FPGA Implementations of the Hummingbird Cryptographic Algorithm

Xinxin Fan and Guang Gong
E&CE Department
University of Waterloo
Waterloo, Ontario, N2L 3G1, CANADA
Email: {x5fan, ggong}@uwaterloo.ca

Ken Lauffenburger
Aava Technology LLC
1206 Donegal Ln
Garland, TX 75044, USA
Email: kenl@aavatech.com

Troy Hicks
Revere Security Corporation
4500 Westgrove Drive, Suite 335
Addison, TX 75001, USA
Email: troy.hicks@reveresecurity.com

***Abstract*—Hummingbird is a new ultra-lightweight cryptographic algorithm targeted for resource-constrained devices like RFID tags, smart cards, and wireless sensor nodes. In this paper, we describe efficient hardware implementations of a stand-alone Hummingbird component in field-programmable gate array (FPGA) devices. We implement an encryption only core and an encryption/decryption core on the low-cost Xilinx FPGA series Spartan-3 and compare our results with other reported lightweight block cipher implementations on the same series. Our experimental results highlight that in the context of low-cost FPGA implementation Hummingbird has favorable efficiency and low area requirements.**

***Index Terms*—Lightweight cryptographic primitive, resource-constrained devices, FPGA implementations.**

I. Introduction

Hummingbird is a recently proposed ultra-lightweight cryptographic algorithm targeted for low-cost smart devices like RFID tags, smart cards, and wireless sensor nodes [3]. It has a hybrid structure of block cipher and stream cipher and was developed with both lightweight software and lightweight hardware implementations for constrained devices in mind. Moreover, Hummingbird has been shown to be resistant to the most common attacks to block ciphers and stream ciphers including birthday attack, differential and linear cryptanalysis, structure attacks, algebraic attacks, cube attacks, etc. [3].

In practice, Hummingbird has been implemented across a wide range of different target platforms [3], [5]. Those implementations demonstrate that Hummingbird provides efficient and flexible software solutions for various embedded applications. However, the hardware performance of Hummingbird has not yet been investigated in detail. As a result, our main contribution in this paper is to close this gap and provide the first efficient hardware implementations of Hummingbird encryption/decryption cores on low-cost FPGAs. Our implementation results show that on the Spartan-3 XC3S200 FPGA device the speed optimized Hummingbird encryption core can achieve a throughput of 160.4 Mbps at the cost of 273 slices, whereas the encryption/decryption core can be implemented in 558 slices and operate at 128.8 Mbps.

II. The Hummingbird Cryptographic Algorithm

Hummingbird is neither a block cipher nor a stream cipher, but a *rotor machine* equipped with novel rotor-stepping rules. The design of Hummingbird is based on an elegant combination of a block cipher and stream cipher with 16-bit block size, 256-bit key size, and 80-bit internal state. Figure 1(a) and Figure 1(b) illustrate the initialization and encryption processes of the Hummingbird cryptographic algorithm, respectively. Both initialization and encryption consist of four 16-bit block ciphers E_{k_i} $(i = 1, 2, 3, 4)$, four 16-bit internal state registers RSi $(i = 1, 2, 3, 4)$, and a 16-stage Linear Shift Feedback Register (LFSR). Moreover, the 256-bit secret key K is divided into four 64-bit subkeys k_1, k_2, k_3 and k_4 which are used in the four block ciphers, respectively.

After a system initialization process as shown in Figure 1(a), a 16-bit plaintext block PT_i is encrypted by passing four identical block ciphers $E_{k_i}(\cdot)$ $(i = 1, 2, 3, 4)$ in a consecutive manner, each of which is a typical substitution-permutation (SP) network with 16-bit block size and 64-bit key as shown in Figure 1(c). The block cipher consists of four regular rounds and a final round. The substitution layer is composed of four S-boxes with 4-bit inputs and 4-bit outputs as shown in Table I.

TABLE I
Four S-Boxes in Hexadecimal Notation

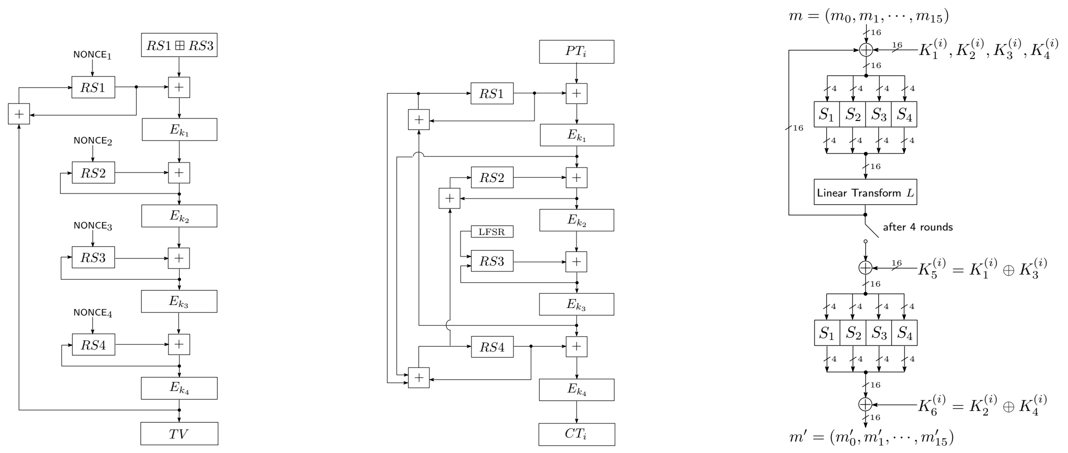
x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(x)$	8	6	5	F	1	C	A	9	E	B	2	4	7	0	D	3
$S_2(x)$	0	7	E	1	5	B	8	2	3	A	D	6	F	C	4	9
$S_3(x)$	2	E	F	5	C	1	9	A	B	4	6	8	0	7	3	D
$S_4(x)$	0	7	3	4	C	1	A	F	D	E	6	B	2	8	9	5

The permutation layer in the 16-bit block cipher is given by the linear transform $L : \{0,1\}^{16} \to \{0,1\}^{16}$ defined as follows:

$$L(m) = m \oplus (m \ll 6) \oplus (m \ll 10),$$

where $m = (m_0, m_1, \cdots, m_{15})$ is a 16-bit data block.

To further reduce the consumption of the area and power of Hummingbird in hardware implementations, four S-boxes used in Hummingbird can be replaced by a single S-box, which is repeated four times in the 16-bit block cipher. The compact version of Hummingbird can achieve the same security level as the original Hummingbird and will be implemented on FPGAs in this paper. For more details about Hummingbird, the interested reader is referred to [3].


NONCE$_1$
$RS1 \boxplus RS3$
$RS1$
E_{k_1}
NONCE$_2$
$RS2$
E_{k_2}
NONCE$_3$
$RS3$
E_{k_3}
NONCE$_4$
$RS4$
E_{k_4}
TV
PT_i
LFSR
CT_i
$m = (m_0, m_1, \cdots, m_{15})$
$K_1^{(i)}, K_2^{(i)}, K_3^{(i)}, K_4^{(i)}$
S_1 S_2 S_3 S_4
Linear Transform L
after 4 rounds
$K_5^{(i)} = K_1^{(i)} \oplus K_3^{(i)}$
$K_6^{(i)} = K_2^{(i)} \oplus K_4^{(i)}$
$m' = (m'_0, m'_1, \cdots, m'_{15})$

(a) Initialization Process (b) Encryption Process (c) 16-bit Block Cipher

Fig. 1. The Hummingbird Cryptographic Algorithm and Its Internal Structure

III. FPGA Implementations of Hummingbird

In this section efficient FPGA implementations of a stand-alone Hummingbird component are described. Note that the choice of different kinds of I/O interfaces has a significant influence on the performance of hardware implementation and is highly application specific. Therefore, we do not implement any specific I/O logic in order to obtain the accurate performance profile of a plain Hummingbird encryption/decryption core and to provide enough flexibility for various applications.

A. *Selection of a "Hardware-Friendly" S-Box*

A "hardware-friendly" S-box is the S-box that can be efficiently implemented in the target hardware platform with a small area requirement. Four 4×4 S-boxes $S_i(x) : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ ($i = 1, 2, 3, 4$) have been carefully selected in Hummingbird according to certain security criteria (see Section II). To implement the compact version of Hummingbird, we need to choose a "hardware-friendly" S-box from four S-boxes listed in Table I. By using the Boolean minimization tool *Espresso* [4] we can obtain the minimal Boolean function representations (BFR) for the four S-boxes in Hummingbird. Note that each S-box can be implemented in hardware by using either a look-up table (LUT) or the Boolean function representations (i.e., combinatorial logic). The exact efficiency of the above two approaches significantly depends on specific hardware platforms and synthesis tools. Therefore, for the proposed architecture of the 16-bit block cipher in Section III-B we investigate two implementation strategies (i.e., LUT and BFR) for the four S-boxes and select one that results in the most area-efficient implementation of the 16-bit block cipher.

B. *Loop-Unrolled Architecture of 16-bit Block Cipher*

The loop-unrolled architecture for the 16-bit block cipher is illustrated in Figure 2. In this architecture, only one 16-bit block of data is processed at a time. However, five rounds are cascaded and the whole encryption can be performed in a single clock cycle. The loop-unrolled architecture consists of 8 XORs, 20 S-boxes, and 4 permutation layers for the datapath. To select a "hardware-friendly" S-box for the compact version of Hummingbird, we implement the loop-unrolled architecture of the 16-bit block cipher on the target FPGA platform and test one S-box candidate from Table I each time. Table II summarizes the area requirement when using different S-boxes and implementation strategies. All experimental results are from post-place and route analysis.

TABLE II
Area Requirement Comparison for the Loop-Unrolled Architecture of 16-bit Block Cipher on the Spartan-3 XC3S200 FPGA

S-box	Implementation Strategy	# LUTs	# FFs	Total Occupied Slices
$S_1(x)$	LUT	186	16	107
	BFR	186	16	109
$S_2(x)$	LUT	193	16	112
	BFR	186	16	107
$S_3(x)$	LUT	**186**	**16**	**101**
	BFR	186	16	106
$S_4(x)$	LUT	190	16	104
	BFR	187	16	109

When comparing different S-boxes and implementation strategies, Table II shows that the loop-unrolled architecture occupies the minimal number of slices provided that the S-box $S_3(x)$ is employed and implemented by a LUT. Therefore, the S-box $S_3(x)$ is chosen for efficient implementation of speed optimized Hummingbird encryption/decryption cores that are described in detail in the following subsections.

C. *Speed Optimized Hummingbird Encryption Core*

The top-level description of a speed optimized Hummingbird encryption core is illustrated in Figure 3. After the chip enable signal changes from '0' to '1', the initialization process (see Figure 1(a)) begins and four rotors RSi ($i = 1, 2, 3, 4$) are first initialized by four 16-bit random nonce through the interface RSi within four clock cycles. From the fifth clock cycle, the core starts encrypting $RS1 \boxplus RS3$ for four times and

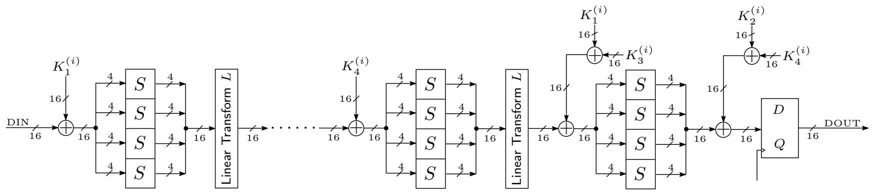


Fig. 2. Loop-Unrolled Architecture of 16-bit Block Cipher

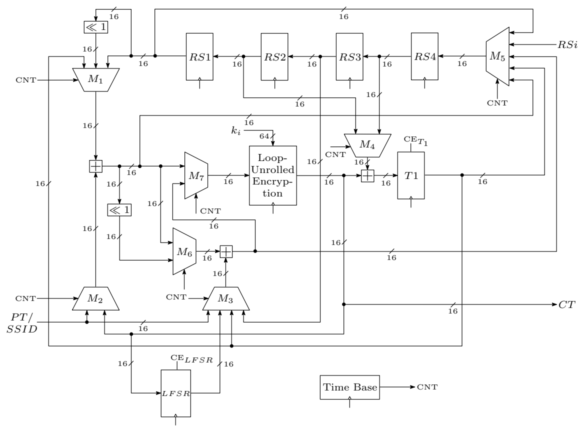


Fig. 3. The Datapath of Speed Optimized Hummingbird Encryption Core

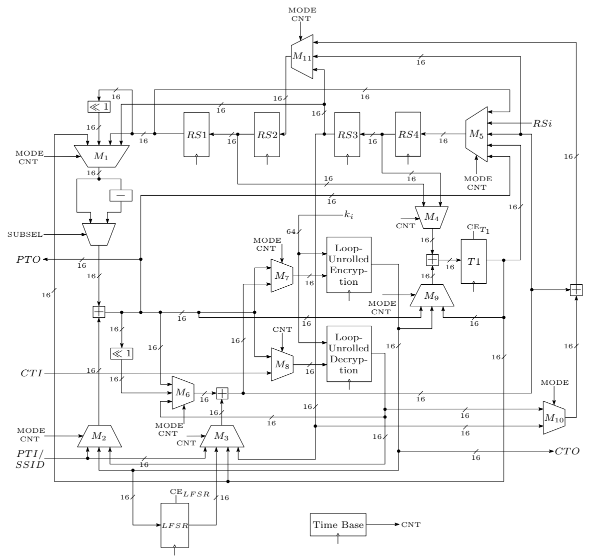


Fig. 4. The Datapath of Speed Optimized Hummingbird Encryption/Decryption Core

each iteration requires four clock cycles to finish encryptions by four 16-bit block ciphers as well as the internal state updating. During the above procedure, the 64-bit subkeys k_i $(i = 1, 2, 3, 4)$ are read from an external register under the control of a key selection signal. Moreover, depending on the value of a round counter, the multiplexer M_5 chooses the correct computation results to update four rotors and other multiplexers select appropriate inputs to feed the 16-bit block cipher. Once the initialization process is done after 20 clock cycles, the first 16-bit plaintext block is read from an external register for encryption. With another four clock cycles, the corresponding ciphertext is output from the encryption core. Therefore, the proposed speed optimized Hummingbird encryption core can encrypt one 16-bit plaintext block per 4 clock cycles, after an initialization process of 20 clock cycles.

D. Speed Optimized Hummingbird Encryption/Decryption Core

We depict the top-level architecture of a speed optimized Hummingbird encryption/decryption core in the following Figure 4. The Hummingbird encryption/decryption core supports the following four operation modes: i) encryption only; ii) decryption only; iii) encryption followed by decryption; and iv) decryption followed by encryption. Both encryption and decryption routines share the same initialization procedure that first takes 4 clock cycles to load four random nonce into rotors through multiplexers M_5 and M_{11}, followed by 16 clock cycles for four iterations. The architecture of the encryption/decryption core is quite similar to that of the encryption-only core except the following several aspects. Firstly, the rotor RS_2 completes the update when encrypting two successive plaintext blocks in the encryption-only core, whereas all rotors are fully updated each time a plaintext block is encrypted or decrypted in order to support the four operation modes in the encryption/decryption core. For this purpose, two multiplexers M_{10} and M_{11} are introduced to fully update the rotor $RS2$ after each encryption/decryption. Secondly, an adder that can perform both modulo 2^{16} addition and subtraction is included, which executes the corresponding arithmetic according to the operation modes of the core. Thirdly, two multiplexers M_7 and M_8 are used to feed correct values to the encryption and decryption routines of the 16-bit block cipher, respectively. Finally, all the other multiplexers select appropriate inputs based on the value of a round counter as well as the operation modes. The workflow of the encryption/decryption core is also similar to that of encryption-only core. Hence, the speed optimized Hummingbird encryption/decryption core can encrypt or decrypt one 16-bit plaintext or ciphertext block per 4 clock cycles, after an initialization process of 20 clock cycles.

E. Implementation Results and Comparisons

A summary of our implementation results is presented in Table III, where the area requirements (in slices), the maximum work frequency, and the throughput are provided. All experimental results were extracted after place and route with the

ISE Design Suite v11.1 from Xilinx on a xc3s200-5ft256 Spartan-3 platform with speed grade -5. From Table III, we note that the speed optimized Hummingbird encryption core can achieve a throughput of 160.4 Mbps at the cost of 273 slices, whereas the Hummingbird encryption/decryption core occupies 558 slices and operates at 128.8 Mbps on the target FPGA platform.

TABLE III
IMPLEMENTATION RESULTS FOR COMPACT VERSION OF HUMMINGBIRD ON THE SPARTAN-3 XC3S200 FPGA

Mode (Enc/Dec)	# LUTs	# FFs	Total Occupied Slices	Max. Freq. (MHz)	# CLK Cycles		Throughput (Mbps)	Efficiency (Mbps/# Slices)
					Init.	Enc/Dec		
Enc	473	120	273	40.1	20	4	160.4	0.59
Enc/Dec	1,024	145	558	32.2			128.8	0.23

TABLE IV
PERFORMANCE COMPARISON OF FPGA IMPLEMENTATIONS OF CRYPTOGRAPHIC ALGORITHMS

Cipher	Key Size	Block Size	FPGA Device	Total Occupied Slices	Max. Freq. (MHz)	Throughput (Mbps)	Efficiency (Mbps/# Slices)
Hummingbird	256	16	Spartan-3 XC3S200-5	273	40.1	160.4	0.59
PRESENT [10]	80	64	Spartan-3 XC3S400-5	176	258	516	2.93
	128	64		202	254	508	2.51
PRESENT [7]	80	64	Spartan-3E XC3S500	271	–	–	–
XTEA [8]	128	64	Spartan-3 XC3S50-5	254	62.6	36	0.14
			Virtex-5 XC5VLX85-3	9,647	332.2	20,645	2.14
ICEBERG [12]	128	64	Virtex-2	631	–	1,016	1.61
SEA [9]	126	126	Virtex-2 XC2V4000	424	145	156	0.368
AES [2]	128	128	Spartan-2 XC2S30-6	522	60	166	0.32
AES [6]			Spartan-3 XC3S2000-5	17,425	196.1	25,107	1.44
			Spartan-2 XC2S15-6	264	67	2.2	0.01
AES [11]			Spartan-2 XC2V40-6	1,214	123	358	0.29
AES [1]			Spartan-3	1,800	150	1700	0.9

Table IV describes the performance comparison of our Hummingbird implementation with existing FPGA implementations of block ciphers PRESENT [7], [10], XTEA [8], ICEBERG [12], SEA [9] as well as AES [1], [2], [6], [11]. Note that numerous AES hardware architectures have been proposed in literature and we only focus on those implementations using low-cost Spartan series FPGA devices with speed grade -5 and above for the purpose of comparison. Moreover, the implementation figures of ICEBERG and SEA are only available on Virtex-2 series FPGAs. We also would like to point out that it is quite difficult to provide a fair comparison among different implementations on FPGAs, taking into account the diversity of FPGA devices and packages, speed grade level, and synthesis and implementation tools. Therefore, we also include additional information such as implementation platform and speed grade level in Table IV.

Our experimental results show that in the context of low-cost FPGA implementation Hummingbird can achieve larger throughput with smaller area requirement, when compared to block ciphers XTEA, ICEBERG, SEA and AES. However, the implementation of the ultra-lightweight block cipher PRESENT is more efficient than that of Hummingbird, although a slightly larger (and hence more expensive) FPGA device Spartan-3 XC3S400 is required. The main reason is due to the complex internal state updating procedure in Hummingbird cipher (see Figure 1(a) and Figure 1(b)). As a result, the control unit is more complicated and the delay of the critical path is much longer in the Hummingbird hardware architecture than those in the PRESENT core.

IV. CONCLUSION

This paper presented the first efficient FPGA implementations of the ultra-lightweight cryptographic algorithm Hummingbird. The proposed speed optimized Hummingbird encryption/decryption cores can encrypt or decrypt a 16-bit message block with 4 clock cycles, after an initialization process of 20 clock cycles. Compared to other lightweight FPGA implementations of block ciphers XTEA, ICEBERG, SEA and AES, Hummingbird can achieve larger throughput with smaller area requirement. Consequently, Hummingbird can be considered as an ideal cryptographic primitive for resource-constrained environments.

ACKNOWLEDGMENT

This work is supported by an NSERC Discovery Grant and an NSERC Strategic Project Grant.

REFERENCES

[1] P. Bulens, F.-X. Standaert, J.-J. Quisquater, and P. Pellegrin, "Implementation of the AES-128 on Virtex-5 FPGAs", *Progress in Cryptology - AFRICACRYPT 2008*, LNCS 5023, pp. 16-26, 2008.

[2] P. Chodowiec and K. Gaj, "Very Compact FPGA Implementation of the AES Algorithm", *The 5th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2003*, LNCS 2779, pp. 319-333, 2003.

[3] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Hummingbird: Ultra-Lightweight Cryptography for Resource- Constrained Devices", to appear in the proceedings of *The 14th International Conference on Financial Cryptography and Data Security - FC 2010*, 2010.

[4] N. N. Espresso. Available at http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/index.htm, November 1994.

[5] X. Fan, H. Hu, G. Gong, E. M. Smith and D. Engels, "Lightweight Implementation of Hummingbird Cryptographic Algorithm on 4-Bit Microcontrollers", *The 1st International Workshop on RFID Security and Cryptography 2009 (RISC'09)*, pp. 838-844, 2009.

[6] T. Good and M. Benaissa, "AES on FPGA from the Fastest to the Smallest", *The 7th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2005*, LNCS 3659, pp. 427-440, 2005.

[7] X. Guo, Z. Chen, and P. Schaumont, "Energy and Performance Evaluation of an FPGA-Based SoC Platform with AES and PRESENT Coprocessors", *Embedded Computer Systems: Architectures, Modeling, and Simulation - SAMOS'2008*, LNCS 5114, pp. 106-115, 2008.

[8] J.-P. Kaps, "Chai-Tea, Cryptographic Hardware Implemenations of xTEA", *The 9th International Conference on Cryptology in India - INDOCRYPT 2008*, LNCS 5356, pp. 363-375, 2008.

[9] F. Mace, F.-X. Standaert, and J.-J. Quisquater, "FPGA Implemenation(s) of a Scalable Encryption Algorithm", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 2, pp. 212-216, 2008.

[10] A. Poschmann, "Lightweight Cryptography - Cryptographic Engineering for a Pervasive World", Ph.D. Thesis, Department of Electrical Engineering and Information Sciences, Ruhr-Universit$\ddot{a}$et Bochum, Bochum, Germany, 2009.

[11] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat, "Compact and Efficient Encryption/Decryption Module for FPGA Implementation of the AES Rijndael Very Well Suited for Small Embedded Applications", *International Conference on Information Technology: Coding and Computing - ITCC 2004*, pp. 583-587, 2004.

[12] F.-X. Standaert, G. Piret, G. Rouvroy, and J.-J. Quisquater, "FPGA Implementations of the ICEBERG Block Cipher", *Integration, the VLSI Journal*, vol. 40, iss. 1, pp. 20-27, 2007.

[13] Xilinx Inc., "Spartan-3 FPGA Family Data Sheet", DS099, December 4, 2009, available at http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf.