A High-Performance Parallel Hardware Architecture of SHA-256 Hash in ASIC

Ruizhen Wu*, Xiaoyong Zhang**, Mingming Wang*, Lin Wang*

**Inspur Electronic Information Industry Co.,Ltd, Xi'an Shaanxi Province China*

***Biren Technology Ltd, Shanghai China*

wuruizhen@inspur.com, xyz8070@126.com, wangmingming02@inspur.com, wanglinlc@inspur.com

Abstract—**The SHA-256 algorithm is used to ensure the integrity and authenticity of data in order to achieve a good security thus is playing an important role in various applications, such as e-transactions and bitcoins. The SHA-256 computation capacity is a main research direction of Hashing Algorithm. In order to improve the computation capacity of hardware, the proposed design first uses pipeline principle and circuitry of timing prediction to find a most efficient architecture for implementation. Then it is optimized with hash function and hardware characteristics to give a high-performance hardware architecture of SHA-256 hash. Three pipelines are used to replace the critical path in the round functions which can shorten the timing path, and divide the computation chain into independent steps. Multi-computation of SHA-256 is working in parallel pipelines, indicating that the computation capacity can be 3 times of that with standard SHA-256 implementation. The proposed SHA-256 hardware architecture has been implemented and synthesized with Intel 14nm technology. Simulation and synthesis results show the proposed SHA-256 hashing throughput can be improved by 3 times with 50.7% power reduction, at an area cost of 2.9 times compared to that of the standard implementation.**

Keyword—**Application specific integrated circuits, Cryptography, High-speed integrated circuits, Low-power electronics**

I. INTRODUCTION

SECURE hashing algorithm is used to ensure the data integrity and authenticity while being stored and transferred. Hash functions take input data of arbitrary length and convert them into some fixed data, called hash value or message digest. The SHA-256 of hashing algorithm is playing an important role in various applications. Almost all e-transactions, high-throughput designs of security schemes are needed. Bitcoin is a new and popular use of SHA-256, as the POW ("Proof of Work" [1]) mentions in the Bitcoin protocol: the POW requests a huge number of SHA-256 computations to find a proper 32-bit number to satisfy the protocol requirement. The first finder is awarded by bitcoin, which means the computation capacity of SHA-256 is the key factor to get awarded, therefor the main research direction.

The SHA-256 hash architecture acts more and more important nowadays thus several improved designs are proposed. To embed a security engine in a RFID tag, two compact SHA-256 implementation are presented, a low area design and a low power design [2]. To achieve the improvement several adder cycles and adder selectors were added in the round computation which made it very suitable for power-area balanced applications.

One application of ideas and techniques from functional languages to the model-driven design and synthesis of hardware artifacts for SHA-256 was proposed in [3]. The co-design of hardware and software not just made the SHA-256 algorithm easier to implement, but also gave a more effective way to optimize the performance of SHA-256 from software to hardware based on designer's need.

However the most challenging request of SHA-256 is high processing speed and low power in hardware. An optimized pipelined architecture of SHA-256 hash function has been implemented in [4] which used custom data path that enforces the reuse of modules based on which novel processor architecture was implemented. Reference [5] proposed a SHA-256 unfolding design based on reconfigurable hardware modules. The complex linear computing of SHA-256 was reconfigured by newly added computation modules. Reference [6] implemented SHA-256 architecture based on operation rescheduling to minimize the critical path delay. Reference [7] proposed a more effective hardware to control the SHA-256 computation, which was using the finite state machine (FSM).

The purpose of this paper is to provide a high

Manuscript received Jan. 10, 2019. This work is a follow-up of the invited journal to the accepted & presented paper of the 21th International Conference on Advanced Communication Technology (ICACT2019), and Grant ID is ICACT-20190011.

Ruizhen Wu is with the Inspur Electronic Information Industry Co.,Ltd, Xi'an, Shaanxi 710071 China. (Phone: +86 15909206044; e-mail: wuruizhen@inspur.com).

Xiaoyong Zhang is with the Biren Technology Ltd, Shanghai 201210 China. (E-mail: xyz8070@126.com).

Mingming Wang is with the Inspur Electronic Information Industry Co., Ltd, Xi'an, Shaanxi 710071 China. (E-mail: wangmingming02@inspur.com).

Lin Wang is with the Inspur Electronic Information Industry Co.,Ltd, Xi'an, Shaanxi 710071 China. (E-mail: wanglinlc@inspur.com).

performance parallel computation hardware architecture in ASIC of SHA-256 hash. The organization of this paper is: Section 2 describes the classic SHA-256 algorithm; Section 3 uses the pipeline principle and circuitry timing prediction to find the most efficient pipeline architecture for SHA-256. Section 4 presents the proposed SHA-256 parallel computation hardware architecture with 3 pipelines. The implementation results and comparison with other designs are in Section 5. The last section provides the conclusions.

II. SHA-256 ALGORITHM

This section describes the function of SHA-256 hash. A detailed description of the SHA-256 hashing algorithm can be found in the official NIST standard [8]. The SHA-256 computation can be divided into 2 steps. The first step is to pre-process the original messages. It involves message padding and expanding the message for the round computation. The padding means appending bits according to some rules until the total length is integer of 512-bit. Afterwards every 512-bit will be expanded to 64*32 bit for SHA-256 round computation.

Here we use "t" to indicate the number of transformation rounds.

When 0≤t≤15: $W_t = input\ message$

When 16≤t≤63:

$$W_t = \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16} \quad (1)$$

The first 16 W_t are input messages. And after that the others are from iterative operation. In equation (1) the σ is calculated by:

$$\sigma_0^{\{256\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) \quad (2)$$

$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \quad (3)$$

In (2) and (3), the $ROTR^n(x)$ means a right rotation of x by n bits, and $SHR^n(x)$ means shift right of x by n bits.

The whole SHA-256 computation is showed in Fig. 1.

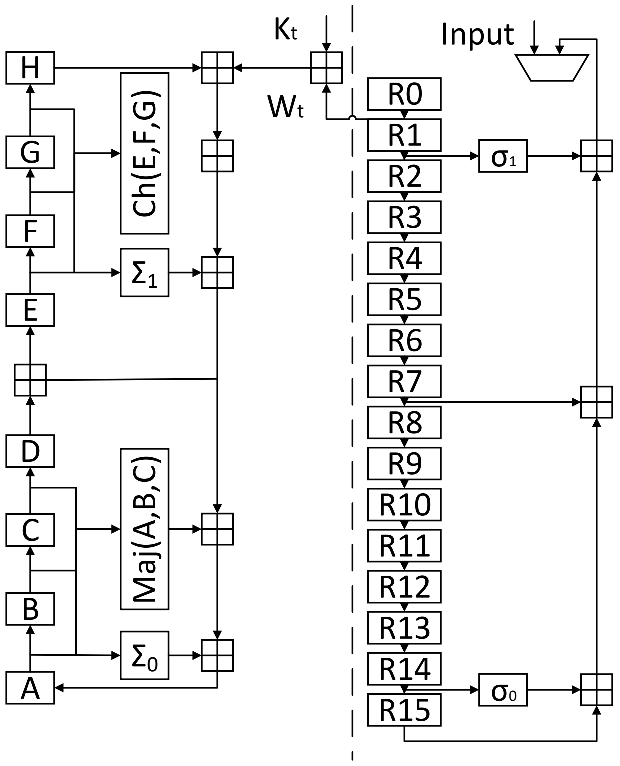

Fig. 1. SHA-256 hashing algorithm

The second step is called round computation shown in left part of dotted line in Fig. 1 is to obtain the "a"~ "h", which can be calculated by:

$$T_1 = h + \sum\nolimits_1^{\{256\}}(e) + ch(e,f,g) + K_1^{\{256\}} + W_1 \quad (4)$$

$$T_2 = \sum\nolimits_0^{\{256\}}(a) + Maj(a,b,c) \quad (5)$$

$$h = g \quad (6)$$

$$g = f \quad (7)$$

$$f = e \quad (8)$$

$$e = d + T_1 \quad (9)$$

$$d = c \quad (10)$$

$$c = b \quad (11)$$

$$b = a \quad (12)$$

$$a = T_1 + T_2 \quad (13)$$

The first round of a, b, c, d, e, f, g and h are assigned by the initial value of SHA-256 definition. The K_t is a constant in 32-bit and 64 values overall. And the four function computations are showed in (14)-(17):

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge y) \quad (14)$$

$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (15)$$

$$\sum\nolimits_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \quad (16)$$

$$\sum\nolimits_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \quad (17)$$

The $\oplus$ represent bitwise XOR operation, the $\wedge$ represent bitwise AND operation and the $\neg$ bitwise complement operation.

Each round of (6)-(13) can generate 8 hash values, they were showed in the right part of Fig. 1 and calculated by:

$$H_0^{(i)} = a + H_0^{(i-1)} \quad (18)$$

$$H_1^{(i)} = b + H_1^{(i-1)} \quad (19)$$

$$H_2^{(i)} = c + H_2^{(i-1)} \quad (20)$$

$$H_3^{(i)} = d + H_3^{(i-1)} \quad (21)$$

$$H_4^{(i)} = e + H_4^{(i-1)} \quad (22)$$

$$H_5^{(i)} = f + H_5^{(i-1)} \quad (23)$$

$$H_6^{(i)} = g + H_6^{(i-1)} \quad (24)$$

$$H_7^{(i)} = h + H_7^{(i-1)} \quad (25)$$

The final output is obtained by the 64th round hash value as below:

$$output = H_0 \,||\, H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4 \,||\, H_5 \,||\, H_6 \,||\, H_7 \quad (26)$$

III. PIPELINE ARCHITECTURE

It is clear from [4]-[7] that to achieve high processing speed and low power in hardware the pipeline architecture is a good choice of implementation which has gained a lot of interests. From hardware design perspective the pipeline architecture can work in various forms as long as the algorithm calculation units can be divided to satisfy the pipeline architecture needs.

So to find a most efficient pipeline architecture for high processing speed and low power in hardware we need to consider the change of delay and area cost in SHA-256 hardware architecture.

A. *State-of-art Pipeline principle*

The pipeline architecture's core concept is to use FFs (Flip-Flops) to divide the serial working flow and rebuild it into a parallel working flow. So the timing model based on the hardware function requirement have to be built first to find out the most efficient hardware architecture solution. From [9-15] we used the state-of-art pipeline method to consider the relationship between delays and timing combinations with a piecewise linear model. Consider the serial working flow's path delay is "D" and area is "G". And the parallel working flow's one stage pipeline path delay is "S" and area is "L". Then the relationship between the serial working flow and parallel working flow can be described as Fig. 2 showed.

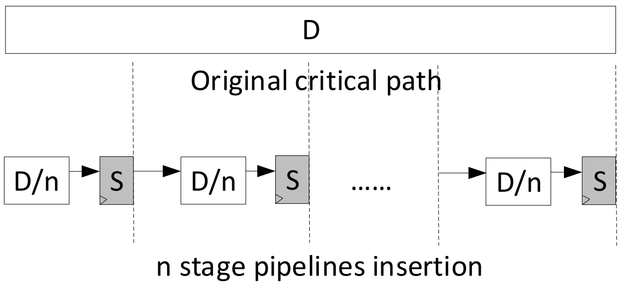

Fig. 2. N stage pipeline insertion into a critical path

The serial working flow can be described as an original critical path which can insert n stage pipelines to work as a parallel working flow. So the new path delay with n stage pipelines inserted is:

$$D' = \frac{D}{n} + S \quad (27)$$

The new area with n stage pipelines inserted can be described as:

$$G' = G + n \times L \quad (28)$$

To give a most efficient pipeline architecture means to find an optimum solution which can have the smallest frequency per area. And that equals to the same question of finding out the smallest area cost for maximum achievable frequency. Considering the maximum achievable frequency can be described as "1/D" so the area cost is:

$$Cost = G \times D \quad (29)$$

Which means the n stage pipelines cost can use (27)-(29) to summarize as:

$$\begin{aligned} Cost' &= G' \times D' \\ &= (G + n \times L) \times \left(\frac{D}{n} + S\right) \\ &= \frac{G \times D}{n} + G \times S + L \times D + L \times S \times n \end{aligned} \quad (30)$$

B. *Update the pipeline principle*

Evolve the theory and model of III.A with II's SHA-256 algorithm to update the pipeline architecture, the N stage pipelines insertion of Fig. 2 can be described as:

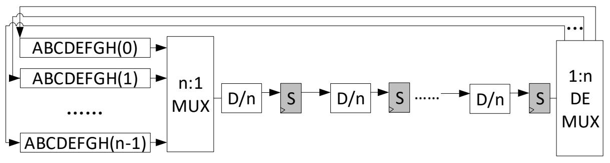

Fig. 3. N stage pipeline insertion with SHA-256

From Fig. 3 it is clear to see that besides the (30) considering FFs' cost there are additional cost needed for MUX and DEMUX's input and output. So for A~H's calculation each pipeline insertion needs an extra 8 registers to temporarily store values of A~H for MUX and DEMUX. No matter how many pipelines are inserted the hardware architecture only needs a pair of MUX and DEMUX. Taking Intel 14nm technology devices' parameter for reference to calculate the area cost with (28), the area of (28) can be updated to:

$$\begin{aligned} G'' &= G' + 8 \times n \times L + 2 \times (n-1) L_{m\&d} \\ &= G + 7.5 \times L + 9.5 \times n \times L \end{aligned} \quad (31)$$

The $L_{m\&d}$ is the area cost of MUX and DEMUX, which is almost 0.25L of (28). The 8*n*L represents the extra 8 registers area cost for n pipelines insertion.

Do the same calculation to find the delay relationship of MUX and DEMUX with (28) to update the (27):

$$D'' = \frac{D}{n} + (0.46 \times \log_2(n) + 1) \times S \quad (32)$$

The 0.46 here is because in STA report it can find that one AND gate's delay is almost 0.46 time of a FF's delay. Use (31) and (32) to update the cost equation as shown in (30):

$$\begin{aligned} Cost'' &= G'' \times D'' \\ &= (G + 7.5 \times L + 9.5 \times n \times L) \\ &\quad \times \left(\frac{D}{n} + (1 + 0.46 \times \log_2(n)) \times S\right) \\ &= (G + 7.5 \times L) \times \frac{D}{n} + 9.5 \times L \times (D + S \times n) \\ &\quad + (G + 7.5 \times L) \times 0.46 \times \log_2(n) \\ &\quad + 4.37 \times L \times S \times n \times \log_2(n) \end{aligned} \tag{33}$$

In (33) only the "n" is a variable, the others are all decided by the technology lib used which can be considered as fixed values here. So the optimum solution means to find out the minimum value of (33).

With the actual technology lib's parameters. the fixed values of (33) can be calculated as:

$$G = 11.2579 \mu m^2 \tag{34}$$

$$D = 1712.57 ps \tag{35}$$

$$L = 0.67 \mu m^2 \tag{36}$$

$$S = 131.71 ps \tag{37}$$

Calculating the (33) with (34)-(37), the relationship curve of cost'' with different n is showed in Fig. 4.

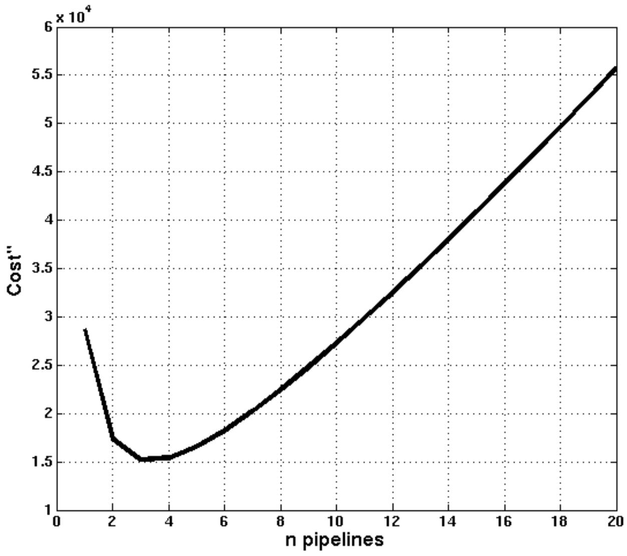

Fig. 4. Relationship curve of cost'' with different n

According to Fig. 4 it is clear to see that when n=3 the cost gets the minimum value which means the most efficient pipeline architecture for SHA-256 calculation with Intel 14nm technology lib is 3 pipelined architecture.

IV. Proposed Design

To realize the 3 pipeline architecture of SHA-256 we need to divide the SHA-256 calculation into 3 parallel working flow in an efficient way. From the SHA-256 hashing algorithm we can see what limits the computation speed most is (4)-(17). In fact the (1)-(3) can be done quite earlier but has to wait a very long time for (4)-(17) to finish a round of whole SHA-256 computation. The proposed design is to optimize the (4)-(17) in order to get a better performance in 3 steps.

A. Critical Path Analysis

To analyze the critical path of SHA-256 we unfold the computation steps, which are showed in Fig. 5.

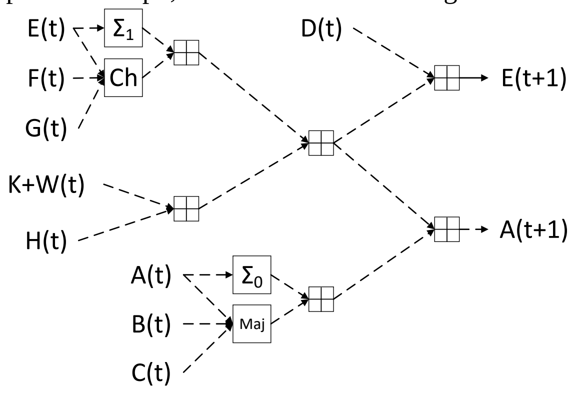

Fig. 5. Critical path

The symbols in Fig. 5 are same as Fig. 1, and the "+" represents the addition modulo 232. As known the most problematic characteristic of SHA-256 is the addition modulo 232, which slows down the speed heavily versus the other calculation steps [16]. Based on this fact, we find the most critical path in SHA-256 round calculation. As Fig. 5 shows, the most critical path is showed in dotted line, which means from calculation e(t) to obtain a(t+1) in each round is the longest path worth optimizing (same as other long path with 3 "+" calculations).

B. Break Critical Path

To optimize the SHA-256 computation it needs to break the critical path into shorter paths thus make it possible for the whole computation chain to work at a higher frequency. For this need, we consider each addition modulo 232 calculation as one basic unit of SHA-256 calculation in each round. To separate all calculations we insert FFs to each minimum unit.

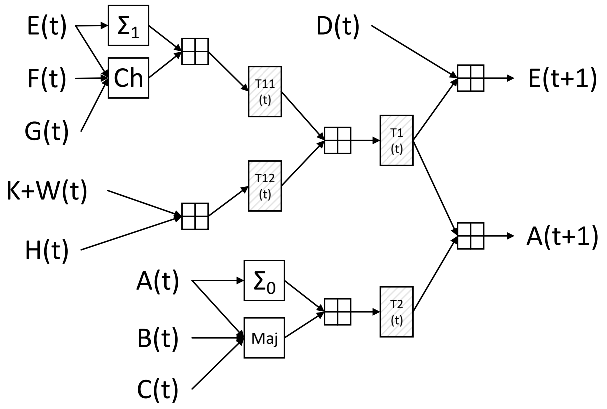

Fig. 6. FFs insertion

As Fig. 6 shows, we insert 4 32-bits FFs (The grey rectangles) in each round of SHA-256 to separate all basic units. With the insertion, there is only one addition modulo 232 calculation between each two FFs. And the T1 (t) and T2 (t) mean the FFs for T1 and T2 calculation as what equation (4) and (5) show, n means the round number of whole SHA-256 computation. T11 (t) and T12 (t) are intermediate results for T1, "t" the same as before.

Because the long path is broken， we can run almost 3 times faster than before, but each round will take 3 cycles now.

C. Reschedule With Parallel Pipeline

In standard SHA-256 computation, variable "A" to "H" are calculated one after another. But most of them are just a bit shifting operation which is much easier compared to addition modulo 232 calculation. Consequently, we reschedule the SHA-256 computation based on step B's basic units with parallel pipeline.

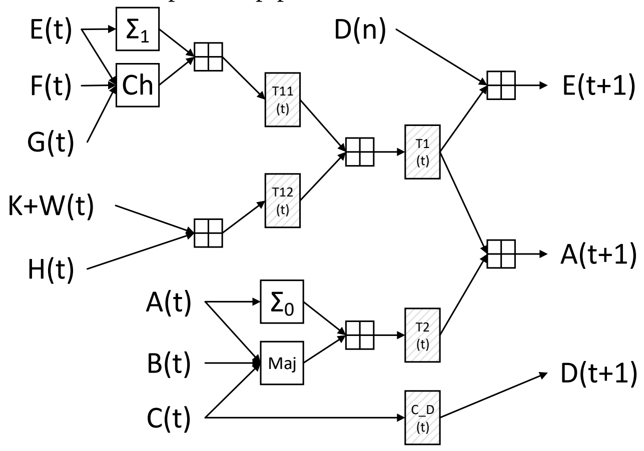

Fig. 7. Variable update

As Fig. 7 showed, to achieve parallel computation, a key intermediate FF "C_D" is added, to preserve C and update D later. With this rescheduling the SHA-256 is separated into 3 individual steps to update "A" ~ "H", in subsequent 3 cycles. The functional diagram in sequence is showed in Fig. 8.

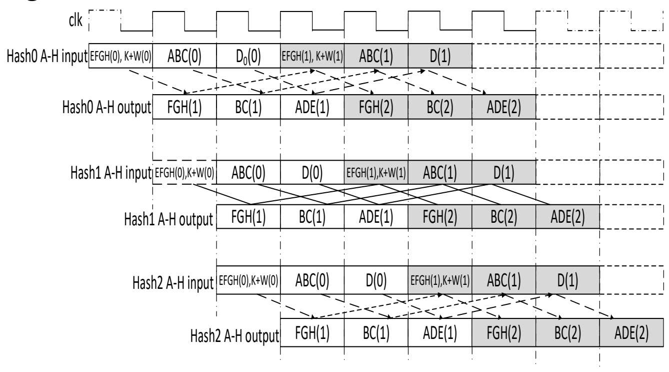

Fig. 8. Rescheduled parallel pipeline SHA-256

As the Fig. 8 shows, with parallel pipelines three SHA-256 hashes can be calculated at the same time. The hardware architecture is showed in Fig. 9.

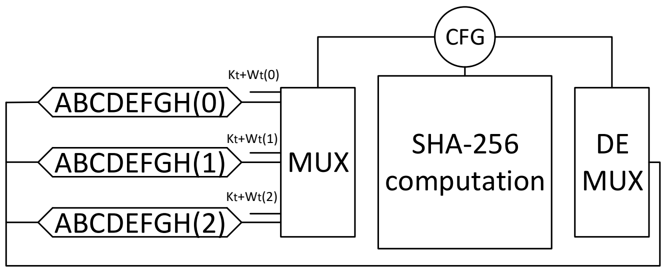

Fig. 9. Parallel computation SHA-256 hardware architecture

The proposed architecture gains 3 times performance with a 3 times higher clock frequency, compared to the standard architecture.

V. Result And Discussion

The proposed high performance parallel computation of SHA-256 is successfully implemented in Verilog. The hardware architecture is fully verified at RTL level and synthesized with Intel 14nm technology lib. The comparison results are showed below:

TABLE I
Hardware Comparison Results

	Clock(ps)	Area(μm2)	Power(mW)
Standard	1959	4916.7	3.3794
Proposed	653	14272.6	6.855

The proposed high performance parallel computation hardware architecture of SHA-256 is 3 times faster than the standard architecture as we expected.

The area cost to achieve this improvement is 2.90 times compare to standard SHA-256, which is because there are reused function modules to save area.

The power of proposed parallel SHA-256 is just 2.03 times of standard SHA-256 to have same 3 times output. That's because the sequential logic consumes much bigger power than the combinational logic, and the proposed architecture can exactly save much sequential logic.

VI. Conclusions

The parallel hardware architecture is the best solution to achieve high processing speed and low power consumption in hardware. This paper first builds the speed and area model with SHA-256 algorithm and technology lib to find the most efficient pipeline architecture is 3 pipeline stages for SHA-256 realization. Then it is dividing and updating the architecture with SHA-256 calculation unfolding by FFs to give a high performance hardware architecture for parallel computation in AISC. The design is synthesized with Intel 14nm technology and the comparison results demonstrated the improvement of 3 times computation speed with 50.7% power consumption, at a cost of only 2.9 times area.

REFERENCES

[1] S. Nakamoto. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] X. Cao and M. O'Neill, "Application-oriented SHA-256 hardware design for low-cost RFID," in *Proc. IEEE International Symposium on Circuits and Systems*, Seoul, 2012, pp. 1412-1415.

[3] W. L. Harrison, A. M. Procter and G. Allwein, "Model-driven design & synthesis of the SHA-256 cryptographic hash function in rewire," in *Proc. IEEE International Symposium on Rapid System Prototyping (RSP)*, Pittsburgh, 2016, pp. 1-7.

[4] M. Padhi and R. Chaudhari, "An optimized pipelined architecture of SHA-256 hash function," in *Proc. IEEE 7th International Symposium on Embedded Computing and System Design (ISED)*, Durgapur, 2017, pp. 1-4.

[5] S. Suhaili and T. Watanabe, "Design of high-throughput SHA-256 hash function based on FPGA," in *Proc. IEEE 6th International Conference on Electrical Engineering and Informatics (ICEEI)*, Langkawi, 2017, pp. 1-6.

[6] I. Algredo-Badillo, C. Feregrino-Uribe, R. Cumplido and M Morales-Sandoval, "FPGA-based implementation alternatives for the inner loop of the Secure Hash Algorithm SHA-256," *Microprocessors & Microsystems*, vol. 37, pp. 750-757, Jun. 2013.

[7] H. Mestiri, F. Kahri, B. Bouallegue, and M. Machhout, "Efficient FPGA Hardware Implementation of Secure Hash Function SHA-2," *International Journal of Computer Network and Information Security*, vol. 7, pp. 9-15, Dec. 2014.

[8] *Secure Hash Standard (SHS), N. I. of Standards and Technology*, FIBS PUB 180-4, 2012.

[9] G. L. Zhang, B. Li, and U. Schlichtmann, "PieceTimer: a holistic timing analysis framework considering setup/hold time interdependency using a piecewise model," in *Proc. 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Austin, 2016, pp. 1-8.

[10] G. L. Zhang, B. Li, Y. Shi, J. Hu, and U. Schlichtmann, "EffiTest2: Efficient Delay Test and Prediction for Post-Silicon Clock Skew Configuration Under Process Variations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 38, pp. 705-718, Apr. 2019.

[11] G. L. Zhang, B. Li, J. Hu, Y. Shi, and U. Schlichtmann, "Design-Phase Buffer Allocation for Post-Silicon Clock Binning by Iterative Learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 392 – 405, Feb. 2018.

[12] G. L. Zhang, B. Li, M. Hashimoto, and U. Schlichtmann, "Virtualsync: timing optimization by synchronizing logic waves with sequential and combinational components as delay units," in *Proc. 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, San Francisco, 2018, pp. 1-6.

[13] G. L. Zhang, B. Li, B. Yu, D. Z. Pan, and U. Schlichtmann, "TimingCamouflage: Improving circuit security against counterfeiting by unconventional timing," in *Proc. 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, 2018, pp. 91-96.

[14] G. L. Zhang, B. Li, and U. Schlichtmann, "Timing with Virtual Signal Synchronization for Circuit Performance and Netlist Security," in *Proc. 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Hong Kong, 2018, pp. 715 – 718.

[15] Y. Yao, *SuperScalar RISC Processor Design*. Bei Jing: Tsinghua University Press, 2014, ch. 1.

[16] L. Dadda, M. Macchetti and J. Owen, "The design of a high speed ASIC unit for the hash function SHA-256 (384, 512)," in *Proc. Design, Automation and Test in Europe Conference and Exhibition*, Paris, 2004, pp. 70-75.

Ruizhen, Wu was born in China, Jan 1st 1986. PhD. The PhD was earned in School of Microelectronics of XIDIAN University, Shaanxi Province, China, in 2014. The major field of study is Asynchronous Circuits design, 5G CODEC and AI.

He has worked in Hangzhou, Zhejiang Province, China, since 2014, in the 2012 communication lab of Huawei at first, then Intel iCDG, and Inspur Electronic Information Industry Co.,Ltd now

Xiaoyong, Zhang was born in China, Nov 5th 1980. Bachelor. The first bachelor degree was earned in automation, in School of Marine Science and Technology, Northwestern Polytechnical University, Shaanxi Province, China, in 2003, and the second bachelor degree was earned in electronic science and technology, in Institute of Microelectronics, Tsinghua University, Beijing City, China, in 2005.

He has worked in Xi'an, Shaanxi Province, China, since 2005, in the wireless department for Infineon Technology at first, and now Biren technology. His current job is SoC HW design manager

Mingming, Wang was born in China, Oct 1st 1986. Master. The Master degree was earned in Computer Application Technology in Xi'an University of posts & Telecommunications, Shaanxi Province, China, in 2011, and the bachelor degree was earned in electronic science and technology, in Xi'an University of posts & Telecommunications, Shaanxi Province, China in 2008.

He has worked in Inspur Electronic Information Industry Co.,Ltd since 2019.

Lin, Wang was born in China, Dec. 1st. 1971. Master of Sci. The master degree was earned in Dept. of Electrical Engineering, Fudan University, Shanghai, China, in 1998. His majority is microelectronics and physics on semiconductor and semiconductor device.

He worked in Shanghai Nortel Semiconductor and Broadcom, focusing on communication chip development after his graduation.

He is now the Director of SoC R&D in Inspur Electronic Information Industry Co.,Ltd.