Efficient FPGA Implementations of PRINT$_{\text{CIPHER}}$

[1]Tadashi Okabe

Information Technology Group
Tokyo Metropolitan Industrial Technology Research Institute, Tokyo, Japan

***Abstract* — This article presents field programmable gate array (FPGA) implementations of PRINT$_{\text{CIPHER}}$, which is an encryption algorithm for protecting important data in restricted computing environments such as lightweight devices and Radio Frequency Identifier (RFID) cards. PRINT$_{\text{CIPHER}}$ has been designed to ensure integrity and authentication with minimum resources. This article is the first work about FPGA implementations of PRINT$_{\text{CIPHER}}$ and proposes firstly efficient hardware architectures for PRINT$_{\text{CIPHER}}$. Hardware implementation results for these architectures have been evaluated in this article. The proposed architectures are synthesizable and implemented on Spartan-3, Spartan-6 and Artix-7 Xilinx FPGAs.**

***Keywords* — Lightweight block cipher, Hardware implementation, FPGA, Hardware architecture, PRINT$_{\text{CIPHER}}$**

I. INTRODUCTION

Currently, the Internet of Things (IoT) paradigm connects all industrial and consumer electronics manufacturers to worldwide communication networks. In the IoT society, both general purpose computers and lightweight portable electronics need higher security. Regarding security, standard block ciphers such as Triple DES [1] or AES [2] are readily available. However, these ciphers cannot be applied to small, lightweight devices that are resource-constrained or have integrated circuits (ICs) with low processing power. However, many lightweight block ciphers are now being proposed for use with radio frequency identification (RFID) tags, small field programmable gate arrays (FPGAs), micro-controllers, and other lightweight devices.

PRINT$_{\text{CIPHER}}$ is a lightweight block cipher designed for IC printing and efficient hardware implementations in resource-constrained compact devices. It is a substitution-permutation network (SPN) structure with a substitution layer that consists of sixteen 3-bit SBOXes. Its permutation layer is the linear diffusion of simple bit permutation. PRINT$_{\text{CIPHER}}$ has 48- or 96-bit block sizes for plaintext or ciphertext, and 80- or 160-bit key sizes for the cipher key, and the number of rounds is 48 or 96. The cipher key is divided into two parts: the first is the XOR function with intermediate round data and the second is the key permutation function. All components of PRINT$_{\text{CIPHER}}$ easily fit into the 4-bit or 6-bit input lookup table (LUT; LUT is 4-bit or 6-bit input function generators and constitute the basic building blocks of most recent reconfigurable devices) of FPGAs, and its key scheduling allows identical round keys in each round. In addition to its compact efficiency in hardware implementation, the identical round key at each round and the small 3-bit SBOX speeds up the PRINT$_{\text{CIPHER}}$ encryption process. In practice, PRINT$_{\text{CIPHER}}$ has potential applications for resource-constrained compact crypt devices and high throughput data encryption fields.

This article presents FPGA implementations of PRINT$_{\text{CIPHER}}$ and compares the performances with recent lightweight block ciphers such as PRESENT, SIMON and several others. The first presented architecture computes one round per clock cycle, while the second is based on the 3-bit word-serial architecture presented in PRINT$_{\text{CIPHER}}$ [3] for application-specific integrated circuits (ASICs). Our main contribution is the architectures that are also 3-bit word-serial by nature and perform PRINT$_{\text{CIPHER}}$ computations based on finite state machine (FSM) models. The first architecture (one round per clock cycle) is actually better than the second one (3-bit word-serial), because it achieves lower area, lower power and better throughput. For this purpose, we investigated various contexts (round iterative rolled and word-serialized iterative rolled implementations, with or without a register key) on the recent Xilinx FPGAs on Spartan-3, Spartan-6 and Artix-7 platforms.

This article is organized as follows. In Section 2, we briefly present the specifications of the PRINT$_{\text{CIPHER}}$ encryption algorithm. Section 3 describes the FPGA design methodology. In Section 4 we describe the implementation results of the proposed hardware architectures and present comparisons with other lightweight block ciphers. Finally, we discuss the conclusions of the study and future work in Section 5.

II. PRINT $_{\text{CIPHER}}$ SPECIFICATIONS

In this section, we describe the specifications and data encryption algorithm of PRINT$_{\text{CIPHER}}$.

A Parameters

PRINT$_{\text{CIPHER}}$ operates on 48- or 96-bit plaintext, cipher text or intermediate data blocks and 80- or 160-bit cipher key blocks. It is an iterative block cipher based on the repetition of 48 or 96 identical key-dependent round functions. More detailed descriptions are given in the following subsections.

B Round Function

The round function is described in Fig. 1, where we distinguish linear diffusion layers and a nonlinear layer. The linear diffusion layer is built from bitwise key additions (denoted as the KEY_XOR Layer in Fig. 1), bit permutations (48-bit wire crossings, denoted as pLayer), sub-key dependent permutations layer (sub-key selected 3-bit wire crossings, denoted as the

KEY_PERM Layer), and least significant 6/7-bitwise round constant additions (denoted as the RC_XOR Layer). The nonlinear layer is built from the parallel application of 3x3 substitution boxes to the intermediate state. For FPGA resource efficiency purposes, these substitution boxes are the same and are constructed from the smaller 3-bit SBOX.

Each round consists of the following five functions:

1. Key XOR with each round input data
2. Linear diffusion
3. Right-most 6/7-bit intermediate data, XORed with round counter constant
4. Key-dependent permutation
5. Nonlinear transformation with SBOX

Each 3-bit word connection in the round function is depicted in Fig. 2. Subsections B.1 to B.5 describe each function layer in detail.

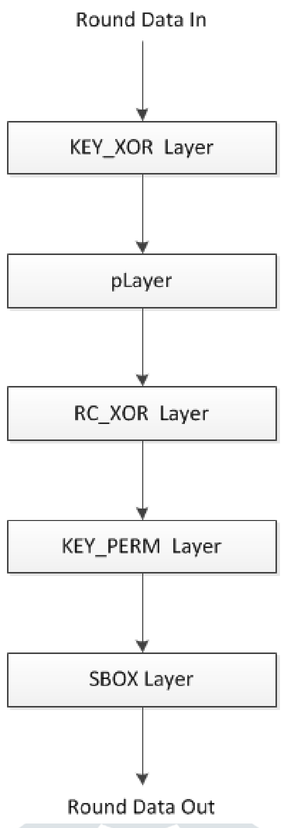


Figure 1 Round function of encryption process Flow

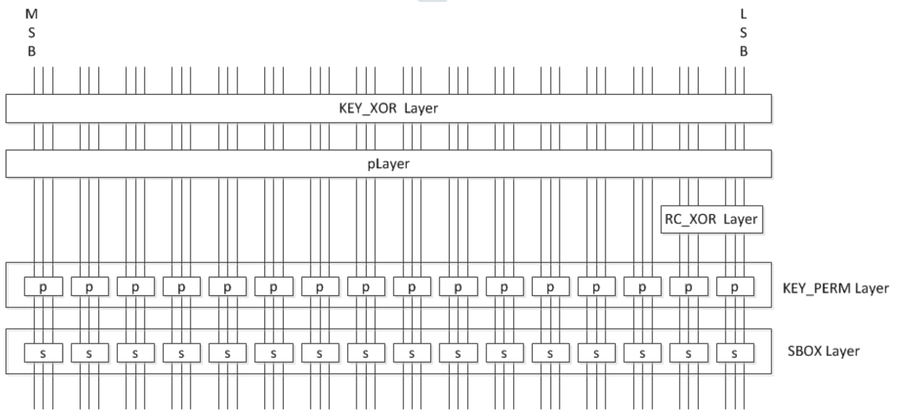


Figure 2 Round Function of PRINT$_{CIPHER}$-48

B.1. Key_XOR Layer This function layer is simply bitwise XOR (exclusive-or) between the current state and 48/96-bit sub-key, sk1. This sub-key is identical in all rounds and is the most significant 48/96-bit sub-key of the cipher key.

B.2. Linear diffusion layer Linear diffusion is described in Fig. 3 as a simple bit permutation layer. Bit *i* of the current state is controlled by the following equation 1

$$P(i) = \begin{cases} 3 \times i \bmod b - 1 & \text{for } 0 \leq i \leq b-2 \\ b-1 & \text{for } i = b-1 \end{cases} \quad (1)$$

Where *b* is the bit size of the cipher blocks, $b \in \{48, 96\}$. In this article, we use the word "pLayer" as the function P(i).

B.3. Round counter The round counter, RC, is specified in the following equation. This counter is the linear feedback shift register (LFSR) sequence and its initial value is 1(hex).

$$\begin{gathered} t = 1 + x_{n-1} + x_{n-2} \\ x_i = x_{i-1} \\ x_0 = t \end{gathered} \quad (2)$$

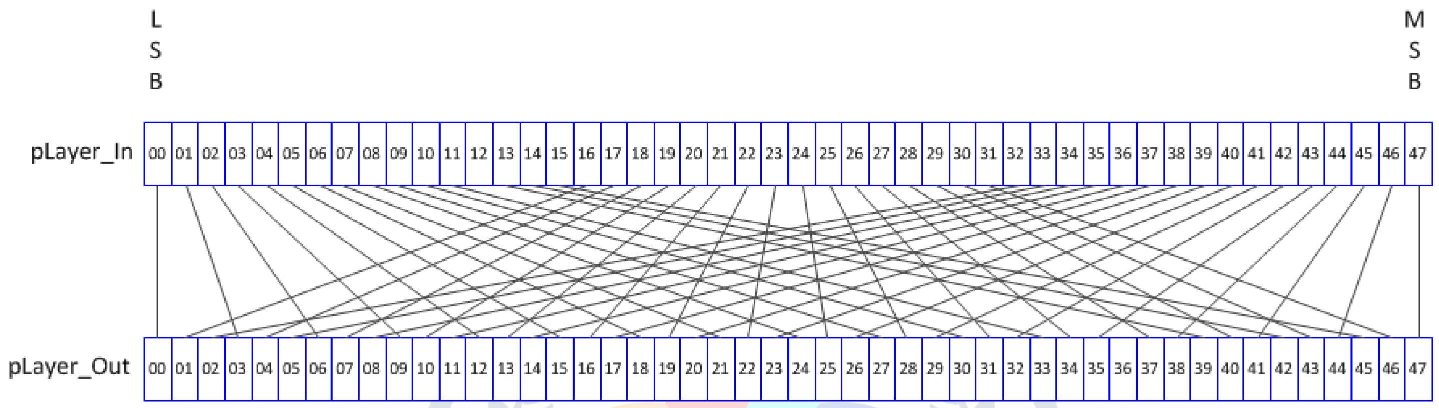
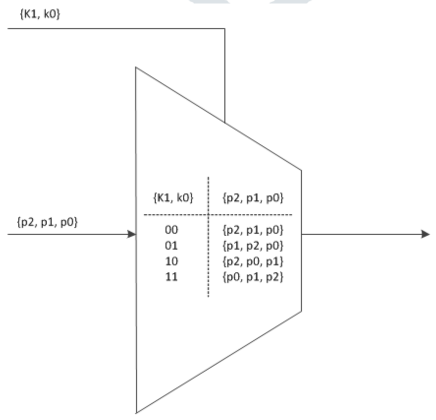


Figure 3 Linear diffusion of PRINT$_{\text{CIPHER}}$-48, pLayer

B.4. RC_XOR Layer This function layer is simply a bitwise XOR (exclusive-or) between the current state and the round counter in equation 2. In this layer, we XORed between the least significant 6/7 bits of intermediate data and the round counter state.

B.5. KEY_PERM Layer Keyed permutation is illustrated in Fig. 4, which is a selector logic block in which the 2-bit sub-key selects the 3-bit input to the 3-bit output corresponding to the permutation law. The permutation law is depicted in Fig. 4. For the keyed permutation layer, 48/96-bit permutations (of each three bits) are picked from a set of four in a key-dependent manner.



{K1, k0}	{p2, p1, p0}
00	{p2, p1, p0}
01	{p1, p2, p0}
10	{p2, p0, p1}
11	{p0, p1, p2}

Figure 4 Key permutations – Block diagram description and truth table for the permutation law

B.6. **SBOX_Layer** The SBOX is described in Table 1. Although most other lightweight block ciphers use a input 4-bit to output 4-bit SBOX, the PRINT$_{CIPHER}$ SBOX is particularly input 3-bit to output 3-bit. This is reasonable for a compact FPGA implementation to fit the nonlinear transformation to LUTs on FPGAs in technology mapping. For the SBOX_Layer, the current state is 16/32 3-bit words. For each word an identical SBOX is used.

Table 1 SBOX table

x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
S(x)	**0**	**1**	**3**	**6**	**7**	**4**	**5**	**2**

C Key Schedule (Round Key and Permutation Key Generation)

The key scheduling process is not used in PRINT$_{CIPHER}$. In each round, the most significant 48/96 bits of the cipher key are the round key that does exclusive-or with round data in the KEY_XOR function. On the other hand, the least significant 32/64 bits of the cipher key is the permutation key, which is a selection flag signal in the KEY_PERM function. Thus, round key and permutation key generation is very simple in the PRINT$_{CIPHER}$ encryption algorithm.

Accordingly, PRINT$_{CIPHER}$ allows encryption/decryption with exactly the same round keys, sk1 and sk2. This means that the hardware of the key schedule is very simple (only the dividing round key and permutation key) and the storage of round keys is not necessary.

D Encryption (Decryption) Process

The complete cipher consists of the iterative 48/96 rounds discussed in subsections B.1 to B.5. Due to the identical round key structures of PRINT$_{CIPHER}$, the key scheduling function only executes in the first encryption (decryption) process. In pseudo C code, we have:

```
Input : plaintext, key;
Output : ciphertext;

PRINTcipher48-encrypt(plaintext, key)
{
    state  = plaintext;
    Key_Schedule(key);
    for (i=0; i<48; i++) Round(state, sk1, sk2, I);
    ciphertext = state;
}

Key_Schedule(key)
{
    sk1 = key[80:32];
    sk2 = key[31:0];
}

Round(state, sk1, sk2, I)
{
    KEY_XOR_Layer(state, sk1);
    pLayer(state);
    RC_XOR_Layer(state, i);
    KEY_PERM_Layer(state, sk2);
    SBOX_Layer(state);
}
```

where key is the 80/160-bit input cipher key, state is the 48/96-bit intermediate round data block and cipher text data after the last round, sk1 is the 48/96-bit round key when it is used in KEY_XOR layer, sk2 is the 32/64-bit permutation key when it is used in the KEY_PERM Layer. The PRINT$_{CIPHER}$ decryption process is simply the reverse of encryption. More details about this particular encryption/decryption algorithm are available in the PRINT$_{CIPHER}$ original paper [3].

III. DESIGN METHODOLOGY

FPGAs are usually composed of reconfigurable logic blocks combined with LUT block memories and arithmetic circuits [4], [5], [6]. The basic logic blocks of FPGAs include a 4/6-input LUT and a storage element, such as an flip flop (FF) or register. Most FPGA manufacturers provide users with fast carry logic and high-performance digital signal processor (DSP) logic blocks, shift registers, and so on.

As reconfigurable components are divided into logic elements and storage elements, efficient implementation is a compromise between both combinatorial logic and sequential logic, and resulting performances and power consumptions. These viewpoints lead to different definitions of implementation efficiency. In this article, we selected the following efficiencies: the first two are used in [7], and the third efficiency, called new power regulation efficiency, is our proposal.

1) In terms of area performances, the efficiency of a block cipher is given by the ratio of throughput (Mbps) to area (slices). The larger the ratio, the better the performance efficiency per unit area.
2) In terms of resource performance, the efficiency is easily tested by computing the ratio of the number of registers to the number of LUTs. Ideally, the ratio should be close to one the number of LUTs. Ideally, the ratio should be close to one. When this value is a value close to one, the circuit resource is more efficient.
3) In terms of power regulation performance, the power efficiency of a block cipher is the ratio of power regulation (mW) to area (slices). When this value is smaller, the power consumption per unit area is smaller.

PRINT$_{\text{CIPHER}}$ is designed to allow IC printing and very efficient FPGA implementations. Our proposed architectures are iterative round function architecture or word-serial architecture, divided into element sub-functions from the round function. These are defined to maximize the hardware efficiency or throughput, and minimize the power regulation, with trade-offs among these performance indexes.

Finally, depending on the optimization criteria, different architectures can be employed. Although optimization for maximum processing speed can ideally be achieved by a fully pipeline architecture, we did not select this architecture for applying PRINT$_{\text{CIPHER}}$ to lightweight devices. In the applications requiring minimum area or power regulation, a loop architecture with only one round or division into element functions in round implementations seems to be the best choice. In this study, we tried to maximize the previously defined efficiencies. In the following sections, we present the results of various FPGA implementations of PRINT$_{\text{CIPHER}}$.

I. IMPLEMENTATION

In this section, we investigate the practical implementation of different architectures for PRINT$_{\text{CIPHER}}$. All the architectures proposed in this section allow the choice of default parameters for every implementation. The area and frequency estimations presented are provided after implementation with Xilinx ISE 14.7 on the Xilinx Spartan-3, Spartan-6 and Artix-7 platforms. The timing constraints were applied according to the reference clock.

A. Round Iterative Roll Architectures

For standard lightweight block cipher applications, we propose the round iterative roll architecture with round-based implementation (see Fig. 5). We applied two key scheduling strategies. If a minimum area is required, a hard-wired key scheduling implementation is provided. On the other hand, if maximum throughput is required, the encryption key input is stored in the registers. We studied two key scheduling models, Std. I and Std. II, for the round iterative roll architecture (see Fig. 6), and selected a simple finite state machine description with Verilog-HDL.

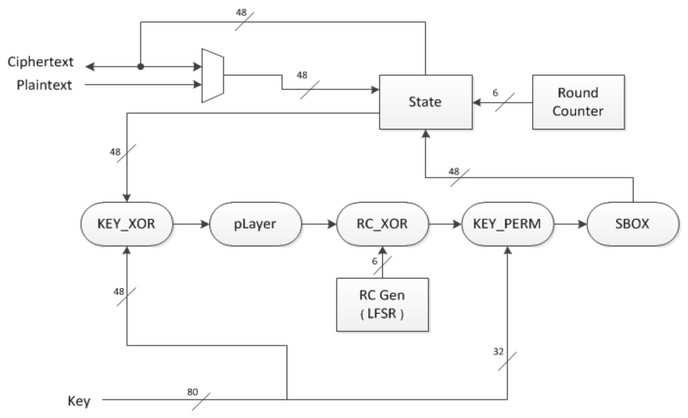


Figure 5 Round iterative roll architecture

Std. I and Std. II

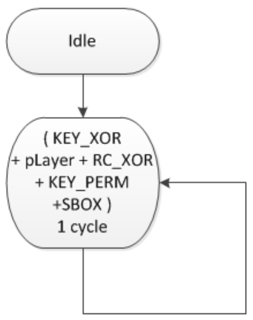


Figure 6.1 cycle Round iterative roll FSM diagram (Std. I and

B. Word Serial Roll Architecture

In the applications requiring minimum area and minimum power regulation, we propose the 3-bit word serial roll architecture (see Fig. 7). To decrease the occupied area and power consumption requirements, we only considered the 3-bit iterative FSM strategy structured by sub-functions in a round. In addition to the efficiency advantages already mentioned, 3-bit word serial structures are especially convenient for loop architectures because they allow the combination of the keyed permutation function with the SBox function in the round. Our proposed FSM is illustrated in Figs. 8 to Fig. 10. For word serial rolled architectures, it is possible to use the FPGA LUT blocks to implement the round SBOX, XOR logic, diffusion layer and round counter. We selected the 3-bit word bus because the keyed permutation and SBOX have a 3-bit input/output signal in PRINT$_{CIPHER}$. We also selected the 48-bit bus width in pLayer to optimize the trade-offs between the processing cycle and logic resources usage on FPGAs. We proposed five types of FSM models. We divided them into several sub-states corresponding to the functions in a round, and we made the sub-states as small as possible in occupied area and power consumption. The proposed-I model describes five simple states divided into each function, and proposed-II merges KEY_XOR, pLayer and RC_XOR functions of the proposed-I model into one “KEY_XOR function. The proposed-III model combines KEY_PERM and SBOX. The proposed-IV model merges pLayer and RC_XOR functions from proposed-III. Finally, the proposed-V model is combined KEY_XOR, pLayer and RC_XOR.

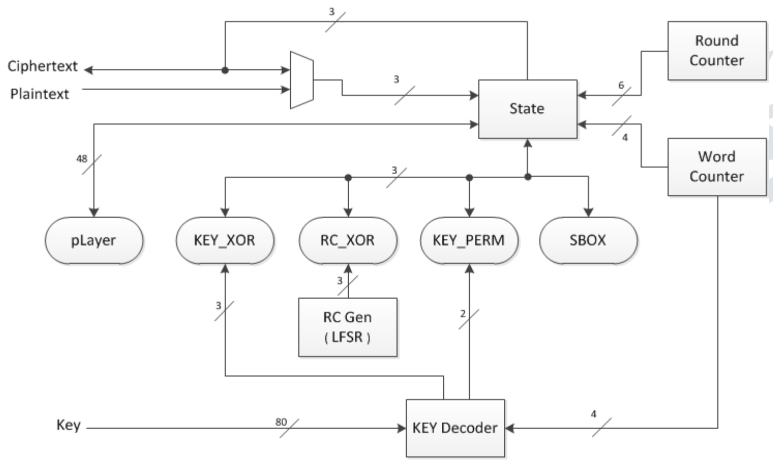


Figure 7 Word serial roll architecture

Std. III and Std. IV

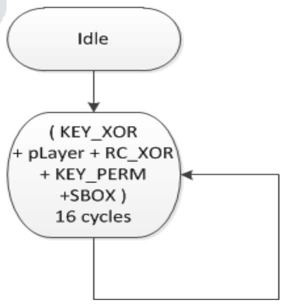


Figure 8 16 cycles word serial roll FSM diagram (Std. III and Std. IV)

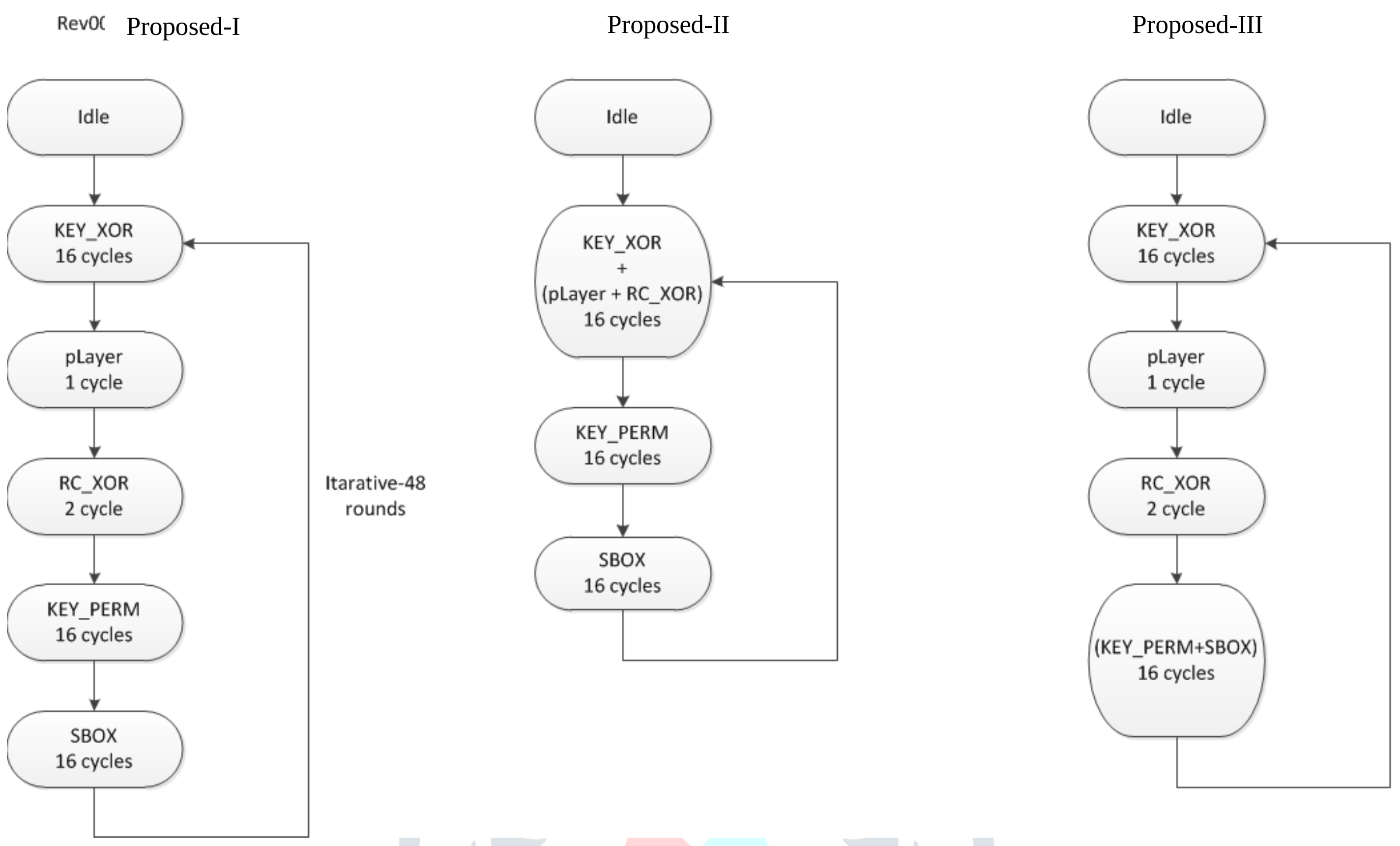


Figure 9 51/48/35 cycles Word serial roll FSM diagram (Proposed-I/II/III)

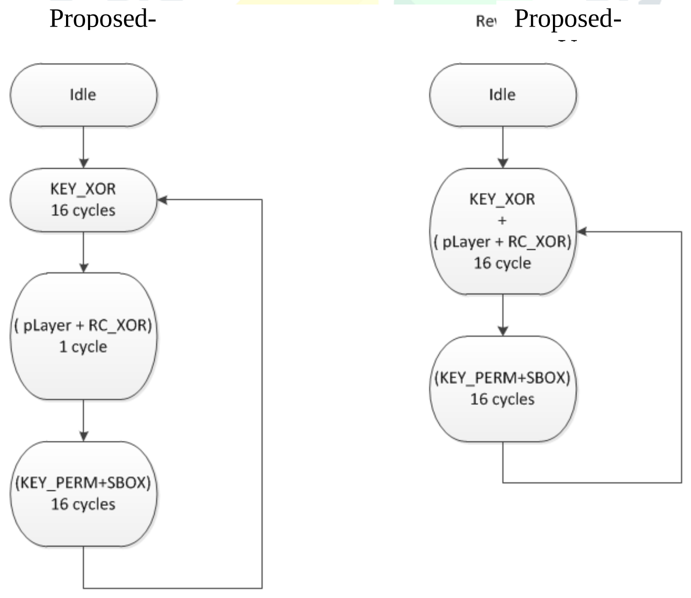

Proposed-
Idle
KEY_XOR 16 cycles
(pLayer + RC_XOR) 1 cycle
(KEY_PERM+SBOX) 16 cycles
Proposed-
Idle
KEY_XOR + (pLayer + RC_XOR) 16 cycle
(KEY_PERM+SBOX) 16 cycles

Figure 10 33/32 cycles Word serial roll FSM diagram (Proposed-IV/V)

II. RESULT AND COMPARISON WITH OTHER LIGHTWEIGHT BLOCK CIPHERS

Our implementation results are presented in Table 2, where the FPGA resource requirements (in LUTs and slices) corresponding to an area, the maximum operation frequency, the maximum throughput and the power regulation are provided. All experimental results were extracted after place and route with ISE Design Suite version 14.7 from Xilinx Inc. on Spartan-3 XC3S700A, Spartan-

6 XC6SLX45, and Artix-7 XC7A100 platforms with speed grades -5/-3/-3, respectively. From Table 2, we note that the area-optimized PRINTCIPHER encryption core (Proposed-V) can achieve a throughput of 3.83 Mbps at the cost of 91 slices on Spartan-3 FPGA platform, whereas the round-rolled iterative encryption core (Std. II) occupies 139 slices and operates at 8.46 Mbps on the same platform.

Table 3 describes the performance comparison of our PRINTCIPHER implementation with existing FPGA implementations of block ciphers LED [13], PRESENT [8], [9], [10], HIGHT [10], AES [11], [12], ICEBERG [7], SEA [14], XTEA [15], SIMON [16] and CLEFFIA [17]. Note that numerous PRESENT, SIMON, and AES hardware architectures have been proposed in the literature, but we only use those implementations on low-cost Spartan and Artix series FPGA devices with speed grades -5 or -3 and above for the purpose of comparison. Moreover, the implementation figures of ICEBERG, SEA and CLEFFIA are only available on Virtex-II or Virtex-4 series FPGAs. We also would like to point out that it is quite difficult to provide a fair comparison among different implementations on FPGAs by taking into account the diversity of FPGA devices and packages, speed grade level, and synthesis and implementation tools. Therefore, we include additional information such as the implementation platform and speed grade level in Table 3.

Our experimental results show that in the low-cost FPGA implementations, PRINTCIPHER can achieve larger throughput with a smaller area requirement, when compared to block ciphers PRESENT, XTEA, SEA and AES. However, the implementation of the ultra-lightweight block cipher SIMON is more efficient than that of PRINTCIPHER. The main reason is the complex internal encryption process in the PRINTCIPHER encryption algorithm. As a result, the encryption unit is more complicated and the delay of the critical path is much longer in the PRINTCIPHER hardware architecture. Although the area of PRINTCIPHER is as compact as that of HIGHT on the Spartan-3 FPGA platform, the throughput of PRINTCIPHER is lower than that of HIGHT. The reason is that PRINTCIPHER has many rounds in the encryption process, and this leads to slower frequency and less throughput than those of HIGHT. Additionally, though an occupied area (slices) of PRINTCIPHER is as compact as the LED area on the Artix-7 FPGA platform, PRINTCIPHER throughput is less than LED throughput, but PRINTCIPHER has a sufficiently small area, lower power regulation and higher throughput on lightweight cipher applications.

Table 2 FPGA implementation results of PRINTCIPHER block cipher with different architectures

Architectures	Devices	Area (Resources)			Speed			Power [mW]	Efficiency		
		Regis-ters	LUTs	Slices	Latency [cycle]	Max Freq. [MHz]	Throughput [Mbps]		Through./Area 1)	No. registers/ No. LUTs 2)	Power reg./Area 3)
Std. I	Spartan-3 XC3S700AN-5	139	262	180	768	123.12	7.70	12.24	0.043	0.531	0.068
Std. II		59	261	139	768	135.28	8.46	7.08	0.061	0.226	0.051
Std. III		135	332	224	48	139.24	139.2	10.67	0.622	0.407	0.048
Std. IV		55	396	210	48	147.73	147.7	11.67	0.703	0.139	0.056
Proposed-I		68	233	121	2448	123.21	2.42	10.85	0.020	0.292	0.090
Proposed-II		65	238	128	2304	111.25	2.32	9.81	0.018	0.273	0.077
Proposed-III		63	222	113	1584	109.87	3.33	5.74	0.029	0.284	0.051
Proposed-IV		69	233	123	1680	114.65	3.28	11.32	0.0267	0.297	0.092
Proposed-V		61	174	91	1536	122.58	3.83	2.82	0.042	0.351	0.031
Std. I	Spartan-6 XC6SLX45 T-3	143	182	90	768	190.91	11.9	8.94	0.133	0.786	0.099
Std. II		62	154	48	768	254.39	15.9	8.13	0.331	0.403	0.169
Std. III		135	270	111	48	165.59	165.6	12.83	1.492	0.500	0.116
Std. IV		55	270	114	48	126.01	126.0	11.63	1.105	0.204	0.102
Proposed-I		68	125	42	2448	129.22	2.53	5.96	0.060	0.544	0.142
Proposed-II		66	107	34	2304	128.35	2.67	6.07	0.079	0.617	0.170
Proposed-III		69	99	35	1584	147.45	4.47	5.00	0.128	0.700	0.143
Proposed-IV		66	135	41	1680	123.05	3.52	5.71	0.086	0.490	0.139
Proposed-V		65	116	43	1536	139.04	4.35	6.23	0.101	0.560	0.145
Std. I	Artix-7 XC7A100 T-3	140	163	77	768	248.26	15.52	8.2	0.202	0.859	0.106
Std. II		60	153	64	768	262.12	16.38	8.13	0.256	0.392	0.127
Std. III		135	280	151	48	220.95	221.0	4.63	1.46	0.482	0.031
Std. IV		55	276	139	48	293.51	293.5	8.2	2.11	0.199	0.059
Proposed-I		61	117	40	2448	176.80	3.47	3.91	0.087	0.521	0.097
Proposed-II		61	100	33	2304	182.28	3.80	3.73	0.12	0.610	0.113
Proposed-III		61	95	29	1584	184.16	5.58	3.09	0.19	0.642	0.107
Proposed-IV		61	114	38	1680	176.18	5.03	3.59	0.13	0.535	0.094
Proposed-V		60	112	38	1536	170.85	5.34	3.75	0.14	0.536	0.099

I. Conclusion

In this article, we presented the first efficient FPGA implementations of PRINT$_{CIPHER}$ and analyzed the feasibility of compact and low power implementations of PRINT$_{CIPHER}$ on FPGAs. We proposed some round-based and word-serial architectures and evaluated their efficiencies. The proposed area-optimized PRINT$_{CIPHER}$ encryption core can encrypt a 48-bit plaintext block with 2.67 Mbps throughput (2,304 clock cycles) on Spartan-6. Moreover, we implemented several possible trade-offs when executing encryption round processes. In particular, we confirmed the efficiency of a proposed word-serialization model in low-cost and low-power aspects. Compared to other lightweight block cipher implementations on FPGAs, XTEA, ICEBERG, SEA, SIMON, LED and AES, PRINT$_{CIPHER}$ can achieve larger throughput with smaller area requirements. Our results show that PRINT$_{CIPHER}$ is a good encryption algorithm for resource-constrained devices. Consequently, PRINT$_{CIPHER}$ can be considered to be an ideal cryptographic primitive for resource-constrained environments. In future work, we will investigate the resistance against side-channel attacks in our implementations and develop countermeasures to maintain resistance and reliability against these attacks.

Table 3 FPGA implementation results of other block ciphers

Block Ciphers	**Devices**	**Area (Resources)**		**Speed**			**Power [mW]**	**Efficiency**		
		LUTs	Slices	Latency [cycle]	Max Freq. [MHz]	Throughput [Mbps]		Through./ Area 1)	No. registers/ No. LUTs 2)	Power reg./Area 3)
ICEBERG [7]	Virtex-II	—	631	—	254	1016	—	1.61	—	—
PRESENT-1 [8]	Spartan-3 XC3S500	—	271	—	—	—	—	—	—	—
PRESENT [9]	Spartan-3 XC3S400-5	—	202	32	254	508	—	2.51	—	—
PRESENT [10]	Spartan-3 XC3S50-5	—	117	256	114.8	28.7	—	0.24	—	—
AES [11]	Spartan-3 XC3S50-5	—	393	534	—	16.86	—	0.04	—	—
AES [12]	Spartan-3 XC3S50-5	—	184	160	45.6	36.50	—	0.20	—	—
HIGHT [10]	Spartan-3 XC3S50-5	—	91	—	163.7	65.48	—	0.72	—	—
LED [13]	Spartan-3 XC3S50-5	148	77	768	119.19	9.93	—	0.13	—	—
	Artix-7 XC7A100T-3	78	37	1120	378	21.6	—	0.58	—	—
SEA [14]	Virtex-II XC2V4000	—	424	—	145	—	—	0.37	—	—
XTEA [15]	Spartan-3 XC3S50-5	—	254	112	62.6	35.77	—	0.14	—	—
SIMON [16]	Spartan-3 XC3S500	72	36	—	136	3.60	—	0.10	—	—
	Spartan-6	40	13	—	—	—	—	—	—	—
CLEFIA [17]	Virtex-4	238	—	—	—	—	—	—	—	—

References

[1] National Institute of Standards and Technology, October 1999, Data Encryption Standard (DES), Federal Information Processing Standards Publications – FIPS 46-3, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[2] National Institute of Standards and Technology, November 2001, Advanced Encryption Standard (AES). Federal Information Processing Standards Publications – FIPS 197, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[3] L. Knudsen, G. Leander, A. Poschmann, M.J.B. Robshaw, "PRINT$_{CIPHER}$: A block cipher for IC-printing", Lecture Notes in Computer Science, Vol. 6225, pp. 16-32, CHES 2010.

[4] Xilinx, Spartan3A FPGA Family: Data Sheet, http://www.xilinx.com/support/documentation/data_sheets/ds529.pdf

[5] Xilinx, Spartan6 FPGA Configuration, http://www.xilinx.com/support/documentation/user_guides/ug380.pdf

[6] Xilinx, 7 Series FPGAs Configuration, http://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf.

[7] F.X. Standaert, G. Piret, G. Rouvroy, J.J. Quisquater, "FPGA implementations of the ICEBERG block cipher", Integration, the VLSI Journal, 40(1):20-27, 2007.

[8] X. Guo, Z. Chen, P. Schaumont, "Energy and performance evaluation of an FPGA-based SoC platform with AES and PRESENT Coprocessors", Lecture Notes in Computer Science, Vol. 5114, pp. 106-115, Embedded Computer Systems:Architectures, Modeling, and Simulation 2008.

[9] A.Y. Poschmann, Lightweight cryptography: Cryptographic engineering for a pervasive world, Ph.D. Thesis, Citeseer, 2009.

[10] P. Yalla, J.P. Kaps, Lightweight Cryptography for FPGAs, Reconfigurable Computing and FPGAs, 2009, ReConFig'09. International Conference on, pp. 225-230. IEEE, 2009.

[11] J.P. Kaps. B. Sunar, “Energy comparison of AES and SHA-1 for ubiquitous computing”, EUC Workshops, pp. 372-381. Springer, 2006~~.~~

[12] J. Chu, M. Benaissa, “Low area memory-free FPGA implementation of the AES algorithm”, 22nd FPL International Conference, pp. 623-626. IEEE, 2012.

[13] N. Nalla Anandakumar, T. Peyrin, A. Poschmann, “A very compact FPGA implementation of LED and PHOTON”, Lecture Notes in Computer Science, Vol. 8885, INDOCRYPT 2014, pp. 304-321, 2014.

[14] F. Mace, F.X. Standaert, J.J. Quisquater, “FPGA implementation(s) of a scalable encryption algorithm”, Very Large Scale Integration (VLSI) Systems, IEEE Transactions, 16(2):212-216, 2007.

[15] J.P. Kaps, “Chai-tea, Cryptographic hardware implementations of xTEA”, Progress in Cryptology, INDOCRYPT 2008, pp. 363-375, Springer, 2008.

[16] A. Aysu, E. Gulcan, P. Schaumont, “SIMON Says, Break the area records for symmetric key block ciphers on FPGAs”, IACR Cryptology ePrint Archive, 2014, Available at:. http://eprint.iacr.org/2014/237.

[17] R. Chaves, “Compact CLEFIA implementation on FPGAs”, Springer Embedded System Design with FPGAs, pp. 225-243, 2012.