© 2017 IJEDR | Volume 5, Issue 1 | ISSN: 2321-9939

FPGA Implementation and Evaluation of lightweight block cipher - BORON

[1]Tadashi Okabe

[1]Information Technology Group,
Tokyo Metropolitan Industrial Technology Research Institute, Tokyo, Japan

***Abstract* –This article presents the first hardware implementations of the BORON lightweight block cipher on general purpose field programmable gate arrays (FPGAs) using hardware description language. Herein, we evaluate several architectures for compatibility with the lightweight block cipher algorithm and report on both the first FPGA implementation results along with a comparison with standard lightweight block cipher algorithms. Of particular note, we confirm that BORON provides an excellent hardware performance algorithm for a lightweight block cipher in terms of high performance per unit area and power consumption.**

***Index Terms* — FPGA, BORON, Lightweight block cipher, Hardware implementation, CPS, Embedded system.**

I. INTRODUCTION

In our current era, cyber-physical systems (CPSs) are advancing worldwide in a variety of industries. CPSs are high level mechanisms that can collect huge amounts of data from numerous physical space sensor networks, analyze the collected data in cyberspace, and ultimately apply the analyzed data to solving social problems and/or activating industries. In such systems, computer and sensor networks are often tightly integrated with the Internet. In a CPS society, both general purpose computers on high performance information processing networks and lightweight portable electronic devices such as sensor nodes need higher levels of security. To this end, standard block ciphers such as the triple data encryption standard (DES) [1] or advanced encryption standard (AES) [2] are widely available, but these ciphers cannot be applied to small, lightweight devices that are resource-constrained or have integrated circuits (ICs) with low processing power. Thus, 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.

BORON [3], which is a lightweight block cipher designed for low power consumption and efficient software and hardware implementations in resource-constrained compact devices, is currently the most common currently proposed lightweight block cipher. It has a substitution-permutation network (SPN) structure with a substitution layer that consists of 16 4-bit substitution boxes (SBoxes) and a permutation layer built from block shuffle, round permutation, and exclusive-or operation layers. BORON utilizes a 64-bit block size for plaintext or ciphertext, and 80- or 128-bit key sizes for the cipher key. The number of rounds is 24. BORON has a unique design that permits shift operation, round permutation, and exclusive-or operations in the round process. These unique elements provide an advantage to allow it to generate a large number of active SBoxes in fewer rounds against linear and differential cryptanalysis. In practice, BORON provides a number of potential applications for resource-constrained compact crypt devices and high throughput data encryption fields.

This article begins by presenting several FPGA implementations of BORON and then compares their performance 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 and third are based on 32-bit and 16-bit word-serial architectures where they run numerous encryption process clock cycles. Our main contribution is the first round base roll architecture that is based on the 64-bit architecture presented in BORON [3], and the second and third are the 16- or 32-bit word-serial architectures that easily and consistently perform BORON computations based on finite state machine (FSM) models. The first architecture (one round per clock cycle) is actually better than the second and third architectures (16- or 32-bit word-serial), because it uses less area and power while providing better throughput. To verify this, we investigated various contexts installed on recent Xilinx FPGAs that were mounted on Spartan-3, Spartan-6 and Artix-7 platforms.

The remainder of this article is organized the following. In Section II, we briefly present the specifications of the BORON encryption algorithms. Section III describes round-base and word serial iterative roll model architectures. In Section IV we describe the FPGA implementation results and present comparison results with other standard lightweight block cipher algorithms. Finally, we discuss the conclusions of this study and our future work in Section V.

II. LIGHT-WIGHT BLOCK CIPHER ALGORITHM

In this section, we describe the BORON specifications and data encryption algorithms.

A Characteristic and Parameters

BORON is a lightweight block cipher substitution-permutation network (SPN) structure equipped with novel encryption function blocks that provide one of the most robust permutation layers and a strong non-linear layer. This structure enhances security against standard cryptographic attacks such as linear and differential cryptanalysis. BORON, which is an iterative block cipher based on the repetition of total 24 rounds, is based on PRESENT cipher key scheduling and operates a 64-bit plaintext/ciphertext and a 80- or 128-bit cipher key. An overview of the encryption algorithm is provided in Fig. 1. More detailed specifications will be given in the following subsections.

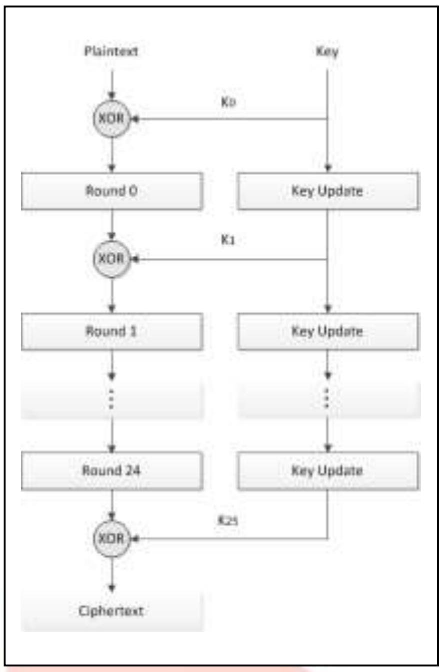

Figure 1: BORON round function

B Round Function

The round function, which includes the key addition, nonlinear permutation, and linear layers, is described in Fig. 2. The key addition layer consists of bitwise key additions (denoted as the Add_round_key in Fig. 2). The nonlinear layer is an S-Box layer (hereafter denoted as the S_Box_Layer) built from the parallel application of 4 × 4 SBoxes to the intermediate state. The linear layer is built from block shuffle (hereafter denoted as the Block_Shuffle), round permutation (denoted as the Round_Permutation), and exclusive-or operation (hereafter denoted as the XOR_Operation).

Each round consists of the following five functions:

1. Key XOR with each round input data
2. Nonlinear transformation with SBOX
3. Block shuffle
4. Round Permutation
5. XOR operation

Each 4-bit word connection in the round function is depicted in Fig. 3. Subsections B.1 to B.5 describe each function layer in detail.

B.1. Add_Round_key Layer

This function is a simple bitwise XOR (exclusive-or) layer between the current state and 64-bit round key, rk[i] at the i-th round. The suffix i ranges from 0 to 25. Detailed key scheduling is given in Subsection C.

B.2. S_Box_Layer

Like most other lightweight block ciphers, BORON uses an input 4-bit to output 4-bit SBox to create a 16 SBox layer. For each 4-bit word, an identical S-Box is used. The BORON S-Box is described in Table 1 and its functions are presented in the following equations:

$$\begin{aligned}
y3 &= (!x3\&!x2\&!x0)|(!x3\&x1\&!x0)|(!x3\&x2\&x0)|(x3\&!x1\&!x0)|(x3\&!x2\&x1\&x0)\\
y2 &= (!x3\&!x2\&!x1)|(!x2\&!x1\&!x0)|(!x3\&x2\&!x0)|(x3\&x1\&x0)|(x3\&x2\&x0)\\
y1 &= (!x3\&!x2\&x0)|(!x3\&!x1\&!x0)|(x2\&x1\&x0)|(x3\&!x2\&x0)|(x3\&x2\&x1)\\
y0 &= (!x3\&!x2\&x1)|(!x2\&x1\&x0)|(!x3\&x2\&!x1)|(x2\&!x1\&x0)|(x3\&!x2\&!x1\&!x0)|(x3\&x2\&x1\&!x0)
\end{aligned}\tag{1}$$

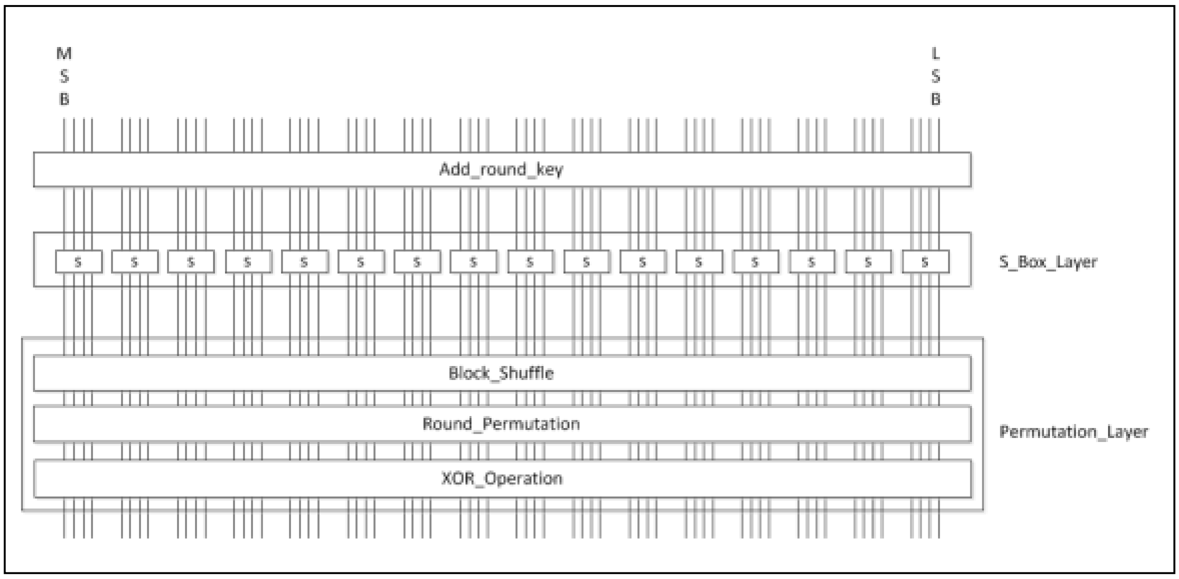

Figure 2: BORON round function

Table 1: S-box table

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
S(x)	E	4	B	1	7	9	C	A	D	2	0	F	8	5	3

B.3. Block_Shuffle

The Block_Shuffle is specified in the following Table 2, which also shows the 8 bit left cyclic shift operation performed for every 16-bit word. The basic Block_Shuffle rule is depicted in Fig. 3.

Table 2: Block_Shuffle: 8 bit left cyclic shift in 16-bit word

j	0	1	2	3
B[j]	2	3	0	1

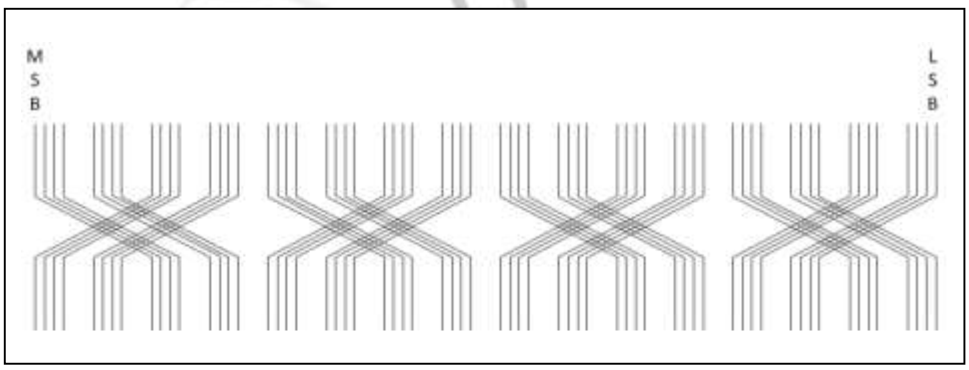

Figure 3: Block_Shuffle

B.4. Round_Permutation

Round permutation is the left circular shift operation performed on each 16-bit word, as shown in Table 3. This table also describes the correspondence between each of the 16-bit words in the 64-bit intermediate data with left circular shift values. A simple depiction of the round permutation law is shown in Fig. 4.

Table 3: Round_Permutation: left circular shift values

j	0	1	2	3
r[j]	1	4	7	9

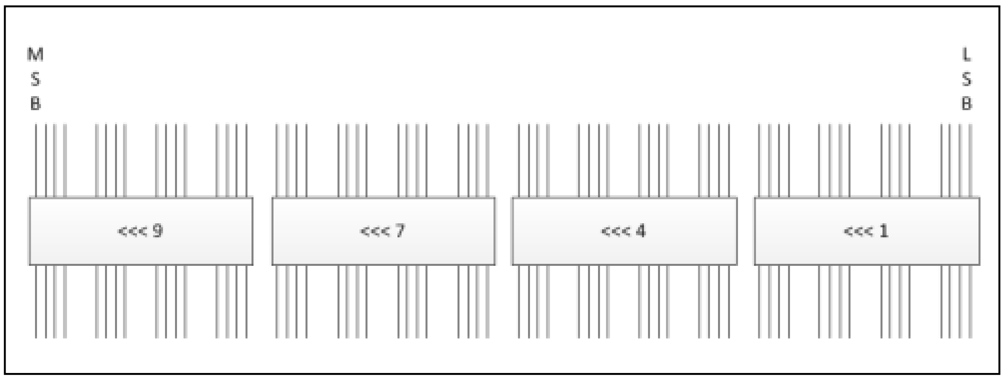

Figure 4: Round_Permutation

B.5. XOR_Operation Layer

This function provides a simple bitwise XOR (exclusive-or) layer between 16-bit words and generates 16-bit 4 word outputs. The layer also generates a 64-bit (16-bit 4-word) output, as described the following equation:

$$W'3 \| W'2 \| W'1 \| W'0 = (W3 \wedge W2 \wedge W0) \| (W2 \wedge W0) \| (W3 \wedge W1) \| (W3 \wedge W1 \wedge W0) \qquad (2)$$

where " ' " presents outputs, "W" means 16-bit each words and "^" refers to exclusive-or operation.

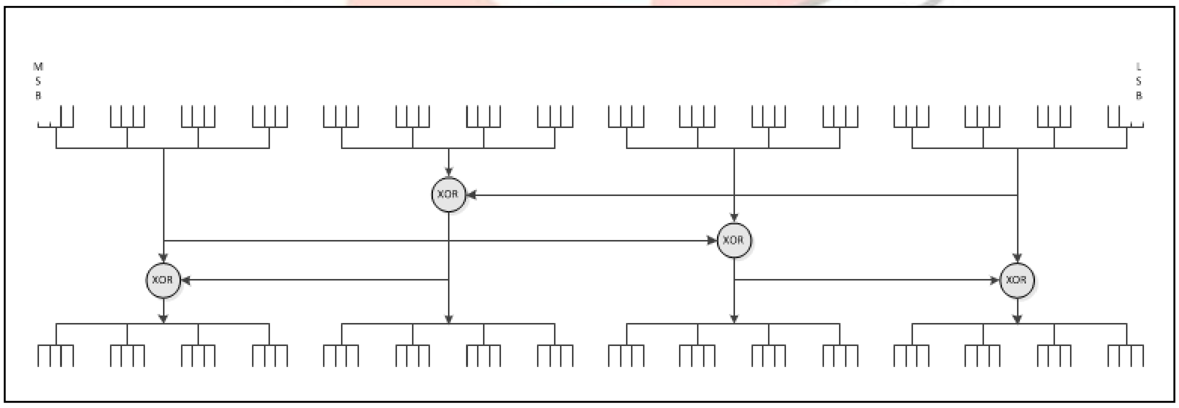

Figure 5: XOR_Operation

C Key Schedule (Round Key and Permutation Key Generation)

BORON key scheduling is similar to the existing PRESENT lightweight block cipher. In each round, the least significant 64 bits of the 80- or 128-bit updated key are the round key that performs exclusive-or with round data in the Add_round_key function. Thus, round key generation is very simple in the BORON encryption algorithm.

In this article, we present only the 80-bit key scheduling rule and the key register stored user defined 80-bit key. BORON uses the least significant 64 bits in the key register at each key scheduling as a round key.

KEY_Register = K79K78...K2K1K0
round_key = K63K62...K2K1K0

The key update of BORON is the following manner.

1. KEY_Register <<<13
2. [K3K2K1K0] <- S([K3K2K1K0])
3. [K63K62K61K60K59] <- [K63K62K61K60K59] ^ RCi

where <<<n means left cyclic shift by n bit operation and RCi means the round counter value at i-th round.

D Encryption (Decryption) Process

The complete cipher consists of the iteration from 0 to 24 rounds discussed in Subsections B.1 to B.5. Due to the round key structures of BORON, the key scheduling function executes the encryption (decryption) process in the manner described in Subsection C in. In pseudo code, we have:

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

BORON-encrypt(plaintext, key)
{
        state  = plaintext;
        Key_Schedule(key);
        for (i=0; i<=24; i++) Round(state, round_key[i]);
        ciphertext = Add_round_key(state, round_key[25]);
}

Key_Schedule(key)
{
        KEY_Register = key;
        for (i=0; i<=25; i++)
        {
                KEY_Register<<<13;
                [K3K2K1K0] = S_Box([K3K2K1K0])
                [K63K62K61K60K59] = [K63K62K61K60K59] ^ i
                round_key[i] =  KEY_Register[K63K62....K1K0];
        }
}

Round(state, rk[i])
{
        Add_round_key_Layer(state, rk[i]);
        S_Box_Layer(state);
        Block_shuffle(state);
        Round_Permutation(state);
        XOR_Operation(state);
}
```

where key is the 80-bit input cipher key, state is the 64-bit intermediate round data block and cipher text data after the last round, KEY_register is the stored 80-bit key, and round_key[i] is the 64-bit round key at an i-th round process. The BORON decryption process is simply the reverse of the encryption. Additional details about this particular encryption/decryption algorithm are available in the original BORON paper [3].

III. Architecture

An FPGA is an integrated circuit that users can configure in multiple ways. Since the reconfigurable components of an FPGA are divided into logic elements and storage elements, efficient implementation is a compromise between both combinatorial and sequential logic, and the resulting performance and power consumption. The evaluation viewpoint leads to different definitions of implementation efficiency. In this article, we selected following efficiencies: the first two are used in [4], and the third is proposed in [12].

A) In terms of area performances, efficiency is given by the ratio of throughput [Mbps] to area [slices].
B) In terms of resource performance, efficiency is the ratio of the number of registers to the number of lookup tables (LUTs).
C) In terms of power regulation performance, efficiency is the ratio of power regulation [mW] to area [slices].

The larger the ratio of A, the better the performance efficiency per unit area. Ideally, the ratio of B should be close to the number of LUTs. Ideally, the ratio should be close to one in order to make the resources more efficient. When the efficiency of C is smaller, the power consumption per unit area is smaller.

Because BORON is a lightweight and low power block cipher, it performs well on hardware and software implementations. Although the original BORON paper [3] showed a hardware implementation result on an application specific integrated circuit (ASIC) platform with a UMCL180 standard cell, it resulted in a lower estimation because, unlike multiplexers, there is no evaluation control-logic circuit.

In the following paragraphs, we will show all BORON hardware implementation results using a control-logic circuit. Our proposed architectures are a 64-bit iterative round function architecture or 16- or 32-bit word-serial architectures that was divided into element sub-functions from the round function. These definitions were selected to maximize the hardware efficiency or throughput, and to minimize the power regulation by means of trade-offs among these performance indexes.

Depending on the optimization criteria, different architectures can be employed. Although optimization for maximum processing speed can ideally be achieved by a pipeline architecture, we did not select this architecture for applying BORON to lightweight devices because a loop architecture with only one round or division into element functions in round implementations seems to be the best choice for lightweight device applications requiring minimum area or power regulation. Additionally, we did not select 4-bit or 8-bit word serial architectures because the round function of BORON consists of 16-bit data blocks in linear layers like the Block_Shuffle, Round_permutation, and XOR_Operation_Layer. In this study, we tried to maximize the previously efficiencies from A) to C). Our proposed BORON architectures are presented below.

A. Round Iterative Roll Architectures

For standard lightweight block cipher applications, we propose a 64-bit round iterative roll architecture with round-based implementation (See Fig. 6). This round-based iterative architecture, which computes one round per clock cycle, is modeled as Proposal-I in the following sections.

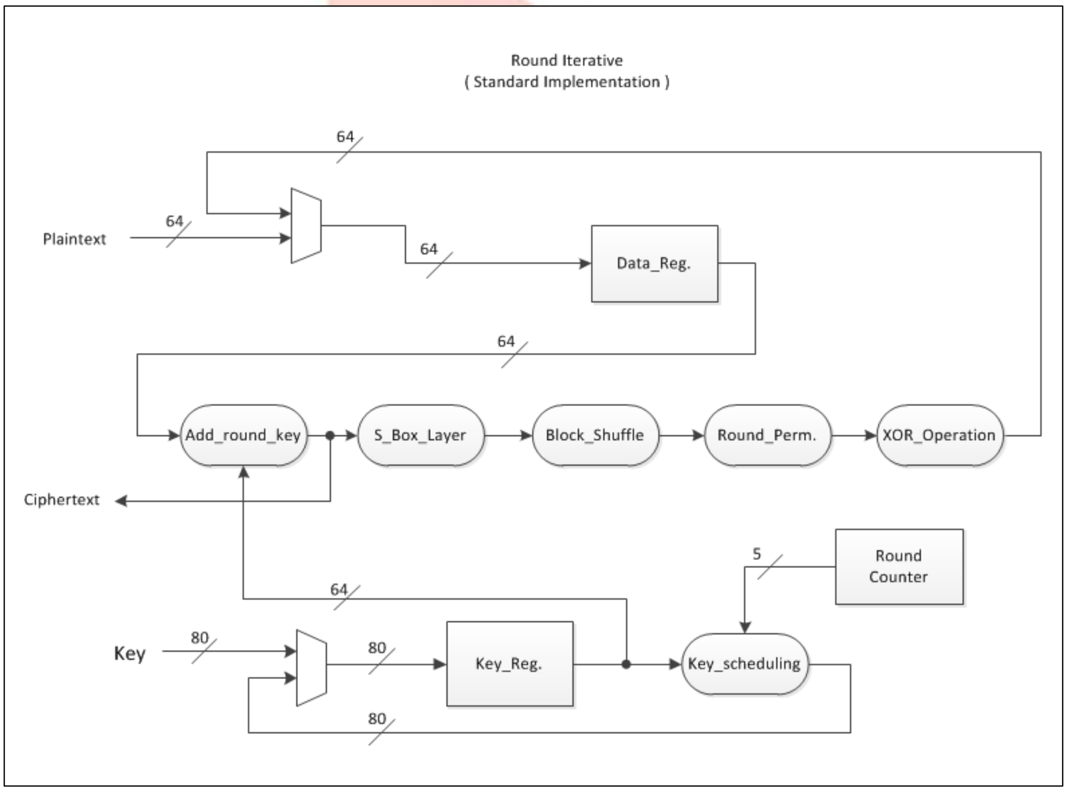

Figure 6: Round iterative roll architecture

B. Word Serial Roll Architecture

For times when space is at a premium, our word-serial roll implementation is proposed. Since this study chose 16- or 32-bit word for word serialization, we propose a 16- or 32-bit word serial roll BORON encryption architecture herein. First of all, we follow the next constraints in the word serial design strategy.

B.I). Merging S_Box_Layer and Block_Shuffle
B.II). Using finite state machine description for word serialization
B.III). Spending 1 clock cycle at Key_scheduling process

In constraint B.I), if the S_Box_Layer and Block_Shuffle are divided, extra cycles are spent for each part. This constraint is simple and effective when compactness and total latency are required in an encryption process. In constraint B.II), a finite state machine (FSM) description is applied to make it easier when designing a complex circuit. On the other hand, if the Key_scheduling is used as the serial process at each 16- or 32-bit word, we spend the same number of cycles as is used in the round process, so constraint B.III) is selected because of the higher throughput it provides in the total encryption process. Following these constraints, we propose the 16- or 32-bit word serial architectures shown in Fig. 7. The FSM function block controls total serial processes in each function block. This 16- or 32-bit word serial architecture model is denoted as Proposal-II in the following Sections.

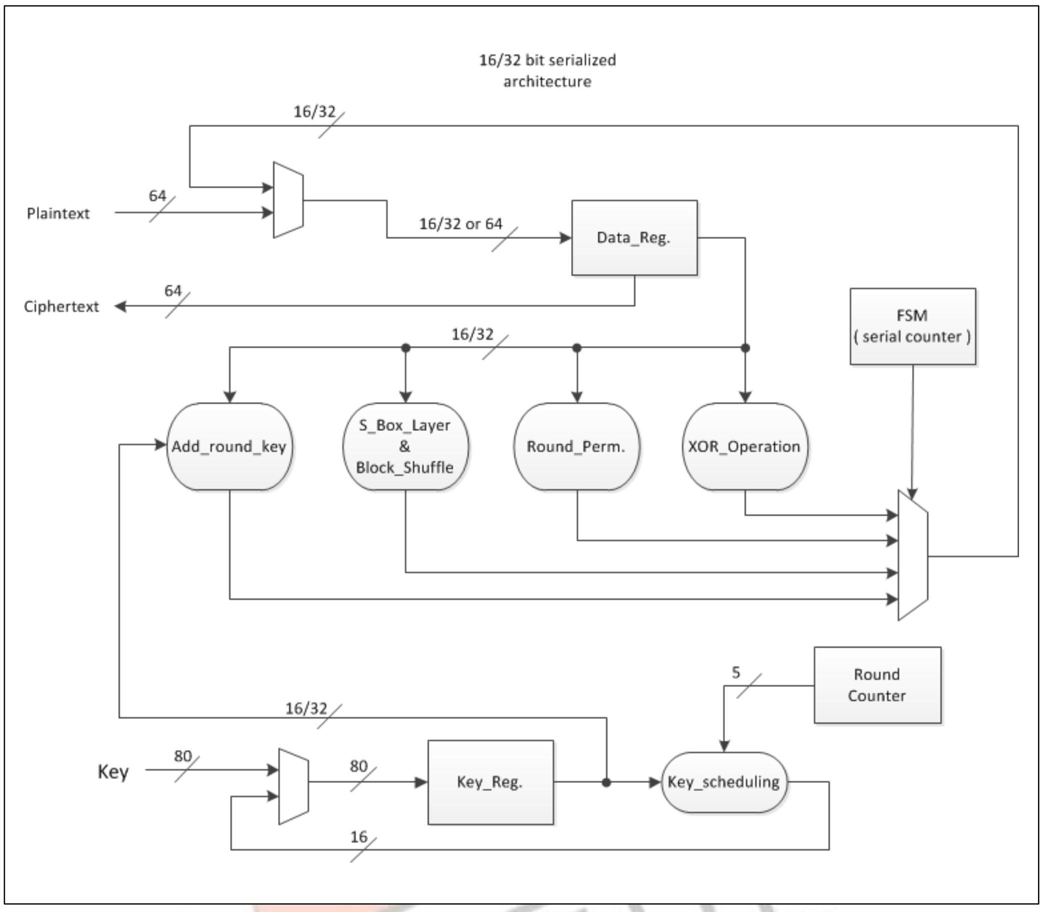

Figure 7: 16- or 32-bit word serial architecture

C. Optimization

Here, we will describe our attempt to further optimize the XOR_Operation function block. Although in Section B, we explained that 16- or 32-bit word serial roll architectures have 16- or 32-bit inputs in XOR_Operation, 64-bit input was chosen for our increased optimization attempt because the XOR_Operation function block processing would be more complex than the 16- or 32-bit word operation utilized by Equation (2). In the following sections, this optimization architecture model is denoted as Proposal III. In addition to this optimization, we tested a more optimal model in which two exclusive-or function blocks were merged between the Add_round_key and XOR_Operation. However the results of this extra optimization attempt were unsatisfactory in comparison with the results of the other proposed architectures mentioned above, so we did not include them in this report.

IV. Result and Evaluation

In this section, we report on the observed practical implementations for each of the above-mentioned BORON architectures. All of the architectures and optimal models proposed in Section III allow the selection of a default parameter for each implementation. For the S_Box_Layer, Table 1 is used to show map circuits into LUTs instead of Equation (1). The presented estimations of area, frequency, and power regulation are provided after logic synthesis, place, and route compilation by Xilinx ISE 14.2 on the Xilinx Spartan-3, Spartan-6 and Artix-7 platforms. The details of these target devices are xc3s700an-5ft484 Spartan-3, xc6slx45t-3fgg484 Spartan-6, and xc7a100t-3fgg484 for the Artix-7 platform, with speed grades of -5, -3, and -3, respectively. Simple timing constraints were applied to the 100 MHz reference clock utilized in these experiments. All designs were verified with static timing analysis (STA) and pre- or post-layout simulation. In one verification, we presented a post-layout simulation waveform for the 64-bit round base roll architecture model (See Fig. 8).

Our results are presented in Table 4, where the FPGA resource requirements (in registers, LUTs and slices) corresponding to area, maximum operation frequency, maximum throughput, and power regulation are also provided. Looking at Table 4, we note that the largest area optimized by the BORON encryption core could achieve a throughput of 10.43 Mbps at the cost of 75 slices on the Spartan-6 FPGA platform in a 16-bit word Proposal-II architecture, whereas the highest throughput optimization occupied 92 slices and operated at 756.73 Mbps on the target Artix-7 FPGA platform in the Proposal-I architecture. Furthermore, the best power reduction optimization resulted in 4.83 mW power regulation at 13.37 Mbps throughput in a 16-bit word Proposal-III architecture. In relation to throughput, round-base architecture was found to have high value on every platform. From these results, it can be said that BORON architectures are very efficient for both high performance and lightweight devices.

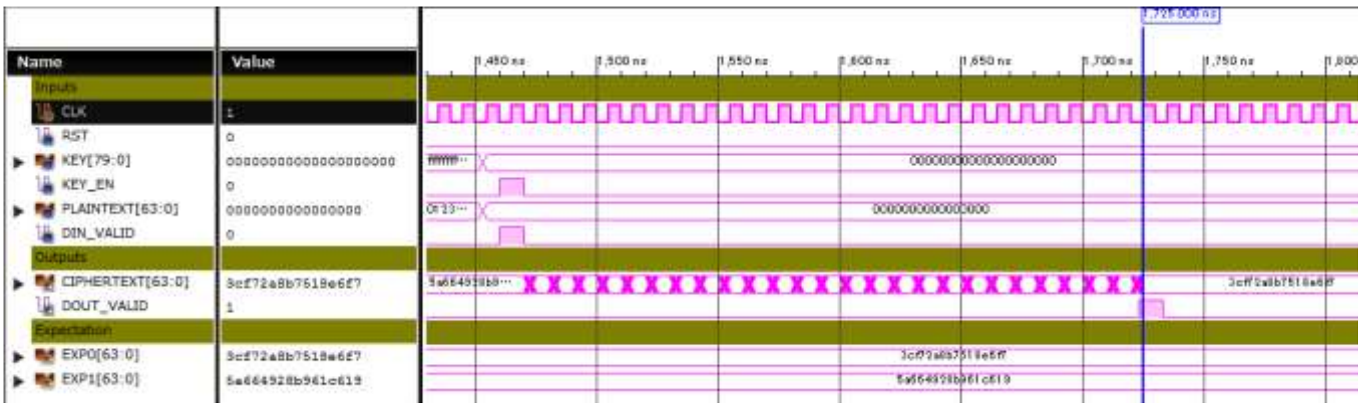
Name
Value
Inputs
CLK 1
RST 0
KEY[79:0] 00000000000000000000
KEY_EN 0
PLAINTEXT[63:0] 0000000000000000
DIN_VALID 0
Outputs
CIPHERTEXT[63:0] 3cf72a8b7518e6f7
DOUT_VALID 1
Expectation
EXP0[63:0] 3cf72a8b7518e6f7
EXP1[63:0] 5a664928b961c619
1,450 ns
1,500 ns
1,550 ns
1,600 ns
1,650 ns
1,700 ns
1,750 ns
1,800

Figure 8: Post-layout simulation waveform - 64-bit round iterative roll architecture

Table 5 enumerates the existing FPGA implementation results of other block ciphers - LED [5], PRESENT [6], [7], [8], HIGHT [8], ICEBERG [4], SEA [9], XTEA [10], Simon [11] as well as PRINT$_{\text{CIPHER}}$[12] for use in performance comparisons with our BORON implementation results. Note that since numerous PRESENT and Simon hardware architectures have been proposed in the literature, we limited our comparison to those implementations using low-cost Spartan or Artix series FPGA devices with speed grade -5 or -3 and above. Furthermore, the implementation figures for ICEBERG and SEA are only available on Virtex-II or Virtex-4 series FPGAs. We would also like to point out that since it is quite difficult to provide a fair comparison among different implementations on FPGAs due to the diversity of such devices, packages, speed grade levels, as well as synthesis and implementation tools, additional information such as implementation platform and speed grade level is included in Table 5.

Our experimental results show that, in the context of low-cost FPGA implementation, BORON can achieve larger throughput with a smaller area requirement when compared to all tested block ciphers except for ICEBERG. However, the implementation of the ultra-lightweight block cipher Simon is more efficient than that of BORON, primarily due to the complex internal encryption process used in the BORON encryption algorithm. This means that the encryption unit is more complicated and the critical path delay is much longer in the BORON hardware architecture than in the Simon core. Also an area of BORON on the Spartan-3 FPGA platform in Proposal-I architecture is as compact as one of PRINT$_{\text{CIPHER}}$ in round-base model, PRESENT [7] and XTEA at Spartan3 FPGA platform. In the point of throughput, a throughput of BORON on the Spartan-3 FPGA platform in Proposal-I architecture is more than other block cipher results except for PRESENT [7] on the other Spartan-3 FPGA platforms. This is because almost all of the other lightweight block ciphers have more rounds in the encryption process, which results in lower frequency and less throughput than BORON. Additionally, even though BORON’s occupied areas (slices) are less compact than those of LED and PRINT$_{\text{CIPHER}}$ on the Artix-7 FPGA platform, BORON throughput is somewhat higher. Furthermore, in terms of the abovementioned efficiencies A) to C), BORON shows better values than other block ciphers in several architectures. Overall, BORON is sufficiently compact, consumes less power, and provides higher throughput on lightweight cipher applications.

As for the first consideration, the serialization of 16- or 32-bit word does not appear to be efficient based on several performance results obtained for LUTs, registers, and slices. On the other hand, the results of general lightweight block ciphers are better than the BORON results, and benefit more from word serialization methodology. This indicates that BORON might be not be suitable for word serialization, and that 64-bit round base roll architecture is the most suitable BORON cipher algorithm because its encryption algorithm has 16-bit blocks in Block_Shuffle, Round_Permutation, and XOR_Operation layers. As for the second consideration, we found that on a Spartan-3 FPGA platform equipped with 4-LUT[1], the 64-bit round base roll architecture is as good as, if not better than, the 16- or 32-bit word serial architectures in all areas except for power consumption. On the other platforms – Spartan-6 and Artix-7 FPGAs equipped with 6-LUT[1], we found that 16- or 32-bit word serial architectures are more compact than the 64-bit round base architecture in terms of occupied LUTs and slices. Naturally, the 6-LUT FPGA platform is more suitable for compact implementation than the 4-LUT platform.

[1]LUTs are 4-bit or 6-bit input function generators and constitute the basic building block of most recent reconfigurable devices like FPGAs [13,14,15].

V. Conclusion and Future Work

In this article, we presented the first FPGA implementations of the BORON lightweight block cipher and analyzed the feasibility of several iterative BORON roll architectures on FPGAs. Additionally, we proposed round-based and word-serial architectures and one optimization method, and evaluated their efficiencies. The proposed round-based roll architecture for the BORON encryption core can encrypt a 64-bit plain text block with 450.78 Mbps throughput (25 clock cycles) on Spartan-6, while the proposed 16-bit word-serial architecture BORON encryption core recorded 10.43 Mbps of throughput with 75 slices on the same FPGA platform. Moreover, we implemented several possible trade-offs when executing encryption round processes. In particular, we confirmed the efficiency of our proposed word-serialization model in terms of low-cost and low-power aspects. When compared to other lightweight block cipher implementations on FPGAs such as XTEA, ICEBERG, SEA, SIMON, LED and PRINT$_{\text{CIPHER}}$, we found that BORON achieved a higher throughput with modestly smaller area requirements.

Our results show that BORON is a good encryption algorithm, not only for resource-constrained devices, but also for high-throughput regions in CPS areas. Consequently, BORON can be considered to be an ideal cryptographic primitive for resource-constrained environments and modest to high-throughput regions in CPS society. In our future work, we will investigate the resistance to side-channel attacks in our implementations and develop countermeasures to maintain resistance and reliability against such attacks.

Table 4: FPGA implementation results for BORON block cipher with different architectures

Architecture	**Device**	**Area (Resources)**			**Speed**			**Power** [mW]	**Efficiency**		
		Registers	LUTs	Slices	Latency [cycle]	Max Freq. [MHz]	Throughput [Mbps]		Effi.A)	Effi.B)	Effi.C)
Proposal I	Spartan-3 XC3S700AN-5	151	377	198	25	152.742	391.02	22.97	1.975	0.401	0.116
Proposal II (32-bit word)		159	583	304	150	113.302	24.17	17.36	0.080	0.273	0.057
Proposal III (32-bit word)		161	559	287	175	131.562	24.06	22.85	0.084	0.288	0.080
Proposal II (16-bit word)		159	570	293	250	134.735	8.62	17.36	0.029	0.279	0.059
Proposal III (16-bit word)		161	559	287	325	122.234	6.02	19.36	0.022	0.308	0.070
Proposal I	Spartan-6 XC6SLX45T-3	151	269	92	25	176.087	450.78	17.00	4.900	0.561	0.185
Proposal II (32-bit word)		165	295	99	150	161.212	34.39	21.29	0.347	0.559	0.215
Proposal III (32-bit word)		173	324	105	175	193.949	35.46	9.73	0.338	0.534	0.093
Proposal II (16-bit word)		159	242	75	250	162.920	10.43	8.62	0.139	0.657	0.115
Proposal III (16-bit word)		166	297	85	325	147.145	7.24	8.54	0.085	0.559	0.101
Proposal I	Artix-7 XC7A100T-3	151	266	92	25	295.596	756.73	11.03	8.225	0.568	0.120
Proposal II (32-bit word)		153	285	99	150	234.632	50.05	5.87	0.506	0.537	0.059
Proposal III (32-bit word)		154	277	104	175	250.941	45.89	4.96	0.441	0.556	0.048
Proposal II (16-bit word)		154	253	92	250	260.078	16.64	5.28	0.181	0.609	0.057
Proposal III (16-bit word)		154	277	104	325	271.518	13.37	4.83	0.079	0.388	0.028

Table 5: FPGA implementation result of other block ciphers

Block Ciphers	**devices**	**Area (Resources)**			**Speed**			**Power** [mW]	**Efficiency**		
		Registers	LUTs	slices	Latency [cycle]	Max Freq. [MHz]	Throughput [Mbps]		Effi.A)	Effi.B)	Effi.C)
ICEBERG [4]	Virtex-II	—	—	631	—	254	1016	—	1.61	—	—
PRESENT[6]	Spartan3 XC3S500	—	—	271	—	—	—	—	—	—	—
PRESENT [7]	Spartan3 XC3S400-5	—	—	202	32	254	508	—	2.51	—	—
PRESENT [8]	Spartan3 XC3S50-5	—	—	117	256	114.8	28.7	—	0.24	—	—
HIGHT [8]	Spartan3 XC3S50-5	—	—	91	—	163.7	65.48	—	0.72	—	—
LED [5]	Spartan3 XC3S50-5	—	148	77	768	119.19	9.93	—	0.13	—	—
	Artix7 XC7A100T-3	—	78	37	1120	378	21.6	—	0.58	—	—
SEA [9]	Virtex-II XC2V4000	—	—	424	—	145	—	—	0.37	—	—
XTEA [10]	Spartan3 XC3S50-5	—	—	254	112	62.6	35.77	—	0.14	—	—
Simon [11]	Spartan3 XC3S500	—	72	36	—	136	3.60	—	0.10	—	—
	Spartan6	—	40	13	—	—	—	—	—	—	—
PRINT$_{\text{CIPHER}}$ [12]	Spartan3 XC3S700AN-5	55	396	210	48	147.73	147.7	11.67	0.703	0.139	0.056
		61	174	91	1536	122.58	3.83	2.82	0.042	0.351	0.031
	Spartan6 XC6SLX45T-3	62	154	48	768	254.39	15.9	8.13	0.331	0.403	0.169
		66	107	34	2304	128.35	2.67	6.07	0.079	0.617	0.170
	Artix7 XC7A100T-3	55	276	139	48	293.51	293.5	8.2	2.11	0.199	0.059
		61	95	29	1584	184.16	5.58	3.09	0.19	0.642	0.107

REFERENCES

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

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

[3] G. BANSOD, N. PISHAROTY, A. PATIL3, "BORON: an ultra lightweight and low power encryption design for pervasive computing", Frontiers of Information Technology & Electronic Engineering, Accepted manuscript available online, Available at: http://www.zju.edu.cn/jzus/iparticle.php?doi=10.1631/FITEE.1500415

[4] 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.

[5] N. Nalla Anandakumar, T. Peyrin, A. Poschmann, "A Very Compact FPGA Implementation of LED and PHOTON", Lecture Notes in Computer Science, Vol 8885, pp.304-321, INDOCRYPT 2014.

[6] 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.

[7] A.Y. Poschmann. LIGHTWEIGHT CRYPTOGRAPHY: Cryptographic Engineering for a Pervasive World. In Ph.D. Thesis. 2009.

[8] P. Yalla, J.P. Kaps. Lightweight Cryptography for FPGAs. In Recon_gurable Computing and FPGAs, 2009. ReConFig'09. International Conference on, pages 225-230. IEEE, 2009.

[9] F. Mace, F.X. Standaert, J.J. Quisquater. "FPGA Implementation(s) of a Scalable Encryption Algorithm". Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 16(2):212-216, 2007.

[10] J.P. Kaps. "Chai-Tea, Cryptographic Hardware Implementations of xTEA". In Progress in Cryptology INDOCRYPT 2008, pages 363-375. Springer, 2008.

[11] 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.

[12] T. Okabe, "Efficient FPGA Implementations of PRINTCIPHER", International Journal of Emerging Technologies and Innovative Research, Vol 3, Issue 4, April-2016, pp.76-85.

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

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

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