Chinese Journal of Electronics
Vol.26, No.3, May 2017

High-Throughput Area-Efficient Processor for Cryptography*

HUO Yuanhong[1] and LIU Dake[1,2]

(1. *School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China*)

(2. *School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China*)

Abstract — Cryptography circuits for portable electronic devices provide user authentication and secure data communication. These circuits should, achieve high performance, occupy small chip area, and handle several cryptographic algorithms. This paper proposes a high-performance ASIP (Application specific instruction set processor) for five standard cryptographic algorithms including both block ciphers (AES, Camellia, and ARIA) and stream ciphers (ZUC and SNOW 3G). The processor reaches ASIC-like performance such as 11.6 Gb/s for AES encryption, 16.0 Gb/s for ZUC, and 32.0 Gb/s for SNOW 3G, *etc* under the clock frequency of 1.0 GHz with the area consumption of 0.56 mm^2 (65 nm). Compared with state-of-the-art VLSI designs, our design achieves high performance, low silicon cost, low power consumption, and sufficient programmability. For its programmability, our design can offer algorithm modification when an algorithm supported is unfortunately cracked and invalid to use. The product lifetime of our design can thus be extended.

Key words — ASIP (Application specific instruction set processor), Cryptographic processor, VLSI (Very large-scale integration), Cryptography.

I. Introduction

Cryptography provides the mechanisms for implementing accountability and confidentiality in communication (*e.g.*, in current fourth generation (4G) and the upcoming fifth generation (5G)[1] mobile networks). Circuits for cryptography generally need to support multiple standards with high throughput and low computing latency under limited power consumption and silicon cost. Cryptography circuits need to support multiple cryptographic algorithms nowadays because a variety of standards are coexistent in a single device and the standards could adopt multiple cryptographic algorithms. For example, EEA1/EIA1 based on SNOW 3G[2], EEA2/EIA2 based on block cipher AES-128[3], and EEA3/EIA3 based on ZUC[4] are adopted in the LTE (Long term evolution) standards[5]. Besides, as demands for secure communication bandwidth grow, high throughput and low computing latency cryptographic processing is becoming increasingly vital to good system performance[6]. For example, an online real-time video play needs low computing latency in the computing chain of radio access control, payload management, decryption, video decoder, and audio-video synchronization, *etc*. In this case, the decryption computing latency shall be low enough. Cryptography circuits thus need to reach the required high throughput and low computing latency. Lastly, products with limited power and area, like portable electronic devices, require low-power and compact circuit designs for cryptography[7]. Designing a single circuit module instead of multiple circuit modules to support multiple cryptographic algorithms can result in a significant silicon cost reduction via hardware sharing[5]. It is thus necessary to design a single circuit module for multiple cryptographic algorithms satisfying all the mentioned essential requirements. However, it is of great challenge to design and implement such a circuit module. Implementations range from Application-specific integrated circuits (ASICs), which achieve very high throughput with low flexibility, to General purpose processors (GPPs), which are highly flexible obtaining low throughput and high power consumption.

In this paper, we introduce a high-throughput area-efficient cryptography processor (CP-ASIP) that specially accelerates three block ciphers and two stream ciphers. Stream ciphers ZUC[4] and SNOW 3G[2] are accelerated because they are used in current wireless communication networks, like 4G LTE[5] and should, in general, be with low computing latency. Block cipher AES (Advanced Encryption Standard)[3] is adopted in almost all security

*Manuscript Received Jan. 12, 2016; Accepted Oct. 19, 2016. This work is supported by the National High-Tech Research and Development Program (863 Program) of China (No.2014AA01A705).

 DOI:10.1049/cje.2017.03.004

protocols and should, in general, be fulfilled with high throughput. AES is thus accelerated by CP-ASIP. Another two block ciphers Camellia[8] and ARIA[9] adopted in TLS/SSL, like AES, are also accelerated for their high security and typicality[10]. Table 1 lists the type, key size, and cases of uses of the algorithms targeted in this paper. CP-ASIP is obtained via hardware/software (HW/SW) codesign methodology, and fulfills all the mentioned essential requirements. Thanks to its programmability, CP-ASIP can offer algorithm modification when an algorithm implemented is unfortunately cracked and invalid to use. Therefore, the product lifetime of our design can be extended.

Table 1. Algorithms implemented

Algorithm	Type	Key size (bit)	Cases of uses
AES	Block Cipher	128, 192, or 256	3GPP LTE, TLS/SSL
ZUC	Stream Cipher	128	3GPP LTE
SNOW 3G	Stream Cipher	128	3GPP LTE
Camellia	Block Cipher	128, 192, or 256	IPsec, TLS/SSL
ARIA	Block Cipher	128, 192, or 256	TLS/SSL, KS X 1213:2004

The rest of this paper is organized as follows. Section II provides a survey of related work on acceleration for cryptographic algorithms. Section III presents the design methodology of CP-ASIP. Section IV illustrates the top-level architecture, the datapath, the instruction set, the memory subsystem, and the pipeline of CP-ASIP. Section V describes area and power consumption of CP-ASIP and evaluates CP-ASIP. The conclusions are provided in Section VI.

II. Related Work

There have been successful methods to accelerate cryptographic algorithms. The research works can be divided into six categories: ASICs, FPGAs (Field programmable gate arrays), GPPs, DSPs (Digital signal processors), GPUs (Graphic processing units), and ASIPs (Application specific instruction set processors).

Verbauwhede *et al.*[11] describe the first AES implementation on silicon, which can provide the throughput of 2.29 Gb/s with a non-pipeline architecture. Mathew *et al.*[12] implement a 53 Gb/s AES accelerator in 45 nm CMOS technology. Zeng *et al.*[13] propose a multiple-term Common subexpression elimination (CSE) algorithm to optimize the AES S-box achieving high computation and optimization efficiency. Gupta *et al.*[5] design an integrated high performance accelerator (HiPAcc-LTE) for SNOW 3G and ZUC with the throughput of 34.9 Gb/s for SNOW 3G and ZUC. ASIC solution, in general, leads to high throughput at low power consumption. However, ASIC is inflexible and is thus only applied to a limited subset of cryptosystems. Except for ASIC, FPGA is another hardware implementation for cryptography. Qu *et al.*[14] demonstrate a 73.7 Gb/s AES system on a Xilinx XC5VLX85 chip running at 570 MHz. Although FPGA implementations are generally more flexible than ASICs, their costs and power consumption are not acceptable for volume products.

Matsui *et al.*[15] propose a bit slice AES implementation on Intel Core 2. It achieves a 9.2 clock cycles per byte throughput for a data chunk longer than 2,048 bytes. Liu *et al.*[16] implement AES on a fine-grained many-core system that can achieve higher throughput per unit of chip area and better energy efficiency compared to other software platforms. GPUs[17] and DSPs[18] are also adopted to implement the AES algorithm. Implementing cryptosystems on GPPs/DSPs/GPUs is flexible. A drawback of software implementation on GPPs/DSPs/GPUs is its high power consumption.

To mitigate the gap between hardware implementations and software implementations, ASIP, a HW/SW codesign implementation, has been introduced. Eslami *et al.*[7] propose a universal cryptography processor for smart-card applications that achieves the throughput of 1.83 Mb/s for AES encryption with area consumption of 2.25 mm^2 (180 nm). Koo *et al.*[19] propose a processor that occupies 19,056 gate counts and encrypts data at 720 Mb/s and 1,047 Mb/s for ARIA and AES, respectively. Sayilar *et al.*[6] propose a highly reconfigurable cryptographic processor that can support 16 cryptographic algorithms and obtains the throughput of 6.4 Gb/s for AES encryption with the silicon area of 6.3 mm^2 (45 nm). Wang *et al.*[20] implement AES, Camellia, DES, SHACAL-1, SMS4, SNOW 3G, and ZUC, *etc* on a reconfigurable architecture for symmetric ciphers achieving the throughput of 51.2 Gb/s for AES encryption with the silicon area of 51.36 mm^2 (65 nm). Previous ASIPs for cryptography either target a small predefined set of algorithms obtaining limited flexibility and/or throughput, or focus primarily on high performance and flexibility leading to an unacceptable chip area for portable electronic devices.

Considering all the methods discussed, it is quite challenging to design and implement a circuit module for multiple cryptographic algorithms fulfilling all the mentioned essential requirements. In this paper, adopting processor architecture, we map block ciphers (AES, ARIA, and Camellia) and stream ciphers (ZUC and SNOW 3G) onto a universal architecture optimizing the degree of hardware sharing among them. This method results in a high performance processor for cryptography achieving the throughput of 11.6 Gb/s for AES encryption and 32.0 Gb/s for SNOW 3G, *etc* with the silicon area of 0.56 mm^2 in 65 nm CMOS. Compared with the previous VLSI design[21] for AES, our design improves the area efficiency by 24.4% with sufficient programmability.

III. Design Methodology

We propose a methodology for designing CP-ASIP. Fig.1 depicts the design flow. The design flow, a HW/SW codesign flow, optimizes the partition of HW (an assembly instruction) and SW (algorithm codes implemented with the proposed instructions) functions cooperatively during the design of CP-ASIP. Firstly, the scope of cryptographic algorithms (AES, ZUC, SNOW 3G, Camellia, and ARIA), throughput, acceptable silicon cost, etc are specified. Then, a datapath, a memory subsystem, and a control path are proposed for CP-ASIP to accelerate the algorithms of the scope.

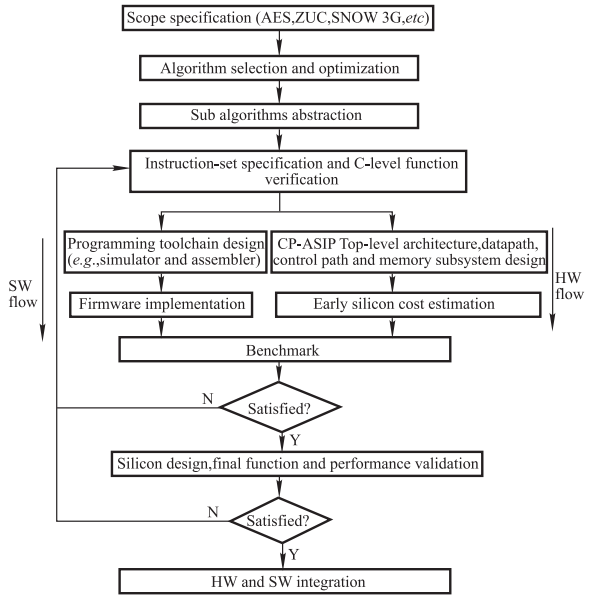


Fig. 1. Design flow of CP-ASIP

The datapath is designed first because the design results of the datapath are the design inputs for other parts of CP-ASIP. According to the hardware requirements of AES, a preliminary datapath is proposed. It is because AES is the most complicate algorithm among the targeted algorithms. In general, the most complicate algorithm of a scope impacts the design of an ASIP for the scope greatly. As AES encryption and decryption are implemented using 16 256×8b LUTs (Lookup tables), respectively, 32 256×8b SRAMs (Static random access memories) are adopted in this design (to be discussed). Then, implementations of other algorithms targeted in this paper are optimized and fine-tuned into the datapath.

Table 2. LUT cost of algorithms implemented

Algorithm	Number of LUT	Size of LUT (bit)
AES	32	256×8
SNOW 3G	16	256×8
ZUC	16	256×8
Camellia	8	256×8
ARIA	16	256×8

Table 2 lists the LUT cost of the targeted algorithms when they are implemented on the datapath. In this paper, common parts among the algorithms are implemented with shared functional blocks of the datapath to obtain low silicon cost. All the algorithms targeted in this paper require no more than 32 256×8b LUTs. Therefore, the 32 256×8b SRAMs for AES can be reused for other algorithms targeted in this paper through software programming. The contents of the LUTs for each algorithm can be obtained according to previous work or the corresponding algorithm specifications. For example, Selent[3] has explained the way to construct the LUTs (called S-box) for AES in detail. Aoki *et al.*[8] and Kwon *et al.*[9] have discussed the LUTs (called S-box or s-boxes) for Camellia and ARIA, respectively. Therefore, the methods to generate the contents of the LUTs are not explained in detail in this paper.

After mapping all the algorithms of the scope onto the datapath incrementally, we fix the datapath. Then, we extract the control signals for the processing routines in the datapath and represent the control signals by a group of control indications, so that a specific instruction set is designed. Afterwards, the algorithms targeted in this paper are implemented adopting the instruction set, so that pseudocode implementations of the algorithms are obtained. According to the addressing and control information of the pseudocode implementations, the specific memory subsystem, the control path, and the top-level architecture of CP-ASIP are designed to ensure that the datapath can handle the algorithms targeted in this paper properly. Based on the specified functional blocks and the instruction set, we develop the Register transfer level (RTL) description of CP-ASIP. Then, we verify the correctness and performance of both the functional design and the silicon layout. Based on the algorithm pseudocode, we develop the firmware codes (assembly program) of the algorithms targeted in this paper. All the algorithms of the previously defined scope are supported via the firmware codes. At last, the software and hardware are integrated and CP-ASIP is designed.

The design flow is recursive. Any requirements previously mentioned not fulfilled may cause a huge work of redesign. Through the HW/SW codesign methodology, we ensure that CP-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 cryptographic algorithms implemented in this paper satisfying all the previously mentioned essential requirements.

IV. Proposed Processor

This section details CP-ASIP and the architecture of the whole system with a host for CP-ASIP. The whole system includes a main processor, a main memory, an interconnection network, and CP-ASIP. The main pro-

cessor gives to CP-ASIP the data to be encrypted or decrypted, loads algorithms to run on CP-ASIP, and starts CP-ASIP. When CP-ASIP completes the task assigned by the main processor, the main processor obtains the data encrypted or decrypted from CP-ASIP. CP-ASIP includes a data path, a memory subsystem, a control path, a Direct memory access (DMA) controller, and a network interface. CP-ASIP fulfills the key generation procedure of each algorithm targeted in this paper together with the main processor and performs the encryption or decryption of data assigned by the main processor independently.

1. Top-level architecture

Single instruction multiple data (SIMD) architecture is adopted in CP-ASIP to meet the requirements on the computational complexity. The top-level architecture of CP-ASIP is composed of control logic, memory subsystem, and datapath as shown in Fig.2(a). The control logic is made up of program flow controller, Program memory (PM), Instruction decoder (ID), and DMA. The control logic reads an instruction from the PM per clock cycle and decodes the fetched instruction into control signals. The control logic performs REPEAT instruction acceleration, too. The memory subsystem is made up of Address generation unit (AGU), Data memory (DM), read permutation network, and write permutation network. The AGU generates addresses for the operands according to the fetched instruction. The generated addresses will be passed to the DM. The DM is made up of four memory blocks. Each memory block is made up of 16 256×8b SRAMs. Therefore, each memory block can provide 16-byte data from the 16 SRAMs per clock cycle. The outputs of the memory blocks will be passed to the read permutation network and then to the datapath. The datapath handles cryptographic algorithms targeted in this paper in parallel. The outputs of the datapath will be passed to the write permutation network and then written to a memory block or register file. The read permutation network and write permutation network are designed for data shuffling according to the requirements of the algorithms implemented in this paper.

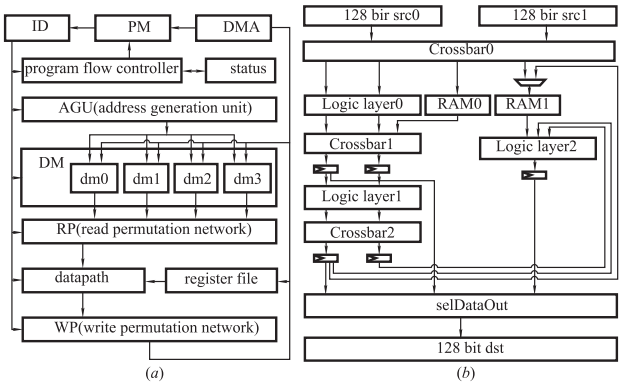


Fig. 2. (a) Top-level architecture; (b) Datapath of CP-ASIP

2. Datapath and instruction set

The datapath of CP-ASIP takes in two 128-bit operands and outputs 128-bit result as shown in Fig.2(b). The datapath contains 9 functional blocks and 3 pipeline stages. This work adopts pipelining technology to achieve high clock frequency. Taking AES encryption as an example, four operations are applied in each intermediate round of AES encryption[3] as shown in Table 3. If all these operations are fulfilled in one clock cycle, the frequency of the circuit will be low. Among these operations, SubBytes() is the critical path of the whole circuit. To improve the frequency of the circuit, the operations of AES encryption are scheduled in two pipeline stages in this design.

Table 3. Operations of an intermediate round of AES encryption (decryption)

Operation name	Function description
(Inv)SubBytes ()	Uses an S-box to perform a byte-by-byte substitution of the current block
(Inv)ShiftRows ()	A simple permutation
(Inv)MixColumns ()	Finite field matrix multiplication
AddRoundKey ()	A XOR operation of the current block with a portion of the expanded key[22]

Among the previously mentioned 9 functional blocks of the datapath, RAM0 and RAM1 are for the previously mentioned LUTs. Crossbar0-Crossbar2 are utilized for shuffling to ensure that vector data can be obtained in parallel. Logic layer0, Logic layer1, and Logic layer2 fulfill the computing functions in pipeline stage one, two, and three of the datapath, respectively for the proposed instructions (to be discussed). The block seldataOut is adopted to choose results from the three pipeline stages of the datapath.

In this design, the operation SubBytes() of AES encryption is fulfilled via RAM0 (Fig.3(a)). RAM0 is made up of 16 256×8b SRAMs. During SubBytes(), the 16-byte data of a block are acted as addresses for the 16 SRAMs of RAM0 and 16-byte substitution of the block can thus be obtained from RAM0. Then, the outputs of RAM0 will be passed to Crossbar1. Crossbar1 fulfills the operation ShiftRows() of AES encryption. Afterwards, the outputs of Crossbar1 will be passed to Logic layer1 to fulfil the operations MixColumns() and AddRoundKey() of AES encryption. The outputs of Logic layer1 will be passed from Crossbar2 to selDataOut. In this work, we did not implement any key scheduler under the assumption that the round keys are stored in the data memory before performing AES encryption. But on-the-fly generation of round keys is possible on CP-ASIP. It costs tens of clock cycles to generate round keys on CP-ASIP.

Different from AES encryption, the four operations of an intermediate round of AES decryption is implemented in the datapath with the order of AddRoundKey(), InvShiftRows(), InvMixColumns(), and InvSub-

Bytes() in this design. The operation InvSubBytes() is performed during the third pipeline stage of the datapath. Therefore, RAM1, made up of 16 256 × 8b SRAMs, is introduced. Fig.3(*b*) shows the blocks adopted by AES decryption. Operations AddRoundKey(), InvShiftRows(), InvMixColumns(), and InvSubBytes() of AES decryption are fulfilled by block Logic layer0, Crossbar1, Logic layer1, and RAM1, respectively. The outputs of RAM1 are passed from Logic layer2 to selDataOut. Utilizing the unified datapath, all the algorithms targeted in this paper can be accelerated.

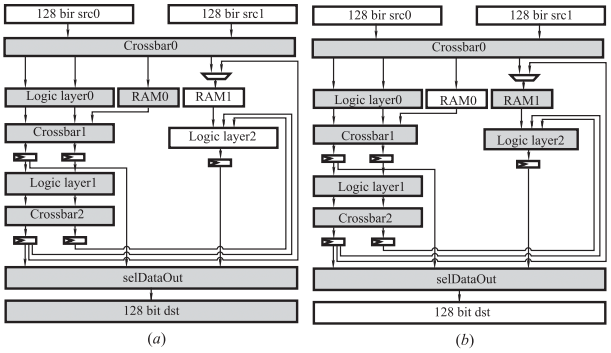


Fig.3. (*a*) Blocks used by aesenc; (*b*) Blocks used by aesdec

Table 4. Selected typical instructions of CP-ASIP

Instruction Mnemonic	Scaled Speedup	Function
aesenc	176	AES encryption intermediate round
aesencf	32	AES encryption first round
aesencl	48	AES encryption last round
aesdec	592	AES decryption intermediate round
aesdecf	48	AES decryption first round
aesdecl	32	AES decryption last round
ariaof	116	ARIA OF
ariaef	116	ARIA EF
camftf	60	Camellia Camellia_Feistel first round
camft	60	Camellia Camellia_Feistel remained round
zuclfsr	65	ZUC LFSRWithWorkMode
zucbrf	78	ZUC BitReorganization() and F()
zucbrfx	79	ZUC BitReorganization() and F() ^ x
snwfsmi	150	SNOW 3G ClockFSM() and SNOW 3G ClockLFSRInitializationMode()
snwfsmk	149	SNOW 3G ClockFSM() and SNOW 3G ClockLFSRKeyStreamMode()

After mapping all the algorithms targeted in this paper onto the datapath, we propose an application specific instruction set for CP-ASIP. The instruction set of CP-ASIP contains 23 SIMD instructions. Among these instructions, 6 are for AES, 3 are for ZUC, 3 are for SNOW 3G, 5 are for ARIA, and 6 are for Camellia. Table 4 lists selected typical instructions of CP-ASIP. Column 1 and 3 present the instruction mnemonics in assembly and the functions of each instruction, respectively. Column 2 describes speedups of the instructions, which are the number of basic operations (*e.g.*, XOR, SHIFT, ADD, and LOAD, *etc*) performed by these instructions. Each operation equals one instruction utilizing GPP while in CP-ASIP, all the operations are merged in one instruction. Utilizing the instruction set, all the algorithms targeted in this paper can be supported via software programming.

3. Memory subsystem

CP-ASIP processes 16-byte vector data per clock cycle. Therefore, 16-byte wanted data should be obtained within one clock cycle to ensure that CP-ASIP can work efficiently. If there are no proper addressing patterns, the 16-byte wanted data cannot be obtained simultaneously all the time. Therefore, a parallel memory subsystem and specific addressing patterns are designed for CP-ASIP. Table 5 describes the addressing patterns supported by CP-ASIP. Column 1 shows the number of these 8 addressing patterns. Column 2 shows the corresponding assembly description. Column 3 describes functions of the addressing patterns. Utilizing these addressing patterns, all the algorithms targeted in this paper can be accelerated.

Table 5. Addressing patterns of CP-ASIP

Addressing Pattern	Assembly Description	Comment
1	imm (*e.g.*, 16)	Point to address represented by an immediate for example, 16
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−−	Point to address represented by register ar, next cycle ar = ar-1
5	ar+=s	Point to address represented by register ar, next cycle ar = ar+s, s is the step
6	ar− =s	Point to address represented by register ar, next cycle ar = ar-s, s is the step
7	ar+=s%	Point to address represented by register ar, next cycle ar = ar+s, s is the step, if (ar ≥ AddrEnd) ar = AddrStart
8	ar− =s%	Point to address represented by register ar, next cycle ar = ar-s, s is the step, if (ar ≤ AddrEnd) ar = AddrStart

4. Pipeline scheduling

The instructions of CP-ASIP are realized in pipelined modules to approach the efficiency limit of the datapath and the memory bandwidth. As shown in Fig.4, CP-ASIP contains at most 8 pipeline stages. Firstly, one instruction will be read out from the PM during Instruction fetch (IF). Then, the fetched instruction will be decoded, and the addresses of operands will be generated during Instruction decoding (ID). During Memory access (Mem), the source operands will be obtained and then passed to the read permutation network. During Permutation (Perm), the operands can be reordered in the read permutation network according to the requirements of the implemented algorithms and then passed to the datapath. The datapath of CP-ASIP costs 1 to 3 pipeline stages according to the requirements of different instructions. To ensure that the datapath can work properly, some logics

are designed to buffer the control signals and the destination operand. A block called Out Ctrl is utilized to select which pipeline stage of the datapath should output results. During the pipeline stage WB (write back result), the results will be stored.

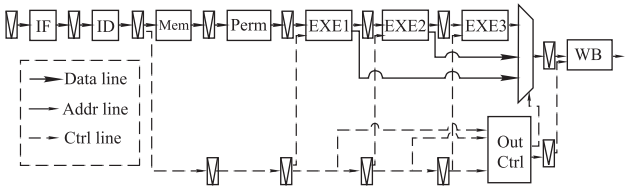
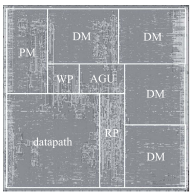


Fig. 4. Pipeline scheduling of CP-ASIP

V. Synthesis Results and Evaluation

The proposed processor is synthesized by Synopsys Design Compiler with STMicroelectronics 65nm low power cell library. Then, the proposed processor is placed and routed by Cadence Encounter. Fig.5 describes the layout result and the specifications. The equivalent gate count for the whole design is 135 kgates and the logic part occupies 36 kgates. The total peak power consumption of our design is 352mW under the clock frequency of 1.0GHz.



technology	65 nm
chip size	0.56 mm^2
logic gates	135kgates
supply voltage	1.2 V
frequency	1.0 GHz
performance	11.6 Gb/s for AES
	16.0 Gb/s for ZUC
	32.0 Gb/s for SNOW 3G

Fig. 5. Chip design results: layout and chip specifications

The synthesis results of CP-ASIP are primarily compared with state-of-the-art VLSI designs for AES due to the lack of published results for other algorithms implemented in this paper. Table 6 lists the comparison results. The choice of operation mode to be feedback (CBC, CFB, and OFB) or non-feedback (ECB and CTR) plays an important role on the throughput of a design[6]. We thus evaluate CP-ASIP in both CTR (Counter) mode (Column 7) and CBC (Cipher block chaining) mode (Column 8). As the throughput of feedback mode provides better insight about the performance of a design, the area efficiency (Throughput/Gate Count) of the listed solutions is provided for CBC mode. Note that in our two-stage pipelined design, the throughput for AES encryption can be maintained if two independent data streams are applied (interleaved). Otherwise, in the feedback cipher modes, the throughput will be scaled down by two.

Among the designs listed in Table 6, Morioka's design[21] and Verbauwhede's design[11] process one round of AES encryption per clock cycle, in a fully parallel non-pipelined fashion, achieving the same throughput in feedback mode as in non-feedback mode. Morioka's AES-only design[21] is the most comparable design to CP-ASIP for AES encryption. With its outer-round pipelined structure and highly tuned datapath, Morioka's design achieves high area efficiency for AES. However, CP-ASIP achieves 19.6% less silicon cost and 24.4% higher area efficiency than Morioka's design. Zhang's design[24], Liu's design[16], Sayilar's design[6], and Hodjat's design[25] achieve higher throughput than CP-ASIP in CTR mode. It is because these designs can process all rounds of AES encryption in parallel in CTR mode. In contrast, CP-ASIP can only process one round of AES encryption each time. However, these designs are not able to provide the same high performance in feedback modes as in non-feedback modes due to their deep pipelines. CP-ASIP achieves higher throughput than these designs in CBC mode. It is because CP-ASIP is programmable and can thus process two independent data streams simultaneously, in feedback modes, achieving the same throughput as in non-feedback modes. The comparison results show that CP-ASIP achieves ASIC-like performance with sufficient programmability.

Table 6. Comparison between the proposed design and existing VLSI designs for AES

Method	Area (mm^2)	Gates (kgates/ Slices)	Power (mW)	Cycles Per Block	Frequency (MHz)	CTR Throughput (Gb/s)	CBC Throughput (Gb/s)	CBC Throug./ Gate Count (Mbps/kgates)	Programmability	Technology (nm)
McLoone[23]	NA	2222	NA	10	54	6.96	0.70	NA	N	FPGA
Zhang[24]	NA	11022	NA	70	168	21.56	0.31	NA	N	FPGA
Liu[16]	6.63	1202	1580	152	1210	153.70	1.02	0.85	Y	65
Sayilar[6]	6.32	2390	6207	20	1000	128.00	6.40	2.68	Y	45
Hodjat[25]	NA	473	NA	41	606	77.57	1.89	4.00	N	180
Verbauwhede[11]	NA	173	56	NA	125	2.29	2.29	13.24	N	180
Morioka[21]	NA	168	1920	10	909	11.64	11.64	69.29	N	130
This work	0.56	135	352	11	1000	11.64	11.64	86.20	Y	65

Except for hardware designs, there are high throughput software designs for AES encryption. Table 7 shows the comparison of our design with selected typical commercial CPUs (Central processing units) for AES encryption in non-feedback modes. Compared with CPUs, CP-ASIP achieves competitive throughput for AES encryption. Besides, for this implementation, there is no need to move the keys among the registers by the main processor. Therefore, less information will leak compared to the software implementations for the attackers who use the

side-channel[7] information to detect the algorithm keys.

Table 7. Comparison with software designs for AES

Architecture	Technology (nm)	Configuration (core x thread)	Frequency (MHz)	Throughput (Gb/s)
POWER8[26]	22	1 x 1	4000	7.60
i7-2600K[27]	32	1 x 1	3400	1.90
i7-2600K[27]	32	4 x 4	3400	7.00
i7-2600K[27]	32	4 x 8	3400	7.50
Our design	65	1 x 1	1000	11.64

Table 8. Performance of CP-ASIP

Algorithm	Block Size (bit)	Cycles Per Block	Throughput (Gb/s)	Bit Per Cycle
AES	128	11	11.6	128
SNOW 3G	32	1	32.0	32
ZUC	32	2	16.0	32
ARIA	128	13	9.8	128
Camellia	128	22	5.8	64

As to other algorithms implemented in this paper, CP-ASIP can also achieve high performance. Table 8 summarizes the performance of CP-ASIP for the implemented cryptographic algorithms. Both encryption and decryption of the three block ciphers (*i.e.*, AES, ARIA, and Camellia) are accelerated, with standard key lengths of 128, 192, and 256 bits. The results of the three block ciphers are obtained when performing encryption on CP-ASIP with the key length of 128 bits.

Compared with state-of-the-art VLSI designs, CP-ASIP achieves high performance, low silicon cost, and obtains sufficient programmability. CP-ASIP obtains high performance for its high clock frequency (1.0 GHz), achieved via pipelining technology and high parallel degree (128-bit), obtained by its SIMD architecture. CP-ASIP achieves low silicon cost via maximizing the number of common blocks among the algorithms implemented in this paper. Due to its small chip area, CP-ASIP can be a valuable option for mobile/portable devices[28]. CP-ASIP supports 5 cryptographic algorithms via hardware sharing because it obtains configurable datapath, programmable architecture, and application specific instruction set. For its programmability, CP-ASIP can offer algorithm modification when an algorithm implemented in this paper is unfortunately cracked and invalid to use.

The LUTs utilized in CP-ASIP can be reused for CRC (Cyclic Redundancy Check) calculation with less than 1 kgates extra silicon cost. The method has been described in detail in our previous work[29]. The throughput for CRC calculation can surpass 50 Gb/s as CP-ASIP contains 8 KB LUTs and can thus process 64-bit data per clock cycle.

VI. Conclusions

This paper presents a high-throughput area-efficient SIMD ASIP for cryptography that supports both block ciphers (AES, ARIA, and Camellia) and stream ciphers (ZUC and SNOW 3G). We achieve this by implementing the five ciphers on a 128-bit table-based datapath optimizing the HW/SW partition. This approach results in a cryptography processor that achieves ASIC-like performance such as 11.6 Gb/s for AES encryption and 32.0 Gb/s for SNOW 3G, etc, obtains sufficient programmability with 23 accelerated instructions, and occupies 0.56 mm^2 in 65 nm CMOS. For its programmability, our design can offer algorithm modification via software programming when an implemented algorithm is unfortunately cracked and invalid to use, extending the product lifetime. Our design is evaluated with state-of-the-art VLSI designs for AES, which reveals its high performance, low silicon cost, and high flexibility. Our design can thus be a valuable option for client-end systems. In our future work, we will extend this design to accelerate more block ciphers and stream ciphers.

References

[1] J.G. Andrews, S. Buzzi, W. Choi, *et al.*, "What will 5G be?", *IEEE Journal on Selected Areas in Communications*, Vol.32, No.6, pp.1065–1082, 2014.

[2] ETSI/SAGE Specification, "Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 2: SNOW 3G Specification", version 1.1, 2006.

[3] D. Selent, "Advanced encryption standard", *Rivier Academic Journal*, Vol.6, No.2, pp.1–14, 2010.

[4] ETSI/SAGE Specification, "Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification", version 1.5, 2011.

[5] S.S. Gupta, A. Chattopadhyay and A. Khalid, "HiPAcc-LTE: An integrated high performance accelerator for 3GPP LTE stream ciphers", *International Conference on Cryptology in India*, Chennai, India, pp.196–215, 2011.

[6] G. Sayilar and D. Chiou, "Cryptoraptor: High throughput reconfigurable cryptographic processor", *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, pp.154–161, 2014.

[7] Y. Eslami, A. Sheikholeslami, P.G. Gulak, *et al.*, "An area-efficient universal cryptography processor for smart cards", *IEEE Transactions on Very Large Scale Integration Systems*, Vol.14, No.1, pp.43–56, 2006.

[8] K. Aoki, T. Ichikawa, M. Kanda, *et al.*, "Specification of Camellia-A 128-bit block cipher", 2000.

[9] D. Kwon, J. Kim, S. Park, *et al.*, "New block cipher: ARIA", *International Conference on Information Security and Cryptology*, Seoul, Korea, pp.432–445, 2003.

[10] J.H. Kong, L.M. Ang and K.P. Seng, "A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments", *Journal of Network and Computer Applications*, Vol.49, pp.15–50, 2014.

[11] I. Verbauwhede, P. Schaumont and H. Kuo, "Design and performance testing of a 2.29 GB/s rijndael processor", *IEEE Journal of Solid-State Circuits*, Vol.38, No.3, pp.569–572, 2003.

[12] S.K. Mathew, F. Sheikh, M. Kounavis, *et al.*, "53 Gb/s native $GF(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45nm high-performance microprocessors", *IEEE Journal of Solid-State Circuits*, Vol.46, No.4, pp.767–776, 2011.

[13] C. Zeng, N. Wu, X. Zhang, *et al.*, "The optimization circuit design of AES S-Box based on a multiple-term common subexpression elimination algorithm", *Acta Electronica Sinica*, Vol.42, No.6, pp.1238–1243, 2014. (in Chinese)

[14] S. Qu, G. Shou, Y. Hu, *et al.*, "High throughput, pipelined implementation of AES on FPGA", *International Symposium on Information Engineering and Electronic Commerce*, Ternopil, Ukraine, pp.542–545, 2009.

[15] M. Matsui and J. Nakajima, "On the power of bitslice implementation on Intel Core2 processor", *International Workshop on Cryptographic Hardware and Embedded Systems*, Vienna, Austria, pp.121–134, 2007.

[16] B. Liu and B.M. Baas, "Parallel AES encryption engines for many-core processor arrays", *IEEE Transactions on Computers*, Vol.62, No.3, pp.536–547, 2013.

[17] S.A. Manavski, "CUDA compatible GPU as an efficient hardware accelerator for AES cryptography", *IEEE International Conference on Signal Processing and Communications*, Dubai, United arab emirates, pp.65–68, 2007.

[18] T. Wollinger, M. Wang, J. Cuajardo, *et al.*, "How well are high-end DSPs suited for the AES algorithm?", *The Third Advanced Encryption Standard Candidate Conference*, pp.94–105, 2000.

[19] B. Koo, G. Ryu, T. Chang, *et al.*, "Design and implementation of unified hardware for 128-bit block ciphers ARIA and AES", *ETRI journal*, Vol.29, No.6, pp.820–822, 2007.

[20] B. Wang and L. Liu, "A flexible and energy-efficient reconfigurable architecture for symmetric cipher processing", *2015 IEEE International Symposium on Circuits and Systems*, Lisbon, Portugal, pp.1182–1185, 2015.

[21] S. Morioka and A. Satoh, "A 10-Gbps full-AES crypto design with a twisted BDD S-box architecture", *IEEE Transactions on Very Large Scale Integration Systems*, Vol.12, No.7, pp.686–691, 2004.

[22] L. Ali, I. Aris, F.S. Hossain, *et al.*, "Design of an ultra high speed AES processor for next generation IT security", *Computers & Electrical Engineering*, Vol.37, No.6, pp.1160–1170, 2011.

[23] M. McLoone and J.V. McCanny, "High performance single-Chip FPGA Rijndael algorithm implementations", *International Workshop on Cryptographic Hardware and Embedded Systems*, Paris, France, pp.65–76, 2001.

[24] X. Zhang and K.K. Parhi, "High-speed VLSI architectures for the AES algorithm", *IEEE Transactions on Very Large Scale Integration Systems*, Vol.12, No.9, pp.957–967, 2004.

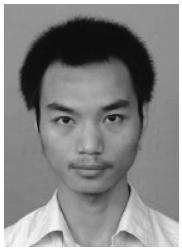
[25] A. Hodjat and I. Verbauwhede, "Speed-area trade-off for 10 to 100 Gbits/s throughput AES processor", *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, Vol.2, pp.2147–2150, 2003.

[26] A. Mericas, N. Peleg, L. Pesantez, *et al.*, "IBM POWER8 performance features and evaluation", *IBM Journal of Research and Development*, Vol.59, No.1, 2015.

[27] N. Nishikawa, K. Iwai and T. Kurokawa, "High-performance symmetric block ciphers on multicore CPU and GPUs", *International Journal of Networking and Computing*, Vol.2, No.2, pp.251–268, 2012.

[28] J. Huang, F. Miao, J. Lv, *et al.*, "Mobile phone based portable key management", *Chinese Journal of Electronics*, Vol.22, No.1, pp.124–130, 2013.

[29] Y. Huo, X. Li, W. Wang, *et al.*, "High performance table-based architecture for parallel CRC calculation", *The 21st IEEE International Workshop on Local and Metropolitan Area Networks*, Beijing, China, pp.1–6, 2015.



HUO Yuanhong was born in 1988. 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 and software-hardware co-design. (Email: hyh@bit.edu.cn)

LIU Dake (corresponding author) was born in 1957. 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 of Linköping University. He is enrolled in the China Recruitment Program of Global Experts. (Email: dake@bit.edu.cn)