An Integrated Hardware Platform for Four Different Lightweight Block Ciphers

Haohao Liao and Howard M. Heys
Department of Electrical and Computer Engineering
Memorial University of Newfoundland

Abstract— **In this paper, we investigate the hardware implementation of four different, but similar, lightweight block ciphers: PRESENT, Piccolo, PRINTcipher and LED. The purpose of this paper is to present a common platform which integrates these four ciphers into one system using a shared datapath, with the objective of reducing the area below the total sum of area consumed by the individual ciphers. The structure and implementation of the platform is clearly stated in the paper with the target technology being the Altera Cyclone IV FPGA.**

I. Introduction

In recent years, with the development of new technologies, such as the RFID tags, more and more resource-constrained environments have security requirements. Under this circumstance, researchers have developed several lightweight block ciphers to meet the strict resource requirements such as PRESENT [1], Piccolo [2], PRINTcipher [3], and LED [4]. These ciphers share a similar structure referred to as a Substitution Permutation Network (SPN) [5]. In some applications, such as RFID tags and other embedded systems, there is a need to integrate several different lightweight block ciphers into a single device to supply higher flexibility for multiple application environment. However, there is always a tradeoff between flexibility and hardware resource consumption. Under this situation, we develop a digital hardware platform which integrates PRESENT, Piccolo, PRINTcipher, and LED without a large increase in hardware resource consumption. The platform is targeted to the Altera Cyclone IV FPGA technology [6]. The resources consumed by the platform are significantly fewer than the total sum of the four ciphers. The resulting resource usage and performance in terms of throughput are presented in this paper. Since in many applications, only the forward or encryption process of the block cipher is required (eg. counter mode), this paper does not discuss the decryption process, although due to the similar structure of encryption and decryption processes, similar results are expected.

II. Background

In this section, we present the necessary background for the work in this paper.

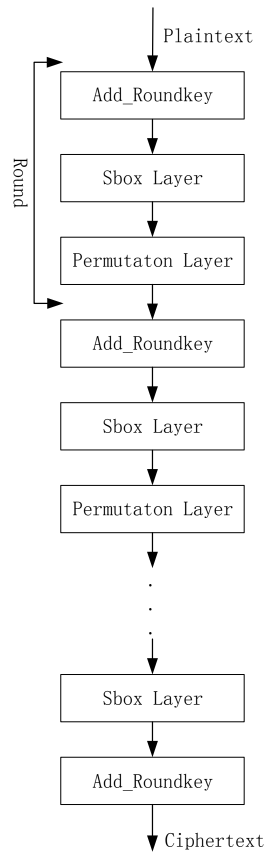

Fig. 1. A typical structure of SPN.

A. Substitution Permutation Networks

The SPN structure is widely used in many ciphers. Figure 1 shows a typical structure of SPN. Basically, an SPN consists of several rounds of three components: Add_Roundkey, substitution or s-box layer and permutation layer. The Add_Roundkey component is actually a bit-by-bit XOR of the data block and the round key. The substitution layer is a non-linear function applied to the data block executed by mapping sub-blocks using a fixed non-linear function. The permutation layer is a bit position change of the data block, usually, at the cost of no hardware resources. A typical lightweight block cipher usually consists of several identical rounds applying a different round key in each round to finally get the ciphertext.

B. Structure of PRESENT, Piccolo, PRINTcipher and LED

Figure 2 shows the top level hardware structure of these four ciphers. The combinational datapath contains the cipher components used in each round and takes as input the round key, K_i, and the cipher data as selected from either the plaintext or the current state. The finite state machine, FSM, controls the flow of data within the system.

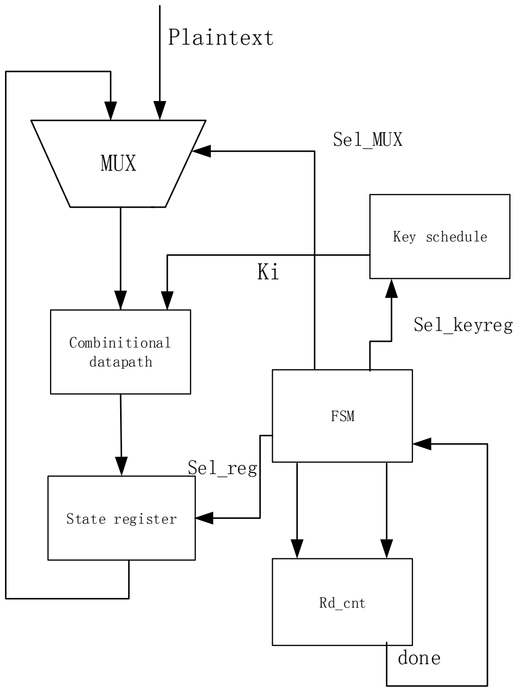

Fig. 2. Structure of Piccolo, PRESENT, Printcipher, and LED.

The major difference between these four ciphers is the combinational datapath and the key scheduling algorithms. This will be discussed in the following section.

1) PRESENT Cipher: The PRESENT cipher is an ultra-lightweight block cipher presented in 2007 [1]. The label PRESENT-80 refers to the cipher structure with an 80 bit length key, while PRESENT-128 uses a 128 bit length key. PRESENT-80 consists of 31 rounds by using a structure of substitution permutation network and it works based on a block size of 64 bits. The key scheduling part of this cipher is comprised of a 61 bit shift, an s-box and an XOR with the round counter. Note that the last round of PRESENT cipher does not include an s-box layer or permutation layer. The last round key will XOR with the last state.

2) Piccolo Cipher: The Piccolo cipher is a lightweight block cipher recently published in 2011 [2]. The cipher is similar to the PRESENT cipher. It also has two different key lengths: 80 bits and 128 bits. The 80-bit key version, Piccolo-80, consists of 25 rounds. The combinational datapath of the Piccolo-80 includes an round function, F, which consists of a diffusion matrix which is taken from AES and two s-boxes. The key scheduling part for Piccolo consists of two different keys. One is whitening key wk_i which is used in the first and last round of the encryption. The other one is round key rk_i which has two different values for each round. For round *i*, the two different round keys are labelled as rk_{2i-1} and rk_{2i}.

3) PRINTcipher: PRINTcipher is a lightweight block cipher published in 2010 [3]. Unlike the PRESENT cipher and Piccolo cipher, the cipher operates on a 48 bit data block. However, PRINTcipher still uses the conventional substitution permutation network except the permutation is selected by the key. It consists of 48 rounds. The key scheduling part for PRINTcipher is more simple than the previous two ciphers. In each round, 48 bits of key, SK_1, are used to XOR with the state and 32 bits of key, SK_2, are used for the keyed-permutation.

4) LED Cipher: The LED cipher is a lightweight block cipher published in 2011 [4]. Similar to the above mentioned three ciphers, LED cipher also uses an SPN structure. The cipher operates on a 64 bit data block. It has four different key sizes: 64 bit, 80 bit, 96 bit and 128 bit and they are labelled as LED-64, LED-80, LED-96 and LED-128, respectively. LED-64 needs 32 rounds to finish encryption while the others need 48 rounds to finish encryption. In the LED cipher, a step is defined as four identical rounds without an Add_RoundKey process. The major difference between the LED cipher and other three ciphers is the key scheduling component. LED cipher does not need a round key for each round. Instead, it needs a round key for each step.

III. Design

The purpose of our design is to integrate PRESENT, Piccolo, PRINTcipher and LED into one platform. Since area and resource usage are the two major factors that we consider, we use an iterative design in our implementation.

Figure 3 shows the top level design of our platform. The components “PRESENT comb”, “Piccolo comb”, “PRINTcipher comb” and “LED comb” are the combinational datapaths of PRESENT, Piccolo, PRINTcipher and LED, respectively. We have two different multiplexers: “MUX_1” is used to select the input from plaintext or the state register, while “MUX_2” is used to load the correct output of the combinational datapaths into the state register. The component “Rd_cnt” is a counter which is used to indicate the end of the encryption process. It also generates the round constant which is used in the key scheduling algorithm. FSM is a finite state machine which is the control unit of our whole platform.

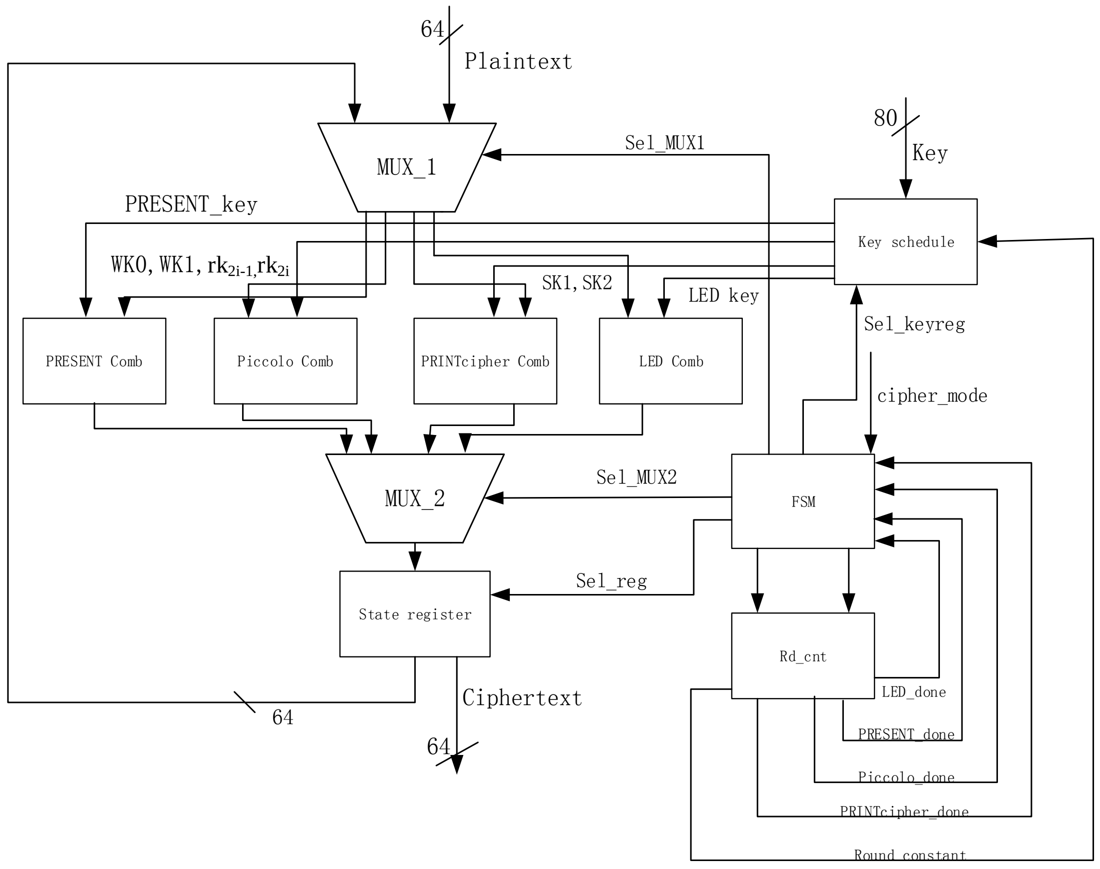

Fig. 3. Top level design of platform.

To decrease the area and resources consumed by this platform, we have tried to share some similar components with different ciphers. For example, the "Add_RoundKey" process for PRESENT and Piccolo are almost the same, so we only use a 64 bit XOR for these two ciphers. However, in this way, we need to use an additional 64 bit 2-to-1 multiplexer and after examining the synthesis result, we found that this approach is not worthwhile. In our final design, we have decided that our approach to save the area is based on sharing the state register within the cipher datapath and the key register inside the key schedule block. The penalty of this structure is that we need to use one more 64 bit 4-to-1 multiplexer to choose the correct data to be loaded into the state register.

Before the encryption process, a 3 bit "Cipher_mode" signal must be loaded into the finite state machine to choose the cipher that would be used in the encryption process. Since the four ciphers need a different number of rounds to finish the encryption process, we need three different done signals from "Rd_cnt" to indicate the end of the encryption process. The block size of PRINTcipher is 48 bits while PRESENT, Piccolo and LED have a 64 bit block. If we choose to use PRINTcipher, the 16 most-significant bits of output will be set to default value of 0.

Figure 4 shows the block diagram of our design. We have 5 input signals. "Cipher_mode" is a 3-bit signal which is used to choose the cipher to be used in the next encryption process. For example, "000" means no cipher is chosen and "001" indicates that the next cipher that will be used is PRESENT. The "Plain_valid" and "Key_valid" signal is used to indicate that plaintext or key are available at the input. For the output, "Cipher_valid" signal is the data valid signal for ciphertext.

Figure 5 shows the state transition diagram of our control unit The following steps show how our platform works:

1) Before the "Cipher_mode" signal is changed to any valid value for one of the four ciphers, the FSM will stay in "IDLE" state.
2) After the "Cipher_mode" signal is loaded into our Platform, the FSM will step into "PRESENT_IDLE", "Piccolo_IDLE", "PRINT_IDLE" or "LED_IDLE" based on the value of "Cipher_mode" signal. For example, if "Cipher_mode" = "001", then the FSM will step into

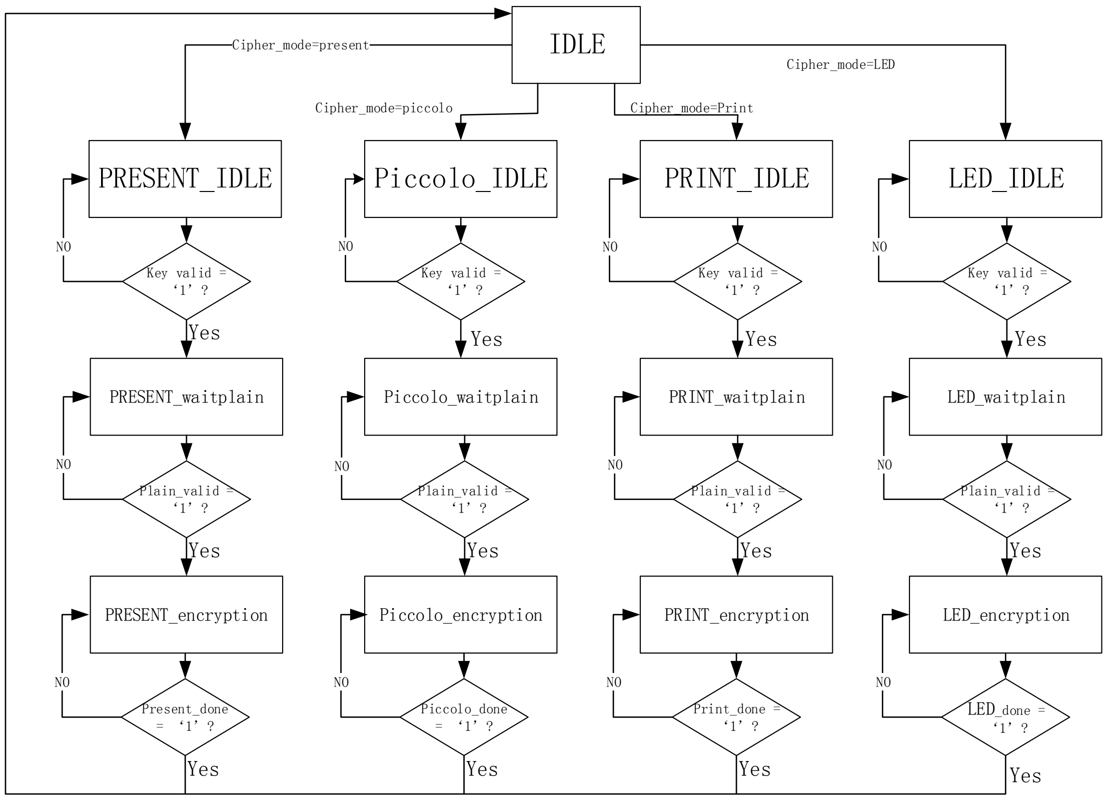

Fig. 5. State transition diagram of Platform

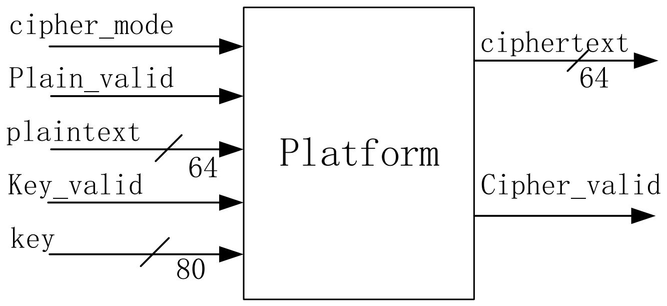

Fig. 4. Block diagram of Platform.

"PRESENT_IDLE" state.

3) In "PRESENT_IDLE", "Piccolo_IDLE", "PRINT_IDLE" or "LED_IDLE" state, the FSM will wait for the key that will be used in the following encryption process. After the key is loaded into the Platform, the FSM will step into "PRESENT_waitplain", "Piccolo_waitplain", "PRINT_waitplain" or "LED_waitplain" state and wait for the plaintext that need to be encrypted.
4) After the plaintext is loaded into the Platform, the FSM will step into the "PRESENT_encryption", "Piccolo_encryption", "PRINT_encryption" or "LED_encryption" state and start the encryption process.
5) In "PRESENT_encryption", "Piccolo_encryption", "PRINT_encryption" or "LED_encryption" state, when the done signal indicate that the encryption process is over, the FSM will step into "IDLE" or one of the four "Cipher_IDLE" states based on the value of "Cipher_mode" signal.

The four ciphers have separate control paths, which is used to ensure that each cipher is not influenced by another. Moreover, to ensure that the whole system works smoothly, we also have fault-tolerant design for the following cases:

- The FSM will stay in IDLE state if there is no "Cipher_mode" signals loaded into it. Any plaintexts and keys will be ignored.

- During the encryption process, any "Cipher_mode" signals, keys, and plaintexts will be ignored.

IV. Implementation and Synthesis Results

We have investigated our design by targetting the Altera Cyclone IV FPGA [6]. First, we synthesize these four ciphers independently by using Quartus II [7] design software. Then we synthesize our platform to give a comparison to the four ciphers. All these four ciphers use an iterative implementation to reduce the area and resource consumption. From Table I, it is clear that our design saves a lot of resources, not only in combinational logic elements (Comb LEs), but also in sequential logic elements (Seq LEs). The total combinational LEs consumed by these four ciphers should be 2964, while the platform only consumes 2296 combinational LEs. This results from simplification that occurs in the key scheduling component when the four ciphers are combined. In fact, we find that the key scheduling algorithms consume a large amount of combinational LEs since the key scheduling algorithms of these four ciphers are complex. Table II shows the number of combinational LEs for the key scheduling consumed by the four ciphers and our platform. This is also the reason why the total combinational LEs of our platform is smaller even though we add an extra 64 bit 4-to-1 multiplexer in the platform. Moreover, the number of sequential LEs is significantly reduced by sharing the state register and key register. The total number of sequential LEs consumed by these four ciphers should be 610, while the platform only consumes 173 sequential LEs.

In Table III, we examine the performance of our system by presenting the resulting throughput of the 4 ciphers using our platform, as determined by the synthesis tools. The data of throughput is based on the maximum frequency of our platform being 209.47 MHz. The throughput of the four ciphers are different from each other since they need a different number of rounds to finish the encryption process. The number of clock cycles that are needed to finish the encryption process is 32, 25, 48 and 48 for PRESENT, Piccolo, PRINTcipher and LED, respectively, after the plaintext is loaded into our platform.

TABLE I
Resources for different ciphers

Cipher	Comb LE	Seq LE
PRESENT	613	153
Piccolo	809	153
PRINTcipher	516	143
LED	1026	161
Total	**2964**	**610**
Platform	**2296**	**173**

V. Conclusion

Our design successfully reduces the area and resource consumption of four lightweight block ciphers. Since users can choose the cipher they want to use by simply change the "Cipher_mode" signal, our design supplies a high flexibility allowing the potential use of the platform for a variety of applications. Since we use an iterative design, the throughput of our design is not very high. However, while resource usage is often critical for embedded systems, required throughput is typically very modest in most lightweight applications.

TABLE II
Number of combinational LEs of Key Scheduling

Cipher	Comb LE of Key Scheduling	Percentage of Comb LEs
PRESENT	328	53.5%
Piccolo	431	53.2%
PRINTcipher	242	46.9%
LED	604	58.9&
Total	**1605**	**54.1%**
Platform	**869**	**37.8%**

TABLE III
Performance of our implementation

Cipher	Throughput
PRESENT Throughput	419 Mbps
Piccolo Throughput	536 Mbps
PRINTcipher Throughput	209 Mbps
LED Throughput	280 Mbps

References

[1] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin and C. Vikkelsoe, PRESENT: An Ultra-Lightweight Block Cipher,vol. 4727, pp. 450-466, Lecture Notes in Computer Science, Cryptographic Hardware and Embedded Systems - CHES 2007.

[2] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita and Taizo Shirai Piccolo: An Ultra-Lightweight Blockcipher, vol. 6917, pp. 342-357, Lecture Notes in Computer Science , Cryptographic Hardware and Embedded Systems - CHES 2011.

[3] Lars Knudsen, Gregor Leander, Axel Poschmann, and Matthew J. B. Robshaw PRINTcipher: A Block Cipher for IC-Printing, vol. 6225, pp. 16-32, Lecture Notes in Computer Science, Cryptographic Hardware and Embedded Systems - CHES 2010.

[4] Guo, Jian., Peyrin, Thomas., Poschmann, Axel., Robshaw, Matt.,"The LED Block Cipher",vol. 6917, pp. 326-341, Lecture Notes in Computer Science, Cryptographic Hardware and Embedded Systems - CHES 2011.

[5] Menezes, A., van Oorschot, P.C., Vanstone, S.: The Handbook of Applied Cryptography. CRC press, Boca Raton, USA(1996).

[6] Altera Cyclone IV Device Handbook, available at http://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf.

[7] Altera Quartus II Handbook, available at http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf.