High-Throughput Message Digest (MD5) Design and Simulation-Based Power Estimation Using Unfolding Transformation

Shamsiah binti Suhaili¹ and Takahiro Watanabe²

¹Faculty of Engineering, Universiti Malaysia Sarawak, 94300 Kota Samarahan, Sarawak, Malaysia
²Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0135, Japan
E-mail: ¹sushamsiah@unimas.my, ²watt@waseda.jp

Abstract The high throughput of the cryptographic hash function has become an important aspect of the hardware implementation of security system design. There are several methods that can be used to improve the throughput performance of MD5 design. In this paper, four types of MD5 design were proposed: MD5 iterative design, MD5 unfolding design, MD5 unfolding with 4 stages of pipelining design and MD5 unfolding with 32 stages of pipelining design. The results indicated that MD5 unfolding with 32 stages pipelining of design provides a high throughput compared with other MD5 designs. By using an unfolding transformation factor of 2, the number of cycles of MD5 design was reduced from 64 to 32. All the proposed designs were successfully designed using Verilog code and simulated using ModelSim. The throughput of MD5 unfolding with 32 stages of pipelining design was increased significantly to 137.97 Gbps, and the power of this MD5 unfolding with 32 stages of pipelining was 750.99 mW. Therefore, it is suggested that an unfolding transformation with a high performance pipelining are applied to MD5 hash function design in order to produce an embedded security system design. This paper is expected to be for improving the maximum frequency and the throughput of MD5 design. Thus, by increasing the number of stages in MD5 unfolding design, the performance of MD5 designs can be improved significantly.

Keywords: maximum frequency, MD5, pipelining, transformation

1. Introduction

A hash function is a computationally efficient function that maps fixed binary chains of arbitrary length to bit sequences of fixed length [1]. The output of a hash function is called a hash code, message digest or hash value. An unkeyed hash function produces a short message with the changes of the bit in the input message. The properties of the hash function are the one-way property or preimage resistance, second preimage resistance and collision resistance. For the one-way property, it is difficult to find message M such that h = H(M). In other words, it is easy to compute the hash code from the message but difficult to obtain the message from the hash code. In the second preimage resistance, it is difficult to find message M₂ when the output hash codes M₁ and M₂ are the same. The third property of a hash function is collision resistance, where it is difficult to find the same hash code H(M₁) = H(M₂) with two different messages M₁ and M₂.

There are several methods that can be used to improve the performance of the MD5 hash function since nowadays all the designs require fast implementation. The high speed of the designs, as well as high throughput, needs to be considered when designing an efficient MD5 hash function. Therefore, unfolding transformation and pipelining design techniques have been chosen to improve the throughput of the MD5 design.

Efficient design also considers power consumption. By obtaining a low-power design, the performance can be improved. The power consumption of a CMOS transistor can be divided into three components: short-circuit, dynamic and static (leakage) powers. Leakage power occurs when the circuit is on because of the sub-threshold currents and reverse biased diodes in a CMOS transistor [2]. Switching power contains dynamic and short-circuit powers. Switching power exists when the signals change their logic states.

2. MD5 Algorithm

A hash function is important for message authentication during data transmission. There are several different types of hash function such as SHA-1, RIPEMD-160 and SHA-2. One of the well-known hash functions is the MD5 algorithm, which was developed by
Rivest [3]. The process of the MD5 algorithm is divided into two parts, namely message preprocessing and hash computation. The preprocessing message starts by padding the message in little-endian format. The message input is padded with a single 1 bit at the end of the message. Then, it is followed by 0 bits until the length of the message is congruent to 448 modulo 512. The remaining 64 bits are reserved for the length of the message. Since MD5 is in little-endian format, the last 64 bits are also the length of the message in little-endian format. Therefore, the overall input messages have a total size of 512 bits. Table 1 shows the input buffer initialization of the MD5 algorithm, that is the values in registers A, B, C and D in little-endian format. After padding the messages, the hash computation takes place. The 512-bit input messages are divided into 16 blocks, where each block consists of 32-bit input messages. Thus, the input messages become \( M[0], M[1], M[2] \ldots \ldots M[15] \). The MD5 algorithm has four different types of nonlinear function, namely F, G, H and I. Each nonlinear function has 16 rounds. Hence, the total number of rounds for the MD5 algorithm is 64.

### Table 1 Buffer initialization of MD5 algorithm

| Reg | Normal Value | Little Endian Format |
|-----|--------------|---------------------|
| A   | 32'h001234567 | 32'h67452301        |
| B   | 32'h89abcdef  | 32'hefcdab89        |
| C   | 32'hfedcba98  | 32'h98badcfe        |
| D   | 32'h76543210  | 32'h10325476        |

Figure 1 illustrates the compression function of the MD5 algorithm. It consists of nonlinear functions (Func\(_t\) (B,C,D)), a message input (\( M_t[k] \)), a constant (K\(_t[i] \)) and a shift (S\(_t\)). This compression function is executed for 64 rounds in order to obtain the 128-bit final MD5 hash code.

![Figure 1](image)

Equations (1) - (4) show the nonlinear functions of the MD5 algorithm, \( F(B,C,D) \) for the first round, \( G(B,C,D) \) for the second round, \( H(B,C,D) \) for the third round and \( I(B,C,D) \) for the fourth round. Symbols \( \wedge, \vee, \neg \) and \( \oplus \) denote logical AND, OR, NOT and XOR operations, respectively.

\[
\begin{align*}
F(B,C,D) &= (B \wedge C) \vee (\neg B \wedge D) \\
G(B,C,D) &= (B \wedge D) \vee (C \wedge \neg D) \\
H(B,C,D) &= (B \oplus C \oplus D) \\
I(B,C,D) &= (C \oplus B \oplus \neg D)
\end{align*}
\]

Let \( t \) denote the step index of the MD5 algorithm. Then, \( M_t[k] \) represents the \( k \)th of 32-bit word of \( M_t \). The symbol \( \ll S \) means a left circular shift by \( S \) bits. Func\(_t\)(B,C,D) represents the nonlinear functions F, G, H and I. Equation (5) shows the outputs A, B, C and D for each iteration of the MD5 algorithm.

\[
T = B + ((A + Func(B,C,D) + M_t[k] + K_t[i]) \ll S) \\
A = D, B = T, C = B, D = C
\]

Finally, the last output of the nonlinear function I(B, C, D) is added to the initial inputs in order to obtain the 128-bit MD5 hash code.

### 3. Proposed Unfolding MD5 Algorithm

One of the DSP application techniques used to obtain new programs is the unfolding algorithm. The unfolding factor \( J \) describes the number of iterations from the original program [4]. This method can increase the throughput of MD5 design when the throughput is important in transmission data processing. An unfolding MD5 algorithm with a factor of 2 is proposed in this paper.

This implementation of the MD5 algorithm reduces the number of cycles from 64 to 32. In other words, the MD5 hash function processes two hash operations in one cycle. This leads to the high throughput of MD5 because of the small amount of latency. The proposed method can achieve better performance than the traditional method [5]. Thus, this improvement of MD5 design is important for HMAC (Hash-based Message Authentication Code) and digital signature applications.
Figure 2 illustrates the compression function of the MD5 unfolding algorithm. From this figure, there are two operations of the MD5 hash function executed in one cycle. The MD5 unfolding algorithm consists of two nonlinear functions (Func_t(B,C,D) and Func_t+1(B,C,D)), two input messages (M_t[k], and M_t+1[k]), two constants (K_t[t] and K_t+1[t]) and two shift values (S_t and S_t+1).

Fig. 2 Compression function of unfolding MD5 algorithm

There are three inputs of the first nonlinear function, B, C and D. Then, the operation is started by adding the message, a constant and input A. This result is shifted to the right by the value S. Output T is obtained by adding input B. This operation is followed by the second nonlinear operation with inputs T, B and C. The same process is carried out to obtain the output Temp. All operations are performed in parallel. By restructuring the architecture of the MD5 compression function into two parallel operations in one cycle, the number of MD5 cycles can be reduced to 32. Some modification of outputs A, B, C and D is required to produce the correct output. The outputs of the MD5 unfolding algorithm are shown in Eq.(6). T represents the first-round calculation of the MD5 algorithm, then it is followed by the second round which outputs Temp.

\[
T = B + ((A + \text{Func}(B,C,D) + M_t[K] + T[i]) \ll S)
\]

\[
\text{Temp} = T + ((D + \text{Func}(T,B,C) + M_{t+1}[K] + T[i]) \ll S)
\]

3.1 Top level of unfolding MD5 architecture

Figure 3 shows the top level of the architecture of the unfolding MD5 design. There are six modules in this MD5 architecture: ABCD_init, input_ABCD, MD5_initial, func_process, getdata and MD5_hash. The top level of the unfolding MD5 architecture has 32 rounds of a nonlinear function. Each nonlinear function consists of 8 rounds, where two hash operations are processed in one cycle. All of these operations and some modifications are executed in the getdata module. Two non-linear functions, two messages and two constants are executed in parallel to apply the unfolding method to this MD5 algorithm. The func_process module generates the data for the message, shift and constant value. The MD5_initial module is used to select the input ABCD for the MD5 algorithm. First, the MD5 algorithm starts with data from the ABCD_init module, then the hash operation uses the data from the input_ABCD module.

3.2 Pipelining and unfolding MD5 algorithm

A combination of pipelining and unfolding methods can improve the performance of the MD5 algorithm. Figure 4 illustrates the architecture of an MD5 compression function with pipelining and unfolding transformation methods. In this paper we propose a novel combination of pipelining and unfolding transformation to increase the throughput of MD5 design. The proposed method achieves better performance because it can provide high-throughput MD5 algorithm based on FPGA. In this design, four registers are required to implement the four stages of pipelining, and 32 registers are required to implement the 32 stages of pipelining. By applying this technique to the MD5 algorithm, the performance of MD5 design increases significantly. The getdata module has been modified to obtain the pipelining and unfolding design for four stages of pipelining with the unfolding transformation and 32 stages of pipelining with the unfolding transformation. The rest of the modules remain the same but with some modification in terms of the input and output data.
4. Results and Discussions

Four types of MD5 design were successfully synthesized and implemented using Verilog code based on Altera Quartus II. The designs were simulated and verified by functional and timing simulation using ModelSim. Table 2 shows a comparison of the MD5 designs with those in previous publications. From this table, the proposed MD5 unfolding with 32 stages of pipelining design provides a high throughput of 137.972 Gbps compared with other designs as well as 23188 combinational ALUTs and a total of 2715 registers. This design uses a 4 ns clock constraint in order to meet the timing requirement of the design. Wang et al.[9] showed the number of combinational ALUTs for unfolding MD5 with 32 stages of pipelining was 36790 and the total number of registers was 26758. This design provided a maximum frequency 66.48 MHz and 32.035 Gbps throughput.

In this project, all designs use Arria II GX: EP2AGX45DF29C4 for the synthesis and implementation process. There are several other FPGA family devices such as Stratix for high-performance FPGAs and Cyclone for low-cost FPGAs. Hence, in order to obtain a balance between power and performance, the Arria II GX device has been chosen because the Arria series provide a low-cost transceiver for FPGA devices. Moreover, they also deliver optimal performance and power efficiency. The throughput of the MD5 design increases significantly because of the latency of the design. The latency for MD5 iterative design is 66 cycles. The latencies of the MD5 unfolding design and MD5 unfolding with pipelining are 33.5 and 34.5, respectively. The throughput of the MD5 design can be calculated by using Eq.(7), where $p$ denotes the number of stages in pipelining.

$$\text{Throughput} = \frac{(512 \times f_{\text{Max}} \times p)}{\text{Latency}}$$  \hspace{1cm} (7)

From Table 2, the efficient design of the MD5 hash function in terms of speed and area can be obtained by using pipelining and unfolding transformations. This method provides small-area implementation as well as the high speed.

| Design                  | Device                        | $f_{\text{Max}}$ (MHz) | Throughput (Mbps) | ALUTs/Total Registers/Slice |
|------------------------|-------------------------------|------------------------|-------------------|----------------------------|
| Proposed MD5 iterative  | Arria II GX: EP2AGX45D F29C4 | 118.9                  | 922.45            | 718/840/-                   |
| Proposed MD5 Unfolding | Arria II GX: EP2AGX45D F29C4 | 72.26                  | 1104.39           | 1344/786/-                  |
| Proposed MD5 Unfolding | Arria II GX: EP2AGX45DF29C4  | 234.0                  | 13892.5           | 3728/980/-                  |
| Proposed MD5 Unfolding | Arria II GX: EP2AGX45DF29C4  | 290.5                  | 137972            | 23188/271-5/                |
| Deepakumara et al.     | Virtex V1000FG680-6           | 21                     | 165               | -/-/880                     |
| Diez et al.            | Virtex 2V3000                 | 60.2                   | 467.3             | -/-/1369                    |
| Jarvinen et al.        | Virtex-II XC2V4000-6          | 78.3                   | 607               | -/-/1325                    |
| Wang_32Pipe [9]*latency 34] | Altera Stratix II EP2SGX90FF | 66.48                  | 32035             | 36790/26758                 |
| Wang_32Pipe [9]        | Altera Stratix II EP2SGX90FF | 86.32                  | 21428             | 36296/257-67               |
| Wang_4Pipe [9]         | Altera Stratix II EP2SGX90FF | 97.95                  | 3039              | 5929/3484                  |
| Wang_ite. [9]          | Altera Stratix II EP2SGX90FF | 102.7                  | 810               | 1352/462/-                 |
| He and Xue [10]        | Cyclone II EP2C35F672C6       | 48.2                   | 4,355             | 30,134/-                   |
| Noaman [11]            | Xilinx XC3S1400A              | n.a                    | n.a               | -/-/7499                   |
| Zhong et al. [12]      | n.a                           | n.a                    | n.a               | n.a                        |
| Irfan et al. [13]      | n.a                           | n.a                    | n.a               | n.a                        |

Other MD5 designs have been implemented on various FPGA family devices such as Altera and Xilinx. The throughput of the MD5 design by Deepakumara et al.[6] was 165 Mbps and that of Diez et al.[7] was 467.3 Mbps. The MD5 designs of Deepakumara et al.[6] and
Diez et al. [7] used Virtex V1000FG680-6 and Virtex 2V3000, respectively. Implementation on Virtex-II by Jarvinen et al. [8] provided a maximum frequency of 78.3 MHz and 607 Mbps throughput. The MD5 design by Wang et al. [9] gave a maximum frequency of 102.9 MHz and 610 Mbps throughput for iterative design. This design used Altera Stratix II EP2SGX90FF. The throughput of the MD5 design with 32 stages of pipelining with unfolding was increased by Wang et al. [9] to 32.03 Gbps. He and Xue [10] used Cyclone II for MD5 implementation and gave 4.4 Gbps throughput. Noaman [11] and Irfan, Landge and Mishra [13] designed MD5 using VHDL but no information was given in terms of speed, area and throughput. Zhong et al. [12] designed MD5 using Java.

Figure 5 shows a comparison of the performance to area ratio of MD5 designs. This graph shows that MD5 unfolding with 32 stages of pipelining gives the highest value. This design provides better results than the other designs. Figure 6 shows the simulation results for the design with MD5 unfolding with 32 stages of pipelining. From the simulation results, we can confirm that the MD5 hash code gives the correct output of the MD5 hash code with the input message.

Table 3 shows the simulation-based power estimation for the four different types of MD5 design evaluated using Altera Quartus II PowerPlay Power Analyzer. This is a power analysis tool that can provide the most accurate power estimation based on simulation [14]. In this table, the total thermal power dissipation is divided into three parts: dynamic thermal power dissipation, static thermal power dissipation and I/O thermal power dissipation. Dynamic power dissipation is caused by signal toggling, static power dissipation by leakage currents and I/O thermal power dissipation by I/O pins, where every possible parameter describing the off-chip board trace at each I/O pin is taken into account [14].

From this table, the total thermal power dissipation for MD5 unfolding design is slightly less than that for the MD5 iterative design. This is because the I/O thermal power dissipation of MD5 unfolding is reduced from 81.64 mW to 63.92 mW. After some modification of the MD5 architecture, the power consumption is reduced by 3.36% compared with the iterative design. However, the total thermal power dissipation for both MD5 unfolding with 4 stages of pipelining and MD5 unfolding with 32 stages of pipelining is increased significantly. From this table, the dynamic thermal power dissipation for MD5 unfolding with 32 stages of pipelining increases to 237.05 mW. This is because of the signal toggling of MD5 unfolding with the pipelining design. The static and I/O thermal power dissipations also increase. As mentioned earlier, the number of cycles of the MD5 unfolding design is reduced from 64 to 32. Therefore, this will affect the power consumption of the MD5 unfolding design, especially the I/O power consumption.

| MD5 Design           | Iterative | Unfolding | Unfolding 4Pipelining | Unfolding 32Pipelining |
|----------------------|-----------|-----------|------------------------|------------------------|
| Core Dynamic Thermal Power Dissipation (mW) | 11.75 | 13.43 | 24.08 | 237.05 |
| Core Static Thermal Power Dissipation (mW) | 317.71 | 319.91 | 320.27 | 334.26 |
| I/O Thermal Power Dissipation (mW) | 81.64 | 63.92 | 82.40 | 179.68 |
5. Conclusion

Four types of MD5 design based on Quartus II Arria II GX: EP2AGX45D F29C4 were successfully synthesized and implemented. The results show that the proposed MD5 unfolding with 32 stages of pipelining provides a high-performance MD5 design in terms of speed, area and throughput. The unfolding transformation gives a high-throughput MD5 design by reducing its latency. Moreover, the pipelined MD5 design has high speed in the maximum frequency. Therefore, an MD5 design combining the unfolding transformation and pipelining can improve the performance significantly. In this study, the maximum frequency (fMax) for MD5 unfolding with 32 stages of pipelining is 290.53 MHz with a throughput of 137.97 Gbps. This maximum frequency is about 144.3% higher than that for an iterative design. The power consumption of this design is 750.99 mW.

In the near future, we suggest the application of various methodologies in order to enhance the maximum frequency of the design such as resource sharing and retiming. Resource sharing can be done by combining the similar nonlinear functions with other hash functions. By carrying out this method, the area requirement can be reduced and a high maximum frequency can be achieved. Furthermore, retiming is one of the methodologies that can be applied to the design to reduce its delay. This method can increase the maximum frequency of the design, where the delay can be reduced by changing the position of the register.

Acknowledgments

This work was supported by Universiti Malaysia Sarawak (UNIMAS) and Graduate School of Information, Production, and Systems, Waseda University.

References

[1] F. R. Henriquez, N. A. Saqib, A. D. Perez and C. K. Koc: Cryptographic algorithms on reconfigurable hardware, Springer Series on Signals and Communication Technology, pp. 189–201, 2006.

[2] P. R. Panda, B. V. N. Silpa, A. Shrivastava and K. Gummidiipudi: Chapter 2, Power-Efficient System Design, Springer Science Business Media, LLC, 2010.

[3] R. L. Rivest: The MD5 message digest algorithm, RFC 1321, MIT Laboratory for Computer Science and RSA Data Security Inc., April 1992.

[4] K. K. Parhi: VLSI digital signal processing systems: Design and implementation, John Wiley & Sons Inc., pp. 119-140, 1999.

[5] S. Suhaili and T. Watanabe: High throughput evaluation of SHA-1 implementation using unfolding transformation, ARPN Journal of Engineering and Applied Sciences, ISSN 1819-6608, Vol.11, No.5, pp. 3350-3355, 2016.

[6] J. Deepakumara, H. M. Heys and R. Venkatesan: FPGA implementation of MD5 hash algorithm, Proceedings of the Canadian Conference on Electrical and Computer Engineering, CCECE 2001, Toronto, Canada, Vol.2, pp. 919-924, May 13-16, 2001.

[7] J. M. Diez, S. Bojanic, L. Stanimirovic, C. Carreras and O. Nieto-Taladrid: Hash algorithms for cryptographic protocols: FPGA implementations, 10th Telecommunications forum TELFOR’2002, Belgrade Yugoslavia, Nov. 26-28, 2002.

[8] K. Jarvinen, M. Tommiska and J. Skytta: Hardware implementation analysis of the MD5 hash algorithm, Proceedings of the 38th Hawaii International Conference on System Sciences, 2005.

[9] Y. Wang, Q. Zhao, L. Jiang and Y. Shao: Ultra high throughput implementations for MD5 hash algorithm on FPGA, High-Performance Computing and Applications, Lecture Notes in Computer Science, Vol. 5938, pp. 433-441, 2010.

[10] D. He and Z. Xue: Multi-parallel architecture for MD5 implementations on FPGA with gigabit-level throughput, 2010 International Symposium on Intelligence Information Processing and Trusted Computing, pp. 535-538, 2010.

[11] M. A. Noaman: A VHDL model for implementation of MD5 hash algorithm, Eng. and Tech. Journal, Vol. 31, No. 6, pp. 1107-1116, 2013.

[12] L. Zhong, W. Wan and D. Kong: Java web login authentication based on improved MD5 algorithm, International Conference on Audio, Language and Image Processing (ICALIP), pp. 131-135, 2016.

[13] A. Irfan, B. Landge and K. Mishra: Hardware based MD5 implementation using VHDL for secured embedded and VLSI based designs, International Conference on Communication and Electronics Systems (ICCES), pp. 1-6, 2016.

[14] Altera Corporation: FPGA power management and modeling techniques, White Paper, 101 Innovation Drive San Jose, CA 95134, December 2010. Access on https://www.altera.com/en_US/pdfs/literature/wp/wp-01044.pdf