Compact ASIC Architectures for the 512-bit Hash Function Whirlpool

Takeshi Sugawara[1], Naofumi Homma[1], Takafumi Aoki[1], and Akashi Satoh[2]

[1] Graduate School of Information Sciences, Tohoku University
[2] Research Center for Information Security,
National Institute of Advanced Industrial Science and Technology
{sugawara, homma}@aoki.ecei.tohoku.ac.jp, aoki@ecei.tohoku.ac.jp,
akashi.satoh@aist.go.jp

Abstract. Compact hardware architectures are proposed for the ISO/IEC 10118-3 standard hash function Whirlpool. In order to reduce the circuit area, the 512-bit function block $\rho[k]$ for the main datapath is divided into smaller sub-blocks with 256-, 128-, or 64-bit buses, and the sub-blocks are used iteratively. Six architectures are designed by combining the three different datapath widths and two data scheduling techniques: interleave and pipeline. The six architectures in conjunction with three different types of S-box were synthesized using a 90-nm CMOS standard cell library, with two optimization options: size and speed. A total of 18 implementations were obtained, and their performances were compared with conventional designs using the same standard cell library. The highest hardware efficiency (defined by throughput per gate) of 372.3 Kbps/gate was achieved by the proposed pipeline architecture with the 256-bit datapath optimized for speed. The interleaved architecture with the 64-bit datapath optimized for size showed the smallest size of 13.6 Kgates, which requires only 46% of the resources of the conventional compact architecture.

Keywords: Hash function, Whirlpool, Hardware architecture, Cryptographic hardware

1. Introduction

Whirlpool [1, 2] is an ISO/IEC 10118-3 [3] standard 512-bit hash function based on the Miyaguchi-Preneel scheme using the SPN-type compression function W with the 512-bit round function $\rho[k]$ that is similar to the block cipher AES [4]. High-performance hardware architectures for Whirlpool were proposed and their performances were evaluated using the ASIC library described in [5]. However, the 512-bit function $\rho[k]$ requires a large amount of hardware resources, resulting in a much larger circuit area compared to other hash functions, such as SHA-256/-512 [3, 6]. Several compact architectures were also examined in [7, 8, 9], but these architectures are limited to the 64- or 8-bit datapath-width on an FPGA supporting a large built-in memory (i.e., Block RAM).

In this paper, we propose compact hardware architectures for ASIC implementations, which partition the function block $\rho[k]$ into sub-blocks with 256-, 128-, or 64-bit datapath-widths. The proposed architectures generate round keys on the fly to eliminate the requirement for memory resources to hold pre-calculated round keys. This feature is especially effective in ASIC implementations in which memory is expensive. The pipelining technique for the compact architectures is also investigated to achieve higher throughput with a smaller circuit area. In total, six hardware architectures for Whirlpool that combine two schemes (interleave and pipeline) with the three datapath-widths are designed. The corresponding throughput and gate count are then evaluated using a 90-nm CMOS standard cell library. Performance comparisons with the Whirlpool hardware using a 512-bit datapath-width [5] and the SHA-256/-512 hardware [10] synthesized using the same library are also presented.

2. The 512-bit Hash Function Whirlpool

The Whirlpool compression function W has two datapaths, and in each path, the 512-bit function ρ[k] is used 10 times, as shown in Fig. 1. One of the paths receives the 512-bit hash value H_{i-1} that was generated from 512-bit message blocks m_1~m_{i-1} in the previous cycles and then outputs ten 512-bit keys K^1~K^{10} by using ten 512-bit constants c^1~c^{10}. Another path processes the current message block m_i using the keys K^1~K^{10}. No hash value is calculated before receiving the first message block m_1, and thus 0 is assigned to H_0.

The function $\rho[k]$ consists of four 512-bit sub-functions γ, π, θ, and $\sigma[k]$, which are similar to *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*, respectively, of AES. Each sub-function treats a 512-bit block as an 8×8-byte matrix. The first function γ is a nonlinear substitution function consisting of sixty-four 8-bit S-boxes, and the S-box is defined as a combination of three types of 4-bit mini-boxes, namely, E, E^{-1}, and R. The following function π rotates each row of the matrix by 0 ~ 7 bytes. The function θ then operates matrix multiplication using the parameters shown in Fig. 1. When the individual input and output bytes of the multiplication are defined as a_{ij} and b_{ij} $(0 \le i, j \le 7)$, respectively, the eight bytes of the column $j = 0$ are calculated as

$$b_{i0} = a_{i0} \oplus (9 \otimes a_{i1}) \oplus (2 \otimes a_{i2}) \oplus (5 \otimes a_{i3}) \oplus (8 \otimes a_{i4}) \oplus a_{i5} \oplus (4 \otimes a_{i6}) \oplus a_{i7},$$

where the byte multiplication is performed over a Galois field $GF(2^8)$ defined by the following primitive polynomial:

$$p_8(x) = x^8 + x^4 + x^3 + x^2 + 1.$$

The last function $\sigma[k]$ is a simple 512-bit XOR operation with the 512-bit value k. In the function $W[H_{i-1}]$ of Fig. 1, k is given by constants $c^1 \sim c^{10}$ for the right-hand 10-round path and is given by keys $K^1 \sim K^{10}$ for the left-hand 10 rounds.

Two different implementations are available for the 8-bit input/output S-box of the function γ: (I) a simple 8-bit input/output lookup table version, and (II) the mini-box

structure shown in the leftmost part of Fig. 1. The mini-boxes E, E^{-1}, and R in (II) are implemented using 4-bit input/output lookup tables. For the pipeline architecture described later, the pipeline register is placed between the mini-boxes E, E^{-1}, and the XOR gates in (II) in order to partition the critical path of $\rho[k]$ within the S-box.

3. Compact Hardware Architectures

3.1 Data manager

The data manager is a circuit component for the function π, which performs byte-wise permutation using a series of shift registers and selectors. Reference [8] used a simple data manager that processes one 8-byte row of the 8×8-byte matrix every cycle. In order to increase throughput using a compact circuit, we extend the data manager to

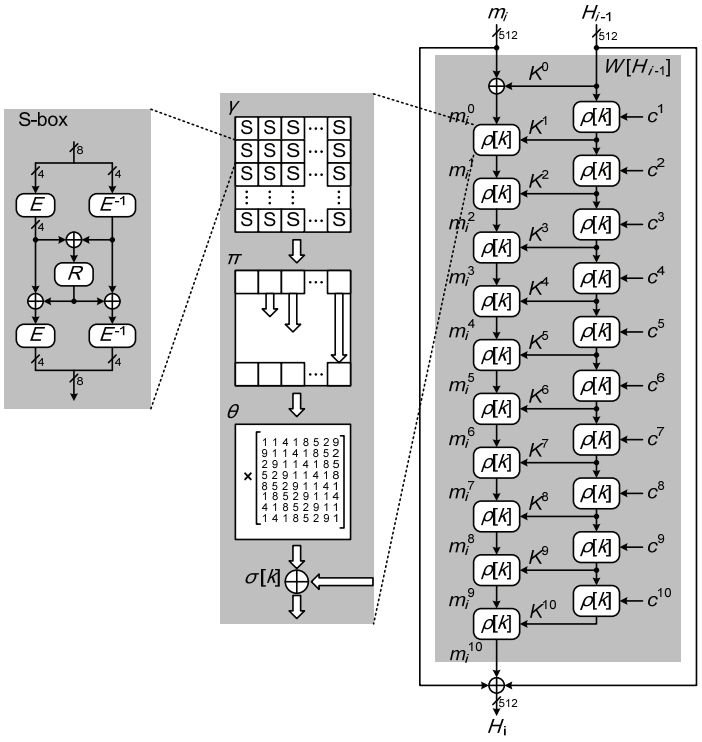

Fig. 1. Whirlpool algorithm

128-bit and 256-bit datapath widths, which have four and two pipeline stages, respectively.

Let us denote the byte permutation of the function π as shown in Fig. 2, and its straightforward implementations are shown in Fig. 3. The function block π is implemented by metal wire interconnection and no transistor device is used, although a large number of selectors and long data buses in the feedback path have an adverse impact on hardware performance with respect to size and speed. In order to settle this problem, the four-stage (128-bit datapath) and two-stage (256-bit datapath) data managers shown in Figs. 4 and 5, respectively, are introduced herein. In these figures, the square boxes denote 8-bit registers. The four-stage data manager receives 16 bytes (128 bits) corresponding to the two rows of the left-hand matrix in Fig. 2 and outputs 16 bytes for the two rows of the right-hand matrix during every cycle. For example, the first 16 input bytes for the data manager are {0, 1, 8, 9, 16, 17, 24, 25, 32, 33, 40, 41, 48, 49, 56, 57}, and the 16 bytes {0, 1, 15, 8, 22, 23, 29, 30, 36, 37, 43, 44, 50, 51, 57, 58} are output after four clock cycles by controlling registers and selectors. The proposed data manager uses only 12 bytes (96 bits) of two-way selectors, whereas the

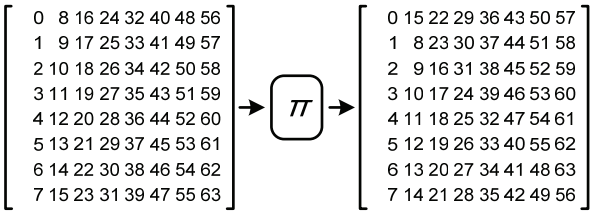

Fig. 2. Input/output relationship of the function π

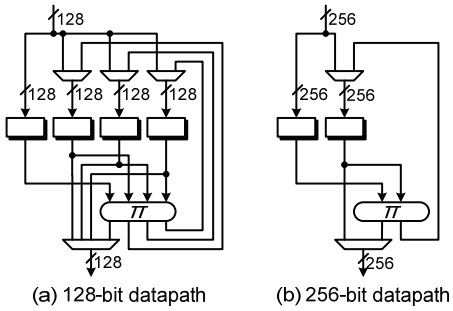

Fig. 3. Straightforward implementations for data managers using 128- and 256-bit datapaths

straightforward implementation in Fig. 3(a) requires 96 bytes (768 bits) of equivalent two-way selectors. By reducing the number of registers, the critical path is shortened, and thus, the new data manager provides a smaller circuit area with a higher operating frequency in comparison to the conventional schemes. The two-stage data manager processes four rows of Fig. 2 simultaneously using 16-byte (128-bit) selectors, while the straightforward implementation in Fig. 3(b) requires four times the number of selectors.

3.2 Datapath Architectures

The compact Whirlpool hardware architectures using the new data managers are

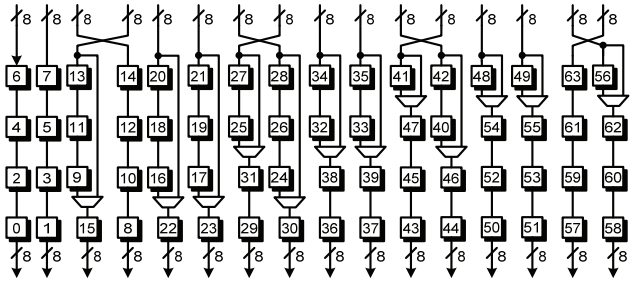

Fig. 4. Four-stage data manager

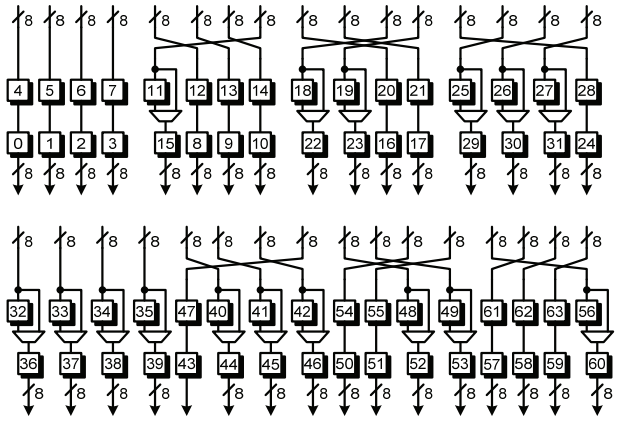

Fig. 5. Two-stage data manager

described in this Section. The 128-bit interleave and pipeline architectures based on the four-stage data manager are described in the following subsections. Other architectures for the two- and eight-stage data managers with a 256- and 64-bit datapath-width, respectively, are depicted in Appendix (Figs. 11~13). The structure of the interleave architectures for the three data managers are all the same except for the size of operating units. In contrast, the pipeline architectures of Figs. 8, 12, and 13 have different numbers of pipeline stages, because the number of operating cycles varies with the datapath width.

3.2.1 Interleave Architecture

The datapath of the 128-bit interleave architecture are shown in Fig. 6. Two four-stage data managers are used for the architecture. A 512-bit message block or a 512-bit key are stored to a 512-bit shift register in each data manager using four clock cycles. There is only one $\gamma\theta$-function block, and it is used alternatively for data randomization and key scheduling every four clock cycles. In the architecture, the order of the substitution function γ and the permutation function π of the data manager are reversed from the original order in Fig. 1 so that the datapath and sequencer logic are simplified. This has no effect on the operations in data randomization, but the data

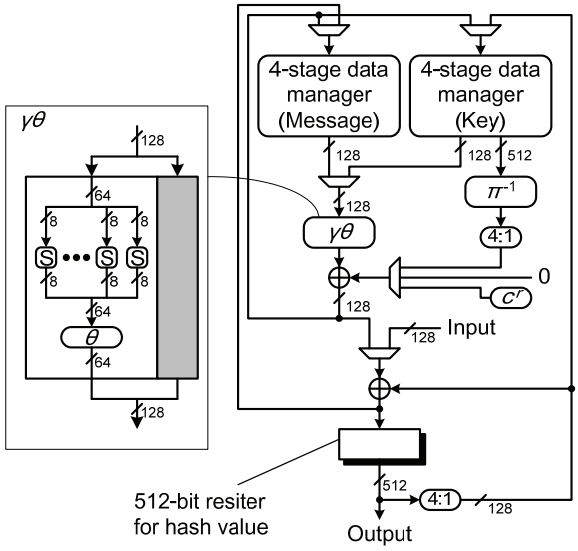

Fig. 6. 128-bit interleave architecture

manager for the key scheduling performs the function π before the key is used in $\sigma[k]$, and thus the inverse function π^{-1} should be applied to the key for adjustment. Fortunately, the function π^{-1} is implemented as rewiring in an ASIC chip in the same manner as the function π and no additional selectors are required in this case. Therefore, the function π^{-1} has no impact on hardware resources. Fig. 7 shows an example operation of the architecture. The figure depicts the architecture processing two $\rho[k]$ functions taking eight cycles. Therefore, four cycles are required to perform the function $\rho[k]$, and thus eight cycles are required for each round of the compression function W. The function W uses 10 rounds to process one 512-bit message block, and an additional four clock cycles is needed for data I/O. As a result, the interleave architecture with the 128-bit datapath and the four-stage data manager requires 84 (= 4 × 2 × 10 + 4) cycles for each message block. In a similar manner, the interleave architecture with 256- and 64-bit datapaths (two- and eight-stage data managers) in Figs. 10 require 42 (= 2 × 2 × 10 + 2) and 168 (= 8 × 2 × 10 + 8) clock cycles, respectively.

3.3.3 Pipeline Architecture

Pipeline architecture divides the functions γ and θ by inserting pipeline registers to

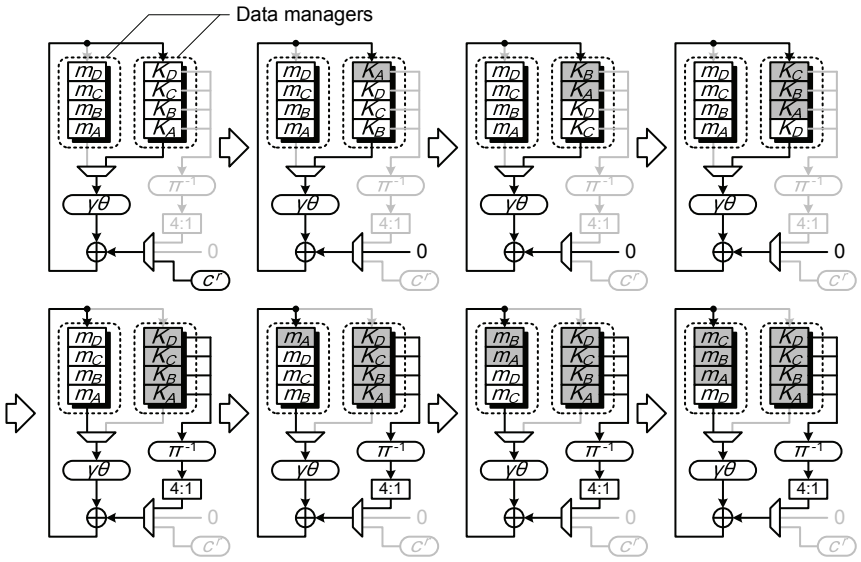

Fig. 7. Example operation of the 128-bit interleave architecture

shorten the critical paths and improve the operation frequency. In the 128-bit datapath architecture, the number of pipeline stages can be increased up to five without causing pipeline stall. The upper limits of the number of pipeline stages are three and nine for the 256-bit and 64-bit versions, respectively. The maximum numbers of stages are used for the performance comparison in the next section.

The datapath and operation of the 128-bit pipeline architecture are shown in Figs. 8 and 9, respectively. The functions γ and θ are partitioned into four sub-blocks as stages 0 ~ 3, and the XOR gates for key addition $\sigma[k]$ followed by selectors correspond to the final stage (stage 5). The partitioned sub-blocks perform message randomization and key scheduling simultaneously, as shown in Fig. 9. The data manager is only used for the message randomization, and the key scheduler uses the 512-bit register with a 4:1 selector. This is because the dependency between key bytes cannot be satisfied, even when using the function π^{-1} as in the interleave architecture. This 128-bit architecture performs two $\rho[k]$ operations in eight cycles and thus requires 80 cycles for the function W, which is the same as the 128-bit interleave

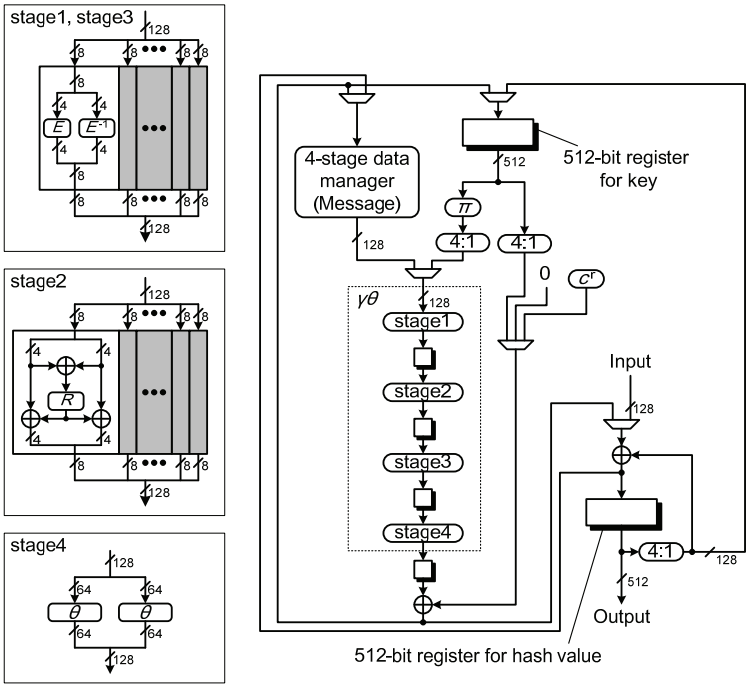

Fig. 8. 128-bit pipeline architecture

architecture. In addition to the 80 cycles, four cycles for data I/O and other four cycles to empty the series of pipeline registers are required. Therefore, one 512-bit message block is compressed in 88 (= 80 + 4 + 4) cycles. Similarly, the 256- and 64-bit versions require 44 (= 40 + 2 + 2) and 176 (= 160 + 8 + 8) cycles, respectively.

4. Performance Evaluation

The proposed Whirlpool architectures were designed in Verilog-HDL and were synthesized by Synopsys Design Compiler (version Y-2006.06-SP2) with the STMicroelectronics 90-nm CMOS standard cell library (1.2-volt version) [11], where two optimization options, size and speed, were specified. Hardware sizes were estimated based on a two-way NAND equivalent gate, and the speeds were evaluated under worst-case conditions. The efficiency is defined as the throughput per gate, and thus higher efficiency indicates better implementation. For performance comparison, the Whirlpool circuits with 512-bit datapath architecture from [5] and the SHA-256/-512 circuits proposed in [10] were also designed and evaluated using the same library.

The synthesis results are shown in Table 1, where the two types of S-box described in Section 2 are referred to as the $GF(2^8)$ and $GF(2^4)$ tables. Only the $GF(2^4)$ table is

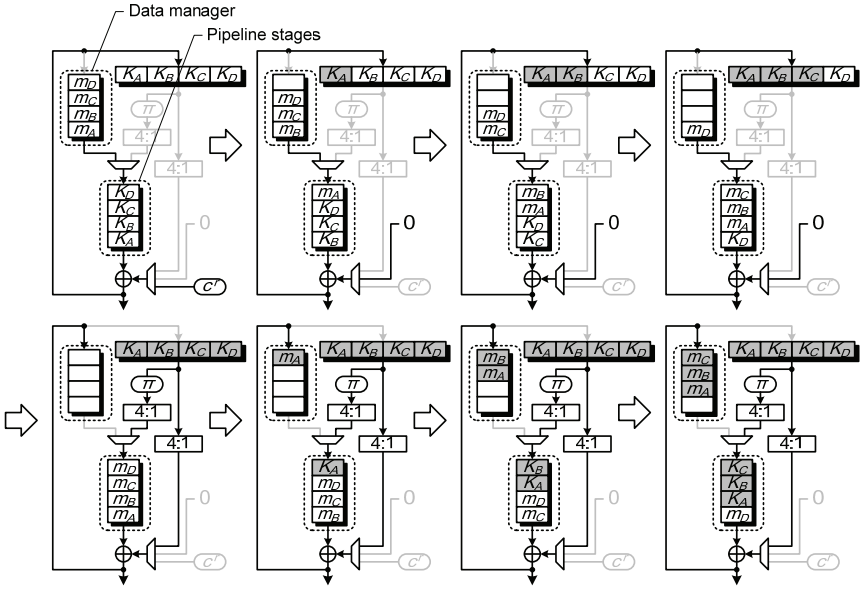

Fig. 9. Example operation of 128-bit pipeline architecture

used for the pipeline architectures so that the pipeline register can be placed in the middle of the S-box. The results are also displayed in Fig. 10 where the horizontal and vertical axes are gate count and throughput. Note that the largest implementation with 179.0 Kgates is outside of this graph. In the figure, the circuit is smaller when the corresponding dot is located in the left region, and is faster when the dot is in the upper region. As a result, implementations plotted at the upper left region of the graph have better efficiency.

The interleave architecture with a wider datapath including the conventional datapath always obtained higher efficiency. This is because throughput is halved if the datapath width is halved, whereas the hardware size cannot be halved due to the constant size of the data registers. In contrast, the 256-bit datapath achieved higher

Table 1. Performance comparison in the 90-nm CMOS standard cell library

Design	Algo -rithm	Message Block (bits)	Cycle	S-box	Datapath architecture	Optimize	Area (gates)	Operating Frequency (MHz)	Through-put (Mbps)	Efficiency (Kbps/ gate)
This work	Whirl -pool	512	42	$GF(2^8)$ Table	256-bit Interleave	Area	36,201	266.67	3,251	89.80
						Speed	64,796	568.18	6,926	106.90
				$GF(2^4)$ Table		Area	21,495	266.67	3,251	151.23
						Speed	42,972	529.10	6,450	150.10
			84	$GF(2^8)$ Table	128-bit Interleave	Area	22,527	269.54	1,643	72.93
						Speed	37,865	571.43	3,483	91.98
				$GF(2^4)$ Table		Area	15,891	268.10	1,634	102.83
						Speed	28,337	537.63	3,277	115.64
			168	$GF(2^8)$ Table	64-bit Interleave	Area	16,675	268.10	817	49.00
						Speed	24,029	568.18	1,732	72.06
				$GF(2^4)$ Table		Area	13,614	268.10	817	60.02
						Speed	20,554	546.45	1,665	81.02
			44	$GF(2^4)$ Table	256-bit Pipeline	Area	21,395	558.66	6,501	303.84
						Speed	35,520	1,136.36	13,223	372.27
			88		128-bit Pipeline	Area	16,677	574.71	3,344	200.50
						Speed	23,230	1,111.11	6,465	278.29
			176		64-bit Pipeline	Area	14,762	564.97	1,644	111.34
						Speed	19,500	1,041.67	3,030	155.40
[5]	Whirl -pool	512	10	$GF(2^8)$ Table	512-bit Parallel	Area	103,633	211.42	10,825	104.45
						Speed	179,035	546.45	27,978	156.27
				$GF(2^4)$ Table		Area	43,726	210.97	10,802	247.03
						Speed	103,408	523.56	26,806	259.23
			21	$GF(2^8)$ Table	512-bit Interleave	Area	58,792	210.08	5,122	87.12
						Speed	97,541	518.13	12,633	129.51
				$GF(2^4)$ Table		Area	29,577	210.08	5,122	173.18
						Speed	64,549	500.00	12,190	188.86
				$GF(2^8)$ Table	512-bit Pipeline (A)	Area	60,066	364.96	8,898	148.14
						Speed	77,312	671.14	16,363	211.65
				$GF(2^4)$ Table		Area	31,938	363.64	8,866	277.59
						Speed	40,476	564.97	13,775	340.31
				$GF(2^4)$ Table	512-bit Pipeline (B)	Area	30,105	363.64	8,866	294.50
						Speed	40,330	574.71	14,012	347.43
[10]	SHA -256	512	72			Area	9,764	362.32	2,576	263.88
						Speed	13,624	490.20	3,486	255.86
	SHA -512	1024	88			Area	17,104	209.64	2,439	142.63
						Speed	27,239	404.86	4,711	172.95

efficiency than 512-bit version for the pipeline architecture. In this case, the proposed deep pipeline scheme allows a much higher operating frequency, and consequently the high throughput with the small circuit resulted in a higher efficiency. However, the operating frequencies of the 128-bit (five-stage) and 64-bit (nine-stage) pipeline architectures were not improved compared with that of the 256-bit architecture, even though the number of pipeline stages was increased. The major reason for this is the increasing additional selectors in the critical path from the key register to the data manager through key addition. Therefore, we must consider the hardware resources and the signal delay time caused by additional selectors for optimizing datapath in deep pipeline operation.

The 64-bit interleave architecture in conjunction with the $GF(2^4)$ S-box and the area optimization option achieved the smallest size of 13.6 Kgates with a throughput of 817 Mbps. The circuits using the $GF(2^4)$ S-box are smaller and have higher efficiency than those using the $GF(2^8)$ S-box. The $GF(2^8)$ S-box leads to a large circuit but is still suitable for high-speed implementation. Generally, the interleave architecture is smaller than the pipeline architecture which requires a number of pipeline registers. The smallest circuit based on the conventional scheme is 29.6 Kgates for the 512-bit pipeline architecture with the $GF(2^4)$ S-box. Therefore, the gate count of 13.6 Kgates obtained by the proposed interleave architecture is 54% smaller than that of the conventional scheme. The 256-bit pipeline version optimized for

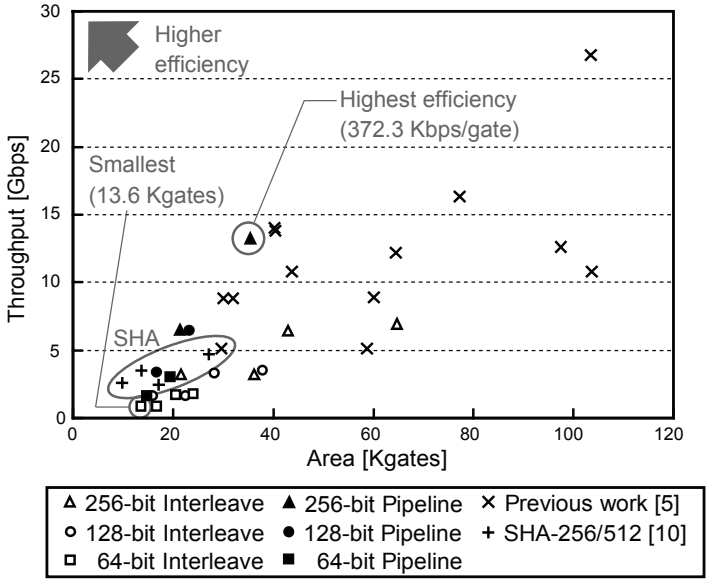

Fig. 10. Throughput versus area for each implementation

speed achieved the highest efficiency of 372.3 Kbps/gate (= 13.2 Gbps/35.5 Kgates) among the Whirlpool implementations. This means that the pipeline architecture provides the optimal balance between speed and size.

In comparison with the SHA-256 and -512 circuits, the smallest Whirlpool circuit is larger than the area-optimized SHA-256 circuit with 9.8 Kgates but is smaller than the SHA-512 circuit with 17.1 Kgates. The highest throughput of the proposed architectures (13.2 Gbps) is 2.8 times higher than that (4.7 Gbps) of the speed-optimized SHA-512, and the highest efficiency of 372.3 Kbps/gate is 1.4 times higher than 263.8 Kbps/gate of the area-optimized SHA-256. The proposed Whirlpool architectures also achieved a wide variety of size and speed performances, while the SHA-256 and -512 circuits have only four implementations. Consequently, the proposed Whirlpool hardware has great advantages in both performance and flexibility.

5. Conclusion

In the present paper, compact hardware architectures with interleave and pipeline schemes were proposed for the 512-bit hash function Whirlpool, and their performances were evaluated using a 90-nm CMOS standard cell library. The fastest throughput of 13.2 Gbps @ 35.5 Kgates, the smallest circuit area of 13.6 Kgates @ 817 Mbps, and the highest efficiency of 372.3 Kbps/gate were then obtained using the proposed architectures. These results indicate that the proposed architectures can provide higher performance with respect to both size and efficiency, as compared to the conventional 512-bit architectures. In addition to the peak performance in size and speed, the flexibility of the proposed architectures enables various design options to meet a variety of application requirements.

Further research to reduce the overhead of additional selectors in the pipeline architectures is currently being conducted. The method will further improve both size and speed.

References

1. “The Whirlpool Hash Function,” http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html.
2. P. Barreto and V. Rijmen, “The Whirlpool Hash Function,” http://planeta.terra.om.br/informatica/paulobarreto/whirlpool.zip.
3. ISO/IEC 10118-3:2004, "Information technology -- Security techniques -- Hash-functions -- Part 3: Dedicated hash-functions," http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39876
4. NIST, “Advanced Encryption Standard (AES) FIPS Publication 197,” Nov. 2001. http://csrc.nist.gov/ publications/fips/fips197/ fips-197.pdf.
5. A. Satoh, “ASIC Hardware Implementations for 512-Bit Hash Function Whirlpool,” Proceedings ISCAS2008, pp. 2917--2920, May 2008.
6. NIST, "Secure Hash Standard (SHS)," FIPS PUB 180-2, Aug. 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.

7. N. Pramstaller, C. Rechberger, and V. Rijmen, “A Compact FPGA Implementation of the Hash Function Whirlpool,” Proceedings of the 2006 ACM/SIGDA, pp. 159 – 166, 2006.
8. M. McLoone and C. McIvor, “High-speed & Low Area Hardware Architectures of the Whirlpool Hash Function,” J. VLSI Signal Processing, vol. 47, no. 1, pp. 47-57, Apr. 2007.
9. T. Alho, P. Hämäläinen, M. Hännikäinen, and T. Hämäläinen, “Compact Hardware Design of Whirlpool Hashing Core,” Proceedings of the DATE ‘07, pp. 1247 - 1252, Apr. 2007.
10. A. Satoh and T. Inoue, “ASIC-hardware-focused comparison for hash functions MD5, RIPEMD-160, and SHS,” Integration, the VLSI Journal, Vol. 40, no. 1, pp. 3-10, Jan. 2007.
11. Circuits Multi-Projets (CMP), CMOS 90 nm (CMOS090) from STMicroelectronics, http://cmp.imag.fr/products/ic/?p=STCMOS090.

Appendix

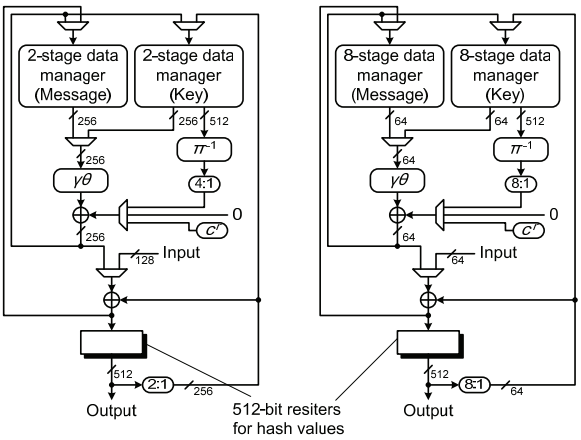

Fig. 11. 256-bit (left) and 64-bit (right) interleave architectures

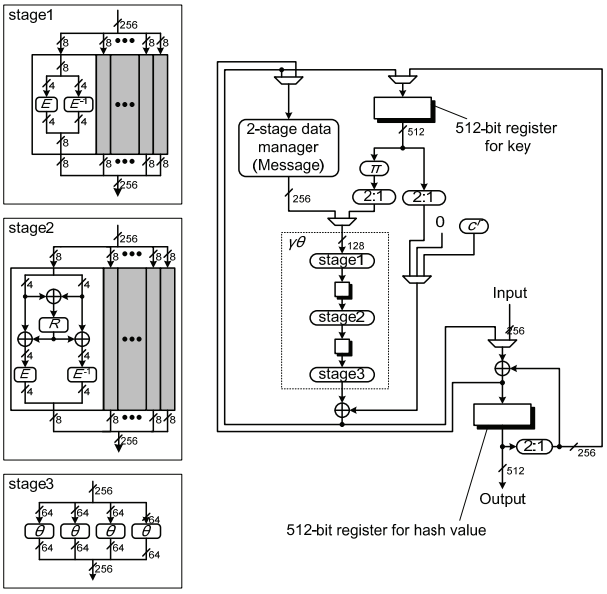

Fig. 12. 256-bit pipeline architecture

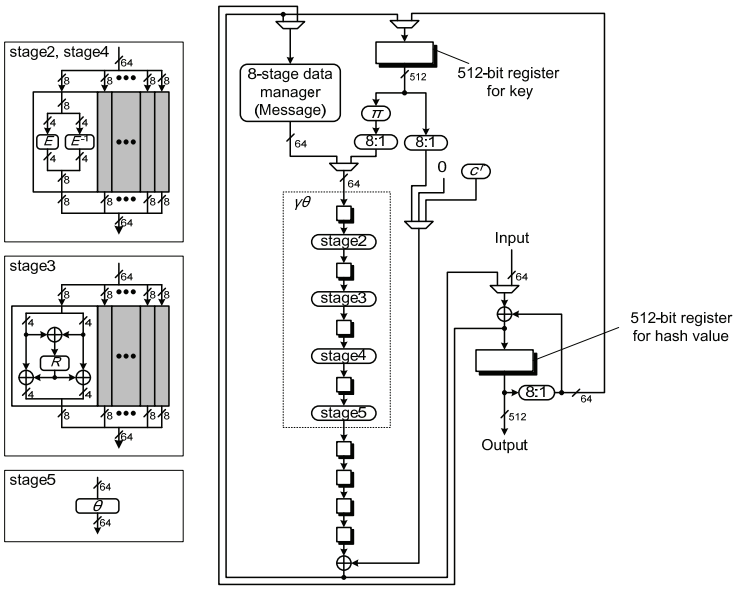

Fig. 13. 64-bit pipeline architecture