The Simeck Family of Lightweight Block Ciphers

Gangqiang Yang(✉), Bo Zhu, Valentin Suder,
Mark D. Aagaard, and Guang Gong

Department of Electrical and Computer Engineering,
University of Waterloo, Waterloo, ON N2L 3G1, Canada
{g37yang,bo.zhu,vsuder,maagaard,ggong}@uwaterloo.ca

Abstract. Two lightweight block cipher families, Simon and Speck, have been proposed by researchers from the NSA recently. In this paper, we introduce Simeck, a new family of lightweight block ciphers that combines the good design components from both Simon and Speck, in order to devise even more compact and efficient block ciphers. For Simeck32/64, we can achieve 505 GEs (before the Place and Route phase) and 549 GEs (after the Place and Route phase), with the power consumption of 0.417 μW in CMOS 130 nm ASIC, and 454 GEs (before the Place and Route phase) and 488 GEs (after the Place and Route phase), with the power consumption of 1.292 μW in CMOS 65 nm ASIC. Furthermore, all of the instances of Simeck are smaller than the ones of hardware-optimized cipher Simon in terms of area and power consumption in both CMOS 130 nm and CMOS 65 nm techniques. In addition, we also give the security evaluation of Simeck with respect to many traditional cryptanalysis methods, including differential attacks, linear attacks, impossible differential attacks, meet-in-the-middle attacks, and slide attacks. Overall, all of the instances of Simeck can satisfy the area, power, and throughput requirements in passive RFID tags.

Keywords: Lightweight · Block cipher · ASICs · Passive RFID

1 Introduction

In recent years, low-end embedded devices have been deployed in an increasing number and used in various applications, such as D.Radio Frequency Identification (RFID) tags and wireless sensor networks (WSNs). Providing security solutions to these widely used devices has attracted a lot of attention from cryptography researchers. These kinds of devices have very limited power consumption, constrained memory and computing capability, and thus applying traditional security solutions, such as TLS and IPsec, in these contexts is often impractical. Hence, lightweight cryptography has been developed in order to provide compact algorithms and protocols that fit in resource-constrained environments.

Numerous lightweight ciphers have appeared. Among them are a large number of block ciphers such as TEA [31], XTEA [26], PRESENT [9], KATAN and KTANTAN [11], LED [16], EPCBC [33], KLEIN [15], LBlock [32], Piccolo [29],

T. Güneysu and H. Handschuh (Eds.): CHES 2015, LNCS 9293, pp. 307–329, 2015.
DOI: 10.1007/978-3-662-48324-4_16

Twine [30], and the more recent SIMON and SPECK [3]. There exist also some lightweight stream ciphers such as Trivium [12], Grain [17] and WG [25], which provide suitable security and small implementations for resource-constrained devices.

The recently proposed lightweight block ciphers, SIMON and SPECK [3], have led to papers concerning their security [1,7,10]. This is partially due to the fact that these ciphers are recognized to be the smallest block ciphers in each of the block/key size categories when used in resource-constrained environments. SIMON is optimized for hardware implementation, while SPECK is optimized for software. Inspired by the designs of SIMON and SPECK, we combine their good components in order to get a new design of block cipher family, called Simeck. We use a slightly modified version of SIMON's round function, and reuse it in the key schedule like SPECK does. Moreover, we take the benefits of using Linear Feedback Shift Register (LFSR) based constants in the key schedule in order to further reduce hardware implementation footprints. The new family of lightweight block ciphers Simeck aims to have comparable security levels but more efficient hardware implementations.

Based on the aforementioned motivations, we have the detailed design goals as follows.

Hardware. First, we want to minimize the area and power consumption of the Application Specific Integrated Circuit (ASIC) implementations. We also want to allow a range of options in the area, throughput, and power consumption. Finally, we want to keep the maximum operating frequency as high as possible.

Applications. Take the application of passive RFID tags for example, Simeck should satisfy the following requirements in order to be used in practice: (1) The area of Simeck should be less than 2000 GEs [2,18]. (2) The power consumption of Simeck should be very small. (3) The typical passive RFID tag's operating frequency is 2 MHz and the data rate is 64 Kbps [14,34], and thus the throughput is $64K/2M \approx 1/32$. Therefore, if the tag's operating frequency is 100 KHz (for benchmarking purpose), the throughput of Simeck should at least be $100\ K \cdot 1/32$ bps ≈ 3.1 Kbps.

Security. Although SIMON and SPECK were designed with small, simple round functions, they are iterated a sufficient number of times in order to resist traditional attacks. We follow the same strategy with Simeck, and due to its similarity with SIMON, we benefit from its analysis carried so far.

In this paper, we offer a wide range of options between area, throughput, and power consumption for the implementations of Simeck. All the Simeck's family members can meet our security, hardware, and applications design goals. We compare our results to the previous constructions with comparable block sizes and key sizes as given in Table 1. Table 1 gives our smallest area results for all the instances of Simeck from before and after the Place and Route (P&R) in CMOS 130 nm and CMOS 65 nm ASICs. In addition, the corresponding throughput and power consumption after the Place and Route are also provided. In particular, Table 1 presents our hardware implementation results of SIMON which cost less

Table 1. Comparison of Hardware Implementations of Lightweight Block Ciphers

Size	Algorithm	Tech (nm)	Area		Throughput	Power	Source
			Before P&R (GEs)	After P&R (GEs)	@100KHz (Kbps)	@ 100KHz (μW)	
32/64	SIMON	130	523	-	5.6	-	[3]
	SPECK		580	-	4.2	-	[3]
	SIMON		**517**	**562**	5.6	0.421	**here**
	Simeck		**505**	**549**	5.6	0.417	**here**
	SIMON	65	**466**	**501**	5.6	1.311	**here**
	Simeck		**454**	**488**	5.6	1.292	**here**
48/96	SIMON	130	739	-	5.0	-	[3]
	SPECK		794	-	4.0	-	[3]
	SIMON		**733**	**796**	5.0	0.579	**here**
	Simeck		**715**	**778**	5.0	0.576	**here**
	SIMON	65	**661**	**711**	5.0	1.812	**here**
	Simeck		**645**	**693**	5.0	1.805	**here**
	EPCBC	180	1008	-	12.1	-	[33]
64/128	SIMON	130	958	-	4.2	-	[3]
	SPECK		966	-	3.4	-	[3]
	SIMON		**944**	**1026**	4.2	0.762	**here**
	Simeck		**924**	**1005**	4.2	0.754	**here**
	SIMON	65	**845**	**908**	4.2	2.336	**here**
	Simeck		**828**	**891**	4.2	2.304	**here**
	LED	180	1265	-	3.4	-	[16]
	PRESENT		1339	-	12.1	-	[33]

area than the original results in [3]. Moreover, the hardware implementations of our Simeck block cipher family are even smaller than our implementations of SIMON in terms of area and power consumption.

More specifically in Table 1, we can achieve a small area of 505 GEs before the Place and Route with a throughput of 5.6 Kbps and 0.417 μW power consumption for Simeck32/64 in CMOS 130 nm ASIC. With a fair comparison (before the Place and Route) in CMOS 130 nm, Simeck32/64 can achieve 2.3 % smaller than our implementations of SIMON32/64, and 3.4 % smaller than the original implementations of SIMON32/64. Correspondingly, we can get an even smaller area of 454 GEs before the Place and Route and 1.292 μW power consumption in CMOS 65 nm ASIC. In this case, Simeck32/64 is 2.6 % smaller than our implementations of SIMON32/64.

Similarly, Simeck48/96, 64/128 are 2.5 %, 2.1 %, respectively, smaller than our implementations of SIMON48/96, 64/128, and they are 3.3 %, 3.5 %, respectively, smaller than the original implementations of SIMON48/96, 64/128 in CMOS 130 nm. Correspondingly in CMOS 65 nm, Simeck48/96, 64/128 are 2.4 %, 2.0 %, respectively, smaller than our implementations of SIMON48/96,

64/128. Moreover, with only a little extra area (GEs) and power consumption, we can increase Simeck's throughput a lot.

This paper is organized as follows. In Sect. 2, we describe the specifications and design rationales of the Simeck family. Section 3 first presents our metrics and design flow in CMOS 130 nm and CMOS 65 nm ASICs. Then, we give two different hardware architectures of Simeck in order to make a trade-off between area, throughput, and power consumption. Later, the hardware evaluations in CMOS 130 nm and CMOS 65 nm are given with a thorough analysis. In Sect. 4, we compare our results of Simeck and SIMON with the results in [3]. Before concluding this paper, we provide a security analysis of our new block ciphers in Sect. 5.

2 Design Specifications and Rationales

In this section, we give the specifications, as well as design rationales, of our block cipher family Simeck. We use the following notations throughout the rest of the paper.

$x \lll c$ denotes the cyclic shift of x to the left by c bits.
$x \odot y$ is the bitwise AND of x and y.
$x \oplus y$ is the exclusive-or (XOR) of x and y.

2.1 Specifications of Simeck

Our lightweight block cipher family Simeck is denoted Simeck$2n/mn$, where n is the word size and n is required to be 16, 24 or 32; while $2n$ is the block size and mn is the key size. More specifically, our Simeck family includes Simeck32/64, Simeck48/96, and Simeck64/128. For example, Simeck32/64 refers to perform encryptions or decryptions on 32-bit message blocks using a 64-bit key. These three size choices of the ciphers aim to fit different applications of embedded systems including RFID systems, and these sizes are also contained in the specifications of SIMON and SPECK families of block ciphers.

Simeck is designed to be extremely small in hardware footprints and to be compact in software implementations as well. The round function and the key schedule algorithm follow the Feistel structure. A plaintext to be encrypted is first divided into two words l_0 and r_0, where l_0 contains the most significant n bits, and r_0 consists of the least significant n bits. Then these two words are processed by the Simeck round function for certain number of rounds, and finally the two output words l_T and r_T are concatenated to form a complete ciphertext, where T denotes the total number of rounds.

Round Function. We define the round function (of the i-th round) as the following function,

$$R_{k_i}(l_i,\ r_i) = (r_i \oplus f(l_i) \oplus k_i,\ l_i),$$

where l_i and r_i are the two words for the internal state of Simeck, k_i is the round key, and the function f is defined as

$$f(x) = (x \odot (x \lll 5)) \oplus (x \lll 1).$$

Fig. 1 illustrates the operations of the round function R_{k_i}.

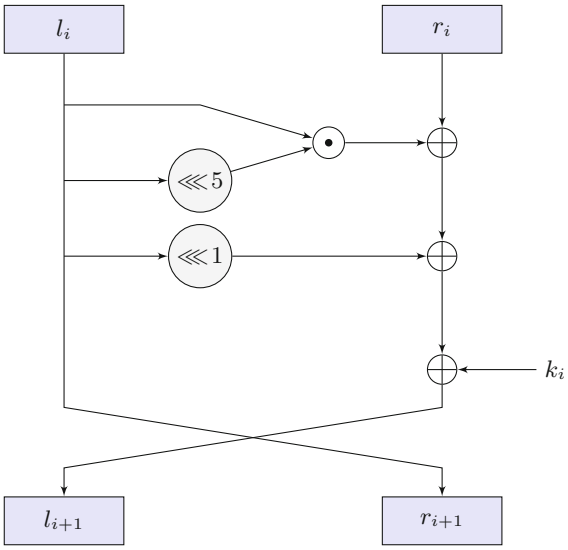


Fig. 1. The Round Function of Simeck

Key Schedule/Expansion. To generate the round key k_i from a given master key K, the master key K is first segmented into four words and loaded as the initial states (t_2, t_1, t_0, k_0) of the feedback shift registers shown in Fig. 2. The least significant n bits of K are loaded into k_0; while the most significant n bits are put into t_2. To update the registers and generate round keys, we reuse the round function with a round constant $C \oplus (z_j)_i$ acting as the round key, i.e. $R_{C \oplus (z_j)_i}$. The updating operation can be expressed as

$$\begin{cases} k_{i+1} = t_i, \\ t_{i+3} = k_i \oplus f(t_i) \oplus C \oplus (z_j)_i, \end{cases}$$

where $0 \leq i \leq T-1$. The value k_i is used as the round key of the i-th round.

The value of the constant C is defined by $C = 2^n - 4$, where n is the word size. $(z_j)_i$ denotes the i-th bit of the sequence z_j. Simeck32/64 and Simeck48/96 use the same sequence z_0, i.e. $j = 0$, which is an m-sequence with period 31 and can be generated by the primitive polynomial $X^5 + X^2 + 1$ with the initial

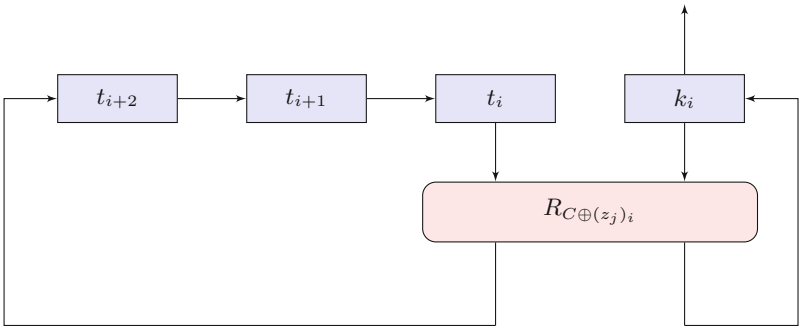


Fig. 2. The Key Expansion of Simeck, where $R_{C\oplus(z_j)_i}$ is the Simeck Round Function with $C \oplus (z_j)_i$ Acting as the Round Key

state $(1, 1, 1, 1, 1)$. When the rounds number is larger than 31, the sequence repeats itself. Simeck64/128 uses another m-sequence z_1 with period 63, which is generated by the primitive polynomial $X^6 + X + 1$ with the initial state $(1, 1, 1, 1, 1, 1)$.

Number of Rounds. The number of rounds T for Simeck32/64, Simeck48/96, and Simeck64/128 are 32, 36, and 44, respectively.

2.2 Design Rationales

In Simeck, we use a slightly simplified version of the round function of SIMON. The round function of SIMON can be expressed as

$$R'_{k_i}(l_i,\ r_i) = (((l_i \lll 1) \odot (l_i \lll 8)) \oplus (l_i \lll 2) \oplus r_i \oplus k_i,\ l_i),$$

where l_i and r_i are the input words, and k_i is the round key. The operations of the round function only contain bitwise AND, XOR and cyclic shifts, and they are very efficient for hardware implementations. In particular, for Simeck, we change these shift numbers from $(1, 8, 2)$ to $(0, 5, 1)$. We choose our shift numbers in order to realize an acceptable trade-off between hardware performance and security. These modifications will improve the efficiency of hardware implementations, but will have comparable security strengths against certain attacks. More discussions will be given in the following sections.

For the key expansion/schedule algorithm of Simeck, we learn the idea of re-using the round function to update the round-key registers from the design of SPECK.

Concerning the number of rounds for Simeck, we choose the same numbers as the corresponding block ciphers in the SIMON family, in order to have comparable security levels and fair hardware implementation evaluations.

To defeat certain self-similarity attacks such as slide attacks and rotational attacks, we add the round constants C and $(z_j)_i$ into the key expansion process.

The constant $C = 2^n - 4$ is also used in the key expansion of SIMON. The polynomials for the two m-sequences z_0 and z_1 are chosen to have minimum numbers of components, such that their hardware implementations will have small footprints.

3 Hardware Implementations

We discuss the hardware implementations of the Simeck family of block ciphers in this section.

3.1 Metrics and Design Flow

We use the Synopsys Design Compiler Version D-2010.03-SP4 to synthesize the RTL of the designs into netlist based on the STMicroelectronics CMOS 65 nm CORE65LPLVT_1.20V and IBM CMOS 130 nm CMR8SF-LPVT Process SAGE v2.0 standard cell libraries with both having a typical 1.2 V voltage, and 25°C temperature. Cadence SoC Encounter v09.12-s159_1 is used to finish the Place and Route phase in order to generate the layout of the designs. We use Mentor Graphics ModelSim SE 10.1a to conduct functional simulation of the designs and perform timing simulation by using the timing delay information generated from SoC Encounter as well. The areas of the designs after the logic synthesis are provided for comparisons with previous ciphers, and a more accurate area after the Place and Route is also provided for using the ciphers in practical cases. The densities used for the Place and Route phase for CMOS 130 nm and 65 nm are 0.92 and 0.93 respectively, in order to make a trade-off between area and maximum operating frequency when the densities are high enough. As usual, the area is measured in gate equivalents (GEs), and one GE is equivalent to the physical area required for the two-input one-output NAND gate with the lowest driving strength of the corresponding technology.

We use SoC Encounter v09.12-s159_1 to generate the accurate power consumption based on the activity information generated from the timing simulation with a frequency of 100 KHz, and a duration time of 0.1s. We do so because the 100 KHz clock frequency is widely used for benchmarking purpose in resource-constrained applications and 0.1 s is long enough to provide an accurate activity information for all the signals.

Moreover, the critical path is obtained after the Place and Route phase, which would be more accurate than the estimated value obtained from logic synthesis. Hence, the maximum clock frequency which can be operated for a specific design is obtained.

In fact, during the analysis of the previous results [3,11,24,27,28], the ASIC results for various implementations differ not only in the basic gate technology but also in the types of flip-flops used. In order to be fair to compare our results with the previous ones, we provide the areas of some basic gates in our specific libraries and the library used in [3] by the researchers from the NSA for SIMON in Table 2. In addition, all the areas of basic gates provided here are the smallest

Table 2. The Areas of Basic Gates in the Libraries

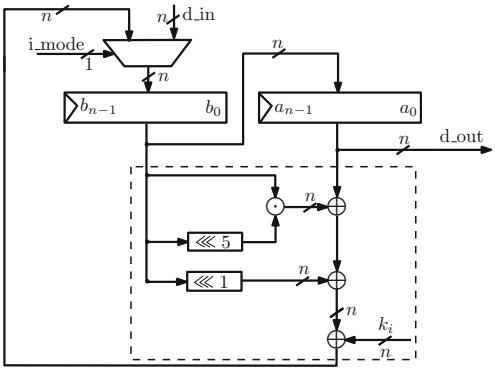
	IBM 130 nm-8RF (NSA [3])	IBM 130 nm-CMR8SF -LPLVT	ST CMOS 65 nm
NAND	1	1	1
AND	1.25	1.25	1.25
OR	1.25	1.25	1.5
NOT	0.75	0.75	0.75
XOR	2	2	2.25
XNOR	2	2	2.25
2-1 MUX	2.25	2.25	2
DFF	4.25	4.25	3.75
1-bit full adder	5.75	5.75	4.5
Scan FF	6.25	5.5	4.75

ones in the library. We observe that our IBM 130 nm library is almost the same as the IBM 130 nm library used by the researchers from the NSA [3] except the scan flip-flops in terms of the areas of the basic gates.

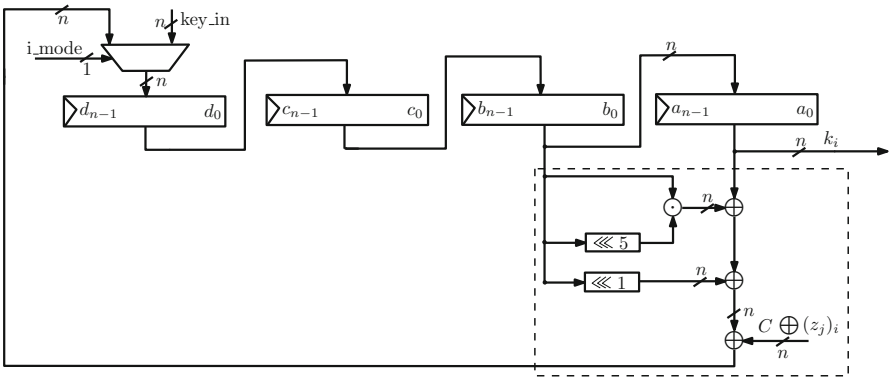
3.2 Two Different Hardware Architectures for Simeck

In this section, we target low-area implementations of Simeck and make a trade-off between area and throughput. Meanwhile, we still keep a very high operating frequency. We give two architectures for the implementations: one is parallel architecture, and another one is fully serialized architecture. Moreover, we provide a block diagram of the top-level I/O interface between the cipher and the outside environment in order to provide a benchmark for the future implementations and comparisons with other ciphers.

Parallel Architecture. The parallel architecture processes one round of the message in one clock cycle, and one round of the key schedule at the same clock cycle, as shown in Fig. 3. This architecture provides a very high throughput while keeping a compact design. The round function in Fig. 3(a) includes three parts: $2n$ flip-flops, a n-bit width 2-to-1 multiplexer, a combinational circuit (dashed box) to compute the feedback data for the multiplexer. Inside the $2n$ flip-flops, n flip-flops are for the message b, and the other n ones are for the message a. The multiplexer is used to select the initial plaintext or the feedback data from the combinational circuit for the message b. The combinational circuit includes one n-bit AND gate, three n-bit XOR gates, and two shift modules (cyclic shift to the left by 5 bits and 1 bit). The shift modules cost no extra hardware resources, because they can be done by rewiring the corresponding signals. When the cipher runs, the n-bit data from the message block b shifts to message block a, and simultaneously, the message block b loads a new n-bit data from the multiplexer



(a) Parallel Datapath for the Round Function



(b) Parallel Datapath for the Key Schedule

Fig. 3. Parallel Architecture for Simeck

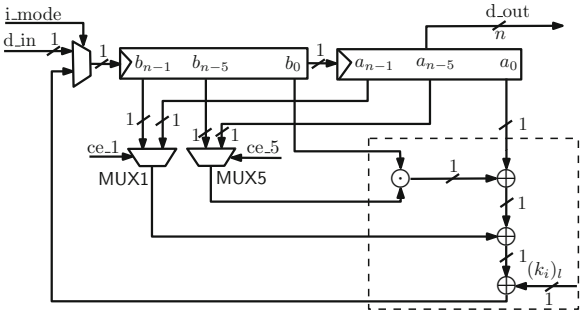
until the cipher stops. The round key k_i in the combinational circuit for every round comes from the key schedule function, which generates a key for every rounds until the cipher outputs the ciphertext.

Different from the round function architecture, the key schedule in Fig. 3(b) has four n-bit key blocks and one input to the combinational circuit (dashed box) is different. This n-bit input to the key schedule is a combination of an $(n-1)$-bit constant and a 1-bit signal generated from the control circuit.

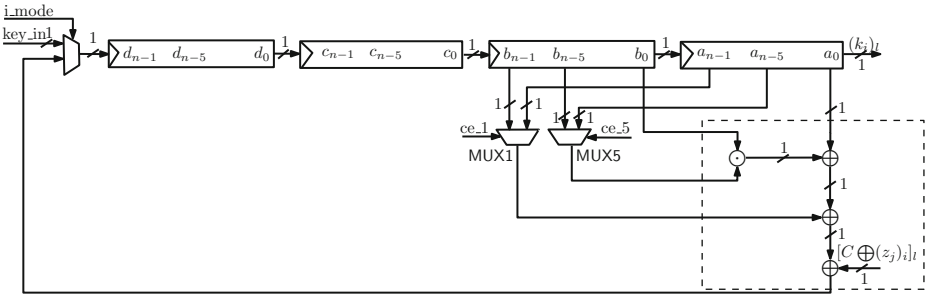
All the flip-flops in the round function and key schedule are standard flip-flops without chip-enable in our architecture. In addition, there are only two n-bit width 2-to-1 multiplexers in total in our architecture to select the initial data or feedback data, where one is for the round function, and the other is for the key schedule. Moreover, the latency for generating a ciphertext using our parallel architecture is $T+4$, where T is the total number of rounds.

Partially Serialized Architecture. In order to make a trade-off between area, throughput, and power consumption, we provide a partially serialized architecture. This architecture processes only several bits in the round function and the key schedule during one clock cycle. The specific partially serialized size (par_sz) of Simeck are summarized as follows:

$$\begin{aligned} \text{Simeck32/64} &: 1, 2, 4, 8, \\ \text{Simeck48/96} &: 1, 2, 3, 4, 6, 8, 12, \\ \text{Simeck64/128} &: 1, 2, 4, 8, 16. \end{aligned}$$



(a) Fully Serialized Datapath for the Round Function



(b) Fully Serialized Datapath for the Key Schedule

Fig. 4. Fully Serialized Architecture for Simeck

Besides the round counter (i in Figs. 3 and 4) in the control circuit, there is another counter to control the rounds of the specific serialized size in the partially serialized architecture. The range of this serialized counter (l in Fig. 4) is between 0 and n/par_sz - 1. In total, the latency for generating a ciphertext is $(n/\text{par\_sz}) \cdot (T + 4)$, where T is the total number of rounds.

A fully serialized architecture is shown in Fig. 4. In this architecture, the multiplexer (MUX), and combinational circuit (dashed box) are all 1-bit width, which save a lot of area. Compared to the parallel architecture, there are two

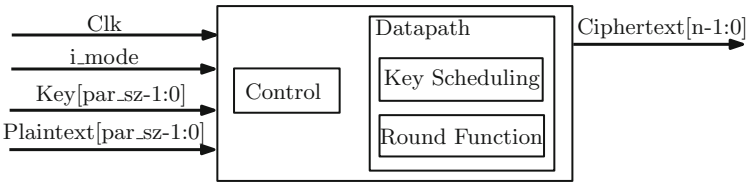


Fig. 5. The Top-level I/O Interface between the Cipher and the Outside Environment

more multiplexers. They are used to select the cyclic shift inputs. The MUX1 is used for the left shift by 1 bit, and MUX5 is used for left shift by 5 bits. The MUX1 selects b_{n-1} as input when the serialized counter equals 0, and chooses a_{n-1} when the serialized counter is larger than 0. Similarly, the MUX5 selects b_{n-5} when the serialized counter is smaller than or equal to 4, and chooses a_{n-5} when the serialized counter is larger than 4.

The partially serialized architecture with par_sz larger than 1 is similar to the fully serialized architecture, where the multiplexer and combinational circuit are par_sz-bit width and the selection signals for the multiplexers (MUXes selection circuitry) are different for various values of par_sz.

The Top-Level I/O Interface for Different Architectures. As discussed in Sect. 3.1, the area of the chip depends on not only the area of the basic gates, but also the adopted types of flip-flops. We provide a top-level I/O interface between the cipher and the outside environment as shown in Fig. 5. We do not have a Finite State Machine (FSM) to control the circuit with the purpose of reducing the entire area as much as possible. In our top-level architecture, the cipher is always running and it is controlled by the outside signal `i_mode`. Therefore, we only have two modes in our architecture: loading phase and running phase. The cipher goes into loading phase when `i_mode` equals 0, and it loads the initial data from the inputs `Key` and `Plaintext`. Later on, the cipher begins running phase when `i_mode` equals 1. The user obtains the `Ciphertext` at the end of the running phase. Then, `i_mode` returns back to 0, another `Plaintext` encryption begins. As our architecture never stops, all the flip-flops in the datapath are standard flip-flops without chip-enable signals. This property makes our design ever smaller in terms of area. This architecture presents a benchmark ASIC implementation of Simeck and can be used to fairly compare with the hardware results of other ciphers.

It is worth mentioning that the parallel architecture can be viewed as a special case of the partially serialized case when par_sz equals n. However, the two cases have different architectures as depicted in Figs. 3 and 4.

Our top-level architecture includes two parts: the control circuit and the datapath. The control circuit for the parallel architecture is used to provide the key constant from the LFSR as described in Sect. 2. However, an extra serialized counter in the control circuit is needed for the partially serialized architecture.

The datapath includes round function and key scheduling, and they are described as above for the parallel architecture and partially serialized architecture.

Recently, LFSR or NLFSR based counters are used to replace binary counter in the control circuit in hardware implementations [20], because they only contain flip-flops and some combinational feedback logics without using a full-adder. Hence, it can reduce the area to some extent if the LFSR or NLFSR counter does not incur extra area in the datapath. However, the serialized counter in our partially serialized architecture is used in two aspects: one is used to count the serialized rounds in the control circuit and another one is used to select the two multiplexers (MUX1 and MUX5) in the datapath. After a theoretical and practical analysis of the effects of the LFSR or NLFSR counter in our partially serialized architecture, we discovered that the total area using binary serialized counter is the smallest one because the LFSR or NLFSR counter results in more additional area in the datapath (i.e., the area of the multiplexers selection circuitry) than the area saved by replacing the binary counter with LFSR or NLFSR counter in the control circuit. Therefore, the binary serialized counter is used for our partially serialized architecture.

3.3 Hardware Evaluations of Simeck

We use three different compilation techniques in the Design Compiler to perform hardware optimizations: simple compile, compile ultra and compile ultra with clock gating. The simple compile option can provide us the hierarchical architectures of the design, and the areas of specific sub-modules. The compile ultra option can make deeper optimizations in a way of optimizing the entire module together, thereby reducing the area and power consumption significantly [11,20]. The clock gating technique can further reduce the area and power consumption [11]. However, we use all standard flip-flops without chip-enable signals for the parallel architecture. Only the LFSR generating the key constant in the control circuit uses the flip-flops with chip-enable signals, which costs 5, 6, and 6 flip-flops for Simeck32/64, Simeck48/96, and Simeck64/128 respectively. Therefore, the clock gating optimization affects only a little of our results in terms of area and power consumption.The ASIC implementation results of Simeck and SIMON in CMOS 130 nm are shown in Tables 3 and 4, and the corresponding results of Simeck and SIMON in CMOS 65 nm are shown in Tables 7 and 8. It is worth noting that these results are obtained without using scan registers.

We provide the best area results before and after the Place and Route phase using compile ultra or compile ultra plus clock gating. These results can be used for comparing with other ciphers or for practical purpose. The maximum frequency corresponding with the best optimization technique is given and it is calculated by using the critical path. The calculated throughput is based on the latency in our architectures and it is the same as SIMON. The difference of the total power consumption among the three different optimizations is marginal. Therefore, we only provide a total power consumption using compile ultra at 100 KHz, which is typical for benchmarking purpose. Since the operating frequency is too small, the static power consumption dominates the total power consumption.

Table 3. Our Implementation Results of Simeck32/64, 48/96, 64/128 in 130 nm

Simeck	Partial serial	CMOS 130 nm				
		Area (GEs)		Max frequency (MHz)	Throughput @100 KHz (Kbps)	Total power @100 KHz (μW)
		Before P&R	After P&R			
Simeck32/64	1-bit	505[a]	549[a]	292	5.6	0.417
	2-bit	510[b]	555[b]	288	11.1	0.431
	4-bit	533[b]	579[b]	312	22.2	0.463
	8-bit	591[b]	642[b]	289	44.4	0.523
	16-bit	695[a]	756[a]	526	88.9	0.606
Simeck48/96	1-bit	715[b]	778[b]	299	5.0	0.576
	2-bit	722[b]	785[b]	294	10.0	0.593
	3-bit	731[b]	794[b]	268	15.0	0.611
	4-bit	748[b]	813[b]	284	20.0	0.628
	6-bit	770[b]	837[b]	287	30.0	0.651
	8-bit	801[b]	871[b]	284	40.0	0.688
	12-bit	858[b]	933[b]	283	60.0	0.742
	24-bit	1027[a]	1117[a]	512	120.0	0.875
Simeck64/128	1-bit	924[a]	1005[a]	288	4.2	0.754
	2-bit	933[b]	1015[b]	303	8.3	0.778
	4-bit	958[b]	1041[b]	271	16.7	0.803
	8-bit	1013[b]	1101[b]	280	33.3	0.834
	16-bit	1132[b]	1231[b]	301	66.7	0.977
	32-bit	1365[a]	1484[a]	512	133.3	1.162

[a] Area obtained by using compile ultra only.
[b] Area obtained by using compile ultra and clock gating.

However, the static power consumption is larger in CMOS 65 nm than in CMOS 130 nm, which is the reason why the total power consumption is larger in CMOS 65 nm as shown in Tables 7 and 8.

Besides having a very small area, our another observation is that most part of the area for all the architectures are built of the sequential logics, especially for the fully serialized architecture. Take Simeck32/64 for example. 86 %, 85 %, 82 %, 76 %, and 70 % of the entire area are sequential logics for the cases that par_sz equals 1, 2, 4, 8, and 16 respectively. From the data provided, we can obtain that the fully serialized architecture is built of about 90 % sequential logics. Similar conclusions can be obtained for Simeck48/96 and Simeck64/128.

We provide a range of options between the area, throughput, and power consumption in our ASIC implementations. Taking Simeck32/64 in CMOS 130 nm for illustration, we can achieve a throughput of 5.6 Kbps at the area cost of 505 GEs (before the Place and Route) and 549 GEs (after the Place and Route)

Table 4. Our Implementation Results of Simon32/64, 48/96, 64/128 in 130 nm

Simon	Partial serial	CMOS 130 nm					
		Area (GEs)			Max frequency (MHz)	Throughput @100 KHz (Kbps)	Total power @100 KHz (μW)
		Before P&R	After P&R	NSA before P&R			
Simon32/64	1-bit	517[b]	562[b]	523	331	5.6	0.421
	2-bit	532[a]	578[a]	535	306	11.1	0.439
	4-bit	563[b]	612[b]	566	283	22.2	0.479
	8-bit	623[a]	677[a]	627	367	44.4	0.540
	16-bit	715[a]	778[a]	722	456	88.9	0.645
Simon48/96	1-bit	733[b]	796[b]	739	258	5.0	0.579
	2-bit	745[b]	810[b]	750	289	10.0	0.601
	3-bit	756[b]	822[b]	763	291	15.0	0.615
	4-bit	778[b]	846[b]	781	287	20.0	0.642
	6-bit	800[b]	869[b]	804	289	30.0	0.670
	8-bit	833[b]	905[b]	839	238	40.0	0.706
	12-bit	895[b]	973[b]	898	307	60.0	0.777
	24-bit	1055[a]	1147[a]	1062	467	120.0	0.929
Simon64/128	1-bit	944[b]	1026[b]	958	225	4.2	0.762
	2-bit	955[b]	1038[b]	968	244	8.3	0.780
	4-bit	988[b]	1074[b]	1000	290	16.7	0.818
	8-bit	1043[b]	1134[b]	1057	296	33.3	0.866
	16-bit	1174[b]	1276[b]	1185	293	66.7	1.024
	32-bit	1403[a]	1524[a]	1417	465	133.3	1.239

[a]Area obtained by using compile ultra only.
[b]Area obtained by using compile ultra and clock gating.

with the power consumption of 0.417 μW. However, a two-fold throughput (11.1 Kbps) can be obtained with only 5 and 6 extra GEs (before and after the Place and Route respectively), and 0.014 μW extra power consumption. With more extra area and power consumption, we can get even higher throughput.

4 Result Comparisons Between Simeck and Simon

We compare our area results before the Place and Route of Simeck and Simon in CMOS 130 nm with the Simon results of the NSA researchers [3]. This is because the NSA researchers only provide the area results before the Place and Route. The comparison is shown in Fig. 6. We can observe that our Simon results are all smaller than that of NSA's results, and our Simeck results are even smaller than Simon for all the cases shown in Fig. 6.

From the theoretical point of view, Simeck is designed to have a smaller area due to the following considerations: the simplified key schedule, the simplified LFSR to generate the key constant, and the decreased shift numbers in the

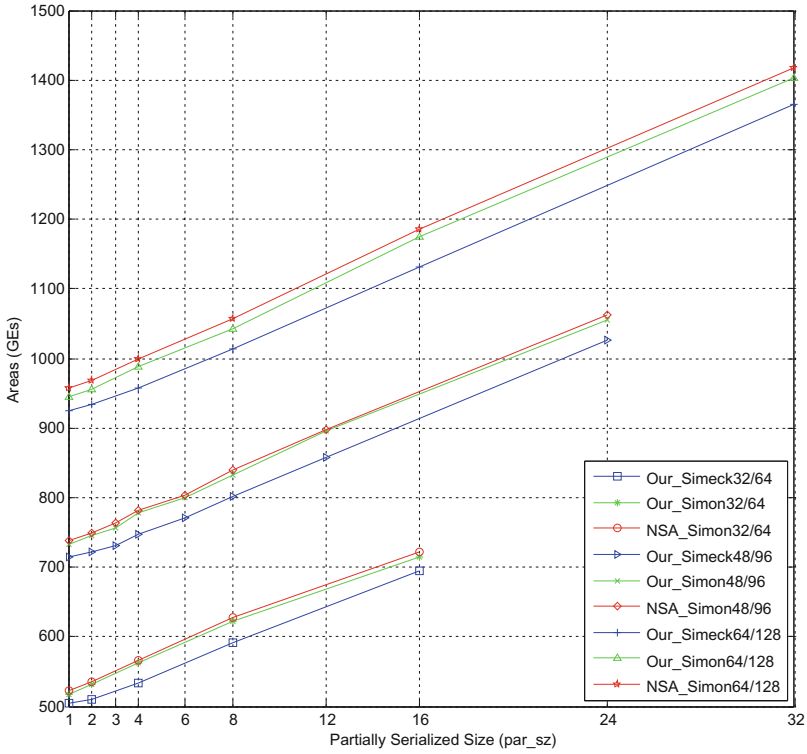


Fig. 6. Comparisons of Areas (before the Place and Route) between the Implementation Results of the NSA Researchers' and Ours in CMOS 130 nm

round function. It is worth noting that the decreased shift numbers do not affect any area in the parallel architecture, and it only affect the area in the partially serialized architecture.

The construction of the combinational circuit in the key schedule of SIMON32/64, 48/96, 64/128 and Simeck32/64, 48/96, 64/128 in the parallel architecture are shown as follows:

SIMON	$(2n+1)$ XOR + $(n-1)$ XNOR
Simeck	$(n+1)$ XOR + $(n-1)$ XNOR + n AND

In general, one XOR gate is larger than one AND gate. Therefore, the key schedule of SIMON is larger than that of Simeck. The LFSRs used to generate the key constants for SIMON32/64 and SIMON48/96 are defined by the primitive polynomial $X^5 + X^4 + X^2 + X + 1$, and the LFSR for SIMON64/128 is defined by $X^5 + X^3 + X^2 + X + 1$. They are all 2 XOR gates (4 GEs) bigger than the ones used in corresponding Simeck, as described in Sect. 2. The decreased shift numbers of the round function and key schedule reduce 1 MUX for the

Table 5. Breakdown of the Implementation Results before the Place and Route in CMOS 130 nm

		Simeck32/64 (130 nm)		Simon32/64 (130 nm)	
Components		Parallel (GEs)	Fully serialized (GEs)	Parallel (GEs)	Fully serialized (GEs)
Control		31	71	35	75
Datapath	Round_combinational circuit	112	7	112	7
	Key_combinational circuit	80	5	96	8
	Sequential + MUXes	474	434	474	443
Totals	Compile simple	697	517	717	533
	Compile ultra	695	505	717	520
	Compile ultra + clock gating	695	506	715	517

inputs to the combinational circuits of the round function and the key schedule respectively (2 MUXes in total, $2 \cdot 2.25$ GEs/MUX = 4.5 GEs), and also some logics to select the MUXes.

From the practical point of view, we break down the area results before the Place and Route in CMOS 130 nm for Simeck32/64, and Simon32/64 in our implementations, as shown in Table 5. For parallel architectures, the differences of the control circuits and the key combinational circuits between Simeck32/64 and Simon32/64 are 4 GEs (key constant) and 16 GEs respectively. The results are almost the same as the theoretical analysis. For the fully serialized architecture, the control circuit is reduced by 4 GEs (key constant), the key combinational circuit (dashed box in Fig. 4) is reduced by 3 GEs, and the 2 MUXes plus the MUXes selection circuitry are reduced by 9 GEs for Simeck32/64 (i.e., a total saving of 16 GEs), compared to that of Simon32/64. Therefore, the practical results match the theoretical analysis. Simeck is smaller than Simon for both parallel architecture and partially serialized architecture.

The main area cost for Simon comes from the registers storing the message block and the key. In order to design a smaller cipher than Simon, we can reduce the areas of only the round function, key schedule, key constant, and multiplexers. For fully serialized architecture of Simon32/64 (see Table 5), the combined area of these blocks is 34.5 GEs ($7 + 8 + 6 + 6 \cdot 2.25$/MUX), which accounts for only about 6.4 % (34.5/533) of the total area. Simeck32/64 reduces this by 16 GEs, a saving of more than 46 %. This reduction leads to 2.3 % smaller total area in comparison to our implementations of Simon32/64 in CMOS 130 nm, and 3.4 % smaller in comparison to the original Simon32/64 results (see Table 1). Similarly, the fully serialized architectures of Simeck48/96, 64/128 are 2.5 %, 2.1 %, respectively, smaller than our implementations of Simon48/96, 64/128 and they are 3.3 % and 3.5 %, respectively, smaller than the original implementation results of Simon48/96, 64/128 in CMOS 130 nm (see Table 1). For the parallel architectures of Simon, these blocks consume a larger fraction (about 29 %) of the total area (see Table 5). Simeck32/64, 48/96, 64/128 achieve the saving of 3.7 %, 3.3 %, and 3.7 % respectively, compared to the original results

of Simon32/64, 48/96, 64/128 (see Tables 3 and 4). The choice of the values of the shift numbers plays a significant role in the area reduction of the partially serialized architecture. Because the parallel architecture does not contain the MUXes for the inputs to the combinational circuit (dashed box), the total area reduction is only slightly greater than the fully serialized architecture.

From Tables 3 and 4, we can also observe that the power consumption of Simeck is smaller than Simon for all the cases in CMOS 130 nm using the same optimizations. This is easy to understand because the area of Simeck is smaller than Simon. This conclusion also holds for CMOS 65 nm in Tables 7 and 8.

In summary, Simeck is smaller than Simon in terms of area and power consumption in both CMOS 130 nm and CMOS 65 nm techniques.

5 Security Analysis

In this section, we give the security analysis of the Simeck family of block ciphers. Due to its similitude with Simon and Speck, most of the next analysis follow from the best known attacks against the Simon and Speck families of block ciphers. As we show in the following, the security level of Simeck is comparable to those of Simon, which is reasonable to be used in practice. Indeed, the number of rounds chosen for Simeck is sufficiently high with respect to the best known attacks on reduced versions. Moreover, it is worth noticing that the ARX (Addition-Rotation-XOR) design of Simeck borrowed from Speck, using the round function as key-schedule, did not lead to a weakness so far. In a recent paper [22], Kölbl *et al.* study the influence of the shifts in Simon-like ciphers. They provide some set of parameters that are optimal with respect to differential and linear properties, and diffusion. Our parameters seem comparable to theirs because we take also into account hardware efficiency and other types of cryptanalysis (e.g., impossible differential cryptanalysis).

Differential/Linear Attacks [6,23]**.** Since the differential and linear behaviors of Simon and Simeck are very closely related, it makes sense to use the best known differential and linear attacks of Simon to evaluate the security of Simeck against these attacks. This is why we have essentially followed the procedure of [7] to evaluate the security of Simeck against differential cryptanalysis. It is then possible to perform an attack on 19 rounds of Simeck32/64 with the time and data complexity 2^{34} and $2^{31.5}$ respectively. It is also possible to attack 20 rounds out of 36 of Simeck48/96 with the time and data complexity 2^{75} and 2^{46} as well as an attack of 26 rounds out of 44 of Simeck64/128 with the time and data complexity 2^{121} and 2^{63}.

For the best cryptanalytic result using linear attacks against Simon, we refer to [1]. Because of the similar structure of Simeck, we verified that those results are also conform with respect to Simeck. For Simeck32/64, we can cover 12 rounds with the data complexity 2^{31}. For Simeck48/96, we can cover 15 rounds with the data complexity 2^{43}. Finally, it is possible to perform a linear cryptanalysis of Simeck64/128 up to 19 rounds with 2^{123} known plaintexts. All these attacks have a success probability of 0.997.

Table 6. Comparison of Impossible Differential Attacks against SIMON and Simeck

Algorithm	#Rounds	Data	Time	Memory
SIMON32/64 [10]	19	2^{32}	$2^{62.56}$	2^{44}
SIMON48/96 [10]	21	2^{48}	$2^{94.73}$	2^{70}
SIMON64/128 [10]	22	2^{64}	$2^{126.56}$	2^{75}
Simeck32/64	20	2^{32}	$2^{62.6}$	2^{56}
Simeck48/96	24	2^{48}	$2^{94.7}$	2^{74}
Simeck64/128	25	2^{64}	$2^{126.6}$	2^{79}

Table 7. Our Implementation Results of Simeck32/64, 48/96, 64/128 in 65 nm

Simeck	Partial Serial	CMOS 65nm				
		Area (GEs)		Max frequency (MHz)	Throughput @100 KHz (Kbps)	Total power @100 KHz (μW)
		Before P&R	After P&R			
Simeck32/64	1-bit	454[a]	488[a]	1754	5.6	1.292
	2-bit	465[b]	500[b]	1428	11.1	1.311
	4-bit	494[b]	531[b]	1388	22.2	1.376
	8-bit	550[a]	592[a]	1250	44.4	1.512
	16-bit	644[a]	692[a]	1428	88.9	1.716
Simeck48/96	1-bit	645[b]	693[b]	1562	5.0	1.805
	2-bit	656[b]	706[b]	1538	10.0	1.825
	3-bit	663[b]	712[b]	1282	15.0	1.857
	4-bit	686[b]	738[b]	1333	20.0	1.886
	6-bit	701[b]	753[b]	1282	30.0	1.919
	8-bit	732[b]	787[b]	1388	40.0	2.009
	12-bit	794[a]	854[a]	1219	60.0	2.212
	24-bit	951[a]	1022[a]	2325	120.0	2.44
Simeck64/128	1-bit	828[a]	891[a]	1369	4.2	2.304
	2-bit	838[b]	901[b]	1408	8.3	2.325
	4-bit	869[b]	935[b]	1098	16.7	2.372
	8-bit	918[b]	987[b]	1190	33.3	2.492
	16-bit	1042[a]	1121[a]	1086	66.7	2.869
	32-bit	1263[a]	1358[a]	1282	133.3	3.316

[a] Area obtained by using compile ultra only.
[b] Area obtained by using compile ultra and clock gating.

Since the best known differential and linear trails found on Simeck, and SIMON, only cover a reduced number of rounds, we believe that the full-round Simeck (any version) is sufficiently secure against differential and linear cryptanalysis.

Table 8. Our Implementation Results of SIMON32/64, 48/96, 64/128 in 65 nm

SIMON	Partial Serial	CMOS 65 nm				
		Area (GEs)		Max frequency (MHz)	Throughput @100 KHz (Kbps)	Total power @100 KHz (μW)
		Before P&R	After P&R			
SIMON32/64	1-bit	466[a]	501[a]	1428	5.6	1.311
	2-bit	476[a]	512[a]	1562	11.1	1.331
	4-bit	506[a]	544[a]	1408	22.2	1.381
	8-bit	570[a]	613[a]	1075	44.4	1.585
	16-bit	666[a]	716[a]	2222	88.9	1.751
SIMON48/96	1-bit	661[b]	711[b]	1204	5.0	1.812
	2-bit	670[b]	720[b]	1136	10.0	1.889
	3-bit	682[b]	733[b]	1086	15.0	1.86
	4-bit	699[b]	752[b]	1041	20.0	1.915
	6-bit	724[b]	779[b]	1369	30.0	1.962
	8-bit	757[b]	814[b]	1282	40.0	2.122
	12-bit	819[a]	881[a]	1176	60.0	2.305
	24-bit	982[a]	1056[a]	2222	120.0	2.542
SIMON64/128	1-bit	845[b]	908[b]	1282	4.2	2.336
	2-bit	858[b]	922[b]	1265	8.3	2.366
	4-bit	887[b]	954[b]	1250	16.7	2.423
	8-bit	944[b]	1015[b]	1265	33.3	2.577
	16-bit	1076[a]	1156[a]	1176	66.7	3.068
	32-bit	1305[a]	1403[a]	1694	133.3	3.398

[a] Area obtained by using compile ultra only.
[b] Area obtained by using compile ultra and clock gating.

Impossible Differential Attacks [4]**.** Impossible differential attacks against Simeck cover few more rounds (depending on the version) than for SIMON as it can be seen in Table 6. This is due to the fact that the diffusion of one bit difference is one round slower for Simeck than for SIMON. Nevertheless, this does not damage the overall security of the Simeck family, since the full versions have more rounds.

Algebraic Degree [21]**.** We computed that after 5 rounds, the algebraic degree of Simeck (any version) is 13, as the one of SIMON. It is sufficient to ensure that after few more rounds, no attack can exploit properties of the algebraic degree, such as algebraic attack or higher-order differential attack.

Meet-in-the-Middle Attacks [13]**.** Because of the key schedule algorithm of Simeck, many key bits of the master key are processed quickly in the round function of Simeck. This should ensure a good resistance of Simeck against Meet-in-the-Middle (MITM) attacks. Moreover, until now SIMON has not shown to be

a good candidate for MITM attacks. As the round function of Simeck is very similar as the one of SIMON, we believe that Simeck will also be resistant against MITM attacks.

Slide Attacks and Rotational Attacks [8,19]**.** The round constant addition and the key schedule design prevent any efficient slide or rotational attacks.

Related-key Differential Attacks [5]**.** Although SIMON and SPECK have been extensively studied in the past years, no concrete attacks in the related-key setting have been shown. Like SPECK, Simeck reuses its round function in the key schedule part. It is reasonable to think that Simeck has also good cryptographic properties in the related key model.

6 Concluding Remarks

In this paper, we have presented Simeck, a new family of lightweight block ciphers. Simeck is very suitable for resource-constrained devices, such as passive RFID tags and wireless sensor networks. We have provided an extensive exploration for different hardware architectures in order to make a balance between area, throughput, and power consumption for SIMON and Simeck in both CMOS 130 nm and CMOS 65 nm techniques. We have shown that it is possible to design a smaller cipher than SIMON in terms of area and power consumption. Moreover, we have improved the hardware implementations of SIMON given in the original paper. In addition, the similarities between SIMON/SPECK and Simeck allow us to have an idea of the actual security offered by Simeck. Even if the round function of Simeck is quite simple, this round function is iterated a sufficient number of time to provide an adequate security against most known attacks. In conclusion, all of the instances in the Simeck family can meet the area, power consumption, and throughput requirements in the passive RFID tags and they are promising candidates for resource-constrained devices.

We have learnt and understood many techniques about designing hardware-oriented ciphers during the process of completing the design of Simeck. It is interesting to see if we can devise a block cipher with even smaller hardware footprints than Simeck. It also interests us whether we can design, from the theoretical point of view, a smallest block cipher with the minimum number of components. This should be very useful for cryptography researchers to get deep insights into designing and analyzing ciphers.

Acknowledgments. The authors would like to thank the anonymous reviewers for their helpful and constructive comments that greatly contributed to improving the final version of the paper. This work is supported by NSERC Discovery Grants, Strategic Project Grant, and Canadian Microelectronics Corporation.

Appendix A ASIC Implementation Results of Simon and Simeck in CMOS 65 nm

Tables 7 and 8 give our results of Simeck and SIMON in CMOS 65 nm.

Appendix B Test Vectors

Here we list some test vectors for the Simeck family of block ciphers, in the same format as the ones in [3].

Simeck32/64

```
      Key: 1918 1110 0908 0100
Plaintext: 6565 6877
Ciphertext: 770d 2c76
```

Simeck48/96

```
      Key: 1a1918 121110 0a0908 020100
Plaintext: 726963 20646e
Ciphertext: f3cf25 e33b36
```

Simeck64/128

```
      Key: 1b1a1918 13121110 0b0a0908 03020100
Plaintext: 656b696c 20646e75
Ciphertext: 45ce6902 5f7ab7ed
```

References

1. Alizadeh, J., Alkhzaimi, H.A., Aref, M.R., Bagheri, N., Gauravaram, P., Kumar, A., Lauridsen, M.M., Sanadhya, S.K.: Cryptanalysis of SIMON variants with connections. In: Sadeghi, A.-R., Saxena, N. (eds.) RFIDSec 2014. LNCS, vol. 8651, pp. 90–107. Springer, Heidelberg (2014)
2. Armknecht, F., Hamann, M., Mikhalev, V.: Lightweight authentication protocols on ultra-constrained RFIDs - myths and facts. In: Sadeghi, A.-R., Saxena, N. (eds.) RFIDSec 2014. LNCS, vol. 8651, pp. 1–18. Springer, Heidelberg (2014)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013). http://eprint.iacr.org/
4. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. J. Cryptology **18**(4), 291–311 (2005)
5. Biham, E., Dunkelman, O., Keller, N.: A unified approach to related-key attacks. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 73–96. Springer, Heidelberg (2008)
6. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. J. Cryptology **4**(1), 3–72 (1991)
7. Biryukov, A., Roy, A., Velichkov, V.: Differential analysis of block ciphers SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 546–570. Springer, Heidelberg (2015)
8. Biryukov, A., Wagner, D.: Slide attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, p. 245. Springer, Heidelberg (1999)
9. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)

10. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: applications to CLEFIA, camellia, LBlock and simon. In: Proceedings of Advances in Cryptology - ASIACRYPT 2014–20th International Conference on the Theory and Application of Cryptology and Information Security, Part I, pp. 179–199. Kaoshiung, Taiwan, R.O.C, December 7–11 (2014)
11. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — a family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
12. De Cannière, C., Preneel, B.: Trivium specifications. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030 (2005)
13. Diffie, W., Hellman, M.E.: Exhaustive cryptanalysis of the NBS data encryption standard. Computer **10**(6), 74–84 (1977)
14. EPCglobal. EPC Class 1 Generation 2 Standard (2013). http://www.gs1.org/sites/default/files/docs/uhfc1g2/uhfc1g2_2_0_0_standard_20131101.pdf
15. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: a new family of lightweight block ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
16. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
17. Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. Int. J. Wireless and Mobile Comput. **2**(1), 86–93 (2007)
18. Juels, A., Weis, S.A.: Authenticating pervasive devices with human protocols. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 293–308. Springer, Heidelberg (2005)
19. Khovratovich, D., Nikolić, I.: Rotational cryptanalysis of ARX. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 333–346. Springer, Heidelberg (2010)
20. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: a block cipher for IC-printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
21. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
22. Kölbl, S., Leander, G., Tiessen, T.: Observations on the simon block cipher family. In: CRYPTO 2015, LNCS. Springer, Heidelberg (2015)
23. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
24. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the Limits: a very compact and a threshold implementation of AES. In: Advances in Cryptology - EUROCRYPT 2011, pp. 69–88. Springer, (2011)
25. Nawaz, Y., Gong, G.: WG: a family of stream ciphers with designed randomness properties. Inf. Sci. **178**(7), 1903–1916 (2008)
26. Needham, R.M., Wheeler, D.J.: TEA Extensions, Technical Report, University of Cambridge, October 1997
27. Plos, T., Dobraunig, C., Hofinger, M., Oprisnik, A., Wiesmeier, C., Wiesmeier, J.: Compact hardware implementations of the block ciphers mCrypton, NOEKEON, and SEA. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 358–377. Springer, Heidelberg (2012)
28. Rolfes, C., Poschmann, A., Leander, G., Paar, C.: Ultra-lightweight implementations for smart devices – security for 1000 gate equivalents. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 89–103. Springer, Heidelberg (2008)

29. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: an ultra-lightweight block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011)
30. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: Twine: a lightweight, versatile block cipher. In: ECRYPT Workshop on Lightweight Cryptography, pp. 146–169 (2011)
31. Wheeler, D.J., Needham, R.M.: TEA: a tiny encryption algorithm. In: Proceedings of Fast Software Encryption: Second International Workshop. Leuven, pp. 363–366. Belgium, 14–16 December 1994
32. Wu, W., Zhang, L.: LBlock: a lightweight block cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011)
33. Yap, H., Khoo, K., Poschmann, A., Henricksen, M.: EPCBC - a block cipher suitable for electronic product code encryption. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 76–97. Springer, Heidelberg (2011)
34. Yeager, D.J., Sample, A.P., Smith, J.R., Smith, J.R.: WISP: a passively powered UHF RFID tag with sensing and computation. In: RFID Handbook: Applications, Technology, Security, and Privacy, pp. 261–278 (2008)