A Low-Power and Performance-Efficient Co-design Implementation of AES Encoder

Ricardo P. Robaina[1], **Fabio L. Ramos**[1], **Bruno S. Neves**[1]

[1]Universidade Federal do Pampa (Unipampa)

ricardorobaina11@gmail.com, {fabioramos,brunoneves}@unipampa.edu.br

Abstract. *Encryption algorithms are required for a higher safety degree when considering, for instance, data exchange in computer networks. AES (Advanced Encryption Standard) is an alternative to cipher data and therefore increase the security of sensitive data communication. A possible approach for a good trade-off of this application regarding processing time, area, power, and time-to-market is to use a co-design fashion, where portions of the algorithm run in a processor, while other parts operate in hardware accelerators. The proposed work, name LP-SB, is a low-power proposal for an AES co-design, where the power-critical step of the algorithm (i.e., SubBytes) is ported to an accelerator and, therefore, two low-power techniques are inserted for that block. Power results running LB-SB on gate-level netlist resulted in around 13% of power reduction when compared with the same block without the low-power techniques. Moreover, the AES co-design proposed achieves 44-times more performance when compared to the software-only solution, which is within the expected improvement when compared to related AES co-design works.*

1. Introduction

The intercommunication of information between several organizations and individuals through the computer networks requires efficient security mechanisms, considering the current environment of data exchange. Therefore, throughout the history of information security, several cryptographic algorithms exist for data protection in different categories; examples of such algorithms are DES, RC4, RSA and AES [Ravi et al. 2002].

The AES (Advanced Encryption Standard) algorithm [NIST 2001] is an alternative for the purpose above, which adoption by the U.S. government makes it an important matter of research, being also the most widely used by the industry and by the academic environment [Wang et al. 2015], [Lin et al. 2009], [C. T. O. Otero and Manohar 2015], [Drimer et al. 2008], [Mangard et al. 2003]. In the current context, the AES receives particular attention, not only for providing a high level of data security, but also for its high efficiency during the process of encryption/decryption of the information, and for its low memory consumption, which enables its massive use in mobile devices.

Hence, there are three different implementation types one may follow to accomplish an AES design: (i) Full Software Implementation: when all computations run at a general purpose processor, providing fast development and update, at the cost of slow execution and high CPU overhead. (ii) Full Hardware Implementation: in this fashion, the whole algorithm executes at a specific hardware design, which requires

much longer development time compared to the software-only version, but with the advantage of higher performance. (iii) Hardware/Software Co-design Implementation: in this hybrid approach, a part of algorithm runs at a general/embedded purpose processor, whereas the other most critical portions of the application run at specific hardware accelerator architectures, usually implemented within an FPGA. In many cases, this approach produces a better tradeoff between design requirements such as execution time, area, power consumption, and system development time [Mangard et al. 2003].

This work presents a low-power approach for a co-design architecture for an AES encoder, name LP-SB, where low-power techniques were inserted into the chosen AES step to be run in the accelerator, achieving around 13% of power dissipation reduction. Moreover, our co-design of AES encoder accomplishes circa 44 times more performance when compared to the software-only version.

The paper organization is as follows: in the next section, a review of the AES encoder algorithm is presented. The related works appear in Section III. Section IV reports the methodology used for the present work. The implementation of the low-power AES encoder co-design is presented in the Section V. Results and comparisons are shown in Section VI, whereas Section VII concludes the paper.

2. AES Algorithm

The Advanced Encryption Standard (AES), by National Institute of Standard and Technology (NIST), is a symmetric 128 bits block cipher that uses a key length of 128, 192 or 256 bits [Ravi et al. 2002]. The AES is a round-based algorithm, where the number of rounds depends on the key length.

As shown in Figure 1, in each round, the AES algorithm performs four macro operations on the input block (named BlockIn in Figure 1), where they are called AddRoundKey (AK), SubBytes (SB), MixCollumns (MC) and ShiftRows (SR). The data input block that is modified by these operations has 128 bits organized in a 4x4 matrix of bytes. The key used in each round (i.e., the round-key) is generated from an initial key expansion step (named ExpKey).

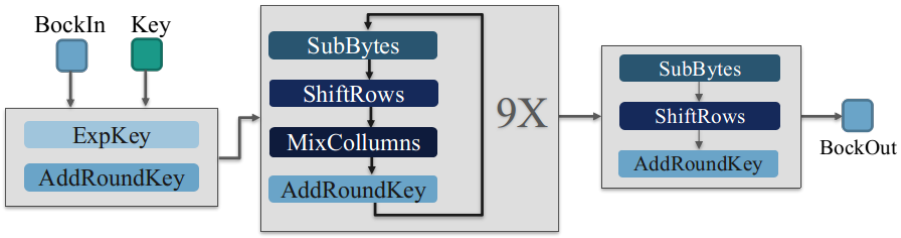

Figure 1. AES encoder operations flow

After the key expansion step, the algorithm executes the AddRoundKey and, in the next nine rounds, the four AES operations are executed repeated times. In the last round, AES executes all its operations, except MixCollumns.

A brief description of AES steps is as follows:

- AddRoundKey (AK): executes a simple bitwise XOR between the round key and the encrypted data block (named state matrix).

- SubBytes (SB): executes a substitution of the bytes in the state matrix by bytes of an auxiliary matrix, named S-Box. The calculation of the position of the element read from S-Box occurs by dividing the current byte from the state matrix into two nibbles. The most significant nibble indicates the line of the element in S-Box, whereas the least significant nibble indicates the column of this element. Figure 2 illustrates this operation.
- ShiftRows (SR): this operation rotates the bytes in each row of the state matrix for a specific number of positions according to row number. Thus, the row number zero remains unchanged (zero rotation is performed on this row). After that, each byte on the row number one jumps one column to the left of the current one. Similarly, on the row number two, each byte jumps two columns to the left and, on the row number three, each byte jumps three columns to the left.
- MixColumns (MC): the computation performed is equivalent to a multiplication of each column of the state matrix by a matrix of constants. In this paper, however, it was used an optimized version of MC operation, in which the matrix multiplication replacement occurs by successive execution of shifts and XORs operations, as better described in [8].

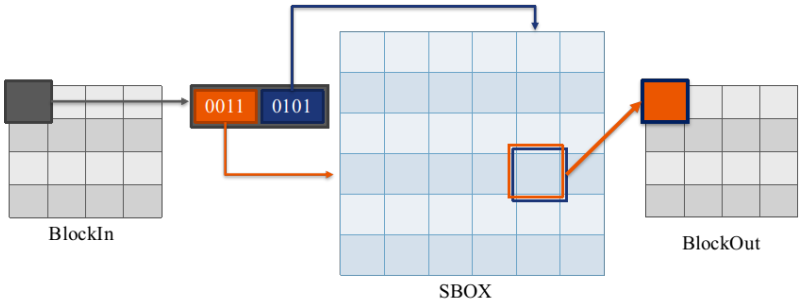

Figure 2. S-Box addressing flow

3. AES Co-Design related works

There are some works found in the literature, which refer to an AES co-design approach. This section is intended to discuss these works. In [Wang et al. 2015] the authors propose a co-design for AES in which the MC operation runs in hardware, while the rest of the AES encoder is executed in a MIPS-based processor, using different multi-cores approach to assess the speed-up achieved by the proposed arranges. The results come from RTL simulation of the proposed scenarios, where a 2x4-core arrange achieves the best speed-up of around 9.78 times when compared to the AES encoder running solely on a single processor.

The work of [Lin et al. 2009] also assesses various scenarios of AES co-design, where combinations of AK+SR and MC, along which different S-Box approaches, are processed in specific hardware accelerators, and the rest of the AES encoder runs on a Nios II soft-core processor for Cyclone FPGA. The combination AK+SR+MC achieves the best result, along with a single S-Box table, reaching a speed-up of 67.7 times compared to the baseline AES running entirely on the Nios II processor.

In [C. T. O. Otero and Manohar 2015], the authors have analyzed all possible combinations of hardware-software co-design for AK, SB and MC operations. For

instance, the authors have made different AES co-designs, putting either one of the AES operations to run on an accelerator, two of the cited operations, or finally all of them running at the accelerator. The AES chosen operations to work at software used two hardcore-processors: ULSNAP and MSP430. ASICs were made for all the AES operations combinations on the accelerators, and run along one of the cited processors. AK + SB + MC working in the accelerators achieve the best result, whereas the rest of the AES encoder operates in the MSP430 processor: a speed-up of around 29-times compared to software-only AES solution. Moreover, in [C. T. O. Otero and Manohar 2015], a power-gating approach is proposed, to save energy during idle moments an AES encoder would undergo in a WSN (Wireless Sensor Network) application. Nevertheless, the proposition seems to be especially beneficial for a full-hardware AES solution, and only to save static power dissipation, as power-gating application is for that purpose.

4. Methodology

This section presents the methodology used for the development of this research, which Figure 3 summarizes. The first step of this flow consists in configuring an embedded softcore processor with the AES algorithm. After this initial configuration, the execution time evaluation of the AES four operations occurred, to gather representative information to execute the next step of the methodology: the hardware/software partitioning of the AES.

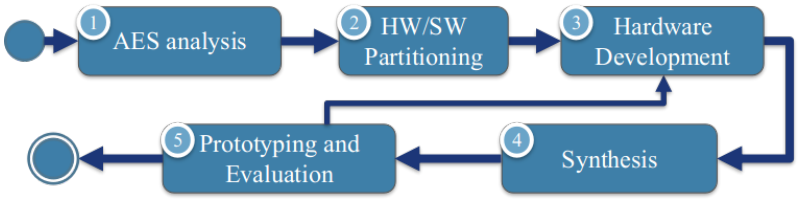

Figure 3. Methodology scheme for the proposed AES co-design

Based on the analysis of the execution time per operation, the second step of the flow consists in detecting the dominant set of AES operations that respond to the majority of the application execution time. If more than one operation corresponds to the majority of AES execution time, the estimation of additional area occupied by each AES step is an important point of the flow, to prevent the final size of the hardware to be similar to the one of a full hardware implementation of the algorithm. High-Level Synthesis (HLS) tool utilization may apply for that purpose, such as the HLS compiler made available by Altera [Altera a].

The third step consists of a manual development of the AES operations, pointed out by partitioning step, to run in hardware (i.e., in the FPGA).

In the fourth step, it is implemented the necessary interconnections to link all the dedicated hardware with the processor. For this purpose, the Nios II soft-core processor tool suite [Altera b] was the platform chosen to implement the co-design. The Nios II executes the software components, whereas the hardware parts connections to this processor happens using Qsys, which is suitable for rapid system IP integration and prototyping [Altera c].

After the coupling of the dedicated hardware blocks, the next step of the flow is to make the co-synthesis of the entire system to analyze area and frequency of the hardware. In sequence, occurs the system prototyping and execution on FPGA, to assess the execution time for the co-design solution. A gate-level evaluation of the power consumption was made on the developed component.

5. Co-Design of the low-power AES encoder

As defined in the methodology, an analysis of the software version of the algorithm was made to identify the main bottleneck of the AES encoder. Figure 4 presents a graph of the impact of each operation in the execution of the algorithm regarding of the percentage of the time.

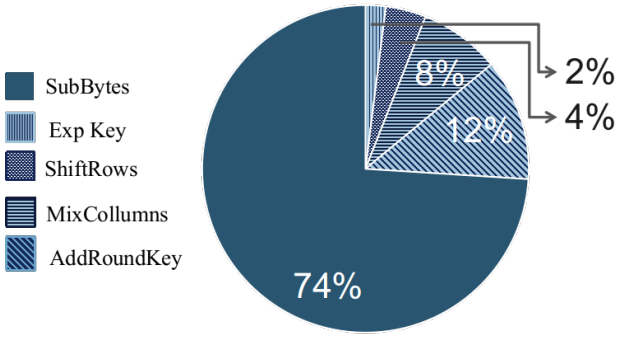

Figure 4. AES operation impact in the software-only execution time

As it is possible to notice, the SubBytes (SB) step corresponds to the major impact on execution time assessed by our analysis (i.e., 74% of the total execution time). Therefore, according to the partitioning step of our methodology, SB was the only chosen AES operation to run in the hardware part of our proposed co-design. The remainder of AES algorithm operates in the processor (i.e., software). Furthermore, as related in [C. T. O. Otero and Manohar 2015], the SB operation is the most critical regarding power dissipation when done in hardware. Therefore, it has the highest power reduction potential. Figure 5 shows the conceptual illustration of our AES co-design choice.

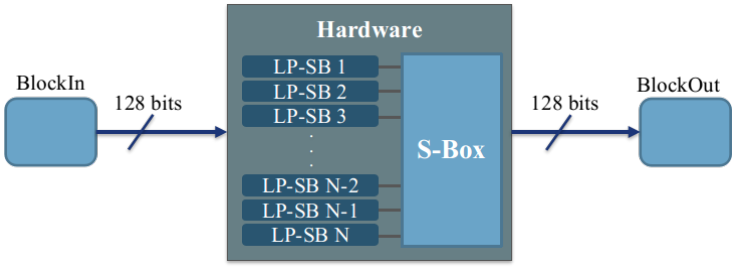

Figure 5. Full LB-SB accelerator architecture

An S-Box is required for the SB step, as already mentioned. Therefore, a study was made of possible S-Box implementation forms. One solution would be to implement the S-Box using FPGA memory (i.e., memory bits). This solution would have less impact on the FPGA logical elements, but would not exploit the parallelism that naturally exists within the operation. Another possibility is to implement the S-Box with a ROM. Hence,

there will be a considerable impact on the component area. However, it is possible to explore the parallelism of the operation. It was chosen to use the ROM on the development of our S-Box to exploit the parallelism and thus obtain an increase of performance that justifies the co-design our S-Box.

A combinational solution of the SB operation was the best choice for implementation since there is no dependency among blocks of encrypted data. Therefore, the entire SB step executes in only one clock cycle. This choice was made to obtain a better result of the performance for the proposed co-design.

In our proposed architecture for the SB, named LP-SB, two power reduction techniques are used. The Operand Isolation technique [Correale 1995] is applied to the input signal of the architecture (i.e., the AND gate in Figure 6), preventing the architecture capacitances from changing state when the computation result of this step is not necessary. We expect to have this portion of the algorithm to be idle for some time since it has to wait for results coming from the CPU, which potentially will last longer to provide the data required for the SB step. Additionally, a Clock Gating [Wu et al. 2000] is applied to the output register, since the final output also will be necessary only at given moments. Therefore there is no need to keep the clock switching the whole time at these registers.

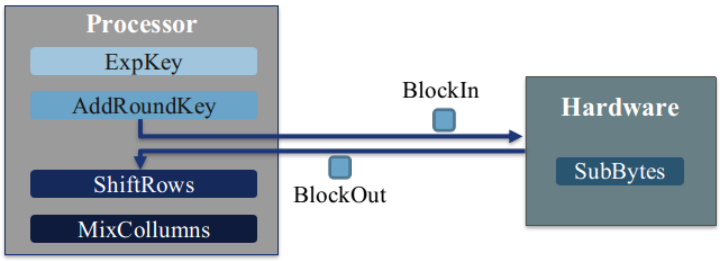

Figure 6. Proposed AES co-design partitioning

Figure 6 shows in details the LP-SB version architecture for one byte of the block. The byte is divided in the middle (i.e., two nibbles are generated). The value contained in the most significant nibble passes through a shift-left by four positions, i.e., a multiplication by 16. The addition of least significant nibble to the shift output occurs, and that is the reason why the S-Box address implementation in memory with contiguous address positions was chosen. Thus, the output signal receives the value whose address was previously calculated at the adder in Figure 6.

Moreover, the component as a whole is given by the concatenation of sixteen of these circuits in parallel, sharing only one S-Box implemented in a ROM memory. Figure 7 presents the full hardware component, where the variable N is equal to 16. One may notice that the use of the low-power techniques proposed is multiplied by the number of LP-SB instances used, which increase the potential of estimated power reduction for our design as a whole.

6. Results

The evaluation was done in a version of the AES encryption algorithm with 128-bits block size and input key. The tool used for synthesis is the Quartus II version 13.1, and the FPGA used in the evaluation of the execution time was the Cyclone III (EP3C25F324c),

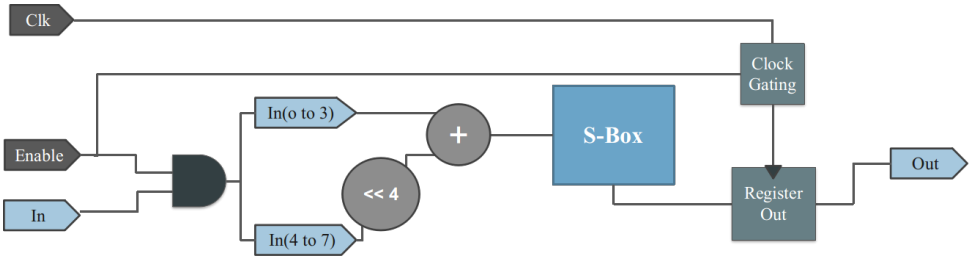

Figure 7. LB-SB architecture

in which the soft-core processor Nios II is instantiated. An enable signal is used to inform when LP-SB has finished its processing and is ready to send data for the rest of AES, and vice-versa.

An analysis to verify the area overhead when using the co-design approach is presented in Table 1. The co-design version, i.e., Nios II processor and LP-SB, generates an increase of 50.55% in the FPGA area compared to the software-only version. These results were obtained by synthesizing both versions for the chosen FPGA.

Estimated real stimuli were applied to our co-design, to evaluate power consumption. These stimuli derive from a NIST reference model [Dworkin 2001]. The values sampled were analyzed with the number of cycles that each operation lasts in the CPU. Figure 8 shows the time proportion that each stage occupies. Our analysis estimated that, in most of the time, the SB operation is idle, because it requires only one cycle, as already mentioned. This scenario justifies the use of power reduction techniques.

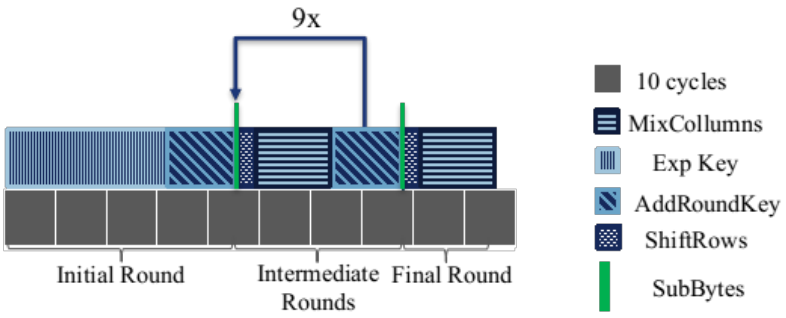

Figure 8. AES operations time estimation within LP-SB design

A gate-level power consumption evaluation was performed comparing LP-SB with a baseline version of SB in hardware, which is the same architecture of LP-SB, but without the insertion of the proposed low-power techniques, named BL-SB. Table 2 presents the results of dynamic power consumption obtained in the tests performed by encrypting a 128-bit block at a frequency of 50 MHz.

Table 1. Logic elements for AES designs

Design	Logical Elements
Full Software version	2352
Co-design version	3541

Table 2. Comparison between aes co-design versions

AES co-design	Max Frequency (MHZ)	Dinamic Power (mW)
BL-SB	100.06	16.76
LP-SB	115.27	14.61

Table 3. Execution times of aes approaches for different input block amounts

Version	1 Block	1K Blocks	10K Blocks
Full Software version	0.148 s	148.771 s	24.57 min
Co-design version	0.003 s	3.453 s	0.575 min

As estimated and expected, the SB stage is active in less than one-quarter of the total execution time of the AES encoder (i.e.only in 22% of the entire block time for one block). During the other 78% of the execution time, the remaining operations process the data that enter SB but are not required at that given moment, generating unnecessary switching in case of the non-application of the low-power techniques. As one may notice, there is a reduction of approximately 13% in the power consumption measured by the use of LP-SB.

Timing analysis was also performed for LP-SB and BL-SB. Table 2 also presents the maximum frequency results achieved by each one of them. LP-SB had a maximum frequency 15.2% higher than BL-SB maximum frequency, which is the opposite of the expected since LP-SB inserts more logic into the critical path of the architecture (i.e., an AND gate is used as Operand Isolation, and its application occurs for every input bit of the adder). The possible explanation is that a more efficient synthesis was done due the insertion of the AND into the design, constraining more logics at less space when compared to BL-SB.

We have submitted the same workload for the AES solution purely in software and our proposed AES co-design, to evaluate the performance increase by using the co-design approach. For both versions, the analyses were done running both AES co-design versions on the chosen FPGA device. Three analyses were made by varying the number of blocks to be coded by the AES algorithm. Three executions were performed for each block size. Table 3 shows the average runtime results for the tests run.

As the runtime data presented, the co-design version performs the same operation 44.15-times faster on average comparing to the software version. This result depicts a considerable increase in performance for the co-design version. Figure 9 shows a logarithmic scale graph of the mentioned results.

Table 4 presents some comparison among related works. The first comparison is regarding which AES operations execute in the hardware of the co-design approach. In [Lin et al. 2009] and [C. T. O. Otero and Manohar 2015], many different combinations of AES operations divisions between software and hardware were reported. Hence, we are considering only the best-case option proposed by the works above. The speed-up of LP-SB co-design is considerable and is within the gains achieved by others co-designs using the different AES co-design partitions. The method of validation and speed-up measurements vary according to each related work, where [Wang et al. 2015]

Table 4. Comparisons among related aes co-designs works

Designs	*[Wang et al. 2015]*	*[Lin et al. 2009]*	*[Otero et al. 2015]*	**LP-SB**
AES Operations in hardware	MC	AK + SR + MC	SB + AK+ MC	**SB**
Speedup	9.78X	67.7X	29X	**44.15X**
Validation Methods	RTL	FPGA	ASIC	**FPGA**

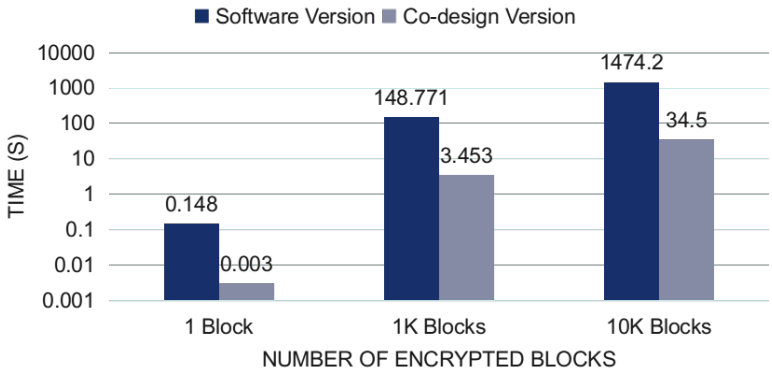

Figure 9. Execution time for different block amounts under AES

has used an RTL level-simulation for its results, our LP-SB and [Lin et al. 2009] used an FPGA platform with Nios II softcore processor, along with the hardware AES operations. In [C. T. O. Otero and Manohar 2015], the authors have used hard-core processors (ULSNAP and MSP430), along with ASIC versions of the chosen AES steps to be ported to the accelerators.

Finally, only LP-SB and [C. T. O. Otero and Manohar 2015] propose a low-power approach, where [C. T. O. Otero and Manohar 2015] utilizes Power-Gating for when the AES hardware operations are idle (i.e., reduction of static power consumption), with significant reduction especially for their AES hardware-only version. Our work implements both Operand Isolation and Clock Gating techniques for the critical power-consuming operation of AES (i.e., the SB step [C. T. O. Otero and Manohar 2015]), achieving significant reduction on dynamic power consumption for the AES co-design, but also is a potential solution for a full- hardware AES design.

7. Conclusion

This work has presented a low-power proposal for an AES co-design, considering the power-critical operation of the cipher algorithm, named LP-SB. Two low-power techniques were inserted into the LP-SB hardware block, to avoid unnecessary switching while the mentioned block has to wait for incoming data from the rest of AES, running on the software. Gate-level simulation results have achieved around 13% of dynamic power reduction when compared to the baseline version without the low-power techniques. The low-power proposal, even if done focusing on a co-design AES, can be further used on a full-hardware AES solution.

Finally, the designed AES co-design has accomplished around 44 times more performance when compared to the software-only AES version, which is within the estimated gain by using a co-design approach, as confirmed by AES co-design related works, using different AES operations for the hardware-software partition.

References

Altera. Intel HSL compiler. Available in: https://www.altera.com/products/design-software/high-level-design/intel-hls-compiler/overview.html.

Altera. Niosii processor. Available in: https://www.altera.com/products/processors/design-tools.html.

Altera. Platform designer (formerly qsys). Available in: https://www.altera.com/products/design-software/fpga-design/quartus-prime/features/qts-platform-designer.html.

C. T. O. Otero, J. T. and Manohar, R. (2015). Aes hardware-software co-design in wsn. In *21st IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 85–92.

Correale, A. (1995). Overview of the power minimization techniques in the ibm powerpc 4xx embedded controllers. In *International Symposium on Low Power Electronics Design (ISLPED)*, pages 75–80.

Drimer, S., Güneysu, T., and Paar, C. (2008). Dsps, brams, and a pinch of logic: Extended recipes for aes on fpgas. In *International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 99–108.

Dworkin, M. J. (2001). Recommendation for block cipher modes of operation: Methods and techniques. *NIST*.

Lin, J. K., Hsiao, C., and Jhan, C. H. (2009). Exploring hw/sw codesign of aes algorithm using custom instructions. In *IEEE 13th International Symposium on Consumers Electronics (ICSE)*, pages 192–195.

Mangard, S., Aigner, M., and Dominikus, S. (2003). A highly regular and scalable aes hardware architecture. In *IEEE Transactions On Computer*, volume 53, pages 483–491.

NIST (2001). Advanced encryption standard(aes). *FIPS PUBS*.

Ravi, S., Raghunathan, A., Potlapally, N., and Sankaradass, M. (2002). System design methodologies for a wireless security processing platform. In *39th Design Automation Conference (DAC)*, pages 777–782.

Wang, J., Wang, W., Yang, J., Hu, Z., Han, J., and Zeng, X. (2015). Parallel implementation of aes on 2.5d multicore platform with hardware software co-design. In *IEEE 11th International Conference on ASIC (ASICON)*, pages 1–4.

Wu, Q., Pedram, M., and Wu, X. (2000). Clock-gating and its applications to low-power design of sequential circuits. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(3):415–420.