Compact FPGA Implementation of PRESENT with Boolean S-Box

J. J. Tay[1], M. L. D. Wong, M. M. Wong
Faculty of Engineering, Computing and Science,
Swinburne University of Technology,
Sarawak Campus, Malaysia.
[1]E-mail: jtay@swinburne.edu.my

C. Zhang, I. Hijazin
Faculty of Engineering and Industrial Sciences,
Swinburne University of Technology, Hawthorn,
Melbourne, Australia.

Abstract

Ever since the conception of the ideology known as the Internet of Things (IoT), our world is slowly approaching the brink of mankind's next technological revolution. The realization of IoT requires an enormous amount of sensor nodes to acquire inputs from the connected objects. Due to the lightweight nature of these sensors, constraints emerge in the form of limited power supply and area for the implementation of information security mechanism. To ensure security in the data transmitted by these sensors, lightweight cryptographic solutions are required. In this work, our goal is to implement a compact PRESENT cipher onto a Field Programmable Gate Array (FPGA) platform. Our proposed design uses an 8-bit datapath to reduce hardware size. Instead of a traditional look-up table (LUT) based S-Box, we have implemented a Boolean S-Box through Karnaugh mapping. Further factorization is also done to reduce the size of the Boolean S-Box. As a result, we have achieved the smallest FPGA implementation of the PRESENT cipher to date, requiring only 62 slices on the Virtex-5 XC5VLX50 platform. Our design also features a respectable throughput of 51.32 Mbps at the maximum frequency of 236.574 MHz.

Keywords

PRESENT, lightweight, block cipher, FPGA, Boolean

1. Introduction

On March 12, 1989, Sir Tim Berners-Lee brought about a revolution to the world in the form of the World Wide Web, a.k.a. the Internet. Ever since then, the Internet has been integrated into our daily lives as an essential need at an incredible pace. It single-handedly destroyed the geographical barriers between nations and realized the concept of "globalization". More importantly, it became the basis of mankind's technological advancement.

Naturally, after riding the success of the Internet for more than two decades, people starts to wonder: What is the next revolution? Will it bring about such an impact to the way we live our lives as the Internet did? The answers lie within the IoT.

Essentially, IoT is the extension of the Internet to connect just about everything on the planet. This includes real, physical objects ranging from household refrigerators to health monitor sensors to industrial controls. As such, these "things" that are connected to the Internet will be able to take actions or make decisions based on the information they obtained from the Internet with or without human intervention. In addition, they also update the Internet with real time information with the help of various sensors. This allows objects connected to the IoT to communicate and work with each other seamlessly.

As mentioned previously, IoT relies heavily on the sensor nodes on the connected objects to obtain real time information. In certain applications, confidentiality of such information might be extremely important. Therefore, cryptography comes into play.

Due to the resource constrained nature of the sensors, the security requirement has led to various attempts to optimize established cryptographic ciphers for small scale applications. Most notably, major efforts had been directed into constructing compact and low power architecture of the Advanced Encryption Standard (AES) block cipher as reported in [1]. At the same time, various new lightweight block ciphers had also been introduced specifically for small scale applications in resource constrained environment. One of the most efficient lightweight ciphers to date is the PRESENT cipher developed by Andrey Bogdanov [2].

In this paper, we provide a hardware implementation of the PRESENT cipher on the FPGA platform. Instead of using LUTs or BRAMs to implement the PRESENT S-Box, we attempted to realize the S-Box using logic circuits consisting of AND and OR gates. This is achieved through the application Karnaugh mapping on each of the four output bits of the PRESENT S-Box. This results in four sum-of-products (SOP) expressions that make up the S-Box. To further reduce the size of the architecture, we factorize the common factors of the four expressions. To the best of our knowledge, our implementation of the PRESENT cipher currently requires the least amount of FPGA slices compared to related works.

The rest of this paper is organized as follows: Section 2 provides an overview of related works on the FPGA implementation of PRESENT cipher. The preliminaries on the PRESENT cipher are given in Section 3. Section 4 provides the information regarding the methodologies used to reduce the hardware requirement of the cipher, namely Karnaugh mapping and factorization. The implementation results and comparisons with related works are provided and discussed in Section 5. Finally, concluding remarks are drawn in Section 6 to summarize the work.

2. Related Works

Since the proposal of the PRESENT cipher in the year 2007, there have been several works concerning the optimization of the PRESENT cipher on FPGA. Sbeiti et al. [3] attempted an efficiency optimized architecture. The authors used a 3-state finite state machine (FSM) to implement the cipher control logic. In addition, the authors also attempt to deviate from the traditional LUT S-Box approach by using a tool named "espresso" to derive Boolean functions that compute the PRESENT S-Box.

Yalla et al. [4] tried a different approach using 16-bit shift registers (SRL16) for efficient memory implementation. A total of four S-Boxes are used for 16-bit datapath implementation. The control signals are generated by a counter and some additional logic.

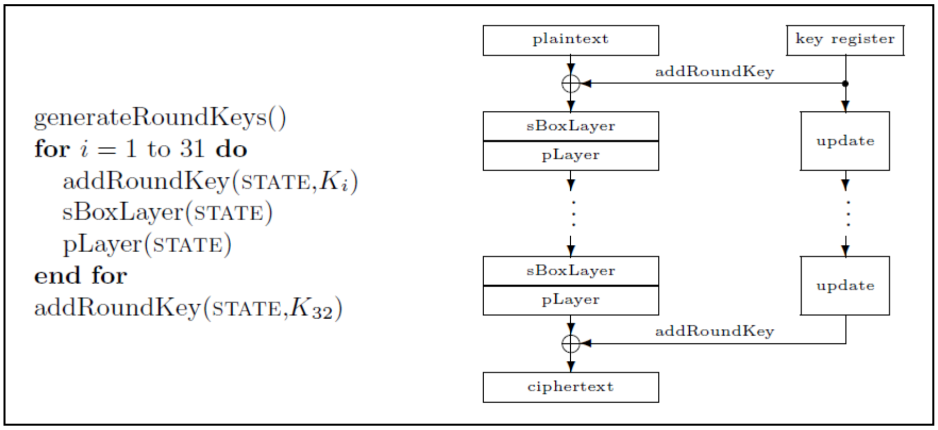
Kavun et al. [5] proposed a RAM-based FPGA implementation of PRESENT. The authors explained that the memory is used in a ping-pong fashion to store the partially encrypted data instead of using registers. Storing the S-Box in RAM as LUT, however, provided undesirable results due to complex control logic.

To the best of our knowledge, the most compact implementation of PRESENT cipher on FPGA is currently achieved by Hanley et al. [6]. The authors attempted two different approaches, namely the iterative architecture and the serial architecture. The iterative approach uses a full 64-bit datapath while the serial approach uses an 8-bit datapath. In fact, the serial architecture is a modification of a previous work by Rolfes et al. [7]. Overall, the iterative architecture achieved much better efficiency than the serial architecture at the cost of slightly larger footprint. However, as noted in [1], the typical upper bound of the throughput requirements of passive sensor nodes (mostly RFIDs) is about 100 kbps. Since both architectures satisfy this requirement, the serial architecture shows tangible benefits in its smaller area.

3. Preliminaries

PRESENT is a symmetric block cipher that shares a rather similar architecture with the popular AES. It was proposed by Bogdanov et al. [2] in the year 2007 and released as the standard for lightweight cryptography under ISO/IEC 29192-2 [8]. The algorithm was designed specifically for small scale applications such as RFID tags, sensor nodes and IoT.

Essentially, PRESENT is constructed based on the substitution-permutation network (SPN) with a block size of 64-bit and accepts a key size of either 80-bit or 128-bit. In this paper, we will be adopting the 128-bit key length for better security and comparison with previous works. A single round of PRESENT encryption includes a key addition step, a substitution layer, and a permutation layer. A complete encryption is accomplished by 31 iterations of the aforementioned processes with an additional key addition after the final round. The overall flow of the PRESENT algorithm can be observed in **Figure 1**.



```
generateRoundKeys()
for $i = 1$ to 31 do
    addRoundKey(STATE,$K_i$)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE,$K_{32}$)
```

Figure 1: Top level description of the PRESENT algorithm.

The PRESENT S-Box is given in **Table 1** and the permutation layer is described in **Table 2**, where bit *i* of the state is moved to bit position *P(i)*.

The PRESENT cipher requires unique 64-bit keys for each round of key addition and these keys are generated from the provided 80-bit or 128-bit keys through key scheduling. In this case, we will be discussing the process for the 128-bit version. To generate the next set of round key from the current one, the 128-bit key is first rotated 61 bits to the left. Then, the leftmost 8 bits of the rotated key are passed through the same S-Boxes used in the encryption process. Finally, the round counter value is XORed with the 66th down to 62nd bit of the key with the least significant bit of the round counter on the right. 64 most significant bits of the generated key are used for the key addition of the respective rounds.

4. FPGA Implementation of PRESENT

The overall architecture of our design follows closely to the serial architecture in [6]. Two sets of registers are used to store the data and the key. We use two S-Boxes in our design to achieve the same 8-bit datapath. The two S-Boxes are responsible for both the encryption of the data blocks and the key scheduling. As a result, eight clock cycles are required to complete the substitution of the 64-bit data blocks and one more clock cycle is required for the most significant 8 bits of the rotated key. Adding up the extra 16 clock cycles needed to register the inputs, the design has a latency of 295 cycles in total.

Both the permutation layer and the bit-rotation of the key scheduling can be achieved in hardware through simple bit-wiring without increasing the size of the design.

The control logic for the encryption process is handled by two counters. Firstly, a 31-step counter is used to coordinate the rounds of encryption. It is mainly required to know when the encryption is done, enable the registers to get new inputs (during round 0), and also for the key scheduling XORs. Secondly, a smaller 9-step counter is used as a selector to the data passing through the two S-Boxes and controls the timing to update the value of the registers in each round of encryption. The overall architecture of our design is depicted in **Figure 2**.

Table 1: PRESENT S-Box.

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

Table 2: PRESENT permutation layer.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(i)	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P(i)	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P(i)	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P(i)	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

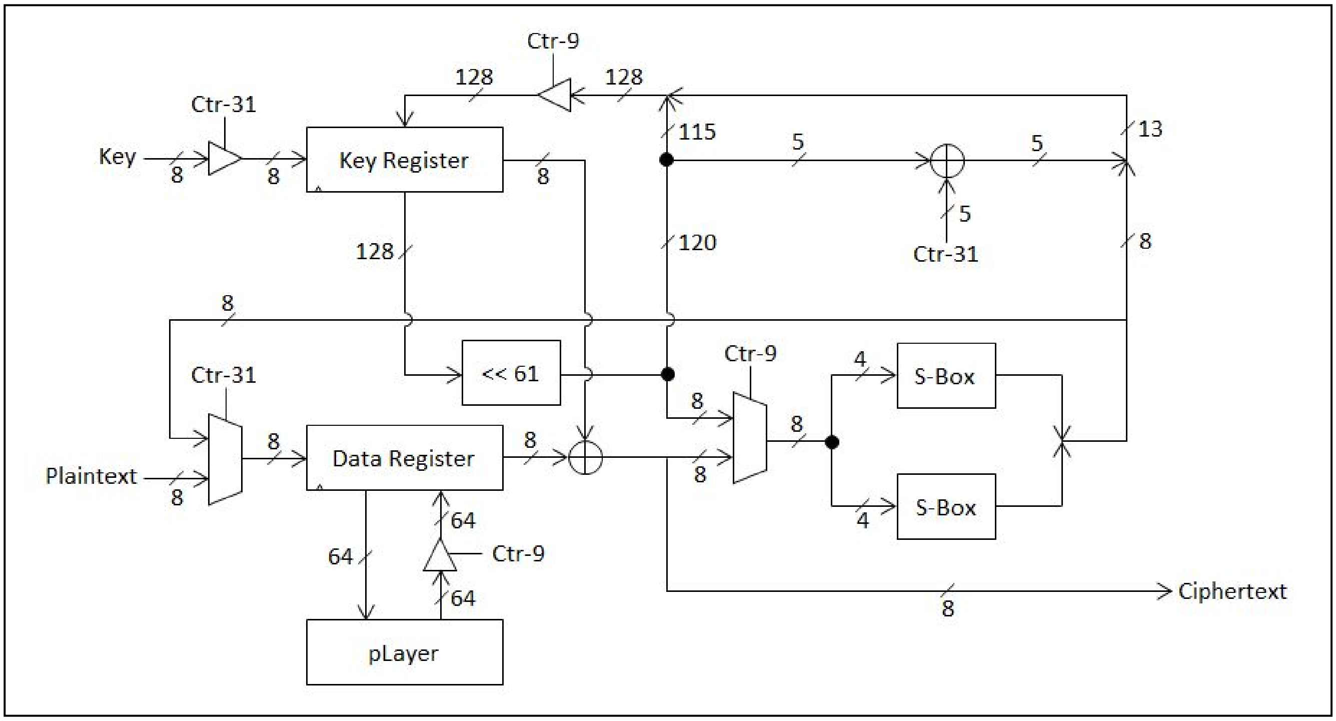


Figure 2: Overall architecture.

4.1. Boolean S-Box

A LUT or BRAM-based S-Box is very rigid in the sense that not much optimization techniques can be further implemented to shrink the size of the architecture. Taking cues from the work in [3], we attempted to construct a Boolean version of the PRESENT S-Box using logic gates. We used Karnaugh mapping to form simplified SOP expressions for each of the four output bits of the S-Box. Throughout this paper, the input bits are denoted as {*a*,*b*,*c*,*d*}, with *a* being the most significant bit. In the same manner we denote the output bits as {*w*,*x*,*y*,*z*}. The resultant expressions are given in **Table 3**.

Table 3: Boolean SOP expressions for PRESENT S-Box generated from Karnaugh mapping.

Output	Expression
w	(*a'c'd'*)+(*a'cd*)+(*ab'd*)+(*ab'c*)+(*a'bc*)
x	(*b'cd'*)+(*abc'*)+(*b'c'd*)+(*a'b'c'*)+(*a'bcd*)
y	(*abd*)+(*ab'c'*)+(*a'cd'*)+(*a'b'c*)+(*ab'd'*)
z	(*a'cd*)+(*acd'*)+(*ab'd'*)+(*a'b'd*)+(*a'bc'd'*)+(*abc'd*)

From the SOP expressions, the S-Box can be constructed using only AND and OR gates. The benefit of being able to express a certain transformation in any form of logical or mathematical expression is that it enables the possibility to reduce the size of the design through simplification of the expression.

In addition, pipeline registers can be placed at multiple levels between the logic gates of the Boolean S-Box to further improve the throughput of the design. In other words, the Boolean S-Box can be pipelined easily shall the need arise.

4.2. Factorization

Observing **Table 3**, we can see that there are common factors between the four expressions. This implies that extra hardware is wasted on calculating the same values in the initial expressions. Instead, we factorized the expressions to reduce the amount of logic gates required. Repeated operations are replaced by a single factor which is calculated only once. The factor is then used in all the expressions that required its value without having to recalculate it. The factorized expressions are shown in **Table 4**.

To see the improvement achieved through factorization, note that the original expressions in **Table 3** required a total of 45 AND gates and 17 OR gates. After factorization, the requirement is reduced to 26 AND gates and 17 OR gates.

Table 4: Expressions for PRESENT S-Box after factorization.

Output	Expression
w	$(f_1)+(f_2)+(df_5)+(cf_5)+(bf_6)$
x	$(b'f_7)+(f_3)+(df_8)+(a'f_8)+(bf_2)$
y	$(df_9)+(c'f_5)+(d'f_6)+(b'f_6)+(f_4)$
z	$(f_2)+(af_7)+(f_4)+(a'b'd)+(bf_1)+(df_3)$
Factor	Expression
f_1	$a'c'd'$
f_2	df_6
f_3	$c'f_9$
f_4	$d'f_5$
f_5	ab'
f_6	$a'c$
f_7	cd'
f_8	$b'c'$
f_9	ab

Table 5: Results and comparison with related works.

Design	Platform	# Flip Flops	# LUTs	# Slices	Max Frequency MHz	Latency Cycles	Throughput Mbps	Efficiency kbps/slice
[2]	Spartan-3 XC3S400	200	300	202	254	32	508	2514.85
[3]	Spartan-3 XC3S50	114	159	117	113.895	256	28.47	243.37
[4]	Spartan-3 XC3S50	-	-	83 + 1 BRAM	129.77	1062	7.82	94.22
[5] Iterative	Virtex-5 XC5VLX50	200	285	87	250.89	47	341.64	3926.87
[5] Serial	Virtex-5 XC5VLX50	203	237	70	245.76	295	53.32	761.68
Our work	Virtex-5 XC5VLX50	201	222	62	236.574	295	51.32	827.81

5. Implementation Results and Comparison

Our proposed PRESENT cipher is implemented on the Xilinx Virtex-5 XC5VLX50-1 FF324 FPGA. This is the same platform used in [6], which is, to the best of our knowledge, the smallest implementation of the PRESENT cipher on a FPGA to date. Thus, we expect a fair comparison. We also include results from other works for a more complete comparison.

As shown in **Table 5**, we have achieved the smallest architecture of PRESENT cipher on FPGA to date. It is noted, however, that our design did not produce the highest efficiency.

In [6], it is noted that the implemented design is unlikely to be able to run at its maximum frequency in actual application. Also, in [9], we have the standard frequency for contactless smart cards as 13.56 MHz. Considering the standard as the operating frequency of the sensor nodes, we recalculated the throughput for the each of the designs and summarized them in **Table 6**.

Table 6: Throughput comparison at 13.56 MHz.

Design	Frequency MHz	Latency Cycles	Throughput Mbps
[2]	13.56	32	27.12
[3]		256	3.39
[4]		1062	0.82
[5] Iterative		47	18.46
[5] Serial		295	2.94
Our work		295	2.94

With reference to **Table 6**, we can see that all of the designs satisfied the requirement in [1], an upper bound throughput of 100 kbps. In other words, the throughput is no longer the main concern in the design space as every design encrypts data at a much faster rate than the rate of data transmission in small scale applications such as RFIDs. As a result, the main competing factor falls on the area requirement of the designs. In this regard, our design achieved the best results, requiring only 62 slices.

In addition, the Boolean S-Box expressions open up opportunities to experiment with various common subexpression elimination (CSE) techniques to further reduce the area requirement of the design. In this work, we have attempted a primitive factorization technique as described in Section 4.2 and achieved the smallest implementation of PRESENT cipher on FPGA to date. We are confident that more complex CSE algorithms are capable of producing even smaller designs in the future. In contrast, LUT-based S-Boxes implemented in most of the other competing designs allow limited room for improvement in comparison.

6. Conclusion

In this paper, we present a compact implementation of the lightweight block cipher PRESENT on FPGA. We reduced the amount of S-Boxes by adopting an 8-bit datapath and utilized simple control logic in the form of two counters. In addition, we managed to reduce the area requirement of the PRESENT S-Box through Karnaugh mapping and further factorization. The resultant Boolean S-Box is also ready for pipelining.

From the comparison against related works, it can be noted that our design is the smallest implementation to date, with the area requirement valued at 62 slices. Despite achieving the smallest slice count, our implementation has a respectable throughput and efficiency as well. At the standard frequency of 13.56 MHz, our design is capable of achieving throughput much higher than the required 100 kbps, making it sufficient for lightweight applications such as sensor nodes and RFIDs.

Acknowledgement

This work was supported in part of the Melbourne-Sarawak Research Collaboration Scheme.

9. References

[1] C. Alippi, A. Bogdanov, and F. Regazzoni, “Lightweight Cryptography for Constrained Devices”, 2014 International Symposium on Integrated Circuits (ISIC), IEEE, Singapore, 2014, pp. 144-147.

[2] 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”, Cryptographic Hardware and Embedded Systems – CHES 2007, Springer, Austria, 2007, pp. 450-466.

[3] M. Sbeiti, M. Silbermann, A. Poschmann, and C. Paar, “Design Space Exploration of PRESENT Implementation for FPGAs”, Programmable Logic, 2009. SPL. 5th Southern Conference, IEEE, Sao Carlos, 2009, pp. 141-145.

[4] P. Yalla, and J.P. Kaps, “Lightweight Cryptography for FPGAs”, 2009 International Conference on Reconfigurable Computing and FPGAs, IEEE, Mexico, 2009, pp. 225-230.

[5] E.B. Kavun, and T. Yalcin, “RAM-Based Ultra-Lightweight FPGA Implementation of PRESENT”, 2011 International Conference on Reconfigurable Computing and FPGAs, IEEE, Mexico, 2011, pp. 280-285.

[6] N. Hanley, and M. O’Neill, “Hardware Comparison of the ISO/IEC 29192-2 Block Ciphers”, 2012 IEEE Computer Society Annual Symposium on VLSI, IEEE, USA, 2012, pp. 57-62.

[7] C. Rolfes, A Poschmann, G. Leander, and C. Paar, “Ultra-Lightweight Implementations for Smart Devices – Security for 100 Gate Equivalents”, Smart Card Research and Advanced Applications, Springer, UK, 2008, pp. 89-103.

[8] “Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers”, ISO/IEC 29192-2:2012, ISO, 2012.

[9] “Identification cards – Contactless integrated circuit cards – Proximity cards – Part 2: Radio frequency power and signal interface”, ISO/IEC 14443-2:2010, ISO, 2010.