High throughput implementation of RIPEMD-160 using unfolding transformation

Shamsiah binti Suhaili[1*], Takahiro Watanabe[2], Maimun binti Huja Husin[1], Kuryati bt Kipli[1], Norhuzaimin bin Julai[1], Rohana binti Sapawi[1]

[1]Department of Electrical and Electronic Engineering, Faculty of Engineering, Universiti Malaysia Sarawak, 94300 Kota Samarahan, Sarawak, Malaysia
[2] Graduate School of Information, Production and Systems, Waseda University,2-7 Hibikino, Wakamatsu-Ku, Fukuoka 808-0135, Japan

[*]Corresponding e-mail: sushamsiah@unimas.my

Keywords: Unfolding; RIPEMD-160; Throughput

ABSTRACT – Cryptographic hash function is important for digital signature, Hash Message Authentication Code (HMAC) and other data security application. There are different types of hash function such as MD5, SHA-1, RIPEMD-160, SHA-2 and others. In this paper, RIPEMD-160 algorithm has been chosen as one of the hash functions because of the parallel inner structure of this algorithm. The objective of this research is to design and implement RIPEMD-160 with high throughput using different types of methodology. Two types of methodologies are iterative and unfolding design. These methodologies were applied to this RIPEMD-160 design in order to analyze the results of maximum frequency, area implementation and throughput of the design on Arria II GX FPGA family device. By using unfolding transformation, the throughput of the RIPEMD-160 can be improved which is about 884.62 Mbps.

1. INTRODUCTION

Hash Function is widely used for some cryptographic application. It receives arbitrary input and provides fixed length of the output based on the structure of hash function. The output of hash function known as hash code or message digests. There is no key for hash function. There are several different types of hash function such as MD5, SHA-1, RIPEMD-160 and others. Many researchers [3-7] have done design and implementation of these hash functions on reconfigurable hardware. Nowadays, efficient design of hash function is important for some application. Therefore, some techniques need to be taken into consideration in order to improve the performance of hash function in terms of frequency and throughput of the design. Motivation of this paper by choosing RIPEMD160 as hash function for some application is because of the structure of hash function where it considers both side left and right line for shift and message values.

2. METHODOLOGY

2.1 RIPEMD-160 Algorithm

The structure of RIPEMD-160 (Race Integrity Primitives Evaluation Message Digest) algorithm produces 160-bit length of the hash code which consists of five 32-bit words. The output of RIPEMD-160 is in little-endian format. There are 80 steps processing for five 32-bit different initial inputs. Five initial inputs H_0, H_1, H_2, H_3, and H_4 will be given to five inputs from left namely *A, B, C, D* and *E* and five inputs from right namely *A', B', C', D' and E'*. There are two parallel process of RIPEMD-160 occur during execution. There are five different non-linear function for f(B,C,D) of RIPEMD-160. There are the non-linear function for five different steps such as f_1, f_2, f_3, f_4 and f_5. In order to produce the output of RIPEMD-160 hash function, ten different constants which are five from the left and five from the right will be used in this design. Since RIPEMD-160 use two parallel lines which are from left, and right, there will be two different values for message selection, *m* and *m'*. In this case, the data needs to be kept in memory first in order to make sure that the data is already in appropriate place. Once the data already is in place, the counter will be used to call the message based on value.

2.2 RIPEMD-160 Unfolding Transformation

Unfolding design is one of the techniques that can be used to obtain a new program that performs more than one iteration of the original program. Figure 1 shows unfolding design for step function of RIPEMD-160. It consists of two non-linear function, two shift rotation over 10 positions, left circular shift with shift value. M and K represent message and constant of RIPEMD-160 respectively. In this design, only one block of step function will be used during the execution process. All the parameters will be used in this design in order to obtain the correct output. Maximum frequency of the design can be achieved by using advisors in terms of resource, timing and power. In addition, by putting register at the end of the design output port can also increase the maximum frequency as we know register-to-register delay in one of the largest delays in modern circuit design.

There are two one non-linear functions block for unfolding design that consists of five different non-linear functions. First, message will be kept in memory with input load which means if there is input load with logic'1', the data will transfer to specific length of register. Then, message will be kept in memory starting from 0 until 15 locations. By using this method, it is

easier to call the message based on message selection ordering. Counter can be used to call the message, *m ,m'* and Shift, *s, s'* value. The iteration of this architecture will be processed until 40 steps because of unfolding factor 2 design.

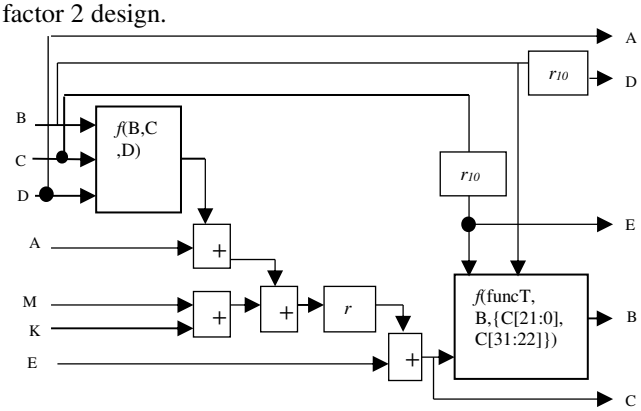

Figure 1 RIPEMD-160 unfolding transformation

3. RESULTS AND DISCUSSION

The proposed RIPEMD-160 for both iterative design and unfolding design were successfully synthesized and implemented by using Altera Quartus II on Arria II GX. The designs were written in Verilog code and simulated in functional and timing simulation using ModelSim-Altera 10.3d. Table 1 shows the synthesis and implementation results of RIPEMD-160 hash function.

Table 1 Synthesis and implementation of RIPEMD-160 hash function

Design	Device	ALUTs /CLB	Reg	FMax (MHz)	Throughput (Mbps)
Proposed RIPEMD-160	Arria II GX	1192 ALUTs	581	134.35	684.45
Proposed RIPEMD-160 unfolding	Arria II GX	1629 ALUTs	657	104.53	884.62
RIPEMD-160[4]	Xilinx Virtex 300E	1004 CLB	-	42.9	65
RIPEMD-160[5]	EPF10 K50SB C356-1			26.6	84
RIPEMD-160[6]	XC2V P30	4410 ALUTs	-	100.05	624

From this table, it shows that the throughput for unfolding design increase significantly from 684.45 Mbps to 884.62 compared with iterative design. The throughput of the design can be calculated using the following Equation (1). In addition, Quartus II timing optimization advisor can also increase the speed of the design. Besides, the usages of register can be optimized by using Quartus II resource optimization advisor.

$$\text{Throughput} = 512 \times \text{FMax} / \text{clock cycles} \quad (1)$$

Figure 2 shows the timing simulation result of RIPEMD-160 unfolding design. From this waveform, the output of hash code provides the correct result that is '8eb208f7e05d987a9b044a8e98c6b087f15a0bfc' with message input 'abc'. In this design, there are six inputs such as clk, rst, start, load, data_in and addr and one output RIPEMD-160 hash function.

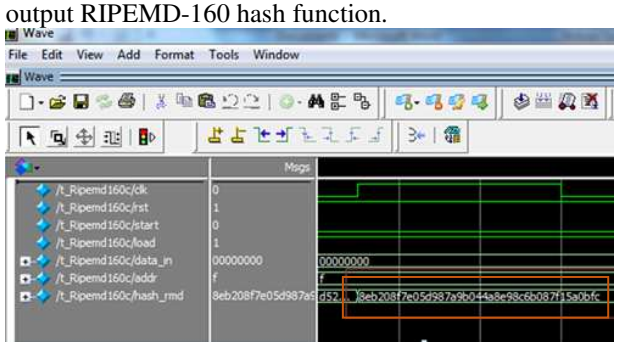

Figure 2 Timing simulation waveform of RIPEMD-160

4. CONCLUSIONS

In conclusion, both proposed designs were successfully synthesized and implemented on Quartus II Arria II GX. By using unfolding design, the throughput of the RIPEMD-160 hash function design can be improved which is about 884.62 Mbps. It is suggested for future work that resource sharing can be done by combining the similar non-linear function with other hash functions. Thus, the area implementation can be reduced.

ACKNOWLEDGEMENT

Authors are grateful to Universiti Malaysia Sarawak for providing facilities to support this research.

REFERENCES

[1] F. Rodriguez-Henriquez, N.A. Saqib, A. Diaz-Perez, C. Kaya Koc, *Cryptographic Algorithms on Reconfigurable Hardware*, Springer Series on Signals and Communication Technology, 2006, pp. 211-242.

[2] K. K. Parhi., VLSI Digital Signal Processing Systems: Design and Implementation, John Wiley & Sons, Inc. 1999, pp. 119-140,.

[3] H. Dobbertin, A. Bosselaers, B. Preneel, *RIPEMD-160, a strengthened version of RIPEMD*, Fast Software Encryption, LNCS 1039, Springer-Verlag, 1996, pp. 71-82.

[4] S. Dominikus, *A hardware implementation of MD4-family hash algorithms*, Proc. 9th Int. Conf. on Electronics, Circuits and Systems, vol. 3, 2002, pp. 1143-1146.

[5] C. Ng, T. Ng and K. Yip, *A Unified Architecture of MD5 and Ripemd-160 Hash Algorithms*, Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS '04,vol.2, ,23-26 May 2004, pp. 889-892.

[6] M. Kneˇzevi´c, K. Sakiyama, Y. K. Lee and I. Verbauwhede, *On the High-Throughput Implementation of RIPEMD-160 Hash Algorithm*, International Conference on Application-Specific Systems, Architectures and Processors,2008. ASAP 2008, 2-4 July 2008, Leuven, pp.85 – 90.