# A High-Throughput Processor for Cryptographic Hash Functions

Yuanhong Huo and Dake Liu
Beijing Institute of Technology, Beijing 100081, China
Email: {hyh, dake}@bit.edu.cn

***Abstract*** —This paper presents a high throughput Application-Specific Instruction-set Processor (ASIP) for cryptographic hash functions. The processor is obtained via hardware/software co-design methodology and accelerates SHA (Secure Hash Algorithm) and MD5 hash functions. The proposed design occupies 0.28 $mm^2$ (66 kgates) in 65 nm CMOS process including 4.5 KB single port memory and 52 kgates logic. The throughput of the proposed design reaches 15.8 Gb/s, 12.5 Gb/s, 12.2 Gb/s, and 19.9 Gb/s for MD5, SHA-1, SHA-512, and SHA3-512, respectively under the clock frequency of 1.0 GHz. The proposed design is evaluated with state-of-the-art VLSI designs, which reveals its high performance, low silicon cost, and full programmability.

***Index Terms***—ASIP, Secure Hash Algorithm, MD5, VLSI (very large scale integration)

## I. Introduction

Cryptographic hash functions play an important role in network security protocols and infrastructures, such as TLS, SSL, SET, IPSec, and PKI [1]. Hash functions can be used to verify integrity of data in transit. Hash functions can also be used as message authentication codes, e.g., in the case of the Hash Message Authentication Code (HMAC) [1]. The cryptographic hash functions MD5 (Message Digest Algorithm 5), SHA-1 (Secure Hash Algorithm 1), and SHA-2 (Secure Hash Algorithm 2) algorithms are widely adopted nowadays. The SHA-3 (Secure Hash Algorithm 3) standard has been released by NIST on August 5, 2015. In the next few years, it is expected that the SHA-3 will become a mandatory or optional cryptographic hash algorithm for all mainstream and future network security protocols and standards [1].

Circuits for MD5 and SHA cryptographic hash functions should, in general, support multiple algorithms with high performance. On the one hand, according to specific protocols proposed for different applications, there are requirements to adopt the hash algorithms of different security level [2]. On the other hand, MD5 and SHA workloads are among the most performance/power-critical workloads due to the iterative nature of hash computation and the high computational complexity. The increasing speeds for wired and wireless data networks require high-throughput hardware implementations of the cryptographic hash algorithms so as to meet the required high performance. The IPv6 network stack and its mandatory IPSec security protocols will push further the requirement for high performance implementations [1].

Several hardware techniques have been adopted to accelerate single or multiple cryptographic hash functions. The techniques include the use of parallel counters and Carry Save Adders (CSA) [3]-[5], loop unrolling to mitigate the serial dependence of hash computation [6], delay balancing [4], embedded memories [7], and pipelining [1]-[4], [8], [9]. These techniques require significant extra hardware resulting in higher area. Up to now, there have been successful VLSI designs for multiple hash functions resulting in significant area savings via hardware sharing. For example, Cao *et al*. [2] and Wang *et al*. [10] propose reconfigurable hardware designs for SHA-1 and MD5 hash algorithms. Ramanarayanan *et al*. [11], Michail *et al*. [9], and Chaves *et al*. [5] present SHA accelerators for five SHA algorithms (i.e., SHA-1/224/256/384/512). However, these designs are implemented with ASICs/FPGAs targeting a small predefined set of hash algorithms.

This paper presents a hash processor (HP-ASIP) for MD5, SHA-1, SHA-2, and SHA-3. Table I lists functions implemented in this paper. HP-ASIP is obtained via hardware/software co-design and achieves ASIC-like performance and full programmability with area consumption of 0.28 $mm^2$ (65 nm). Thanks to its programmability, HP-ASIP can offer changes to the implemented algorithms via software programming when one of them is cracked to extend chip lifetime.

TABLE I: FUNCTIONS IMPLEMENTED IN THIS WORK

| Function | Output size | Security strengths in bits | | |
|---|---|---|---|---|
| | | Collision | Preimage | 2nd preimage |
| MD5 | 128 | < 64 | NA | NA |
| SHA-1 | 160 | < 80 | 160 | 105-160 |
| SHA-224 | 224 | 112 | 224 | 201-224 |
| SHA-512/224 | 224 | 112 | 224 | 224 |
| SHA-256 | 256 | 128 | 256 | 201-256 |
| SHA-512/256 | 256 | 128 | 256 | 256 |
| SHA-384 | 384 | 192 | 384 | 384 |
| SHA-512 | 512 | 256 | 512 | 394-512 |
| SHA3-224 | 224 | 112 | 224 | 224 |
| SHA3-256 | 256 | 128 | 256 | 256 |
| SHA3-384 | 384 | 192 | 384 | 384 |
| SHA3-512 | 512 | 256 | 512 | 512 |

Manuscript received May 25, 2016; revised July 18, 2016.
This work was supported by the National High Technical Research and Development Program of China (863 program) 2014AA01A705.
Corresponding author email: dake@bit.edu.cn.
doi:10.12720/jcm.11.7.702-709

The rest of this paper is organized as follows. Section II provides the design of HP-ASIP. Section III presents the top-level architecture, the datapath, the memory subsystem, the pipeline scheduling, and the instruction set of HP-ASIP. Section IV describes the area and power consumption of HP-ASIP. Section V evaluates HP-ASIP. Section VI concludes the paper.

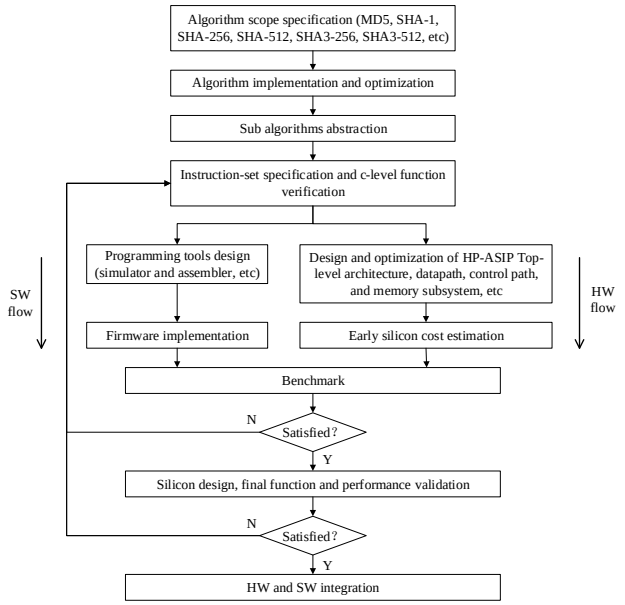



Fig. 1. Design flow of HP-ASIP.

## II. Design of HP-ASIP

We propose a methodology to design HP-ASIP. Fig. 1 depicts the design flow. The design flow, a hardware/software (HW/SW) co-design flow, optimizes the partition of HW and SW functions cooperatively during the design of HP-ASIP. Firstly, the scope of algorithms (e.g., MD5, SHA-1, SHA-256, SHA-512, and SHA3-512, etc), throughput, etc are specified according to application requirements. Then, a datapath, a memory subsystem, and a processor architecture are designed to accelerate the algorithms of the scope.

### A. Design and Optimization of HP-ASIP

The datapath of HP-ASIP is first designed as other parts of HP-ASIP can be designed only when the datapath is fixed. There are multi-mode hash accelerators for SHA [11] and MD5 [2]. In this work, firstly, a preliminary datapath for SHA-1/224/256/384/512 is proposed according to [11]. The datapath can process two independent data streams in parallel when performing SHA-1/224/256. Then, we map one step of MD5 [2] onto the datapath optimizing the degree of hardware sharing between MD5 and SHA-1/224/256. At last, we map one round of SHA-3 [1] onto the datapath incrementally.

Through our research, the critical path of the datapath for SHA-3 is much shorter than the critical path of the datapath for SHA-1/224/256/384/512 and MD5. To achieve high clock frequency, we optimize the datapath into two pipeline stages. Taking the datapath for SHA-1 (Fig. 2(a)) and the datapath for SHA-256 (Fig. 2(b)) as examples, we explain how to optimize the datapath into two pipeline stages.

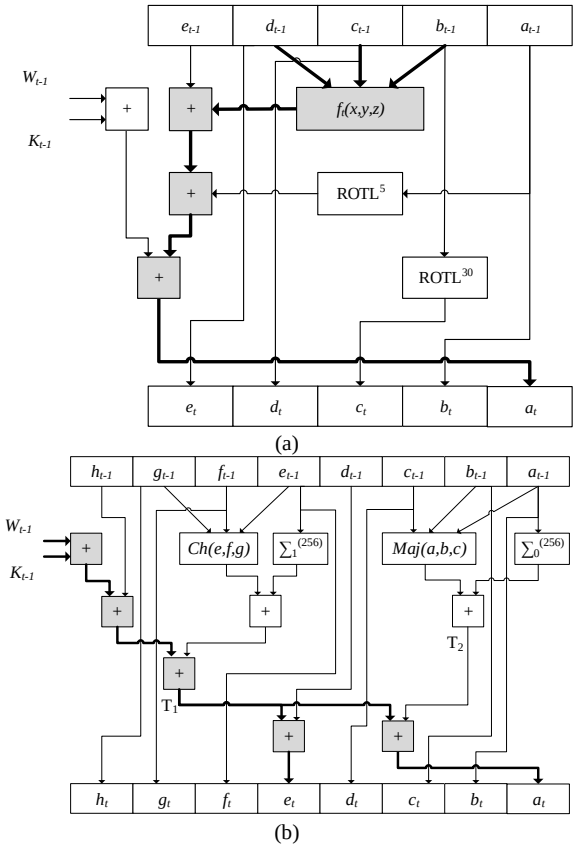
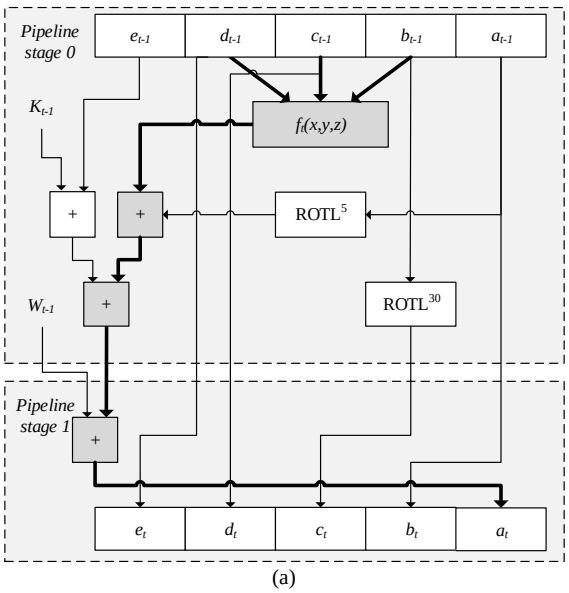
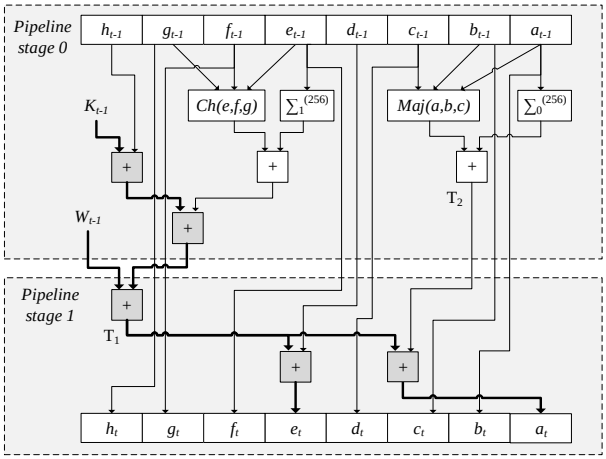



Fig. 2. Fundamental transformation rounds: SHA-1(a) and SHA-256(b).

Firstly, we identify the critical path of the datapath for SHA-1 and SHA-256, respectively as shown in Fig. 2. Then, we optimally implement the operations of the critical path into two pipeline stages as shown in Fig. 3. Fig. 3(a) describes the implementation of SHA-1 on HP-ASIP and Fig. 3(b) describes the acceleration of SHA-256 on HP-ASIP. The rest of algorithms implemented in this paper are optimally implemented in a similar way.






(b)

Fig. 3. Optimized transformation rounds: SHA-1(a) and SHA-256(b).

TABLE II: SIZE OF PARAMETERS FOR MD5 AND SHA FUNCTIONS

| Algorithm | Internal state size | Message block size | Iteration constants | Iterations |
|---|---|---|---|---|
| MD5 | 4×32b | 16×32b | 64×32b | 64 |
| SHA-1 | 5×32b | 16×32b | 0 | 80 |
| SHA-224 | 8×32b | 16×32b | 64×32b | 64 |
| SHA-256 | 8×32b | 16×32b | 64×32b | 64 |
| SHA-384 | 8×64b | 16×64b | 80×64b | 80 |
| SHA-512 | 8×64b | 16×64b | 80×64b | 80 |
| SHA-512/224 | 8×64b | 16×64b | 80×64b | 80 |
| SHA-512/256 | 8×64b | 16×64b | 80×64b | 80 |
| SHA3-224 | 25×64b | 18×64b | 24×64b | 24 |
| SHA3-256 | 25×64b | 17×64b | 24×64b | 24 |
| SHA3-384 | 25×64b | 13×64b | 24×64b | 24 |
| SHA3-512 | 25×64b | 9×64b | 24×64b | 24 |

In this design, SHA-3 requires only one pipeline stage of the datapath while SHA-1/224/256/384/512 and MD5 need two pipeline stages. To fully adopt the two-stage pipelined datapath, we introduce odd and even register contexts for hash values and message schedulers. Firstly, we analyze the parameters of the targeted algorithms. Table II lists the size of the internal state, the size of each message block, the size of iteration constants, and the iterations of the targeted algorithms. The maximum size of internal state for MD5 and SHA-1/224/256/384/512 is 512b and the maximum size of message block for MD5 and SHA-1/224/256/384/512 is 1024b. We thus introduce 512b register and 1024b register for both the odd and the even register contexts (to be discussed). When performing SHA-3, we adopt 1600b of the odd and even register contexts for the internal state of SHA-3.

In this work, we introduce data memory (to be discussed) for the iteration constant ($K_t$) [11] and the index for the message schedule ($W_t$) of the targeted algorithms so that we can adopt them via software programming when performing MD5 and SHA message digest computation. Besides, we propose instructions to adopt the odd and even register contexts (to be discussed). HP-ASIP can thus process two independent data streams simultaneously when performing SHA-384/512 and process four independent data streams simultaneously when performing MD5 and SHA-1/224/256.

In this design, common operations among the targeted algorithms are implemented by shared functional blocks to achieve low silicon cost. After mapping all the algorithms targeted in this paper onto the datapath incrementally, we fix the datapath. Then, we extract and represent the control signals for the processing routines in the datapath by a group of control indications and propose a specific instruction set for the targeted algorithms. As the rounds of message digest computation for SHA-1 and MD5 can be divided into 4 parts [2], we introduce 4 instructions to fulfill the hash computation of SHA-1 and MD5, respectively (to be discussed).

Afterwards, algorithm pseudocode for the algorithms targeted in this paper are developed adopting the instruction set. We extract the addressing and control information of the algorithm pseudocode and propose a specific memory subsystem, a control path, and a top-level architecture for HP-ASIP. Based on the specified functional blocks and the instruction set, we develop the RTL (Register Transfer Level) description of HP-ASIP. Then, the correctness and performance of the functional design and the silicon layout are verified. Based on the algorithm pseudocode, we develop the assembly codes of the algorithms, offering the support of all algorithms targeted in this paper. At last, the hardware (assembly instruction set) and software (assembly codes) are integrated and HP-ASIP is thus designed.

As shown in Fig. 1, the design and optimization flow of HP-ASIP is recursive. Any previously mentioned essential requirements not fulfilled may cause a huge work of redesign. Adopting the HW/SW co-design methodology, we ensure that HP-ASIP can achieve low silicon cost via optimizing the degree of hardware sharing among the targeted algorithms. This method results in an ASIP for the targeted cryptographic hash algorithms satisfying all the previously mentioned essential requirements.

### *B. Data Block Expansion for SHA Function*

The SHA-1 algorithm computation steps described in Fig. 2(a) are performed 80 times (rounds). Each round adopts a 32-bit word obtained from the current input data block. As each input data block only contains 16 32-bit words (512 bits), we need to obtain the remaining 64 32-bit words via data expansion. The data expansion is performed via the computation described in (1), where $M_t^{(i)}$ denotes the first 16 32-bit words of the $i$ th data block.

$$W_t = \begin{cases} M_t^{(i)}, & 0 \le t \le 15 \\ RotL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}), & 16 \le t \le 79 \end{cases} \quad (1)$$

For the SHA-2 algorithm, the computation steps shown in Fig. 2(b) are performed for 64 rounds (80 rounds for SHA-512). In each round, a 32-bit word (64-bit for SHA-512) from the current input data block is adopted. As the input data block only contains 16 32-bit words (64-bit for SHA-512), we need to expand the initial data block to

obtain the remaining words. The expansion is performed via the computation described in (2), where $M_t^{(i)}$ denotes the first 16 words of the $i$ th data block and the operator describes the arithmetic addition operation.

$$W_t = \begin{cases} M_t^{(i)}, & 0 \le t \le 15 \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}, & 16 \le t \le 63\{or 79\} \end{cases} \quad (2)$$

For efficiency reasons, this work accelerates data block expansion in hardware. Taking SHA-256 as an example, we expand the 512 bits of each data block in hardware. The input data block expansion described in (2), can be implemented with registers and ADD operations. The output value is selected between the original data block (for the first 16 rounds) and the computed values (for the remaining rounds). Fig. 4 depicts the implemented structure for SHA-256. As the datapath for MD5 and SHA-1/224/256/384/512 is two-stage pipelined, we implement the data block expansion adopting two pipeline stages so that the datapth and the data block expansion circuit can work synchronously.

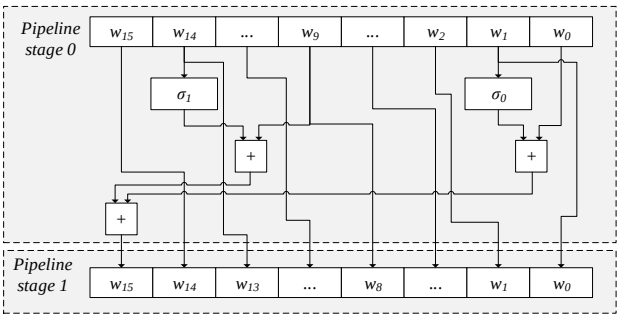



Fig. 4. Data block expansion of SHA-256.

### C. Message Padding

To ensure that the input data block is a multiple of 512 bits as required by the MD5 and SHA-1/224/256 specifications (1024 bits for SHA-384/512, etc), the original message needs to be padded. Taking the padding procedure for a 512-bit input data block as an example, it is performed as follows: for an original message composed of *n* bits, the bit "1" is appended at the end of the message, followed by *k* zero bits, were *k* is the smallest solution to the equation $n+1+k \equiv 448 \bmod 512$. The last 64 bits of the 512-bit input data block are filled with the binary representation of *n*. For the SHA-512 message padding, 1024-bit data blocks are utilized and the last 128, not 64 bits, are reserved for the binary value of the original message [5]. The message padding operations can be efficiently implemented in software.

## III. The Proposed Processor

This work adopts SIMD (Single Instruction Multiple Data) architecture to meet the requirements on the computational complexity. A two-stage pipelined datapath and a SIMD instruction set are proposed for the targeted algorithms. Multiple memory banks are designed to fulfill the bandwidth requirements of the datapath. We also introduce a variable depth pipeline to approach the efficiency limit.

### A. Top-level Architecture

The top-level architecture (Fig. 5) of HP-ASIP is made up of three parts: control logic, memory subsystem, and datapath. The control logic includes PC FSM, PM (program memory), ID (instruction decoder), DMA, and status registers. The control logic reads an instruction from the PM, a 256 × 80b SRAM, per clock cycle and decodes the machine code (i.e., the instruction fetched) into control signals. The control logic also performs loop acceleration. The memory subsystem is composed of AGU, RPN, WPN, and DM (data memory). The AGU generates addresses for operands according to the machine code. Then, the addresses generated will be passed to the DM. The DM contains four memory blocks. Each memory block contains 16 32 × 8b SRAMs and can provide 16-byte data per clock cycle. The outputs of these memory blocks will be passed to the RPN and then to the datapath. The datapath accelerates the hash algorithms implemented in this paper. The outputs of the datapath will be passed to the WPN. The RPN and WPN are introduced for data shuffling to ensure that the vector data can be accessed in parallel without access conflict. The outputs of the WPN will be written to a memory block.

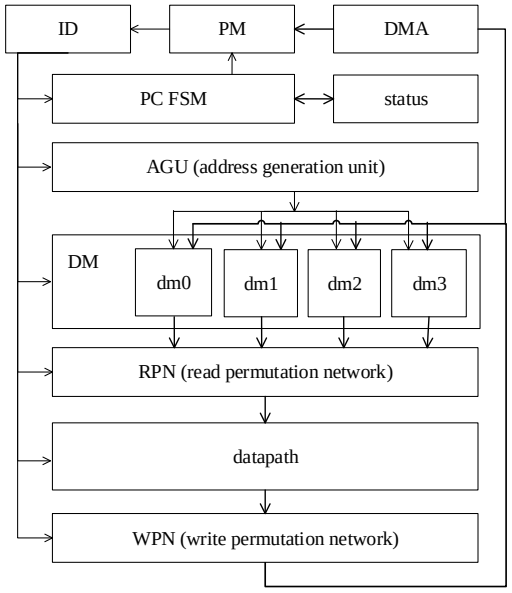



Fig. 5. Top-level architecture of HP-ASIP.

### B. Datapath

The datapath of HP-ASIP contains 2 pipeline stages and 5 blocks (Fig. 6). Among these blocks, the block *SHA-1/224/256/384/512 first stage & SHA-3* fulfills functions of SHA-3 and the first pipeline stage of MD5 and SHA-1/224/256/384/512, etc. The block *SHA-1/224/256/384/512 second stage* fulfills the computing tasks of the second pipeline stage of MD5 and SHA-1/224/256/384/512, etc. The block *Hash Register* is made up of two 512b registers and is utilized for the hash values of MD5, SHA-1/224/256/384/512, and SHA-3.

The block *Message Scheduler* consists of two 1024b registers and is adopted for the messages of MD5 and SHA-1/224/256/384/512. The block *selDataOut* is utilized to choose results from the two pipeline stages of the datapath.

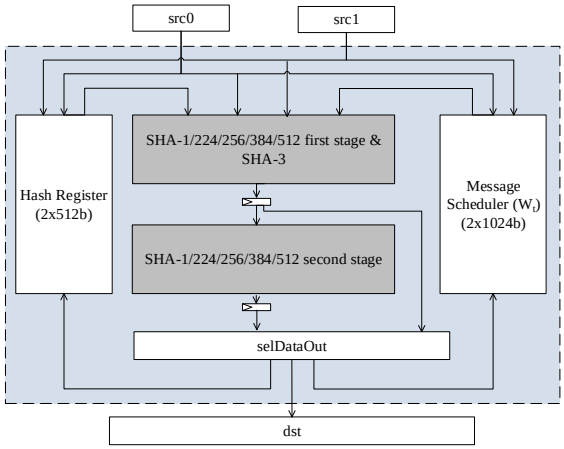



Fig. 6. Datapath of HP-ASIP.

When performing SHA-3 on HP-ASIP, 1600-bit of *Hash register* and *Message Scheduler* are adopted for the SHA-3 hash values. The block *SHA-1/224/256/384/512 first stage & SHA-3* handles one round of SHA-3 [1]. When performing MD5 and SHA-1/224/256/384/512 on HP-ASIP, two pipeline stages of the datapath are adopted. To fully adopt the two-stage pipelined datapath, we process two independent messages each time. Taking SHA-512 as an example, when performing hash computing, two 512-bit hash values can be stored in *Hash register* and two 1024-bit messages can be stored in *Message Scheduler*. Therefore, we can process two independent data streams simultaneously via software programming (i.e., interleaved).

TABLE III: ADDRESSING PATTERNS OF HP-ASIP

| Addressing pattern | Assembly description | Comment |
|---|---|---|
| 1 | imm (e.g., 18) | Point to address represented by an immediate e.g., 18 |
| 2 | ar | Point to address represented by register ar, next cycle ar remains |
| 3 | ar++ | Point to address represented by register ar, next cycle ar = ar+1 |
| 4 | ar+=s | Point to address represented by register ar, next cycle ar = ar+s, s is the step |
| 5 | ar+=s% | Point to address represented by register ar, next cycle ar = ar+s, s is the step; if (ar > AddrEnd) ar = AddrStart |

## C. Memory Subsystem

The operands of HP-ASIP are 16-byte vector data. The 16-byte vector data should be obtained within one clock cycle to ensure that the datapath of HP-ASIP can work efficiently. We thus propose a parallel memory subsystem and specific addressing patterns for HP-ASIP. Five addressing patterns are proposed for HP-ASIP as shown in Table III. All the algorithms targeted in this paper can thus be supported adopting the 5 addressing patterns and the parallel memory subsystem.

To ensure that the vector data can be obtained in parallel, we introduce the RPN and WPN. Fig. 7 describes an example of the RPN. Without the RPN, 16-byte data stored in addresses 19 to 34 can't be obtained simultaneously in sequential order. Utilizing the RPN for shuffling, the vector data can be allocated in parallel for the datapath. The WPN works in a similar way.

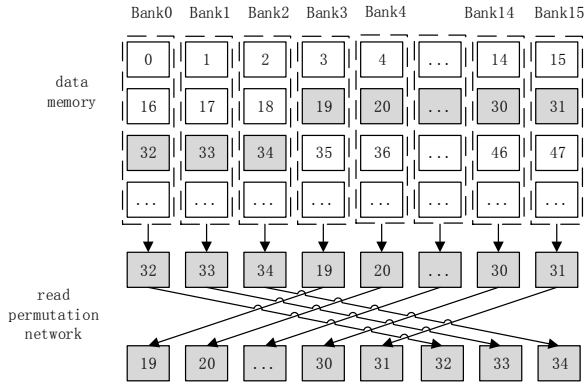



Fig. 7. An example of the RPN.

## D. Pipeline Scheduling

To approach the efficiency limit of the datapath, the instructions of HP-ASIP are realized in pipelined modules. The HP-ASIP contains 7 pipeline stages as shown in Fig. 8. Firstly, an instruction will be read out from the PM and decoded into control signals during IF and ID, respectively. During ID, the addresses of operands will be generated and then passed to the DM. The source operands will be obtained from the DM during Mem. The obtained operands will be passed to the RPN and permuted if necessary during Perm. Afterwards, the outputs of the RPN will be passed to the datapath.

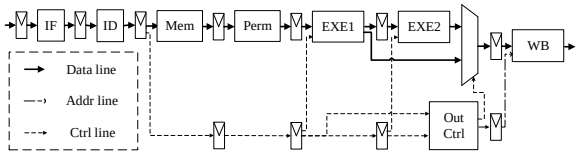



Fig. 8. Pipeline scheduling of HP-ASIP.

The datapath can consume 1 or 2 pipeline stages to fulfill the requirements of different instructions. Some logics are designed to buffer the control signals and the addresses of destination operand to ensure that the datapath can work properly. The block Out Ctrl is utilized to select which pipeline stage of the datapath should output results. Finally, the results will be stored during WB.

## E. Instruction Set

To support multiple algorithms targeted in this paper, we propose an instruction set for HP-ASIP. The instruction set of HP-ASIP consists of 24 SIMD instructions. Among these instructions, 2 are for SHA-3, 3 are for SHA-384/512, 3 are for SHA-224/256, 8 are for SHA-1, and 8 are for MD5. Table IV lists selected typical

instructions of HP-ASIP. Column 1 shows the instruction mnemonics in assembly. Column 2 shows functions of the instructions. Adopting the instruction set introduced, all the algorithms targeted in this paper can be efficiently accelerated.

TABLE IV: SELECTED TYPICAL INSTRUCTIONS OF HP-ASIP

| Instruction mnemonic | Function |
|---|---|
| SHA5120 | One step of SHA-384/512 (adopting the odd register context) |
| SHA5121 | One step of SHA-384/512 (adopting the even register context) |
| SHA2560 | 2x one step of SHA-224/256 (adopting the odd register context) |
| SHA2561 | 2x one step of SHA-224/256 (adopting the even register context) |
| SHA110 | 2x one step of the second round of SHA-1 [2] (adopting the odd register context) |
| MD500 | 2x one step of the first round of MD5 [2] (adopting the odd register context) |
| MD530 | 2x one step of the fourth round of MD5 [2] (adopting the odd register context) |
| SHA3 | One round of SHA-3 [1] |

As loop control under software flow consumes much resource, we introduce an efficient branch-cost-free loop acceleration in hardware. In this design, the instruction REPEAT and nested loops are hardware accelerated. All the loops implemented in this design are performed with no branch cost. We achieve this adopting two mechanisms. Firstly, the microcode for each instruction of HP-ASIP contains a special part indicating how many times the instruction requires to be repeated. The special part of an instruction can be configured via appending an option "-i Imm" at the end of the instruction in assembly code. For example, if we want to repeat an instruction 24 times, an option "-i 24" can be adopted. Secondly, we propose an instruction REPEAT to repeat a block of instructions several times.

```
…
REPEAT 20 {
   SHA100 dm0[ar0].b
   SHA101 dm0[ar0++].b
}
REPEAT 20 {
   SHA110 dm0[ar0].b
   SHA111 dm0[ar0++].b
}
REPEAT 20 {
   SHA120 dm0[ar0].b
   SHA121 dm0[ar0++].b
}
REPEAT 20 {
   SHA130 dm0[ar0].b
   SHA131 dm0[ar0++].b
}
…
```

Fig. 9. Slices of assembly code for SHA-1.

Fig. 9 shows how to perform SHA-1 on HP-ASIP with the proposed instructions. Data hazard avoidance assembly coding is adopted to enhance performance. The indexes for the message schedule are stored in *dm0* before performing hash computation. The REPEAT instruction repeats the following 2 instructions 20 times.

Fig. 10 shows how to perform SHA-3 on HP-ASIP with the proposed instructions. C code of SHA-3 and the corresponding assembly code are presented. The round constants are stored in *dm0* before performing hash computation. The instruction SHA3 is adopted. To process one data block, the instruction SHA3 repeats 24 times with the option "-i 24".

```
C code:
…
for (round = 0; round < 24; round++)
{
      keccak_theta(state);
      keccak_rho();
      keccak_pi(state);
      keccak_chi(state);
      *state ^= keccak_round_constants[round];
}
…

Assembly:
…
SHA3 dm0[ar0+=8].h –i 24
…
```

Fig. 10. Slices of assembly code for SHA-3.

## IV. AREA AND POWER CONSUMPTION

The proposed design is synthesized by Synopsys Design Compiler with STMicroelectronics 65 nm low power cell library. Table V lists the area consumption of each component of HP-ASIP. The overall area cost of HP-ASIP is 0.28 $mm^2$ in 65 nm CMOS technology, wherein the datapath consumes 0.19 $mm^2$. The equivalent gate count for the whole design is 66 kgates and for the logic part is 52 kgates. The total peak power consumption of HP-ASIP is 103.7 mW under the clock frequency of 1.0 GHz. The power estimation is provided by Synopsys Design Compiler.

TABLE V: AREA CONSUMPTION OF COMPONENTS

| Component | Area ($\mu m^2$) | | | |
|---|---|---|---|---|
| | Sum | Percentage | Combinational | Noncombinational |
| AGU | 7024 | 2.53% | 7024 | 0 |
| ID | 346 | 0.12% | 346 | 0 |
| PC FSM | 1138 | 0.41% | 542 | 596 |
| PM | 26549 | 9.57% | 0 | 26549 |
| DM | 32145 | 11.58% | 95 | 32050 |
| RPN | 12692 | 4.57% | 12692 | 0 |
| WPN | 5926 | 2.14% | 5926 | 0 |
| datapath | 191655 | 69.07% | 148221 | 43434 |
| total | 277476 | 100.00% | 174847 | 102629 |

Among all the modules of HP-ASIP, the datapath, the DM, and the PM are the modules consuming most of the area. The datapath costs 69.1% of the area. The DM with 4 memory blocks consumes 11.6%. The PM and the AGU cost 9.6% and 2.5%, respectively. The permutation networks, including read/write permutation network, cost

6.7%. Processor top-level and control path consume negligible area.

TABLE VI: COMPARISON BETWEEN HP-ASIP AND EXISTING VLSI DESIGNS

| Implementation | Hash function | Frequency (MHz) | Throughput | Gate count | Area / Scaled area | Power (mW) | Program mability | Technolo gy |
|---|---|---|---|---|---|---|---|---|
| [10] | SHA-1 and MD5 | 66 | 417.0 Mb/s for SHA-1<br>520.0 Mb/s for MD5 | 21K | NA | NA | N | 0.25 μm |
| [2] | SHA-1 and MD5 | 104 | 16.0 Mb/s for SHA-1<br>28.0 Mb/s for MD5 | 9K | NA | NA | N | 0.18 μm |
| [1] | SHA3-224, SHA3-256, SHA3-384, SHA3-512 | 412 | 9.9 Gb/s for SHA3-512 | NA | 1,115 slices | NA | N | Virtex 6 XC6VLX 760 |
| [11] | SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 | 1,400 | 18.0 Gb/s for SHA-1,<br>23.0 Gb/s for SHA-224,<br>23.0 Gb/s for SHA-256,<br>18.0 Gb/s for SHA-384,<br>18.0 Gb/s for SHA-512 | NA | 62,500 $\mu m^2$ /<br>130,401 $\mu m^2$ | 50 | N | 45 nm |
| This work | MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512 | 1,000 | 15.8 Gb/s for MD5,<br>12.5 Gb/s for SHA-1,<br>15.5 Gb/s for SHA-224,<br>15.5 Gb/s for SHA-256,<br>12.2 Gb/s for SHA-384,<br>12.2 Gb/s for SHA-512,<br>34.9 Gb/s for SHA3-224,<br>33.0 Gb/s for SHA3-256,<br>26.8 Gb/s for SHA3-384,<br>19.9 Gb/s for SHA3-512 | 66K | 277,476 $\mu m^2$ /<br>277,476 $\mu m^2$ | 104 | Y | 65 nm |

## V. EVALUATION

The synthesis results of HP-ASIP are compared with successful VLSI designs for hash algorithms. Table VI summarizes the comparison results. Four VLSI designs are chosen for comparison, because they are very comparable: [2] and [10] are successful SHA-1/MD5 processor cores that support both the SHA-1 and MD5 hash functions; [1] is one of the most efficient SHA-3 methods proposed recently; [11] accelerates SHA-1/224/256/384/512, and it is the most comparable method to ours for SHA functions.

Note that the gate counts of [2] and [10] listed in Table VI exclude the cost of memory [2]. The area of [11] is obtained in 45 nm CMOS technology while HP-ASIP is synthesized in 65 nm CMOS technology. To compare with [11] more fairly, we scale the area provided by [11] to 65 nm CMOS technology. The scaled area is figured out by multiplying the area obtained at 45 nm CMOS technology with $(\frac{65}{45})^2$.

When performing SHA-1/224/256 and MD5, HP-ASIP processes four independent payloads simultaneously. For example, when performing SHA-1, HP-ASIP simultaneously processes four 512b messages producing four 160b hashes with the latency of 2 × (80 + 2) clock cycles, resulting in the SHA-1 throughput of 12.5 Gb/s. When performing SHA-384/512, HP-ASIP processes two 1024b messages simultaneously producing two 384b/512b hashes with the latency of 2 × (80 + 4) clock cycles, resulting in the throughput of 12.2 Gb/s. When performing SHA3-224, HP-ASIP processes one 1152b message producing a 224b hash with the latency of 9 + 24 clock cycles, resulting in the throughput of 34.9 Gb/s. The remained throughput of HP-ASIP listed in Table 6 is figured out in a similar way.

In terms of throughput, Ramanarayanan's design [11] achieves better throughput than HP-ASIP for SHA-1/224/256/384/512 because Ramanarayanan's design obtains high clock frequency. However, HP-ASIP is programmable and supports MD5 and SHA-3 because HP-ASIP obtains programmable architecture and application specific instruction set.

Compared with state-of-the-art ASICs/FPGAs, our design achieves competitive throughput for MD5 and SHA functions with full programmability. For its programmability, HP-ASIP can offer changes to the algorithms implemented in this paper to extend its chip lifetime. For example, when one of the implemented cryptographic hash algorithms is cracked, HP-ASIP can still work properly via updating software.

## VI. CONCLUSIONS

This paper presents a SIMD ASIP for cryptographic hash functions that accelerates MD5, SHA-1, SHA-2, and SHA-3. Adopting processor architecture, we map the hash algorithms onto a two-stage pipelined datapath, optimizing the degree of hardware sharing among the algorithms. This approach results in a hash processor that achieves the throughput of 15.8 Gb/s for MD5, 12.5 Gb/s for SHA-1, 12.2 Gb/s for SHA-512, and 19.9 Gb/s for SHA3-512, occupying 0.28 $mm^2$ in 65 nm CMOS. Compared with state-of-the-art VLSI designs, our design achieves ASIC-like performance, full programmability, and low silicon cost.

ACKNOWLEDGMENT

The finance supporting from National High Technical Research and Development Program of China (863 program) 2014AA01A705 is sincerely acknowledged by authors.

REFERENCES

[1] H. E. Michail, L. Ioannou, and A. G. Voyiatzis, "Pipelined SHA-3 Implementations on FPGA: Architecture and Performance Analysis," in *Proc. Second Workshop on Cryptography and Security in Computing Systems*, 2015, pp. 13-18.

[2] D. Cao, J. Han, and X. Zeng, "A reconfigurable and ultra low-cost VLSI implementation of SHA-1 and MD5 functions," in *Proc. 7th International Conference on ASIC*, 2007, pp. 862-865.

[3] 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, 2007.

[4] M. Macchetti and L. Dadda, "Quasi-Pipelined hash circuits," in *Proc. 17th IEEE Symposium on Computer Arithmetic*, 2005, pp. 222-229.

[5] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis, "Cost-Efficient SHA hardware accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 8, pp. 999-1008, 2008.

[6] Y. K. Lee, H. Chan, and I. Verbauwhede, "Throughput optimized SHA-1 architecture using unfolding transformation," in *Application-Specific Systems, Architectures and Processors*, 2006, pp. 354-359.

[7] M. McLoone and J. V. McCanny, "Efficient single-chip implementation of SHA-384&SHA-512," in *Proc. IEEE International Conference on Field-Programmable Technology*, 2002, pp. 311-314.

[8] G. S. Athanasiou, G. Makkas, and G. Theodoridis, "High throughput pipelined FPGA implementation of the new SHA-3 cryptographic hash algorithm," in *Proc. 6th International Symposium on Communications, Control and Signal Processing*, 2014, pp. 538-541.

[9] H. E. Michail, G. S. Athanasiou, G. Theodoridis, and C. E. Goutis, "On the development of high-throughput and area-efficient multi-mode cryptographic hash designs in FPGAs," *Integration, the VLSI Journal*, vol. 47, no. 4, pp. 387-407, 2014.

[10] M. Wang, C. Su, C. Huang, and C. Wu, "An HMAC processor with integrated SHA-1 and MD5 algorithms," in *Proc. Asia and South Pacific Design Automation Conference*, 2004, pp. 456-458.

[11] R. Ramanarayanan, *et al*., "18Gbps, 50mW reconfigurable multi-mode SHA hashing accelerator in 45nm CMOS," in *Proc. ESSCIRC*, 2010, pp. 210-213.

**Yuanhong Huo** was born in Henan Province, China, in 1988. He received the B.Sc. degree from Zhengzhou University, Zhengzhou, China, in 2011. He is currently pursuing the Ph.D. degree in computer science and technology with the Beijing Institute of Technology, Beijing, China. His current research interests include Application Specific Instruction Set Processors (ASIP) design, Software Defined Radio (SDR), software-hardware co-design and VLSI implementation.

**Dake Liu** received the D.Tech. degree from Linkoping University, Linkoping, Sweden, in 1995. He has experiences in the design of communication systems and radio frequency CMOS integrated circuits. He is currently a Professor and the Head of the Institute of Application Specific Instruction Set Processors (ASIP), Beijing Institute of Technology, Beijing, China, and also a Professor with the Computer Engineering Division, Department of Electrical Engineering, Linkoping University. He is the Co-Founder and Chief Technology Officer of Freehand DSP AB Ltd., Stockholm, Sweden, and the Co-Founder of Coresonic AB, Linkoping, which was acquired by MediaTek, Hsinchu, Taiwan. He has authored over 150 papers on journals and international conferences and holds five U.S. patents. His current research interests include high-performance low-power ASIP and integration of on-chip multiprocessors for communications and media digital signal processing. Dr. Liu is enrolled in the China Recruitment Program of Global Experts.