Technology-agnostic power optimization for AES block cipher

Kais Chibani, Adrien Facon, Sylvain Guilley, Youssef Souissi

Technology-agnostic power optimization for AES block cipher

Kais Chibani*, Adrien Facon*,†, Sylvain Guilley*,†,‡ and Youssef Souissi*

* Secure-IC S.A.S † École Normale Supérieure ‡ TELECOM ParisTech

***Abstract*—On the one hand, IoT applications require low-power consumption. On the other hand, they also need to embed strong cryptographic algorithms to protect the data they manipulate. We address this issue in the context of the AES block cipher. The challenge is to design an AES module consuming as little as possible, mostly by leveraging on the architecture. This paper presents two contributions. First of all, we present design guidelines for low-power AES, including lazy dataflow, glitch reduction techniques in combinational logic and algebraic simplification of diffusion operations. These optimizations allow to divide by two the power consumption of the AES module. Second, we show a methodology to improve the power consumption using a high-level technology-independent power and security evaluation tool (Virtualyzr).**

Key words: IoT, low-power, AES architecture, high-level power optimization, LORA, BLE

I. Introduction

Recent advances in technologies of wireless communication, sensors, and embedded microprocessors have laid a solid foundation for the emerging pradigm of Internet of Things (IoT). It is a quickly growing market, driven by new applications such as smart devices (e.g., sensor networks, RFID tags, smart bulbs, gadgets, etc).

Low-power devices are required as they will typically be deployed for years without being connected to a power source. Some leverage on energy scavenging, but others such as battery-powered systems, shall live with limited energy. At the same time, the continuous development of IoT raises concerns about the security which is, nowadays, crucial and must no longer be seen as an additional feature for many applications. Consequently, low power devices shall communicate in a secure way. Transmission of information, especially the uplink, consumes a lot of energy when it is wireless. Moreover, this information shall be protected, hence cryptography is required.

Software implementations of cryptographic functions reduce the overall throughput and increase power consumption. Therefore, implementing these functions in hardware can be used to optimize throughput and power consumption [2] and to achieve good security, power, and area trade-offs. In IoT applications, the Advanced Encryption Standard (AES) [9] is a core security element used in the current IoT proposals such as LoraWan [8] and Sigfox [11]. The same remark applies to Bluetooth Low Energy (BLE) technology which already uses the AES-GCM cipher to provide data encryption and integrity over the wireless link. Moreover, AES is a conventional block cipher primitive which is currently adopted by standard Internet security protocol, such as IPSec and TLS. Therefore, we focus our analysis on AES, even though this block cipher is not the most lightweight. In this respect, we propose a methodology based on a set of architectural power optimizations that can be used to implement low-power AES crypto module. Our main goal is to optimize the architecture of data encryption on custom silicon (ASIC) libraries with energy-efficiency as a target. We show also a new method to improve the power consumption of several architectures independently of the target technology.

The rest of the paper is structured as follows. Section II describes the methodology to optimize the power by applying various activity minimization techniques. Implementation results will be discussed in Section III. The conclusion and perspectives are in Section IV.

II. High-level Power Optimizations

In this section, we explain how to optimize the power at a high-level description. We do not address high-level power estimation, which is a different topic. Neither do we address technology-level power improvements, such as sub-/near-threshold circuit techniques [12].

A. Considerations for power evaluation

Power can be static, caused by leakage, or dynamic, caused by switching. Switching activity is crucial because dynamic power is, after all, proportional to the value of the toggle count switching activity in the design. In general, the toggle count equals the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions of a design object (for example, a net, pin, or port) per unit of time.

The majority of the power in digital systems is dissipated dynamically, i.e., static leakage is ignored and dynamic power is estimated as the number of gates toggles. This means that each gate which toggles consumes the same amount of power. Obviously, those two assumptions are approximations of the truth. Indeed, static power does exist, and also contributes to the leakage. Besides, dynamic power is indeed increasing when the number of toggling gates increase, however, each gate has its own leaking profile. One should weight each gate independently from the others. This is why we focus on reducing dynamic power consumption. Several factors can have a simultaneous impact on the power consumption in AES such as the chosen operation mode (e.g., ECB, GCM, etc.) or the use of techniques for improving area efficiency (e.g., the Common Subexpression Elimination (CSE) algorithms adopted to eliminate the redundant gates). By investigating the power consumption of each primitive component in AES, it appeared that the key scheduling and the SubBytes functions consume much of the total as shown in [1]. Thus, the toggle count parameter and thereby the power dissipation extremely depends on the approaches used to implement these functions.

B. Low-Power Architecture

The fundamental method to reduce the power is to size the AES to its strict minimum. Therefore, all power-hungry features are banned. The encryption AES core we propose in this context is restricted to the following features: 128-bit key, ECB mode of operation, no countermeasure against physical attacks and the round key is computed in advance (i.e., the key is already unrolled hence no energy is lost to compute it).

The AES is operated column-wise: one round is scheduled in four clock cycles. Therefore, 128-bit encryption is performed in 40 clock cycles. The property of the proposed architecture is that each gate which toggles must contribute to advance the computation in such a way that total toggle count is greatly minimized. This will be our metric for improvement over classical architecture. Two strategies are leveraged to minimize the energy consumption at architectural level:

At state-register level, keep all inactive registers in enable; in practice, only four bytes out of the sixteen making up the datapath are updated at each clock operation.

At combinational logic level, the number of dynamic hazards (i.e., glitches) should be reduced to a minimum since they impact directly the number of toggles during execution. To this end, we design the AES to perform AddRoundKey transformation (i.e., XOR operations) as close as possible to the registers. In addition, the SubBytes and MixColumns transformations are combined together into a single step, nicknamed T-box [4], using a logic style which minimizes glitches. The AES T-box based algorithm was first proposed for a software implementation using 32-bit processor [5]. Then, it was adapted for hardware implementations [6].

The architectural datapath block of this architecture is shown in Fig. 1. In the following, we list optimizations which have been performed on the basis of the proposed architecture.

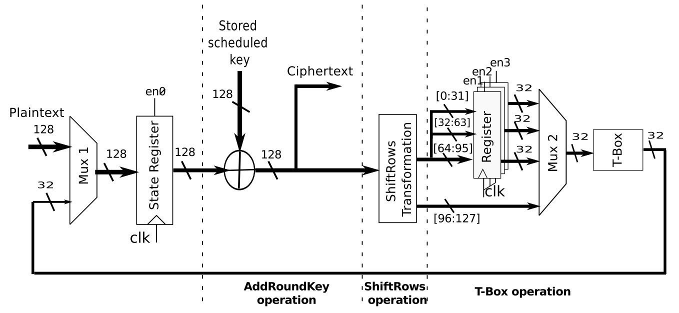

Figure 1. Architectural data path block for AES encryption

*1) **Keep unused registers in enable state**:* Each of the 16 bytes of the state are kept in enable, except with the four which are written to. Those registers are not the same as the original registers of the column, owing to ShiftRows. This strategy is termed "lazy datapath", as only those register bits to be updated indeed have their values changed; the rest of the datapath remains still. We construct column by column the new state. A scratch-pad register of 3×32 bits (96 bits) is required, as shown in Fig. 1, but this extra hardware allows to save some more power.

*2) **Reduce glitches from the combinational logic**:* As illustrated in Fig. 1, placing the flip-flops containing the AES encryption state just before XOR gates (notoriously amenable to glitching) minimizes the chance the signal arrival times are different between two inputs of each XOR gate. The T-box approach allows for power savings, as it consists in a pre-computation. This transformation is based on four T-tables T_0, T_1, T_2, T_3 which are of the size of 8×32 bits. An output column of this transform equals:

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = T_0[a_0] \oplus T_1[a_1] \oplus T_2[a_2] \oplus T_3[a_3], \text{ where:}$$

$$T_0[a_0] = \begin{pmatrix} 2S(a_0) \\ S(a_0) \\ S(a_0) \\ 2S(a_0) \end{pmatrix} \qquad T_1[a_1] = \begin{pmatrix} 3S(a_1) \\ 2S(a_1) \\ S(a_1) \\ S(a_1) \end{pmatrix}$$

$$T_2[a_2] = \begin{pmatrix} S(a_2) \\ 3S(a_2) \\ 2S(a_2) \\ S(a_2) \end{pmatrix} \qquad T_3[a_3] = \begin{pmatrix} S(a_3) \\ S(a_3) \\ 3S(a_3) \\ 2S(a_3) \end{pmatrix}$$

and where the bytes a_i, for $i \in [0..3]$, represent the input of T-box module as illustrated in Fig. 1. Obviously, T-box software implementations manipulate $3S(.), S(.), S(.), 2S(.))$, but for hardware, we need simply the concatenation of $(S(.), 2S(.), 3S(.))$, where S is the $8 \to 8$ S-box. The last round is special, since there is no MixColumns. Hence only output $S(.)$ of the T-box is used, 4 times. However, the T-box must be implemented in a glitch-less manner. Bertoni et al. propose in [3] an interesting method to implement AES S-box on ASIC using a synthesis methodology composed of three main blocks: a multi-level decoder followed by a permutation block, which does the S-Box computation, and finally an encoder. A decoder is used to convert one 8-bit input into 256 lines of one bit. The permutation block is used to rewire the output lines of decoder stage. It infers no logic and involves only routing of the signals. The structure of this method is shown in Fig. 2.

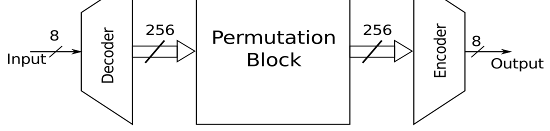

Figure 2. The structure of the S-box in [3]

Only one bit of the decoder output and one bit of the encoder input are in logic ’1’ level at the same time. The others are in logic ’0’ level. When an arbitrary number of inputs switch, only a certain number of bits are activated (i.e., have some switching activity) between decoder, permutation and encoder blocks. The same decode-permute-encode scheme can be made to compute the remaining $(2S(.), 3S(.))$ needed by T-box module. In this case, each input byte of the T-box module requires one decoder, three permutation blocks and three encoders to obtain $(S(.), 2S(.), 3S(.))$.

*3) **Algebraic optimization**:* The above mentioned multiplication of permutation blocks and encoders per T-box input byte increases substantially the area of the AES core and causes more toggles leading to additional power consumption. In order to limit these costs, we optimize the scheme presented in Fig. 2 as shown in Fig. 3. The main goal of this optimization is to compute $(2S(.))$ based on $(S(.))$ and then, deduct $(3S(.))$ from $(S(.), 2S(.))$, since $3S(.) = S(.) \oplus 2S(.)$. The added 11 XOR gates do glitch, but they consume less power than the required $256 \rightarrow 8$ encoders.

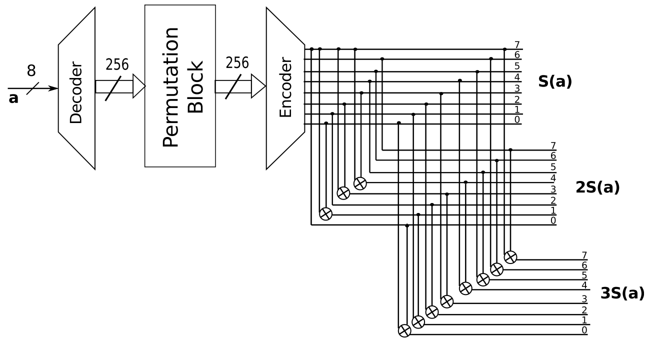

Figure 3. Optimization of the S, 2S, 3S computation for T-box module

*4) **Finite State Machine by using one-hot encoding**:* The control part of the AES core can be optimized too. The one-hot encoding is a greedy encoding for finite state machines (FSMs). Indeed, only one of the bits of the state variable is "1" (hot) for any given state. All the other bits are "0". Hamming distance of this techniques is 2. As a result, the state machine is already decoded and determined simply by finding out which flip-flop is active. This encoding technique reduces the width of the combinatorial logic and leads to a minimum number of transitions between states, hence to reach a local minimum in terms of power consumption of the control part.

*5) **Applying Clock-gating**:* In synchronous digital circuits, the clock is responsible for significant part of dynamic power consumption, up to 40% according to [7]. Clock continuously consumes power because it induces dissipation while changing values at the capacitive input of flip-flops, even if those are disabled. Reduction of clock switching by clock gating is thus desirable. Clock gating can be inserted to prevent state register clock from toggling since only four bytes out of sixteen are updated at each clock cycle.

III. Implementation Results

Experiments have been carried out on 5 various implementations whose the specifications are given in Tab. I. In Imp_2, the T-box module (i.e., the computation of $(S(.), 2S(.), 3S(.)))$ is implemented using the Galois Field (GF) combinational logic whose synthesis is delegated to the silicon compiler. Imp_5 includes clock-gating system during synthesis process at gate-level. These implementations are synthesized on two CMOS technologies. The first one is 65 nm, 1.0 V. The second one is 28 nm, 0.9 V. The main goal is to see the impact of the proposed optimizations presented in Sec. II.

Table I
Studied implementations

	State registers	T-Box	Clock-gating
Imp_1	After ARK*	LUTs	No
Imp_2	After ARK*	GF(128)	No
Imp_3	Before ARK*	scheme in Fig. 2	No
Imp_4	Before ARK*	scheme in Fig. 3	No
Imp_5	Before ARK*	scheme in Fig. 3	Yes

*AddRoundKey.

The area expressed as the number of gate equivalents and the leakage/dynamic power consumption under typical operating conditions (25°C) are presented in Tab. II. The power consumption estimations are based on the cell information of the target ASIC library and switching activities, which are captured with gate-level simulations using random test vectors as the stimuli. It is a very accurate method because the effects of glitches can be reflected during timing simulations and thus in the power estimation results.

Clearly, we can observe that Imp_5 which regroups all optimization techniques is the best low-power implementation. In fact, for both evaluated ASIC technologies, this implementation offers the minimal power consumption with a very low standard deviation, that demonstrates a weak variation with respect to the various sets we used as test. We can also remark that the new positioning of registers and the use of decode-permute-encode schemes have actively contributed to reduce power consumption. The power dissipation inside combinational logic is progressively minimized thanks to optimizations used on Imp_3, Imp_4 and Imp_5 which are successful in reducing glitches. On the other side, classical implementations (i.e., Imp_1 and Imp_2) seem to be the most power consuming due to their high switching activity. The consumption of the combinational gates has been reduced way below that of the registers. Therefore, only marginal gain is to be expected from the further decrease of combinational gates including removing more glitches with respect to sequential logic which cannot be optimized because it is necessary in the computation of AES to keep a 128-bit state register. Nevertheless, we notice the non-negligible effect of clock-gating on reducing the consumption of registers in Imp_5.

The Virtualyzr tool [10] estimates power consumption based on toggle count. Using this tool, we compute the total toggle count for all implementations which have been synthesized on a generic library in order to determine the effect of optimizations independently of the target technology. The Virtualyzr generic library is composed of classical standard gates (e.g., DFF, INV, AND, OR, XOR). Fig. 4 summarizes, for each implementation, the average dynamic power (Tab. II) according to the number of toggles obtained after gate-level simulations based on the generic library. This figure shows that Imp_5 leads to a minimum number of toggles. This justifies the positive impact of our optimizations in terms of toggles reduction and again confirms the linearity between

Table II
IMPLEMENTATION RESULTS AND COMPARISON (65 NM CMOS TECHNOLOGY, 1.0 V / 28 NM CMOS TECHNOLOGY, 0.9 V)

Implementation	**Average dynamic power [μW/MHz]**	**Deviation standard of dynamic power [μW/MHz]**	**Combinational power [μW/MHz]**	**Sequential power [μW/MHz]**	**Leakage power [μW/MHz]**	**Area logic [kGE]**
65 nm CMOS technology, 1.0 V						
Imp_1	8,471	0,174	6,362	2,109	0,228	8,451
Imp_2	13,251	0,097	10,889	2,362	0,133	4,339
Imp_3	4,388	0,082	2,539	1,849	0,127	7,260
Imp_4	4,146	0,057	2,319	1,827	0,089	5,479
Imp_5	3,426	0,044	1,952	1,474	0,088	5,384
28 nm CMOS technology, 0.9 V						
Imp_1	1,524	0,018	1,003	0,521	0,508	8,344
Imp_2	2,444	0,025	1,856	0,588	0,209	4,350
Imp_3	1,065	0,009	0,530	0,535	0,351	7,034
Imp_4	0,960	0,007	0,425	0,535	0,237	5,371
Imp_5	0,825	0,009	0,438	0,387	0,241	5,235

the dynamic power consumption and the switching activity. Therefore, the methodology based on the computation of toggles is useful to quickly estimate and compare the power consumption of several implementations early in the design flow, especially that the generic library does not depend on any specific technology.

The best low-area implementation is Imp_2 since mathematical theorems over Galois fields are possible and various methods for constructing compact inversion circuits can be used. However, the difference is not very large compared to Imp_5 ensuring less power dissipation and a lower toggle coverage, i.e., around 62.5% versus 96% in case of Imp_2.

IV. CONCLUSION

We propose a set of architectural optimizations which allow to optimize considerably the power consumption of an AES implementation by reducing the number of toggles. The variant of AES with GF S-boxes show that sometimes area and power optimizations contradict. However, our careful high-level optimizations allowed us to save both on the power ($\approx 2.5\times$ less) and on the area (reduced by $\approx 40\%$).

A simulation-based framework (namely, the Virtualyzr tool) allows to easily and rapidly improve the power of an implementation based on the toggle count metric from a technology-independent library (i.e., synthesize and simulate). This method is validated on two technologies and enables more interactive validation of code-level optimizations without resorting to the lengthy and costly classical power estimations from a technology (synthesize, extract parasitics, simulate and estimate power). The same framework can be reused to test the safety and the security of the AES design. Further work includes evaluation of the proposed low-power implementation against side channel attacks and comparison with state of the art lightweight symmetric cryptography implementations.

ACKNOWLEGMENTS

This work has benefited from a partial funding via TeamPlay (website at: https://teamplay-h2020.eu/), a project from European Union's Horizon2020 research and innovation program, under grant agreement N° 779882.

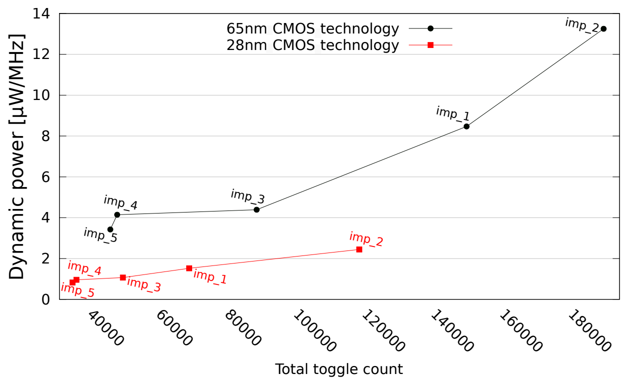

Figure 4. Dynamic power and total toggle count using the generic library

REFERENCES

[1] S. Banik, A. Bogdanov, and F. Regazzoni. Exploring energy efficiency of lightweight block ciphers. In *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, pages 178–194, 2015.
[2] L. Batina, A. Das, B. Ege, E. B. Kavun, N. Mentens, C. Paar, I. Verbauwhede, and T. Yalçin. Dietary Recommendations for Lightweight Block Ciphers: Power, Energy and Area Analysis of Recently Developed Architectures. In *RFIDSec*, 2013.
[3] G. Bertoni, M. Macchetti, L. Negri, and P. Fragneto. Power-efficient ASIC Synthesis of Cryptographic Sboxes. In *Proceedings of the 14th ACM Great Lakes Symposium on VLSI*, GLSVLSI '04, pages 277–281.
[4] P. Bulens, F.-X. Standaert, J.-J. Quisquater, P. Pellegrin, and G. Rouvroy. Implementation of the AES-128 on Virtex-5 FPGA. In *Progress in Cryptology – AFRICACRYPT 2008*. Springer Berlin Heidelberg, 2008.
[5] J. Daemen and V. Rijmen. AES Proposal: Rijndael. https://www.nist.gov/programs-projects/lightweight-cryptography, 1999.
[6] V. Fischer and M. Drutarovský. Two Methods of Rijndael Implementation in Reconfigurable Hardware. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 77–92. Springer Berlin Heidelberg.
[7] S. Jeff. Analyzing Clock Trees, SNUG Boston 2005.
[8] Lora Alliance. LoraWan Specification, 2015.
[9] N. I. of Standards and Technology. Advanced encryption standard. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.
[10] Secure-IC. http://www.secure-ic.com/solutions/virtualyzr/.
[11] Sigfox Technology. http://sigfox.com.
[12] W. Zhao, Y. Ha, and M. Alioto. AES architectures for minimum-energy operation and silicon demonstration in 65nm with lowest energy per encryption.