Cryptoraptor: High Throughput Reconfigurable Cryptographic Processor

Gokhan Sayilar
The University of Texas at Austin
Email: gokhan@utexas.edu

Derek Chiou
Microsoft and The University of Texas at Austin
Email: derek@ece.utexas.edu

***Abstract*—This paper describes a high performance, low power, and highly flexible cryptographic processor, Cryptoraptor, which is designed to support both today's and tomorrow's symmetric-key cryptography algorithms and standards. To the best of our knowledge, the proposed cryptographic processor supports the widest range of cryptographic algorithms compared to other solutions in the literature and is the only crypto-specific processor targeting future standards as well. Our 1GHz design achieves a peak throughput of 128Gbps for AES-128 which is competitive with ASIC designs and has 25X and 160X higher throughputs per area than CPU and GPU solutions, respectively.**

I. Introduction

As the demand for secure communication bandwidth is growing at an unprecedented pace, efficient and high throughput cryptographic processing is becoming increasingly critical for good overall system performance. Implementations range from application-specific integrated circuits (ASICs), which are very high throughput but inflexible, to general purpose processor (GPP) based software, which is highly flexible, but is low throughput and requires high area and power.

In this paper, we propose a high performance, power efficient, and highly flexible cryptographic processor, *Cryptoraptor*, which supports a wide range of existing ciphers and cryptographic hash functions, and has high potential to support future algorithms. The proposed architecture with its reconfigurable substrate provides flexibility on a non-reconfigurable platform. Besides its flexibility, our design is competitive with fully-optimized Advanced Encryption Standard (AES) cores presented in the literature [45, 4, 50, 31, 14, 69, 57, 67, 33]. We also provide detailed timing, power, and area analysis on both cryptographic functional primitives and the processor. We believe that such analysis may help both cryptographic algorithm developers and hardware designers to evaluate trade-offs during the design and implementation.

II. The Processor Design Methodology

Our goal is to define a complete, flexible, and high throughput cryptographic processor that supports a wide range of ciphers and cryptographic hash functions. We focused on the architecture for overall throughput performance and flexibility and have not yet optimized the design for area or power.

We studied 148 symmetric key encryption algorithms consisting of 96 block ciphers, 26 stream ciphers, and 26 cryptographic hash functions found in widely used security protocols such as IPsec, TLS/SSL, WTLS, SSH, S/MIME, and OpenPGP, and cryptographic libraries such as OpenSSL and GNU Crypto. The results of our algorithm analysis guided the overall processor architecture which was designed to address the needs of a wide range of algorithms and balance the flexibility and performance of the system.

Since achieving high performance while having a high degree of flexibility requires special considerations, we considered both FPGA and ASIC implementations. However, our preferred target is an ASIC implementation since the design itself provides the flexibility.

III. Cryptoraptor: Cryptographic Processor

At a high level, Cryptoraptor architecture consists of an *Execution Tile* as the functional part, *State Engine (SE)* as the front-end, and a 256-entry register file.

The front-end of Cryptoraptor is controlled by a hardware state machine that is configured as part of the initial setup and remains constant as long as the algorithm does not change. The SE consists of a state counter and a small control memory block. By eliminating fetch and decode stages of conventional processors, the majority of the area and power is consumed by the functional parts of Cryptoraptor, yielding higher area and power efficiency.

The Execution Tile, which encapsulates all functional parts of the architecture, consists of multiple identical *stages*, each containing a number of *Processing Elements (PEs)* connected to the next stage by *Connection Row (CR)*. As shown in Figure 1, it consists of a number of PEs and CRs, and loopback connections from each stage to a register file. Our analysis suggests that 87.2% of cryptographic algorithms require four or less parallel PEs for maximum performance, while only 19 out of 148 algorithms benefit from eight-way and sixteen-way PEs. An eight-way or sixteen-way crypto-processor will potentially result in underutilization of resources for the most of the algorithms and increase the complexity and cost of the communication across PEs. Therefore, Cryptoraptor consists of four parallel and independently configurable PEs in each stage, called *PE row*.

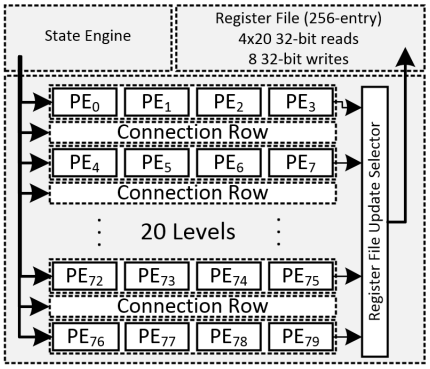

Fig. 1. Overview of the proposed architecture

Even though all algorithms can be implemented with one repeatedly used stage, it is generally the case that multiple stages can improve performance by exploiting the parallelism inherent in most algorithms. The largest number of stages observed in the analyzed algorithms is 40. In the light of our algorithm analysis, we decided that 20 stages of PEs and

978-1-4799-6278-5/14/$31.00 ©2014 IEEE

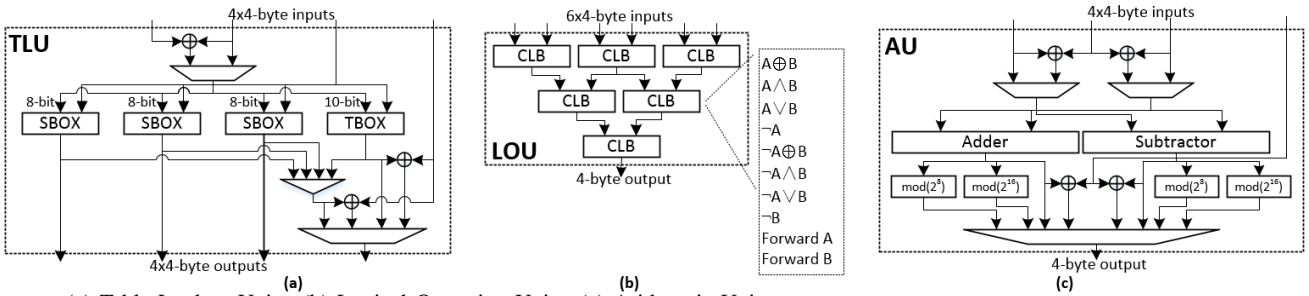

Fig. 2. (a) Table Lookup Unit - (b) Logical Operation Unit - (c) Arithmetic Unit

CRs is the optimal configuration in the Execution Tile with a 256-entry register file. Therefore, the processor can provide up to 80 logical stages by fully utilizing the control memory that provides four distinct stage images. A smaller number of pipeline stages can be realized using register file loopbacks and partial configuration of PEs in each stage.

Besides controlling connections between PEs, a CR is also responsible for controlling up to eight outputs from all stages to be stored to the register file. By providing eight parallel register file write port, we doubled the requirements of existing symmetric-key encryption algorithms. The flexibility of storing intermediate results from any stage enables us to utilize any portion of Cryptoraptor and to have a different number of pipeline stages for each algorithm. Thus, any PE individually and/or PE row can be turned on/off as needed by the algorithm. The control structure enables users to configure only required PEs and corresponding PE connectors or even a small portion of a single PE and PE connector.

A. Connection Row (CR)

The CR is a crossbar connecting outputs of each PE from stage i to inputs of each PE in stage $i+1$. Although the crossbar has a significant impact on cycle time (Section V), it increases the flexibility of Cryptoraptor. This unit could have been structured with a predefined set of connections between FUs. However, doing so would limit the processor's ability to support future algorithms. Similar to the control signal memory of PE, a memory block is placed next to each CR to provide configuration bits to the CR as needed.

B. Processing Element (PE)

Our algorithm analysis shows that functional primitives used in cryptographic algorithms can be clustered into five operation classes: logical, shift/rotate, table lookup, arithmetic, and permutation/expansion, are used 95.9%, 66.9%, 56.8%, 51.4%, and 41.9% of the algorithms respectively. Thus, a PE consists of five bypassable and independently configurable functional units (FUs) that can work concurrently: an Arithmetic Unit (AU), a Logical Operation Unit (LOU), a Table Lookup Unit (TLU), a Shifter-Rotator Unit (SRU), and a Permutation-Expansion Unit (PEU). Since one fourth of the algorithms use only byte-wise rotation, a selectable one, two or three byte rotate operations are added to the outputs of AU, LOU, and TLU, which provides both original and rotated outputs. Doing so reduces the total number of cycles with negligible overhead on the processor cycle time.

A small memory block is associated with each PE to store control signals and enable reconfiguration of that PE. Those signals mostly remain constant throughout the execution of the algorithm. Our evaluation assumes that such structures can store up to four sets of signals that can be sequenced by the SE of Cryptoraptor to provide more capacity, by remapping a stage cycle-by-cycle, or to provide the ability to switch to another algorithm in a single cycle.

a. Table Lookup Unit (TLU): The table sizes, entry widths, addressing schemes, number of different tables, and the number of parallel tables vary greatly between cryptosystems. Therefore, the table lookup structure of a generic reconfigurable cryptographic processor should ideally be capable of supporting table size requirements of all existing algorithms for both the size of one table (4KB) and total size of tables in algorithm (16KB), as well as maximum number of parallel lookup operations (16 operations). Since more than 70% of the algorithms using table lookup uses 256x4B tables, we use that structure. However, to support a wider range of cryptographic algorithms, we provide three 256x4B and one 1024x4B tables (named as SBOX and TBOX, respectively) addressed as shown in Figure 2. The TLU provides three ways to perform a table lookup: (i) one table lookup with up to a 10-bit address that returns a 4B output, (ii) 4 table lookups with each byte of a 4B input used as one address that returns four 4B outputs, and (iii) one table lookup with each byte of a 4B input as an address that outputs each byte of one 4B output. Larger lookup tables can be supported by splitting them into multiple SBOXes/TBOXes, and using multi-stage lookup operations.

In more than 69% of symmetric-key encryption algorithms, table lookup operations are preceded and/or followed by an XOR operation. Thus, bypassable XOR operations are placed before and after the memory lookup blocks. The TLU defines the critical path of Cryptoraptor (Section V).

b. Arithmetic Unit (AU): Our studies show that 75.3% of modular arithmetic operations use (2^{32}) as the base value, while more than 90% of modular arithmetic operations can be realized by masking the result of a standard integer arithmetic with an AND operation. The current AU is capable of doing addition and subtraction on 8, 16, and 32-bit granularities. Doing so supports 94.6% of studied cryptographic algorithms efficiently while 8 out of 148 algorithms require either a combination of existing FUs or new support for bases other than (2^8), (2^{16}), and (2^{32}). Due to its insignificant improvement on flexibility but high impact on overall cycle time (Section V), we choose not to include a multiplier in our current datapath and multiplication can be realized using a combination existing FUs. However, a two-staged multipler can be integrated for higher algorithm coverage without affecting the critical path.

Since more than half of algorithms use XOR and arithmetic operations back-to-back, XORs are placed before and after arithmetic operations (Figure 2).

c. Logical Operation Unit (LOU): Since the TLU defines the critical path, the LOU can contain a lot of functionality without affecting the critical path. Our analysis shows that 82.4% of cryptographic algorithms process two consecutive logical operations while 58.8% of all algorithms process three, and 57.7% of the cryptographic hash functions process

four or more. To support sequences of logical operations the LOU is separated from the AU to process more operations concurrently and, therefore, achieves higher throughput. As shown in Figure 2, the LOU consists of a three-level reduction tree with six independently configurable and bypassable logic blocks (CLBs). Each CLB is capable of performing four logic primitives (AND, OR, NOT, and XOR) on its operands as well as applying bitwise inversion to any operand (i.e. ¬A&B). The LOU can support most functions found in common cryptographic hash functions and complex logic reduction functions with up to six inputs.

$$F(X,Y,Z) \quad = \quad (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge z) \qquad (1)$$

For example, Equation 1 is a 6-input function that is used in MD4, SHA-1, and SHA-2. Such a simple function requires 5 iterations to produce the result using a simple arithmetic logic unit. However, it can easily be implemented in a single cycle with the three-level reduction tree structure in the LOU.

d. Permutation/Expansion Unit (PEU): Bit manipulation is the key operation to provide non-linearity in some cryptographic algorithms such as DES. The PEU can merge, manipulate, and expand two 32-bit inputs to generate one up to 64-bit output (two 32-bit outputs). Each bit of the output can be set to any input bit. The crossbar structure of PEU increases the flexibility of Cryptoraptor for both existing and future algorithms.

e. Shifter/Rotator Unit (SRU): The 32-bit unit supports a variable shift/rotate operation in both directions. Since the rotation on 8 or 16-bit data is not common among cryptographic algorithms, and can be implemented using PEU, smaller granularity rotations are not supported by this unit. Since more than 40% shift/rotate operations are preceded and/or followed by a logical operation, the flexibility of performing logical operations (AND, OR, NOT, and XOR) before and after shift/rotate operations is provided.

IV. Algorithm Mapping & Mapper Tool

The reconfigurability and performance of Cryptoraptor are evaluated by using a custom toolchain to map 16 cryptographic algorithms: (i) 10 block ciphers; AES, Blowfish, Camellia, Cast128, DES, GOST, Kasumi, SEED, RC5, and Twofish, (ii) 2 stream ciphers RC4 and Phelix, and (iii) 4 cryptographic hash functions MD4, MD5, SHA-1, and SHA-2. This algorithm selection is based on mostly used security protocols (IPsec, TLS/SSL, WTLS, SSH, S/MIME and OpenPGP) and cryptographic libraries (OpenSSL and GNU Crypto). Some algorithms that are not in existing protocols and libraries are also mapped to stress the flexibility of Cryptoraptor. The Table I summarizes our rationale behind including these 16 algorithms in the mapping process.

To enable easier implementation, more efficient and optimized algorithm mapping, and higher throughput, a simple cryptography assembly language is introduced (Table II), which allows users to define and use variables, arrays, tables, constants, and permutation tables. Instructions can also operate on 32-bit immediate values. Since the multiplication and modular arithmetic with arbitrary modulo bases are not currently supported, they are not included in the language. We also implemented a custom toolchain which is fully aware of the underlying processor architecture and optimizes the input mapping for throughput.

TABLE II. Instruction List

Operation Class	Instructions
AU	ADD, ADD8, ADD16, ADDi, ADD8i, ADD16i, SUB, SUB8, SUB16, SUBi, SUB8i, SUB16i
LOU	AND, OR, XOR, NOT, ANDi, ORi, XORi
TLU	SBOX, SBOX_M, SBOX_P, STR
SRU	SHR, SHL, SHRi, SHLi, ROTR, ROTL, ROTRi, ROTLi, BROTR, BROTL
PEU	PERM, PERM32_64, PERM64_32, PERM64_64
Helper	REPEAT, MOVE, SWAP

The toolchain unrolls round loop(s), generates a dataflow graph, and optimizes the operation sequence for underlying hardware. It issues the operations to available FUs as soon as their operands are ready. Besides its own optimization process, the assembly language and toolchain also allow users to hand-tune their implementation. The automated toolchain enables achieving throughput and resource utilization that are greater than or equal to well-studied hand-based mapping.

Our experience leads us to believe that mapping algorithms is straightforward as long as all of the necessary functional

TABLE I. Implemented Algorithms

	Security Protocols	Cryptographic Libraries	Speciality
AES [52]	Almost all	Almost all	Most widely used algorithm, stresses maximum parallel lookup table
Blowfish [64]	IPsec	OpenSSL	Base structure for some block ciphers, example Arithmetic-XOR pattern
Camellia [6]	IPsec, TLS	Crypto++, Cryptospecs, OpenSSL	Example algorithm that requires special attention to achieve high performance, example byte-wise rotator
CAST-128 [2]	IPsec, PGP	Crypto++, Cryptospecs, OpenSSL, GnuPG	Base structure for some block ciphers, example changing round structures, and Arithmetic-XOR pattern
DES [1]	IPsec, SSL, TLS	Almost all	Example bit permutation and unorthodox operation width
GOST [55]	-	Crypto++, Cryptospecs	Doesn't fit well, example Shift-Logic pattern
Kasumi [47]	as A5/3 in UMTS, GSM, GPRS	Cryptospecs	Data dependent and fairly complex structure due to its unbalanced rounds and unorthodox table sizes (7 and 9-bit addressing)
RC5 [60]	S/MIME	Crypto++, Cryptospecs, OpenSSL	Good on software, doesn't fit well in Cryptoraptor
SEED [42]	CMS, IPsec, SSL, S/MIME, TLS	Crypto++, Cryptospecs, OpenSSL	Recursive round structure, example algorithm that requires special attention to achieve high performance
Twofish [65]	OpenPGP	GnuPG	Complex structure and example operation patterns
RC4 [74]	TLS, WEP, WPA	Crypto++, GnuPG, Cryptospecs, OpenSSL	Most widely used stream cipher, table updates, register file usage
Phelix [75]	-	-	Very complex structure, use of patterns
MD4 [59]	PGP, S/MIME	Crypto++, Cryptospecs, OpenSSL	Base structure for several hash functions, stresses LOU structure, changing round structures
MD5 [58]	IPsec, NTLM, SSL, S/MIME, TLS	Almost all	Base structure for several hash functions, stresses LOU structure, changing round structures
SHA-1 [15]	IPsec, PGP, SSH, SSL, S/MIME, TLS	Almost all	Most widely used hash function, changing round structures
SHA-2 [20]	Bitcoin, IPsec, PGP, SSH, SSL, S/MIME, TLS	Almost all	Complex round structure, replacing SHA-1

blocks are available. Multiplication, used in 11.5% of the analyzed algorithms, is a notable exception. Our algorithm analysis suggests that 86.5% of studied algorithms can be mapped efficiently onto current Cryptoraptor architecture (Section V). Since even multiplication can be performed using existing FUs and we provided more functionality than studied algorithms require, we strongly believe that Cryptoraptor can support all existing algorithms.

V. Processor Analysis

A. Implementation

We have developed a highly modular, fully configurable architecture using Verilog HDL. We synthesize our RTL code into a gate level structure using Synopsys Design Compiler (DC) and FreePDK45^{TM} v1.4 45nm standard-cell CMOS technology [68]. However, due to the lack of a memory compiler, register arrays are used for the memory blocks while analyzing the maximum frequency of Cryptoraptor. Even though the use of register arrays potentially increases the cycle time, it achieves a frequency of 1GHz, where each pipeline stage consists of four PEs, and a CR. We have not yet optimized our design for timing, area, and power. The CACTI 6.5 Memory Model [51] is used to get area metrics for the memory blocks. The area numbers are generated from the functional area numbers produced by DC, the memory area numbers produced by CACTI, plus 10% for error.

Even though implementing Cryptoraptor on FPGAs is somewhat redundant, since the reconfigurability is provided within Cryptoraptor itself, we also synthesize our RTL code to a Xilinx Virtex-6 FPGA using ISE Design Suite 14.6 without applying any FPGA-related optimizations. We use Block RAMs (BRAMs) in the FPGA for memory blocks. Even though an FPGA could be reconfigured as the algorithm changes, we do not assume that is possible. Despite the fact that the excessive use of multiplexers in the design has a negative impact on our FPGA cycle time, we achieve 203.8MHz with the same configuration used in ASIC version.

B. Timing, Area, and Power Analysis

We synthesize FUs and structures in the design separately to find out which FU or structure creates a bottleneck in Cryptoraptor. Such analysis also allows us to further improve the design other FUs which are not on the critical path. We used "-uniquify", "-ungroup", and "-flatten" DC optimizations, which removed module boundaries and synthesized the design as a whole block.

TABLE III. The timing, area, and power analysis of funtional units

	Cycle time (ns)	**Area** (mm^2)	**Internal Power** (mW)	**Switching Power** (mW)	**Leakage Power** (mW)	**Dynamic Power** (mW)
AU	0.51	0.0107	3.97	3.07	0.0479	7.08
LOU	0.48	0.0080	2.49	2.30	0.0269	4.81
TLU	0.58	0.0110	51.54	1.35	0.0399	53.55
SRU	0.55	0.0093	1.90	1.84	0.0303	3.78
PEU	0.23	0.0156	14.35	4.07	0.0748	18.50
Byte rotator(BR)	0.07	0.0011	0.58	0.52	0.0021	1.10
AU (with BR)	0.56	0.0117	4.06	3.22	0.0513	7.33
LOU (with BR)	0.53	0.0087	2.62	2.41	0.0282	5.06
TLU (with BR)	0.63	0.0115	51.75	1.48	0.0427	53.89
AU (w/o bundle)	0.42	0.0035	1.23	0.87	0.0147	2.12
SRU (w/o bundle)	0.30	0.0063	1.86	1.82	0.0220	3.71
Multiplication	0.91	0.0075	4.09	3.06	0.0374	7.18

Due to its memory operations and complex structure, TLU is the longest path in the PE design (Table III). Since we cannot make the TLU design faster, the functionality of other units are improved, yielding higher compute capabilities. Thus, we bundle arithmetic operations with XOR and shift/rotate operations with logical operation blocks. The Table III also summarizes the overhead byte rotator and the operation bundles introduced in the design. Integrating byte rotators to AU, LOU, and TLU has a negligible effect (0.04-0.05ns when merged) on cycle time, area, and power.

A single-staged multiplier requires 0.91ns to multiply two 32-bit integer, which would significantly increase the cycle time of PE, yielding lower clock frequency for Cryptoraptor. Our analysis shows that supporting multiplication operation would only increase the algorithm coverage of Cryptoraptor by 8.1 percent. Thus, a single-stage multiplier results in 25% decrease in clock frequency and does not improve algorithm coverage significantly. Unlike its effects on cycle time, the multiplication unit does not have a serious impact on the area.

Like its effects on cycle time, TLU requires the highest dynamic power by far with 53.5mW due to its memory blocks. However, there is also a huge gap between the power usage of PEU and other FUs (Table III). This finding clearly shows the trade-off between flexibility and power usage.

The Table IV shows cycle time, area requirements, and power consumption of each sub-structure as well as Cyrptoraptor as a whole. As discussed before, full crossbar structure between PE rows has a significant impact on the cycle time of Cryptoraptor. The functional part of a row forms only 65% of the overall cycle time. Even though one *Full Row*, which is a merged version of one PE and CR, requires 0.95ns, the register file connections and capacitance effects of all connections increases the total cycle time to 1.0ns.

The CR between PE rows does not have a significant impact on power usage. The majority of total power is consumed by the FUs. The DC and optimization flags result in different power predictions and make it more complicated than a simple calculation combining sub-modules. For example; one PE row, CR, and pipeline registers require 277.17mW, 2.60mW, and 292.57mW dynamic power respectively when they are synthesized separately. However, DC reports 360.37mW dynamic power usage for one Full Row (which consists of one pair of a PE row and CR with pipeline registers), where only 119.39mW is attributed to pipeline registers.

TABLE IV. The timing, area, and power analysis of sub-modules

	Cycle time (ns)	**Area** (mm^2)	**Internal Power** (mW)	**Switching Power** (mW)	**Leakage Power** (mW)	**Dynamic Power** (mW)
PE	0.63	0.0344	63.42	4.21	0.15	68.40
PE Row	0.63	0.1448	254.37	18.35	0.60	277.17
Connection Row	0.47	0.0627	1.58	0.86	0.17	2.60
Pipeline Register	0.09	0.0221	288.74	3.71	$1.20\text{x}10^{-7}$	292.57
Full Row	0.95	0.2115	336.20	19.35	0.96	360.37
Execute Tile	1.00	4.5428	5229.32	303.75	19.96	5602.65
Cryptoraptor	1.00	6.3244	5802.80	303.75	23.36	6207.04

The functional and execution parts of Cyrptoraptor use 71.8 percent of overall die area. With a 256-entry register file (1.78mm^2), Cryptoraptor requires only 6.32mm^2, which is approximately 34X and 78X smaller than existing CPUs and GPUs, respectively. The Table IV also shows that 90.6 percent of total power is consumed by functional and execution part of Cryptoraptor. In an in-order Reduced Instruction Set Computer (RISC), a large fraction of energy dissipation can be attributed to the instruction supply; 37% for fetching, 18% for decoding, and 14% for issuing an instruction [29]. Using a compact finite state machine representation for control flow in algorithms improve energy and area efficiency by simplifying the front-end structure of a traditional processor.

C. Performance Analysis

Since the most of prior art is hard-coded, hand-tuned, and not parametrized, fair performance comparisons are not easy. The limited architectural insight and incomplete data provided in research papers make comparisons even harder. Moreover, there is no standard set of base settings to present the performance of cryptographic algorithms. The choice of operation mode to be feedback (CBC, CFB, and OFB) or non-feedback (ECB and CTR) plays a crucial role on the throughput of a design. Even though feedback mode throughput provides better insight about the performance of a design, the throughput of existing solutions is provided for both modes as well as their clock frequencies, lithographies, and pipeline depths. The presented results are not scaled to a particular technology or device to avoid unrealistic advantages/disadvantages [45, 41].

Even though we aim to achieve high throughput for all algorithms, we can only compare our performance on AES due to the lack of published results for any other algorithm. Though Cryptoraptor does not have the highest performance, it is competitive with ASIC solutions, while providing much more flexibility. However, we should emphasize that we compare our flexible processor against to the designs that are hand-tuned specifically for AES and do not support other algorithms.

TABLE VI. AES PERFORMANCE ON ASICS

	Lith. (nm)	Parallel Stream	Cycles	Freq. (MHz)	CBC ($Gbps$)	CTR ($Gbps$)
Saravanan [63]	180nm	1	80	333.0	0.53	10.66
Amphion [5]	180nm	1	1	200.0	25.60	25.60
EnSilica [18]	65nm	1	11	500.0	5.82	64.00
Mathew [46]	45nm	4	20	2615.0	16.74	66.94
Hodjat [31]	180nm	1	41	606.0	1.89	77.57
Swankoski [69]	160nm	1	50	680.3	1.74	87.07
Ip Cores, Inc [35]	90nm	1	10	824.0	10.55	105.47
Morioka [50]	130nm	1	10	909.0	11.64	116.35
Ali [4]	180nm	1	21	1015.0	6.19	129.92
Liu [45]	65nm	1	152	1210.0	1.02	153.70
Cryptoraptor	45nm	1	20	1000.0	6.40	128.00

Table VI shows that an ASIC implementation of Cryptoraptor achieves competitive throughput results compared to AES-specific ASIC cores. With its outer-round pipelined structure and highly tuned datapath, Morioka's AES-only implementation [50] on an old 130nm technology achieves high throughput on both CBC and CTR modes, 11.64Gbps and 116.35Gbps respectively, while our reconfigurable processor achieves peak AES throughputs of 6.40Gbps and 128Gbps. Since the number of pipeline stages has a negative impact on CBC throughput, our throughput is roughly half of Morioka's.

With its higher frequency, 152 pipeline stages, and its many core structure, Liu's AES processor [45] achieves the highest AES-128-CTR throughput of 153.70Gbps. Due to its very deep pipeline, however, it is not able to provide high performance on feedback modes. Liu's AES processor (6.63mm^2) only supports AES while Cryptoraptor (6.32mm^2) is capable of supporting a wide range of algorithms. Cryptoraptor achieves similar throughput per area as Liu's many core solution, 20.25Gbps/mm^2 and 23.18Gbps/mm^2 respectively.

Amphion's AES core [5] running at 200MHz achieves the highest AES-CBC throughput by processing 10 rounds in a single cycle. Even though its internal structure is not publicly available, it is obvious that the core cannot achieve a high CTR throughput due its low clock frequency. The 10 stage pipelined 824MHz AES core on 90nm technology introduced by Ip Cores provides a balanced performance on both CBC and CTR mode. Both designs clearly show the importance of the balance between the pipeline depth and the clock frequency.

TABLE VII. AES PERFORMANCE ON FPGA-BASED SOLUTIONS

	FPGA	Lith. (nm)	Cycles	Freq. (MHz)	CBC ($Gbps$)	CTR ($Gbps$)
Jarvinen [38]	Virtex 2	130nm	41	139.10	0.43	17.80
Swankoski [70]	Virtex 2	130nm	10	147.00	1.88	18.82
Hodjat [32]	Virtex 2	130nm	41	168.30	0.53	21.54
Zhang [79]	Virtex 1	130nm	70	168.40	0.31	21.56
Good [25]	Spartan 3	90nm	70	196.10	0.36	25.10
Iyer [37]	Virtex 2	130nm	50	206.84	0.53	26.48
Good [26]	Virtex 2	130nm	240	222.90	0.12	28.53
Rizk, [61]	Virtex 4	90nm	20	223.00	1.43	28.54
Yoo [78]	Virtex 2	130nm	30	232.60	0.99	29.77
Tim Good [26]	Virtex 3	90nm	120	240.90	0.26	30.84
Fan [19]	Virtex 2	90nm	50	250.00	0.64	32.00
EnSilica [18]	Virtex 6	40nm	11	275.00	3.20	35.20
Ali [4]	Stratix II GX	180nm	21	282.50	1.72	36.16
Wang [73]	Virtex 6	40nm	66	344.12	0.67	44.05
Mercoratech [49]	Virtex 6	40nm	10	357.00	4.57	45.70
Deshpande [62]	Spartan 6	45nm	80	430.00	0.69	55.04
Hossain [33]	Stratix II GX	90nm	23	450.05	2.50	57.61
Swankoski [69]	Virtex 4	90nm	50	519.18	1.33	66.46
Soliman [67]	Virtex 5	65nm	40	557.00	1.78	71.30
Qu [57]	Virtex 5	65nm	10	576.07	7.37	73.74
Chen [14]	Virtex 4	90nm	10	645.70	8.26	82.65
Cryptoraptor	Virtex 6	40nm	20	203.80	1.30	26.09
Cryptoraptor	45nm	45nm	20	1000.0	6.40	128.00

There exists a rich literature on high performance AES hardware architectures targeting FPGAs, as summarized in Table VII. The AES core proposed by Chen [14] adapts outer-round pipelining scheme, and is stated to achieve a very high frequency of 645.70MHz on a Xilinx Virtex-4 FPGA, producing the highest AES-CTR throughput of 82.65Gbps. We try to replicate the work using the source code provided by authors and synthesize to exact same FPGA family with highest speed grade with all optimizations on, using ISE Design Suite 14.6. However, the highest clock frequency that we are able to produce is 284.43MHz, yielding 36.41Gbps as opposed to 82.65Gbps. Similarly, 10-staged AES core reported by Qu [57] claims an astonishingly high clock frequency on Virtex-5 while Soliman's solution [67] is not able to achieve such a high clock frequency on the same FPGA even with its deeper inner-round pipelined structure.

On the other hand, even commercial high performance AES cores introduced by EnSilica [18] and Mercoratech [49]

TABLE V. AES PERFORMANCE ON GPPS

	Architecture	Lith. (nm)	Config. ($core/warp$ x $thread$)	Freq. (MHz)	Through-put ($Gbps$)	Area (mm^2)	Through-put per area ($Gbps/mm^2$)	Power per area (W/mm^2)
Nishikawa [53]	Core i7-2600K	32nm	1 x 1	3400	1.90	216	0.009	0.4398
Nishikawa [53]	Core i7-2600K	32nm	4 x 8	3400	7.50	216	0.035	0.4398
Nishikawa [53]	Core i7-2600K	32nm	1 x 1	3400	25.10	216	0.116	0.4398
Nishikawa [53]	Core i7-2600K	32nm	4 x 4	3400	44.20	216	0.205	0.4398
Akdemir [3]	Core i7-980X	32nm	1 x 1	3300	6.30	239	0.026	0.5439
Akdemir [3]	Core i7-980X	32nm	6 x 12	3300	72.30	239	0.303	0.5439
VIA Tech. [72]	VIA C7	90nm	1 x 1	2000	25.00	30	0.833	0.6666
Zola [54]	GTX 260	55nm	27 x 256	1242	30.00	576	0.052	0.3159
Iwai [36]	GTX 285	55nm	60 x 512	1500	35.20	490	0.072	0.4163
Nishikawa [53]	GTX 285	55nm	60 x 512	1242	35.20	490	0.072	0.4163
Nishikawa [53]	Tesla C2050	40nm	28 x 512	1150	48.60	539	0.090	0.4415
Bos [9]	GTX 295	55nm	120 x 512	1240	59.60	470	0.127	0.6148
Cryptoraptor	ASIC	45nm	1 x 1	1000	128.00	6.32	20.253	0.9777

TABLE VIII. PERFORMANCE AND UTILIZATION ON PROPOSED PROCESSOR

	Algorithm Specification			Manual mapping			Automated mapping			Manual mapping						Automated mapping					
	Parallel Stream	Round	Block size	Cycles	CBC ($Gbps$)	CTR ($Gbps$)	Cycles	CBC ($Gbps$)	CTR ($Gbps$)	LOU	TLU	AU	SRU	PEU	PE	LOU	TLU	AU	SRU	PEU	PE
AES	1	16	64	20	6.40	128.00	20	6.40	128.00	50%	0%	50%	0%	0%	100%	50%	0%	50%	0%	0%	100%
Blowfish	4	16	64	48	5.33	85.33	48	5.33	85.33	0%	67%	33%	0%	0%	100%	33%	67%	33%	0%	0%	100%
Camellia	2	18	128	80	3.20	64.00	73	3.51	64.00	80%	0%	20%	3%	0%	60%	71%	0%	25%	3%	0%	71%
CAST128	4	16	64	80	3.20	64.00	80	3.20	64.00	6%	54%	20%	20%	0%	100%	6%	54%	20%	20%	0%	100%
DES	2	16	64	48	2.67	42.67	48	2.67	42.67	0%	0%	17%	0%	50%	50%	0%	0%	33%	0%	50%	83%
GOST	4	32	64	96	2.67	51.20	98	2.61	51.20	34%	33%	33%	33%	0%	100%	34%	33%	33%	33%	0%	100%
Kasumi	1	6	64	64	1.00	16.00	64	1.00	16.00	25%	0%	20%	9%	5%	50%	25%	9%	20%	0%	5%	48%
RC5	4	12	64	48	5.33	85.33	48	5.33	85.33	0%	50%	0%	50%	0%	100%	0%	50%	0%	50%	0%	100%
SEED	1	16	128	160	0.80	16.00	152	0.84	16.00	13%	8%	30%	0%	0%	50%	14%	8%	32%	0%	0%	50%
Twofish	2	16	128	80	3.20	64.00	80	3.20	64.00	20%	40%	30%	20%	0%	90%	21%	40%	40%	20%	0%	81%
RC4	4	4	32	32	4.00	-	32	4.00	-	0%	38%	50%	0%	0%	100%	0%	38%	50%	0%	0%	100%
Phelix	2	1	32	10	6.40	-	10	6.40	-	0%	70%	0%	80%	20%	100%	0%	70%	0%	80%	20%	100%
MD4	2	48	128	145	1.77	-	144	1.78	-	17%	50%	0%	17%	0%	66%	22%	50%	0%	17%	0%	73%
MD5	2	64	512	257	3.98	-	254	4.03	-	12%	50%	0%	12%	0%	62%	19%	63%	0%	13%	0%	85%
SHA-1	2	80	512	240	4.27	-	225	4.55	-	17%	83%	0%	33%	0%	100%	18%	75%	0%	28%	0%	99%
SHA-2	1	64	512	320	1.60	-	320	1.60	-	20%	40%	0%	30%	0%	55%	25%	35%	0%	30%	0%	50%
									Average utilization:	18%	36%	19%	18%	5%	80%	21%	37%	21%	18%	5%	84%

achieve significantly lower clock frequencies on a faster FPGA family with the same number of pipeline stages. Table VII clearly shows the impact of the number of pipeline stages on AES-CTR throughput. Despite their higher frequency, while most of the AES cores targeted FPGA has poor performance in feedback modes, they achieve very high throughput on non-feedback modes due to their deep pipelines ranging from 40 to 240 stages. The performance results of FPGA solutions suggest that an outer-round pipelined architecture yields better overall performance for AES by balancing the pipeline depth and the clock frequency of a design. Because the Cryptoraptor design is tuned for an ASIC implementation, it contains an aggressive connection network and excessive use of multiplexers. Even though it only runs at 203.8MHz it still achieves reasonable throughput of 1.30Gpbs AES-CBC and 26.80Gbps AES-CTR.

Current generation CPU and GPU solutions also generate throughput results of up to 72.30Gbps and 59.60Gbps respectively (Table V). However, current generation GPPs are large, up to 539mm^2. Even though we have not put much effort to minimize the area, we achieve orders of magnitude higher throughput per area than commercial GPPs. A similar analysis on ASICs and FPGAs is not available due to missing or inconsistent area constraints. Also, it is hard to compare numbers from different FPGA generations, as the underlying technology performance and area differ significantly.

As mentioned earlier, there is no literature on hardware implementations of other algorithms that we mapped. For that reason, we are unable to provide such a comparison. However, the performance of other algorithms on Cryptoraptor is summarized in Table VIII. Our automated toolchain is able to achieve mostly higher, or equal throughput compared to hand-tuned mapping by extracting more parallelism.

D. Resource Utilization

To provide a high degree of flexibility, redundant units and connections are introduced in Cryptoraptor. Therefore, not all algorithms can utilize all resources available in Cryptoraptor. The Table VIII summarizes resource utilization of the algorithms that are mapped onto Cryptoraptor. The utilization analysis includes only FUs and PEs that are processing some operations; thus, operand forwarding is not included in resource utilization.

The Table VIII also shows the resource utilization comparison between the toolchain and hand-tuned mapping. Our naive automated toolchain produces slightly different FU patterns than ones in manual mapping; however, is better at finding parallelism between rounds. Thus, it achieves slightly different FU utilization for each algorithm while improving overall PE utilization by 4% on the average.

Even though the average PE utilization is pretty high, utilization of individual FUs is low on both manual and automated mappings. The added redundancy for flexibility and unpredictability of future algorithms increase area significantly. However by power gating unused FUs, which we have not yet done, we believe that power can be significantly reduced.

E. Algorithm Coverage

Though not all algorithms are implemented due to the time required to do so, we strongly believe that 86.5% of the studied algorithms can be easily supported with the current structure of Cryptoraptor. The remaining 13.5% require additional hardware or logic to be supported efficiently (Table IX).

TABLE IX. ALGORITHMS COVERAGE OF CRYPTORAPTOR

	Block Ciphers	Stream Ciphers	Hash Functions	All
Not supported (Mult only)	10.42%	3.85%	3.85%	8.11%
Not supported (Mod only)	2.08%	0.00%	3.85%	2.03%
Not supported (Mod&Mult)	3.13%	0.00%	7.69%	3.38%
Supported	84.38%	96.15%	84.62%	86.49%

The lack of modular arithmetic support for arbitrary choice of modulus and the lack of dedicated multiplication units prevent us to cover all algorithms. However, we should emphasize that both multiplication and modular arithmetic on bases other than (2^8), (2^{16}), and (2^{32}) can be realized using a combination of existing FUs for higher algorithm coverage in exchange for performance.

VI. RELATED WORK

Many GPPs include crypto instructions. For example, Intel added six SSE instructions and hardware support in their CPUs to speed up AES [3]. IBM [10] and Oracle [24] introduced crypto engines in their latest processors to accelerate a predefined set of cryptosystems: AES, RC4, DES, Kasumi, Camellia, MD5, SHA-1, and SHA-2. Both the IBM and Oracle designs consist of algorithm-specific instructions and dedicated hardware units for each algorithm, restricting their capabilities to the supported algorithms. Several research projects [7, 12, 16, 27, 39, 43, 66, 76] propose instruction set architecture extensions (ISEs) or hardware extensions to GPPs; however, they are also restricted. Even though the Parallel Table Lookup [21] and Parallel Read instructions [44] work are intended to be algorithm-independent extensions, they are not useful for the algorithms that do not have table lookup operations and are not sufficient to implement all steps of algorithms that do have table lookup operations. The ISE proposed by Grabher [27] accelerates a wider range of cryptographic algorithms. Nevertheless, it is also limited to a subset

of cryptosystems, specifically ones that operate on data in a bit-oriented manner rather than in a word-oriented manner. Even though ISE proposals improve the software performance of cryptographic algorithms, the added functionality is generally limited to only speeding a limited subset of cryptosystems.

In addition to existing ISE solutions, there are many algorithm-specific designs for high throughput and area and power efficiency, specifically for AES [4, 8, 14, 19, 25, 26, 32, 31, 33, 37, 38, 45, 46, 50, 57, 61, 62, 63, 70, 73, 79]. Optimized hardware implementations have also been described for Camellia [77], DES [48], Twofish [40], Blowfish [19], RC4 [23], SHA1, and SHA2 [13]. Although the objectives of such solutions is not the same as ours, they inspired us to design a more optimized processor and to redesign algorithms to be better suited for Cryptoraptor.

To bridge the gap between GPPs and ASIC solutions, FPGA-based solutions and application-specific cryptographic processors have been proposed, each with its own ISA, datapath, and targeted cryptography standards. There are only a handful of configurable cryptographic processors with generic modules suitable for a set of cryptographic algorithms [11, 71, 28, 22, 30]. However, they provide a limited flexibility and/or throughput. COBRA [17] is a notable exception. It is a reconfigurable array structure, akin to a specialized FPGA, for block ciphers whose design is based on the analysis of 41 block ciphers. It is, however, restricted to block ciphers that operate on plaintext with block sizes of 64 and 128 bits. The main difference between COBRA and other configurable cryptographic processors is that its datapath needs to be recompiled/reconfigured for each algorithm, resulting in different clock frequencies and areas. Whereas, Cryptoraptor is more like a microprocessor that can be programmed for different cryptographic algorithms. Our clock frequency does not change regardless of the implemented algorithm.

IBM has introduced the IBM PCIe Cryptographic Coprocessor [34] that provides a high-security, high-throughput cryptographic subsystem with specialized hardware to perform only AES, DES, 3DES, RSA, SHA-1, and SHA-2. Even though its advantages include better area, power and cost efficiencies, and higher throughput than GPP solutions, it can support only a small and predefined set of cryptographic algorithms.

The existing configurable cryptographic processors provide limited flexibility and/or low performance since their structures rely on either dedicated hardware module for each algorithm aimed to support or FUs and structures that are tailored to a small and predefined set of cryptosystems; mostly focus on AES, DES, MD5, SHA-1, and SHA-2. The most common structural limitations in FUs are (i) limited block size support, (ii) insufficient lookup table structure, (iii) insufficient bit-wise permutations, and (iv) fixed modulus in modular arithmetic units. Even though proposed architectures are designed for high throughput and multi-algorithm support, none of them has been evaluated for cryptographic algorithms other than a couple of targeted ones.

There are also projects [29, 41, 56] designing a domain independent configurable processor intended to lower design effort and eliminate hardware modifications when requirements change. The main purpose of such systems is to enable efficient high performance computing. Despite the fact that the flexibility and ease of design are listed as the key benefits of these systems, the applicability of the proposed techniques has not been evaluated on more than one application or domain. Generic reconfigurable processors either fail to achieve very high crypto throughput due to the lack of crypto-specific instructions or require recompiling and re-synthesizing hardware to adapt their internal structures to cryptographic algorithms.

Beside limited flexibility and low performance, most of the proposed alternative solutions rely on traditional instruction fetch and decode structures to make control decisions, which increases the complexity of hardware, requires more area, and consumes the majority of total energy used in the whole processor [29]. With our processor architecture, we improve energy efficiency by simplifying the front-end structure of a traditional processor. We use a compact finite state machine for control flow to reduce the energy and area requirements.

VII. Limitations & Future Work

We provided a comprehensive timing, area, and power analysis on architectural structures of the reconfigurable cryptographic processor, Cryptoraptor. However, there are still possible improvements for the FUs as well as overall processor design to achieve a higher degree of flexibility and the highest possible throughput for all.

One major limitation is the limited addressing structure of the TLU. The current structure and limits are solely based on our analysis of existing algorithms. Since it is almost impossible to predict the requirements of future standards, the current TLU and limitations may or may not be an impediment to efficient implementations of future algorithms.

Since the multiplication is one of the main limitations of Cryptoraptor, which restricts the algorithm coverage for both existing and future cryptographic algorithms, a pipelined multiplication unit is a good candidate for introduction and would yield an 8.1% increase in the algorithm coverage, but would add 0.33ns to the cycle time if introduced naively. The lack of support for varying modulo in modular arithmetic operations is another important factor that limits the flexibility of Cryptoraptor. Even though modular arithmetic operations can be implemented using existing FUs, the design can be extended to support varying and unorthodox modulo bases, which may increase the algorithm coverage by 2% while having both multiplication unit and modular arithmetic support may increase algorithm coverage by 13.5%.

Our current toolchain has limited capabilities and does not provide aggressive optimizations. A more powerful automatic mapping structure for cryptographic algorithms using a crypto-specific language or ideally high level language like C/C++ can be developed for robust and high performance mapping of the algorithms. Such a toolchain might also solve the issues related to multiplication and modular arithmetic automatically by mapping these operations to existing hardware on the fly.

Public key cryptography and the security of the processor are currently beyond the scope of this work and extending our hardware support for public key encryption is left as future work.

VIII. Conclusions

In conclusion, we provided a comprehensive literature review on cryptographic algorithms and detailed analysis on the specifications and requirements of various cryptosystems. Such a detailed analysis might also help both cryptographic algorithm developers and hardware developers while designing new algorithms, standards, and hardware implementations.

We have also developed a highly reconfigurable cryptographic processor that can support a wide range of cryptographic algorithms efficiently and has high potential for supporting future algorithms. The performance of our architecture is competitive with high-end ASIC and FPGA cores while achieving 25X and 160X higher throughput per area than the best CPU and GPU solutions, respectively. To the best of our

knowledge, Cryptoraptor supports more cryptographic algorithms than any other, and the only crypto-specific processor that can support future algorithms.

We also provided detailed analysis of Cryptoraptor in terms of performance, area, power, and the algorithm coverage. We believe providing such detailed study and evaluation may enable both cryptographic algorithm developers and researchers to explore performance, power, and area tradeoffs while designing new algorithms.

IX. Acknowledgement

This work was supported by the Semiconductor Research Corporation under contract 2013-HJ-2456 and 2013-TJ-2416.

References

[1] Data Encryption Standard (DES). *FIPS PUB 46*, pages 46–2, 1977.

[2] C. Adams. RFC2144: The CAST-128 Encryption Algorithm. *Network Working Group*, 1997.

[3] K. Akdemir, M. Dixon, W. Feghali, P. Fay, V. Gopal, J. Guilford, E. Ozturc, G. Worlich, and R. Zohar. Breakthrough AES performance with Intel AES new instructions. *White paper, June*, 2010.

[4] L. Ali, I. Aris, F. S. Hossain, and N. Roy. Design of an ultra high speed AES processor for next generation IT security. *Comput. Electr. Eng.*, 37(6), 2011. 1160-1170.

[5] Amphion Semiconductor. *CS5210-40 high performance AES encryption cores*. Amphion Semiconductor.

[6] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita. Specification of Camellia-A 128-bit block cipher. 2000.

[7] G. Bertoni, L. Breveglieri, F. Roberto, and F. Regazzoni. Speeding up AES by extending a 32 bit processor instruction set. In *ASAP'06*, 2006. 275-282.

[8] A. Biryukov and J. Grossschadl. Cryptanalysis of the full AES using GPU-like special-purpose hardware. *Fundam. Inf.*, 114(3-4), 2012. 221-237.

[9] J. W. Bos, D. A. Osvik, and D. Stefan. Fast implementations of AES on various platforms. *SPEED-CC'09*, page 501, 2009.

[10] J. D. Brown. *The IBM Power Edge of Network™ Processor*. IBM Corporation.

[11] R. Buchty, N. Heintze, and D. Oliva. Cryptonite - a programmable crypto processor architecture for high-bandwidth applications. In *ARCS'04*, 2004. 184-198.

[12] J. Burke, J. McDonald, and T. Austin. Architectural support for fast symmetric-key cryptography. In *ASPLOS'00*, 2000. 178-189.

[13] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis. Cost-efficient SHA hardware accelerators. *IEEE Transactions on VLSI Systems*, 16, 2008. 999-1008.

[14] D. Chen, G. Shou, Y. Hu, and Z. Guo. Efficient architecture and implementations of AES. In *ICACTE'10*, volume 6, 2010. 295-298.

[15] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). *RFC 3174, September*, 2001.

[16] A. Elbirt. Fast and efficient implementation of AES via instruction set extensions. In *AINAW'07*, 2007. 396-403.

[17] A. Elbirt and C. Paar. An instruction-level distributed processor for symmetric-key cryptography. *Parallel and Distributed Systems, IEEE Transactions on*, 16(5):468–480, 2005.

[18] EnSilica. *eSi-8110 product brief*. EnSilica.

[19] C.-P. Fan and J.-K. Hwang. Implementations of high throughput sequential and fully pipelined AES processors on FPGA. In *ISPACS'07*, 2007. 353-356.

[20] FIPS NIST. 180-2: Secure Hash Standard (SHS). Technical report, Technical report, (NIST), 2001.

[21] A. Fiskiran and R. Lee. On-chip lookup tables for fast symmetric-key encryption. In *ASAP'05*, 2005. 356-363.

[22] D. Fronte, A. Perez, and E. Payrat. Celator: A multi-algorithm cryptographic co-processor. In *ReConFig'08*, pages 438–443, 2008.

[23] M. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, O. Koufopavlou, and C. Goutis. Comparison of the hardware architectures and FPGA implementations of stream ciphers. In *ICECS'04*, 2004. 571-574.

[24] R. Golla and P. Jordan. T4: a highly threaded server-on-a-chip with native support for heterogeneous computing, August, 2011. Slides of a talk given at Hot Chips: A Symposium on High Performance Chips.

[25] T. Good and M. Benaissa. AES on FPGA from the fastest to the smallest. In *CHES'05*, volume 3659, 2005. 427-440.

[26] T. Good and M. Benaissa. Pipelined AES on FPGA with support for feedback modes (in a multi-channel environment). *Information Security, IET*, 1(1), 2007. 1-10.

[27] P. Grabher, J. Grossschadl, and D. Page. Light-weight instruction set extensions for bit-sliced cryptography. In *CHES'08*, 2008. 331-345.

[28] M. Grand, L. Bossuet, G. Gogniat, B. Le Gal, J.-P. Delahaye, and D. Dallet. A reconfigurable multi-core cryptoprocessor for multi-channel communication systems. In *IPDPSW'11*, pages 204–211, 2011.

[29] S. Gupta, S. Feng, A. Ansari, S. Mahlke, and D. August. Bundled execution of recurring traces for energy-efficient general purpose processing. In *MICRO-44*, 2011.

[30] W. Haixin, B. Guoqiang, and C. Hongyi. Zodiac: System architecture implementation for a high-performance network security processor. In *ASAP'08*, pages 91–96, 2008.

[31] A. Hodjat and I. Verbauwhede. Speed-area trade-off for 10 to 100 Gbits/s throughput AES processor. In *ASILOMAR'03*, volume 2, 2003. 2147-2150.

[32] A. Hodjat and I. Verbauwhede. A 21.54 Gbits/s fully pipelined AES processor on FPGA. In *FCCM'04*, 2004. 308-309.

[33] F. Hossain, M. Ali, and M. Al Abedin Syed. A very low power and high throughput AES processor. In *ICCIT'11*, 2011. 339-343.

[34] IBM Corporation. *IBM 4765 PCIe Cryptographic Coprocessor*. IBM.

[35] IP cores, Inc. AES-GCM MACsec (IEEE 802.1AE) and FC-SP Cores GCM1/GCM2/GCM3, 2013.

[36] K. Iwai, N. Nishikawa, and T. Kurokawa. Acceleration of AES encryption on CUDA GPU. *International Journal of Networking and Computing*, 2(1), 2012.

[37] N. Iyer, P. Anandmohan, D. Poornaiah, and V. D. Kulkarni. High throughput, low cost, fully pipelined architecture for AES crypto chip. In *Annual IEEE India Conference*, 2006. 1-6.

[38] K. U. Järvinen, M. T. Tommiska, and J. O. Skyttä. A fully pipelined memoryless 17.8 Gbps AES-128 encryptor. In *FPGA'03*, 2003. 207-215.

[39] C. Jenkins, M. Schulte, and J. Glossner. Instruction set extensions for the advanced encryption standard on a multithreaded software defined radio platform. *IJHPSA'10*, 2(3/4), 2010. 203-214.

[40] X. Lai. *On the design and security of block ciphers*. ETH SERIES in Information Processing. Hartung Gorre Verlag, v.1 edition, 1992.

[41] I. Lebedev, S. Cheng, A. Doupnik, J. Martin, C. Fletcher, D. Burke, M. Lin, and J. Wawrzynek. MARC: a many-core approach to reconfigurable computing. In *ReConFig'10*, 2010. 7-12.

[42] J. Lee, J. Park, S. Lee, and J. Kim. The SEED encryption algorithm. *SEED*, 2005.

[43] R. Lee, Z. Shi, and X. Yang. Efficient permutation instructions for fast software cryptography. *Micro, IEEE*, 21, 2001. 56-69.

[44] R. B. Lee and Y.-Y. Chen. Processor accelerator for AES. In *SASP'10*, 2010. 16-21.

[45] B. Liu and B. M. Baas. Parallel AES encryption engines for many-core processor arrays. *IEEE Transactions on Computers*, 62(3), 2013. 536-547.

[46] S. Mathew, F. Sheikh, M. E. Kounavis, S. Gueron, A. Agarwal, S. Hsu, H. Kaul, M. Anders, and R. Krishnamurthy. 53 gbps native gf$(2^4)^2$ composite-field aes-encrypt/decrypt accelerator for content-protection in 45nm high-performance microprocessors. *Journal of Solid-State Circuits*, 46(4), 2011. 767-776.

[47] M. Matsui. New block encryption algorithm MISTY. *Fast Software Encryption*, 1267, 1997. 54-68.

[48] M. McLoone and J. McCanny. High-performance FPGA implementation of DES using a novel method for implementing the key schedule. *IEEE Proceedings of Circuits, Devices and Systems*, 150, 2003. 373-378.

[49] Mercora Technologies. *AES ultra fast IP core for Xilinx FPGAs* . Mercora Technologies.

[50] S. Morioka and A. Satoh. A 10 Gbps full-AES crypto design with a twisted-BDD S-Box architecture. In *ICCD'02*, 2002. 98-103.

[51] N. Muralimanohar, R. Balasubramonian, and N. Jouppi. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In *MICRO'07*, 2007. 3-14.

[52] National Institute of Standards and Technology (NIST). Advanced encryption standard (AES). *FIPS Publication*, 197, 2001.

[53] N. Nishikawa, K. Iwai, and T. Kurokawa. High-performance symmetric block ciphers on multicore CPU and GPUs. *IJNC'12*, 2(2), 2012. 251-268.

[54] W. Nunan Zola and L. De Bona. Parallel speculative encryption of multiple AES contexts on GPUs. In *Innovative Parallel Computing*, 2012. 1-9.

[55] J. Pieprzyk and L. Tombak. Soviet encryption algorithm. *Preprint*, pages 94–10, 1993.

[56] T. Pionteck, T. Staake, T. Stiefmeier, L. Kabulepa, and M. Glesner. Design of a reconfigurable AES encryption/decryption engine for mobile terminals. In *ISCAS'04*, volume 2, pages 545–556, 2004.

[57] S. Qu, G. Shou, Y. Hu, Z. Guo, and Z. Qian. High throughput, pipelined implementation of AES on FPGA. In *IEEC'09*, 2009. 542-545.

[58] R. Rivest. RFC 1321: The MD5 message-digest algorithm, April 1992. *Status: INFORMATIONAl*.

[59] R. Rivest. The MD4 Message-Digest Algorithm, RFC 1320. 1992.

[60] R. L. Rivest. The RC5 encryption algorithm. In *Fast Software Encryption*, pages 86–96. Springer, 1995.

[61] M. R. M. Rizk and M. Morsy. Optimized area and optimized speed hardware implementations of AES on FPGA. In *IDT'07*, 2007. 207-217.

[62] D. Sagar and G. Leelavathi. Design and implementation of extended version of AES algorithm with DSP units. In *IJEAT'13*, volume 2-6, 2013. 360-364.

[63] P. Saravanan, N. R. Devi, G. Swathi, and D. P. Kalpana. A high-throughput ASIC implementation of configurable advanced encryption standard (AES) processor. *IJCA Special Issue on Network Security and Cryptography*, NSC(3), 2011. 1-6.

[64] B. Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In *Fast Software Encryption*, pages 191–204. Springer, 1994.

[65] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Twofish: A 128-bit block cipher. *NIST AES Proposal*, 15, 1998.

[66] Z. Shi and R. Lee. Bit permutation instructions for accelerating software cryptography. In *ASAP'00*, 2000. 138-148.

[67] M. I. Soliman and G. Y. Abozaid. FastCrypto: parallel AES pipeline extension for general-purpose processors. *Neural, Parallel Sci. Comput.*, 18(1). 47-58.

[68] J. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. Davis, P. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal. FreePDK: an open-source variation-aware design kit. In *MSE'07*, 2007. 173-174.

[69] E. Swankoski and V. Narayanan. Dynamic high-performance multi-mode architectures for AES encryption. In *MAPLD'05*, 2005. 1-9.

[70] E. J. Swankoski, R. Brooks, V. Narayanan, M. Kandemir, and M. Irwin. A parallel architecture for secure FPGA symmetric encryption. In *IPDPS'04*, 2004. 132-.

[71] D. Theodoropoulos, A. Siskos, and D. Pnevmatikatos. Ccproc: A custom vliw cryptography co-processor for symmetric-key ciphers. In *Reconfigurable Computing: Architectures, Tools and Applications*, pages 318–323. Springer, 2009.

[72] VIA Technologies, Inc. VIA C7 processor, 2013.

[73] Y. Wang and Y. Ha. FPGA-based 40.9-Gbits/s masked AES with area optimization for storage area network. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(1), 2013. 36-40.

[74] R. Wash. Lecture notes on stream ciphers and RC4. *Reserve University*, pages 1–19, 2001.

[75] D. Whiting, B. Schneier, S. Lucks, and F. Muller. Fast encryption and authentication in a single cryptographic primitive. *ECRYPT*, 27(200):5, 2005.

[76] L. Wu, C. Weaver, and T. Austin. CryptoManiac: a fast flexible architecture for secure communication. In *ISCA'01*, 2001. 110-119.

[77] P. Yalla and J. Kaps. Compact FPGA implementation of Camellia. In *FPL'09*, 2009. 658-661.

[78] S.-M. Yoo, D. Kotturi, W. D. Pan, and J. Blizzard. An AES crypto chip using a high-speed parallel pipelined architecture. *Microprocessors and Microsystems*, 29, 2005. 317-326.

[79] X. Zhang and K. Parhi. High-speed VLSI architectures for the AES algorithm. *IEEE Transactions on VLSI Systems*, 12(9), 2004. 957-967.