DESIGN SPACE EXPLORATION OF PRESENT IMPLEMENTATIONS FOR FPGAS

Mohamad Sbeiti, Michael Silbermann, Axel Poschmann, Christof Paar

Horst Görtz Institute for IT Security
Embedded Security Group
44801 Bochum, Germany
{mohamed.sbeiti,michael.silbermann}@rub.de
{poschmann,cpaar}@crypto.rub.de

ABSTRACT

In this paper we investigate the performance of the block cipher PRESENT on FPGAs. We provide implementation results of an efficiency (i.e. throughput per slice) optimized design and compare them with other block ciphers. Though PRESENT was originally designed with a minimal hardware footprint in mind, our results also highlight that PRESENT is well suited for high-speed and high-throughput applications. Especially its hardware efficiency, i.e. the throughput per slice, is noteworthy.

1. INTRODUCTION

PRESENT is a recently proposed lightweight block cipher that was specifically designed with a low hardware footprint in mind [1]. Besides this, another major design goal was simplicity which eases implementation. The authors claim that PRESENT is resistant against known attacks and provide proofs for linear and differential cryptanalysis. First cryptanalytic results of independent researchers underline these claims [2].

Up to now there have been only a few papers published that deal with implementations of PRESENT. Rolfes *et al.* investigate different architectures for ASIC implementations in [3]. Graber *et al.* provide implementation results for a lightweight Instruction Set Extension for bit sliced software implementations [4]. Guo *et al.* investigate the overall performance of a System-On-Chip (SoC) platform that uses a PRESENT implementation as a cryptographic co-processor in [5]. However their focus lies on hardware software co-design rather than on a plain hardware implementation.

Up to now no design space exploration of PRESENT on FPGAs has been published. Our main contribution is to close this gap and provide the first implementation results of such kind. The remainder of this work is organized as follows: In Section 2 the PRESENT algorithm is briefly recalled. Subsequently, in Section 3 our implementation is described and our results are presented and compared with FPGA implementations of different block ciphers. Finally, in Section 4 this paper is concluded.

2. THE PRESENT ALGORITHM

PRESENT is symmetric encryption algorithm that was first published in [1]. It was specifically designed with ultra-constrained applications such as passive low-cost RFID-tags in mind. PRESENT is a so-called substitution-permutation network (SPN) with a block size of 64 bits and two different key sizes: 80 or 128 bits. From here on we refer to the version with an 80 bit key as PRESENT-80 and the one with a 128 bit key as PRESENT-128. PRESENT has 31 regular rounds and a final round that only consists of the key mixing step. One regular round consists of a key mixing step, a substitution layer, and a permutation layer.

The substitution layer consists of 16 S-Boxes that each have 4-bit input and 4-bit output (4x4): $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$. The S-Box is given in hexadecimal notation according to Table 1. The bit permutation used in PRESENT is given by Table 2. Bit i of STATE is moved to bit position P(i). In other words: the bit on position 0 of the input to the P- Layer is moved to position 0, the bit on position 4 is moved to position 1, the bit on position 8 to position 2 and so on.

The key schedule of PRESENT-80 consists of a 61-bit left rotation, an S-Box, and a XOR with a round counter. Note that PRESENT uses the same S-Box for the datapath and the key schedule, which allows to share resources. The user-supplied key is stored in a key register and its 64 most significant (i.e. leftmost) bits serve as the round key. The key register is rotated by 61 bit positions to the left, the

Table 1. S-Box Layer of PRESENT

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	2	1

Table 2. Permutation layer of PRESENT

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

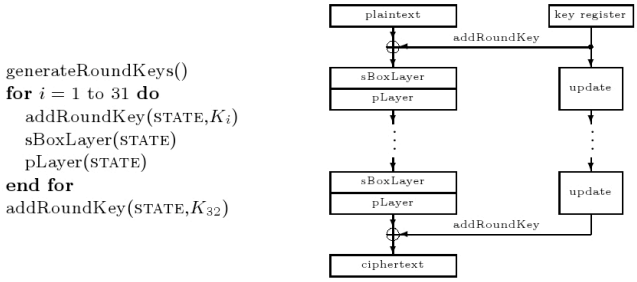

Fig. 1. A top-level algorithmic description of PRESENT.

left- most four bits are passed through the PRESENT S-Box, and the round counter value i is exclusive-ored with bits $k_{19}k_{18}k_{17}k_{16}k_{15}$ of K with the least significant bit of the round counter on the right. The key schedule of PRESENT-128 is slightly different from the one of PRESENT-80. The only difference is that it contains two S-Boxes and consequently the 8 MSB are processed by the S-Boxes (and not only the 4 MSB as in PRESENT-80). All other components, i.e. the 61-bit left rotation, an S-Box, and an XOR with a round counter, and also the order of processing stays the same. For further details, the interested reader is referred to [1]. Testvectors, visualizations, and implementations of PRESENT can be downloaded from the website `www.lightweightcrypto.org/present`.

3. FPGA IMPLEMENTATION OF PRESENT

The main design goals of the PRESENT block cipher described in Section 2 were simplicity and high performance/area ratio, so that all cipher components can be easily mapped in hardware. First, we describe our implementation of the encryption algorithm of PRESENT. The top level design overview is shown in Fig. 3 and the interface of the cipher top module is shown in Fig. 4. As can be seen from the latter one our PRESENT-80 and PRESENT-128 entities have 212 and 270 I/O pins, respectively. We did not implement any I/O logic such as a UART interface in order to achieve implementation figures for the plain PRESENT core. The interface usually strongly depends on the target application.

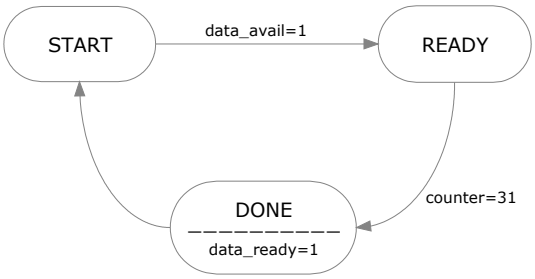

Fig. 2. Finite State Machine of the encryption core.

We deliberately use additional I/O pins for a parallel key input. There are two reasons why we abandon the options of hard-coding the key inside the cipher module or implementing serial interface to supply the key to the algorithm. First, we want to reduce the control logic overhead to a minimum to be able to present the results reflecting the performance of the ciphering algorithm only. Secondly, most applications will use PRESENT as an independent cipher module inside a larger top entity, so that the key can be supplied externally and in that perspective our implementation model offers the best flexibility.

Unfortunately, the low-cost Spartan-III XC3S200 FPGA has no package with more than 173 I/O pins [6]. Therefore we decided to move to the more advanced Spartan-III XC3S400 which features a package (FG456) with 264 I/O pins. Larger Spartan FPGAs such as the Spartan-III XC3S1000 feature even more I/O pins but also contain more logic resources. Since we focus on lightweight and low-cost implementations of PRESENT in this paper we chose the smallest possible device Spartan-III XC3S400 which is only slightly larger (and hence more expensive) than the Spartan-III XC3S200.

The entire cipher control logic was implemented as a 3-state finite-state machine (see Fig.2). After reset the first round begins and the two inputs of the algorithm, plaintext and user-supplied key are read from the corresponding registers. The 64- and 80-bit multiplexers select the appropriate input depending on the value of the round counter, i.e. initial values for plaintext and key are valid only in round 1. Both 64- and 80-bit D-flip-flops are used for round synchroniza-

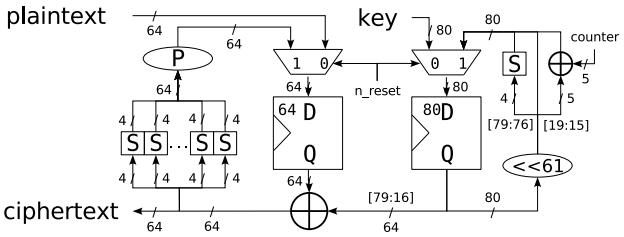

Fig. 3. The data path of an area-optimized version of the PRESENT-80 encryption unit.

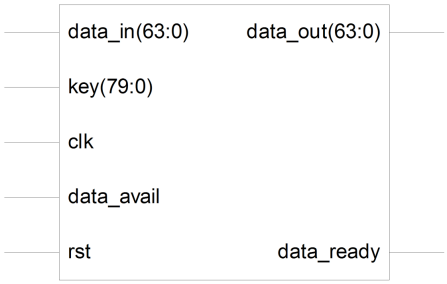

Fig. 4. Interface of the PRESENT-80 top module.

tion between the round function output and the output of the key schedule. Part of the round key is then XOR-ed with the plaintext. Key schedule and round function run in parallel for each round $1 \leq i \leq 32$.

Implementation of both permutation and bit-rotation is very straightforward in hardware, which is a simple bit-wiring. The highly non-linear PRESENT S-Box function is the core of the cryptographic strength of the cipher, and is the only design component that takes a lion's share of both computational power and area. Two implementation options for the PRESENT S-Box were taken in consideration in order to optimize the efficiency of the cipher. Using Look-Up Tables (LUTs) for bit substitution is the most obvious one and was implemented first. An alternative considered next was determining a minimal non-linear Boolean function

$$\begin{aligned} S_i : \quad \mathbb{F}_2^4 &\mapsto \mathbb{F}_2 \\ (x_3x_2x_1x_0) &\mapsto y_i, \qquad 0 \leq i \leq 3 \end{aligned}$$

for each bit output of the PRESENT S-Box using only standard gates, i.e. AND, OR and NOT. A tool named `espresso` [7] helped us produce such minimal Boolean functions for the PRESENT S-Box.

Interestingly, in some cases this modification yielded performance boost in terms of max. frequency/throughput and area requirements measured in occupied slices. E.g., for PRESENT-80 with `espresso`-optimized S-Box ISE showed significant decrease in critical path delay due to routing as compared to the S-Box implementation with LUTs. From our results we conclude that `espresso` and its minimal Boolean functions can yield better resources utilization and may in some cases outpace ISE's internal synthesis mechanisms.

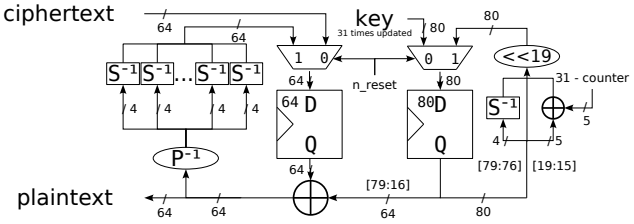

Fig. 5. The data path of an area-optimized version of the PRESENT-80 decryption unit.

The decryption unit of PRESENT is very similar to the encryption. The decryption data path is presented in Fig. 5. The first round of decryption requires the last round key of the encryption routine. For optimal performance we assume that this last round key is precomputed and available at the beginning of the decryption routine. The assumption is fair since we have to perform this step only once for multiple cipher texts.

We implemented both encryption and decryption functions in VHDL for the Spartan-III XC3S400 (Package FG456 with speed grade -5) FPGA core from Xilinx. We used Mentor Graphics ModelSimXE 6.2g for simulation purposes and Xilinx ISE v10.1.03 WebPACK for design synthesis.

Table 3 summarizes the performance figures for our implementations. All figures presented are from Post Place & Route Timing Report. To achieve optimal results both Synthesis and Place & Route Effort properties were set to High and Place & Route Extra Effort was set to Continue on Impossible.

Numerous FPGA implementations of AES block cipher exist. Some of them are tuned to maximize data throughput, whereas others were designed for optimization of area requirements and power consumption. There are also block ciphers that were designed specifically for hardware (SEA [8]) or even FPGA (ICEBERG [9]) applications. We compare our PRESENT implementation with different existent FPGA implementations of those ciphers. In some cases it is hard to make a fair comparison, so additional information on implementation platform and boundary conditions is provided. Table 4 shows the results. Our results show that in the field of low-cost FPGA cores, PRESENT offers both the smallest area requirement and highest hardware efficiency compared to AES as well as ICEBERG and SEA implementations. Note, that our implementation does not require any Block RAM units while most AES implementations do. For this matter we show the total equivalent slice count for each implementation to highlight the real area requirements.

The speed grade of the Spartan devices has significant

Table 3. Performance results for encryption and decryption of one data block with PRESENT for different key sizes and S-Box implementation techniques.

Key size	enc/dec	S-box w/	#LUTs	#FFs	Total equiv. Slices	Max. freq. (MHz)	#CLK cycles	Throughput (Mbps)	Efficiency (Mbps/#Slices)
80	enc	espresso	253	152	176	258	32	516	2.93
		LUT	350	154	202	240	32	480	2.38
	dec	espresso	328	154	197	240	32	480	2.44
		LUT	328	154	197	238	32	476	2.42
128	enc	espresso	299	200	202	250	32	500	2.48
		LUT	300	200	202	254	32	508	2.51
	dec	espresso	366	202	221	239	32	478	2.16
		LUT	366	202	221	239	32	478	2.16

Table 4. Performance comparison of PRESENT, AES, ICEBERG and SEA FPGA implementations.

Algorithm	Block Size	Device	Max. freq. (MHz)	Throughput (Mbps)	Total equiv. Slices	Efficiency (Mbps/Slice)
PRESENT-128	64	Spartan-III XCS400-5	254	508	202	2.51
PRESENT-80, [5]	64	Spartan-IIIE XC3S500	-	-	271	-
ICEBERG, [9]	64	Virtex-II	-	1016	631	1.61
$SEA_{126,7}$, [8]	126	Virtex-II XC2V4000	145	156	424	0.368
AES, [10]	128	Spartan-II XC2S30-6	60	166	522	0.32
AES, [11]	128	Spartan-III XC3S2000-5	196.1	25,107	17,425	1.44
AES, [11]	128	Spartan-II XC2S15-6	67	2.2	264	0.01
AES, [12]	128	Spartan-II XC2V40-6	123	358	1214	0.29
AES, [13]	128	Spartan-III	150	1700	1800	0.9

impact on the max. frequency of the cipher. Switching from speed grade 4 to speed grade 5 gave us up to 20% max. frequency increase. There are also different packages for each device platform with varying pinout count. Those facts make a fair inter-platform comparison even harder. Hence, for comparison's sake we picked only AES implementations on Spartan devices with speed grade 5 and above. For ICEBERG and SEA there are only Virtex-II implementations available. We also chose PRESENT version with 128 bit key size for the same reason even though the implementation figures for PRESENT-80 are more encouraging.

4. CONCLUSION

In this work we have presented a design space exploration for FPGA implementation of the lightweight block cipher PRESENT. Though PRESENT was designed with a minimal hardware footprint in mind, i.e. targeted for low-cost devices such as RFIDs, our results also highlight that PRESENT is well suited for high-speed and high-throughput applications. Especially its hardware efficiency, i.e. the throughput per slice, is noteworthy. Furthermore, interestingly the old-fashioned Boolean minimization tool espresso resulted in one case in an implementation that was significantly smaller than a standard LUT based implementation.

5. REFERENCES

[1] A. Bogdanov, G. Leander, L. R. Knudsen, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT - An Ultra-Lightweight Block Cipher," in *Proceedings of CHES 2007*, ser. LNCS, no. 4727. Springer-Verlag, 2007, pp. 450 – 466. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74735-2_31

[2] M. Wang, " Differential Cryptanalysis of Reduced-Round PRESENT," in *Proceedings of AFRICACRYPT 2008*, ser. LNCS, no. 5023. Springer-Verlag, 2008, pp. 40 – 49. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-68164-9_4

[3] C. Rolfes, A. Poschmann, G. Leander, and C. Paar, "Ultra-lightweight implementations for smart devices - security for 1000 gate equivalents," in *Proceedings of the 8th Smart Card Research and Advanced Application IFIP Conference – CARDIS 2008*, ser. LNCS, vol. 5189. Springer-Verlag, 2008, pp. 89–103.

[4] P. Grabher, J. Großschädl, and D. Page, "Light-weight instruction set extensions for bit-sliced cryptography," in *Cryptographic Hardware and Embedded Systems*

— *CHES 2008*. Springer Verlag LNCS 5154, August 2008, pp. 331–345. [Online]. Available: http://www.cs.bris.ac.uk/Publications/Papers/2000890.pdf

[5] X. Guo, Z. Chen, and P. Schaumont, "Energy and Performance Evaluation of an FPGA-Based SoC Platform with AES and PRESENT Coprocessors," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, ser. Lecture Notes in Computer Science, vol. 5114. Springer-Verlag, 2008, pp. 106–115.

[6] X. Inc., "Spartan-3 FPGA Family Data Sheet," available online via http://www.xilinx.com, June 2008.

[7] N.A., "Espresso," available online via http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/index.htm, November 1994.

[8] F. Macé, F.-X. Standaert, and J.-J. Quisquater, "FPGA implementation(s) of a scalable encryption algorithm," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 2, pp. 212–216, 2008.

[9] F.-X. Standaert, G. Piret, G. Rouvroy, and J.-J. Quisquater, "FPGA implementations of the ICEBERG block cipher," *Integration*, vol. 40, no. 1, pp. 20–27, 2007.

[10] P. Chodowiec and K. Gaj, "Very Compact FPGA Implementation of the AES Algorithm," in *Proceedings of CHES 2003*, 2003, pp. 319–333.

[11] T. Good and M. Benaissa, "AES on FPGA from the Fastest to the Smallest," in *Proceedings of CHES 2005*, 2005, pp. 427–440.

[12] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat, "Compact and Efficient Encryption/Decryption Module for FPGA Implementation of the AES Rijndael Very Well Suited for Small Embedded Applications," in *ITCC (2)*, 2004, pp. 583–587.

[13] P. Bulens, F.-X. Standaert, J.-J. Quisquater, P. Pellegrin, and G. Rouvroy, "Implementation of the AES-128 on Virtex-5 FPGAs," in *AFRICACRYPT*, 2008, pp. 16–26.