistance, that it is computationally infeasible to have two inputs with the same hashed output $(x \neq y\ and\ H(x) \neq H(y))$ [3]. Besides, hash functions are highly sensitive. A slight change in the input message will completely change the output hash.

SHA-256 takes any message up to 2^{64} and outputs a hash with a fixed length of 256 bits with 128 bits of security against collision attacks. Other hash functions provide different security and output lengths such as SHA-224, SHA-384, and SHA-512. Although SHA-512 provides better security, it consumes a greater memory and takes a longer computational time, making it unsuitable for some applications. For SHA-256 high security, high collision resistance, and its computational time, it is used in various important applications such as blockchain. The blockchain utilizes SHA-256 to authenticate and verify the transactions included in a block before joining the blockchain. It also prevents any alteration in the contents of the block after being introduced to the chain, as any trivial change in the data will significantly alter the hash of the block and the following blocks connected to it, which will be easily detected.

An FPGA (field-programmable gate array) will be used for implementation to exploit the hardware acceleration. It is suitable to implement cryptographic algorithms and to perform the tasks more efficiently allowing design optimization. Encryption using FPGA is nearly 20 times faster than the dual-core processor while reducing the CPU usage by 85%. Moreover, it offers shorter design time, more flexibility, and lower costs [4].

In this paper, investigations are made to implement a proposed design for SHA-256 along with optimization techniques to achieve low power and area on Xilinx Virtex-7 FPGA (*xc7v2000t fhg1761-2*), using Vivado Design Suite, so that the design can be of future use for hardware-accelerated blockchains on FPGAs. Most papers focused only on increasing the SHA-256 throughput, while this paper's main target is to reduce the power consumed and the area utilized by the architecture while maintaining good throughput.

The paper is organized as follows. Section II describes the SHA-256 algorithm and how it works. Section III discusses the implementation of the proposed architecture and the optimization methods used. Section IV presents our results and comparisons with other papers. Finally, future recommendations and the conclusion are given in section V.

II. SHA-256 ALGORITHM

The SHA-256 algorithm aims to encrypt an intermediate hash value using the message as a key. To hash a certain message, it has to go through three distinct operations, padding, message scheduler, and compression function [3].

A. Padding

In padding, the message has to be padded such as it becomes a multiple of 512, parsed into 512-bit blocks, and an initial hash value H^0 must be set. To start padding, for a message of length ℓ, the bit "1" is added to its end, followed by k zero-bits where k is the smallest non-negative solution to $\ell + 1 + k \equiv 448 \, mod \, 512$, then the message length ℓ is expressed in binary in the last 64-bits.

After obtaining a padded message as a multiple of 512, it is parsed into N 512-bit blocks $M[N:1]$ so that each block goes through the message scheduler and the compression function. Before starting to compute the hash, an initial hash value H^0 is set to be the first 32-bits of the fractional parts of the square roots of the first eight primes.

B. Message scheduler

In the message scheduler, a 32-bit message words W_t $(t = 0, \ldots, 63)$, generated N times for each of the 512 padded blocks, given by the following equation, are initialized. W_t equals $M_t^{(i)}$ for the first 16 words, and equals $\sigma_0(W_{t-15}) + \sigma_1(W_{t-2}) + W_{t-7} + W_{t-16}$ for the remaining 48 words.

C. Compression Function

The hash computation proceeds as follows:

$For \; i = 1 \; to \; N$

{

The working variables, registers $A, B, \ldots, H$ are initialized with the $(i-1)^{st}$ intermediate hash value (the initial hash value when $i = 1$). Apply the SHA-256 compression function to update $A, B, \ldots, H$

For $j = 0$ to 63

{

Compute W_j, $Ch(E,F,G)$, $Maj\,(A,B,C)$, $\sum_0(A)$, $and \; \sum_1(E)$ (their definitions are below)

$$T_1 = K_j + W_j + H + \sum_1(E) + Ch(E,F,G)$$

$$T_2 = Maj\,(A,B,C) + \sum_0(A)$$

$$H = G$$

$$G = F$$

$$F = E$$

$$E = D + T_1$$

$$D = C$$

$$C = B$$

$$B = A$$

$$A = T_1 + T_2$$

}

Compute the i^{th} intermediate hash value $H^{(i)}$ as the sum of the previous hash and the registers $A, B, \ldots, H$

}. After the N iterations, the hash of the message is $H^{(N)} = (H_1^{(N)}, H_2^{(N)}, \ldots, H_8^{(N)})$.

Where the definitions of the logical functions are:

$$\sigma_0(x) = S^{18}(x) \wedge S^7(x) \wedge R^3(x)$$
$$\sigma_1(x) = S^{19}(x) \wedge S^{17}(x) \wedge R^{10}(x)$$
$$\sum_0(x) = S^{22}(x) \wedge S^{13}(x) \wedge S^2(x)$$
$$\sum_1(x) = S^{25}(x) \wedge S^{11}(x) \wedge S^6(x)$$
$$Ch(x,y,z) = (\neg x \,\&\, z) \wedge (x \,\&\, y)$$
$$Maj(x,y,z) = (x \,\&\, y) \wedge (y \,\&\, z) \wedge (x \,\&\, z)$$
$$R^n \; is \; right \; shift \; by \; n \; bits$$
$$S^n \; is \; right \; rotation \; by \; n \; bits$$

The sequence of constant words $K_0, \ldots, K_{63}$, are the first thirty-two bits of the fractional parts of the cube roots of the first sixty-four primes.

III. IMPLEMENTATION

To design a low power and low area SHA-256 architecture, the proposed design was implemented along with various optimization techniques to further decrease consumption.

The proposed design is based on partitioning modules and hierarchies to small building blocks to further optimize them and keep the area usage to a minimum, as shown in Fig.1.

A. Padding

In our design, in the *padding block*, on the rising edge of the first clock, the input message is padded and parsed into a 512-bit block, it then goes through the *bit selection block* to be sent to the *message scheduler* during the course of the next 16 clock cycles in the form of 32-bit words.

B. Message Scheduler

The main role of the *message scheduler block* is to determine W_t in order to compute the working variables and the intermediate hash values. In normal designs, the 64 message words are computed, then are used to determine the 64 working variables. On the contrary, in our proposed design, only one message word is computed and used to determine its corresponding working variables, all in one cycle. This in turn had a huge effect on throughput, area, and power. Instead of computing all working variables in 128 clock cycles, 64 for message words, and 64 for the actual working variables, the whole operation now only takes 64 cycles, nearly doubling the throughput. It also decreased the area used in the W_t register and its switching activity, resulting in huge power reduction.

C. Compression Function

The *compression block* contains combinatorial functions used to calculate the variables, and a register memory storing the values of K constants, which reduces the access time to the K_{const}.

The block receives one W_t per cycle, along with values from K_{const}, *Ch, Maj, sum1*, and *sum2*, then it computes the 32-bit temporary variables which in turn update the working variables at the rising edge of each clock. At the end of the 64^{th} clock cycle, the working variables are added to the intermediate hash values to output the final hash.

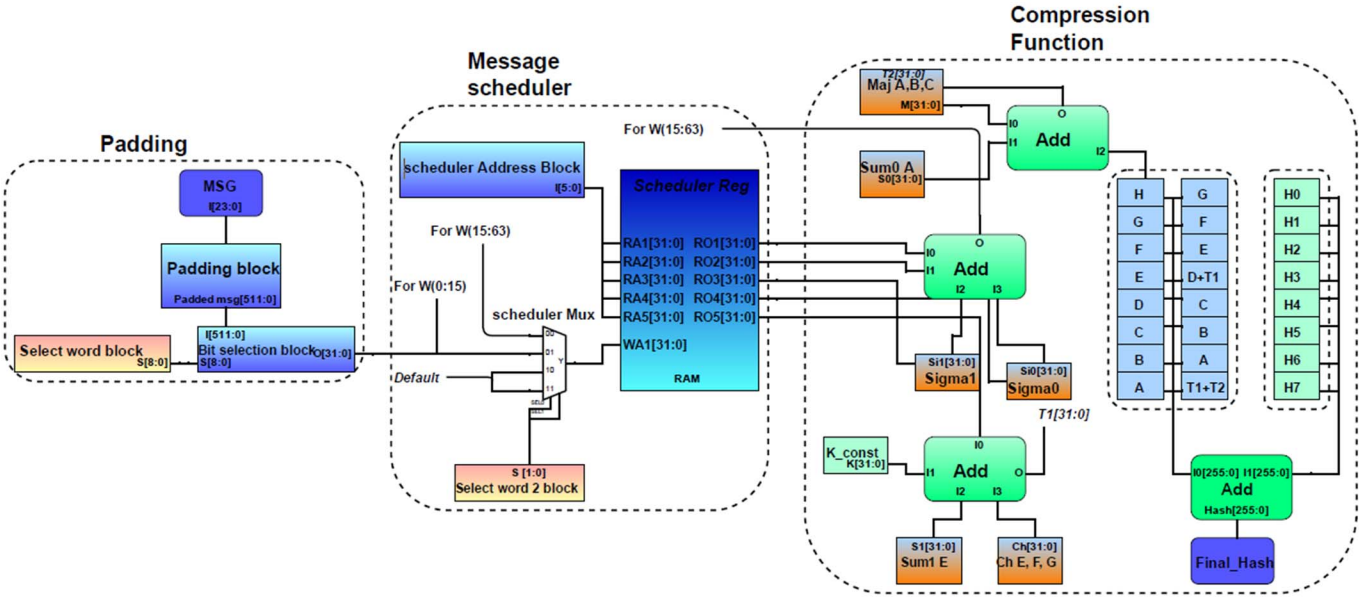

Fig.1. Simplified Schematic Architecture of SHA-256

We used component inference in constructing our modules. This allowed our design to reach a low power consumption, to be optimized way further by the synthesis tool, and to be portable onto various architectures without modifications.

To further reduce and optimize the power consumption, we used resource sharing for the arithmetic components, due to the large number of operations performed, which contributed to area and power reduction. Moreover, by decreasing the logical resources and introducing registers in-between, the critical path also decreased leading to an increase in the maximum allowed frequency and the overall efficiency. Further, power gating technique was implemented to achieve a lower power consumption, however the area usage increased because of the added logic gates and a clock skew was introduced. As a result, a gated clock conversion from logic to flops was introduced to reduce the area and keep the power as it is. Also, a clock-tree synthesis is added to compensate for the skew through balancing clock buffers and routing.

The results of the mentioned optimization techniques differed when implemented on different FPGAs.

IV. The Results

The proposed SHA-256 design was described using SystemVerilog. The design was synthesized and implemented on Xilinx Virtex-7 FPGA (*xc7v2000t fhg1761-2*), using Vivado Design Suite.

The proposed SHA-256 architecture processes a 512-bit block within 67 clock cycles. The message padding takes 1 cycle, hash computation takes 64 cycles, and outputting the final hash takes 2 cycles. Fig.2 shows the post-implementation timing simulation results of hashing the messages abc (616263) and aaa (616161).

The implemented hardware achieved a maximum frequency of 83.33 MHz with a static power of 148 mV and a dynamic power of 13 mV using 275 slices, 763 LUTs, and 568 ffs. A throughput of 0.637 Gbits/s and an efficiency of 2.32 Mbits/s per slice were calculated using equations (1), (2) as shown below [5]

$$Throughput = \frac{(Block\ Size \times\ Frequency)}{Cycles\ per\ block} = 0.637\ Gbits/second \quad (1)$$

$$Efficiency = \frac{Throughput}{Slices} = 2.32\ Mbits/(second.slice) \quad (2)$$

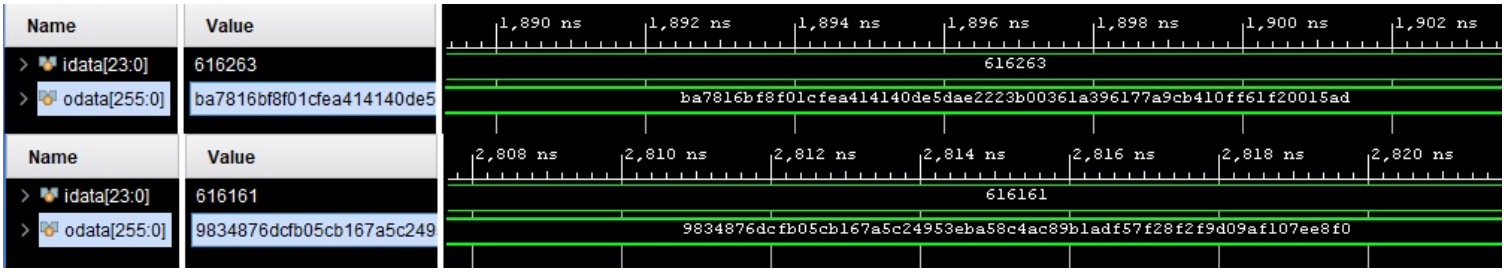

Fig.2. The post-implementation timing simulation results of hashing the messages 616263 and 616161

Table I. Detailed Utilization Report

Resources	Utilization	Available	Percentage of Utilization
LUT	763	1221600	0.062%
FF	568	2443200	0.023%
Slices	275	305400	0.09%

For accurate power estimation, a switching activity interchange format (SAIF) file was generated during a post-implementation timing simulation and used to generate a power report under maximum condition. The average dynamic power consumed for hashing 512-bit block was 13 mW. The clocks, signals, logic, and I/O consumed 3 mW, 5 mW, 4 mW, and 1 mW respectively. Table I summarizes the proposed design results.

Throughout literature, there were inconsistencies in reporting the results of proposed designs. Papers estimated their area utilization using slices, others used LUTs and Flip Flops. Thus, the comparison report includes different factors and parameters to have a comparison, as fair as possible with other papers. Power consumption was rarely mentioned in literature as most papers focused on optimizing speed, represented in achieving high frequency and throughput. So some of the previous work we compare our design with will be old, as we could not find more recent publications.

Only [6] and [7] mentioned the estimated power. [6] compared the power consumption at a fixed throughput of 0.2 Gbits/s between ASIC and FPGA, thus their FPGA results were chosen. While [7] tried applying different techniques and estimating the power consumption of each one, their minimum power was chosen to compare with. Though, each of the two papers reported higher power consumption than our proposed design.

In [8], their main target was decreasing power by decreasing the area. They hashed two 512-bits blocks, but no power results were mentioned, only their area utilization which is larger than our proposed design. [4] implemented and compared an FPGA encrypt engine with SHA-256 and AES IP. Their SHA-256 was implemented on different FPGA devices, thus the Virtex platform was chosen to compare with. Though having a better throughput, they had a larger area and a lower efficiency than our proposed design. [9] implemented their SHA-256 on Virtex-5 operating at 179.08 MHz using 2796 slices. They did not mention any power consumption results.

Table II summarizes the comparison between our work and other existing work. Most referenced papers reported higher operating frequencies than our design. That is because most papers targeted throughput optimization and speed, whereas we focused on power optimization which results in lower speed, which is what to be expected.

V. Conclusion & Future Recommendations

A new hardware implementation of SHA-256 on Xilinx FPGA has been proposed in this paper. The SHA-256 is chosen because of its complex security, high collision resistance, acceptable computation time along with its utilization in different applications. Optimization techniques were used based on gated clock conversion, arithmetic resource sharing, and structural modeling of small building blocks.

The proposed design along with the optimization techniques used have resulted in a significant power and area reduction with a relatively large efficiency while maintaining a decent maximum frequency and throughput in comparison with other related work. The hardware operated at 83.33 MHz frequency with 0.637 Gbits/s throughput while utilizing 275 slices and consuming 13 mW of dynamic power. Such design with low power and area can be used to hash messages on a portable device opening a whole new area for different applications and opportunities.

For future work, we recommend implementing the code on actual FPGA hardware and measuring the dynamic power consumption to get a more accurate power estimation. Also, implementing the same optimization techniques used on different hardware is recommended, as the effect of different techniques on the power and area may depend on the hardware used.

Table II. Implementation results and comparison with previous work

Paper	Our Work	[4]	[6]	[7]	[8]	[9]
Slices	275	973	740	-	-	2796
LUTS	763	-	-	-	1803	-
Flip Flops	568	-	-	-	714	-
Frequency (MHz)	83.333	184.46	201.1	66	-	179.08
Throughput (Gbits/s)	0.637	1.388	0.2	-	-	-
Efficiency (Mbits/(s.slice))	2.32	1.43	0.27	-	-	-
Power (mW)	13	-	73.47	88.5	-	-
Platform	Virtex-7	Virtex-5	Virtex-5	Virtex-2	Virtex-6	Virtex-5

VI. ACKNOWLEDGEMENT

This work was supported by the Egyptian Information Technology Industry Development Agency (ITIDA) under ITAC Program.

REFERENCES

[1] M. Bahnasawi, A., K. Ibrahim, A. Mohamed, M. Khalifa, A. Moustafa, K. Abelmonim, Y. ismail, and H. Mostafa, "ASIC-Oriented Comparative Review of Hardware Security Algorithms for the Internet of Things Applications", *IEEE International Conference on Microelectronics (ICM 2016)*, Cairo, Egypt, pp. 285-288, 2016.

[2] N. Samir, A. S. Hussein, M. Khaled, A. N. ElZeiny, M. Osama, H. Yassin, A. Abdelbaky, O. Mahmoud, A. Shawky, and H. Mostafa, "ASIC and FPGA Comparative Study for IoT Lightweight Hardware Security Algorithms", *Journal of Circuits, Systems, and Computers (JCSC)*, vol. 28, no. 12, pp. 1-13, 2019.

[3] Q. Dang, "Federal Information Processing Standard (FIPS) 180-4, Secure Hash Standard," Aug. 2015.

[4] C. Li, Q. Zhou, Y. Liu, and Q. Yao, "Cost-Efficient Data Cryptographic Engine Based on FPGA," *2011 Fourth International Conference on Ubi-Media Computing*, 2011.

[5] R. García, I. Algredo-Badillo, M. Morales-Sandoval, C. Feregrino Uribe, and R. Cumplido, "A compact FPGA-based processor for the Secure Hash Algorithm SHA-256," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 194–202, 2014.

[6] X. Guo, et al., "On The Impact of Target Technology in SHA-3 Hardware Benchmark Rankings," *IACR Cryptology ePrint Archive,* vol.2010, pp. 536, Jan. 2010.

[7] R. G. Dimond, et al., "Combining Instruction Coding and Scheduling to Optimize Energy in System-on-FPGA," *2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2006.

[8] M. Thakur, "Low Power Implementation of Secure Hashing Algorithm (SHA-2) using VHDL on FPGA of SHA-256", *International Journal for Research in Applied Science and Engineering Technology*, vol. 6, no. 5, pp. 2298-2303, 2018.

[9] C. Jeong and Y. Kim, "Implementation of efficient SHA-256 hash algorithm for secure vehicle communication using FPGA," *2014 International SoC Design Conference (ISOCC)*, 2014.