March - April 2020
ISSN: 0193-4120 Page No. 107 - 113

Efficient Power Reduction Scheme for AES using Hardware-Software Co-Design

[1] Padmini G Kaushik, [2] Dr.S.M.Gulhane
[1] Assistant Professor,[2] Professor
[1]padmini.jp@gmail.com,[2] smgulhane67@rediffmail.com

Article Info
Volume 83
Page Number: 107 - 113
Publication Issue:
March - April 2020

Article History
Article Received*: 24 July 2019*
Revised*: 12 September 2019*
Accepted*: 15 February 2020*
Publication*: 12 March 2020*

Abstract
Embedded product design is driven by various design constraints like Device surface area utilization, volume, power consumption, and performance. These designs may even need to concurrently address the hardware and software requirements to bring out the product features at each stage while factoring in that, these requirements may additionally constrain or contribute to each other. All of these ought to be accomplished preserving in context time to Market and the Market timing. Hardware/Software Co-design methodology is an frequently employed way to deal to design embedded systems to reduce product development time and power consumption. Co-design offers the adaptability of design, as there are numerous Soft-core processors and Hardware development platforms that are offered in the market. This paper proposes a new Hardware/Software Co-design methodology, which has been used for enforcing the Advanced Encryption Standard (AES) algorithm for encrypting and decrypting 128,198 and 256 bits of data using NIOS II processor, from ALTERA to be imposed in FPGA retaining pace, area, and thermal dissipation as the focal point.

Keywords: *AES, NIOS-II, Hardware/software co-design, Rijndael algorithm.*

1. Introduction

Embedded systems having significant computing capability and supporting various functionalities and applications are continuously evolving and are increasingly becoming prevalent.

Design techniques of this embedded system are also evolving with the growth in complexity of the embedded system. Co-Design which approaches the embedded system solution as a combination of software and hardware is known to advance the system performance.

The well-known cryptographic AES algorithm, also known as Rijndael, process the data and encrypts at the sender (encipher) and again processes and decrypt (decipher) the information at the receiver end. Since this has both compute and memory-intensive tasks, we have chosen this algorithm to illustrate the Co-design approach. Running the cryptography algorithms in software can cause lack of processing speed and hence the performance of the overall system. Hence the processing activities are done in the Hardware and the rest of the activities are executed via the Software in our illustration.

1.1 Co-Design

Hardware/software co-design is described as the consolidated implementation of hardware and software. The goal of co-design is to compact the manufacturing time, while

plunging the structure endeavor and expenditure of the planned items [2].

A platform for implementing the Co-design can be an embedded system and processor is the main part of any embedded system. As software is more lithe and inexpensive than hardware, numerous benefits can be achieved using processors. This flexibility of software opens overdue design modifications and debugging. In addition, the chance of reutilization of programs by targeting it to different processors lessens an opportunity to-advertise and the design exertion [2] [4]. When processors are not ready to fulfill the essential, the hardware is utilized to vanquish the design challenge. This tradeoff among hardware and software represents the enhancement aspects of the co-design problem. Co-design is an interdisciplinary activity, bringing insight and thoughts from assorted fields together, e.g. system-level modeling, hardware design and software design [5].

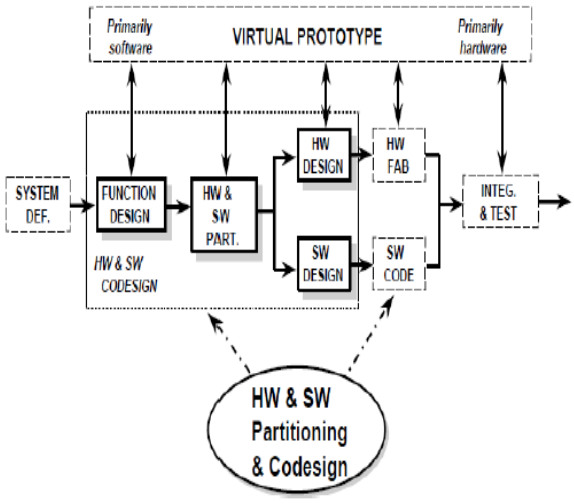

Fig. 1 Design Flow for HW-SW co-design

A general structure for a co-design methodology is illustrated in fig. 1

Step1: System-level behavior of the co-design is specified in this step.

Step 2: After verification of all algorithms, a pure software system will be evolved.

Step 3: System bottlenecks are to be determined through analyzing the performance of the system.

Step 4: Determine the partitioning of the system for hardware and software implementation. This partitioning is merely relying at the complexity of the system, computations required and memory used by the system. Some design bottleneck will get swapped by means of hardware to advance the overall performance.

Step 5: Hardware and software parts will be designed based on the outcome of step 4.

Step 6: After combining hardware and software, analyze the system performance in terms of speed, power dissipation, and area.

Step 7: If essential results are not met through the step 6, then a new scheme and design will be applied for HW/SW partitioning.

Plan tradeoffs, rehashed confirmation during the structure cycle, and common impact of both HW and SW from the get-go in the structure cycle

1.2 Motivation

The co-design technique is often used to reduce the duration spent on the design cycle and the troubleshooting process. Partitioning of the system (Co-design) is the most ideal approach to accomplish structure tradeoffs, repeated affirmation during the design cycle, and common impact of both hardware and software early in the design cycle [5].

The incorporation of composite system on a Chip (SoC) is facilitated by modernization in ASIC and FPGA technologies. The utilization of various Co -design strategies is application-specific. In the proposed method, we are

targeting for a fast and low power system. For FPGA implementation with the highest operating frequency, NIOS-II soft-core processor is suitable, as it also permits salvage of code and extremely configurable. These facilities provoked us to illustrate Co-design methodology with NIOS-II soft core processor by using AES Algorithm implementation to study the system performance mainly focusing on power reduction.

2. AES Algorithm

The co-design method is frequently used to reduce the duration spent on the design cycle and investigating the process. Partitioning of the system (Co-design) is the most ideal approach to accomplish design tradeoffs, repeated affirmation during the design cycle, and shared impact of both hardware and software early in the design cycle [5].

The consolidation of composite framework on a Chip (SoC) is encouraged by modernization in ASIC and FPGA technologies. The utilize of various Co-design strategies is application-specific. In the proposed method, we required a fast and low power system. For FPGA implementation with the highest operating frequency, NIOS-II soft-core processor is appropriate, as it also permits salvage of code and extremely configurable. These facilities provoked us to illustrate Co-design methodology with NIOS-II soft core processor by using AES Algorithm implementation to study the system performance mainly focusing on power reduction.

AES algorithm is implemented in three ways, namely AES - 128, AES - 192, and AES-256. The number in each case describes the key size (in bits) used for encryption/decryption. Depending on the range of the block and the key size chosen, the above modes are finished in 10, 12 or 14 rounds. AES just permits a data of size 128 bits and separated into four operational blocks. These blocks work on 4*4 matrix arrays which are generally called as the states.

For both encryption and decryption algorithms, the flow initialized with an Add round key stage pursued by nine rounds of four stages and the tenth round of three stages [1] [3]. Following four stages are used to implement these rounds:

In the Encryption process, every round comprises of four operations: Sub-Bytes, Shift-Rows, Mix-Columns, and Add-Round-key. For 128 bit key, there will be a total of 10 rounds. In any case, in the last round, Add-round-key operation is not performed.

Sub-Bytes: The format required for input data and the key is a matrix with the size of a byte. To make a new matrix, ***X-OR*** operation is performed among data and key. S-box byte is substituted for each byte in the matrix, where S-box is a standard substitution table in the algorithm.

Shift-Rows: After the Sub-byte operation, the elements in the array are moved to case side. The starting row of the array stays unaltered, the subsequent row is moved left by 1 byte, the third row by 2 bytes and the last row is shifted by 3 bytes to frame a new matrix.

Mix-Columns: This stage of transformation is depends on Galois Field multiplication. Each element of this column is altered with another value that is an each element of four bytes in the given column. The Mix-Column transformation executes on the State column-by-column, regarding every column as a four-term polynomial GF (28).

Add-round-key: In this round, State matrix is added with a Round Key by performing bitwise operation on plain text and input key. Key schedule generation provides round Key

March - April 2020
ISSN: 0193-4120 Page No. 107 - 113

of size ***Nb***. Columns of the States are then added with these ***keys***. The lat round i.e.10th round of Mix columns stage is not included.

The four stage operations for initial rounds of the algorithm implementation are given by Inverse Shift rows, Inverse Substitute Bytes , Add round key, Inverse Mix columns. All these four stages are completed in 9 rounds. The last round Inverse Mix columns stage i.e. tenth in this case, is not covered again.

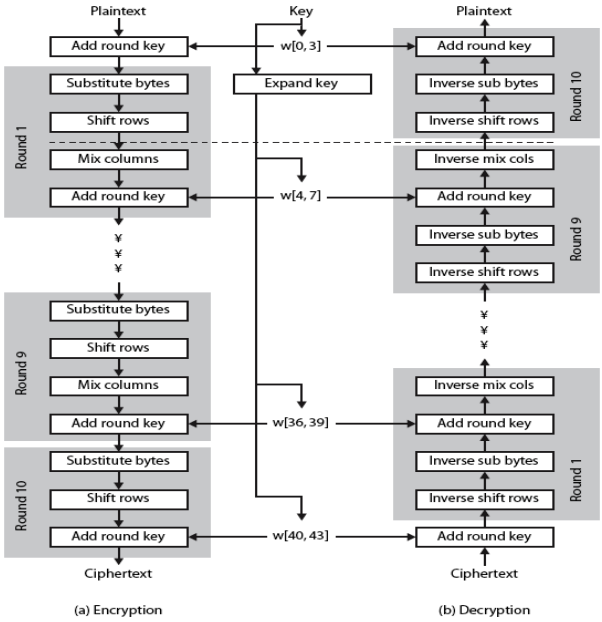

Figure 2-AES Flow

3. System Design Using Soft Core Processor

Altera provides soft core processor NIOS-II as 32-bit processor. It includes a processor (NIOS II), a set of memory on-chip and accessories, GPIOs, all linked with Avalon bus to generate a system [9, 10]. It is a reconfigurable soft-core processor, as contrasting to a off-the-shelf, fixed, microcontroller. As the processor is configurable adding and eliminating components or features on a system is simple to satisfy overall performance desires in phrases of area. NIOS-II can be focused to any FPGA family of Altera [7].

Following are the features of Altera soft core processors:

• It is a general-purpose RISC processor core with full 32-bit instruction set, data path, and address space, 32 general-purpose registers, Access to a variety of on-chip peripherals, 32 interrupt sources, External interrupt controller interface for more interrupt sources, and interfaces to off-chip memories and peripherals, Single-instruction 32 × 32 multiply and divide producing a 32-bit result, Optional memory protection unit (MPU), Instruction set architecture (ISA) perfect over all NIOS- II processor systems, Performance up to 250 DMIPS, Software advancement environment dependent on the GNU C/C++ tool chain and NIOS IDE,.

3.1 Implementation

Essential components required for building AES design using FPGA soft core processors are NIOS II Processor, Input, Output GPIO'S, JTAG UART, Performance Counter, SRAM MEMORY.

The interconnection of the building blocks to incorporate the parts in the hardware system is automatically generated by SOPC Builder. The selection of these blocks can be chosen from the list provided in NIOS- II EDS.

The figure 3 & 4 shows the component selection to build and generate the design. The components used to build the design are accessed by SOPC Builder is shown by the figure 3and successful generation of the system is shown in figure 4.

March - April 2020
ISSN: 0193-4120 Page No. 107 - 113

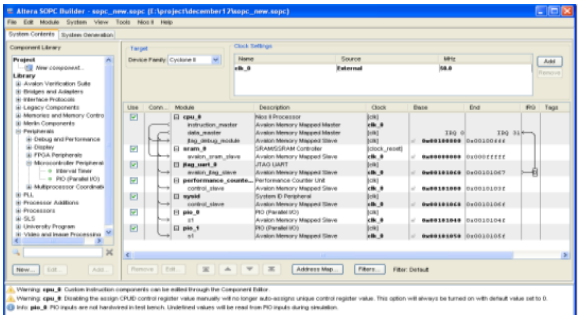

Fig. 3 System contents in SOPC Builder

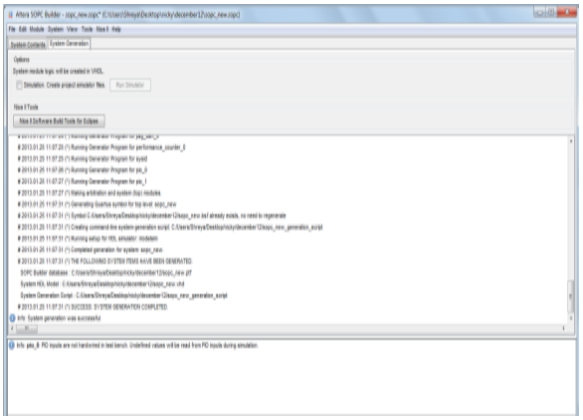

Fig. 4 Successful System generation in SOPC Builder

Figure 5 shows, the execution of encryption with 128-bit key and 128-bit Plaintext/Ciphertext in NIOS -II.

After successful generation of SOPC *model*, NIOS II IDE will be activated to implement the design by targeting "*.sopcinfo"* file as a hardware platform.

The AES Algorithm is written in C by using the inbuilt project template "*Hello_world.c* file (Encryption & Decryption separately). AES project is then compiled /build using the command "*Build Project"*. The built project next targeted to CYCLONE II (EP2C35F672C6) FPGA by running the command "*Run as NIOS II Hardware*". Subsequently, encrypted cipher text will be found on output Window of NIOS II platform.

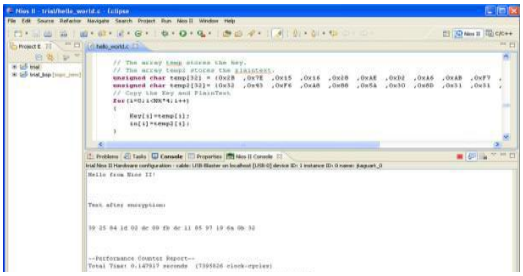

Fig. 5 Execution of AES encryption on NIOS-II

In Figure 5, shows output of AES encryption and timing analysis when executed using NIOS-II processor. This execution uses the soft core processor available in Altera.

4. Implementation of Hw/Sw Codesign o On Cyclone-II:

A digital ASIC implementation of the hardware-software co-design consequences in consistent, steady and highly performing circuit design, at the same time as the strength optimized hardware-software co-design fashion leads to Power efficient architecture.

As Hardware works parallel, speed will be accomplished as significant preferred position On account of programming, each move is made by the microcontroller, so it runs in consecutively.

Programming advancement is quick and it shows up very straightforward when we are dealing with the entangled task however equipment improvement gets intense if there should be an occurrence of the volumetric venture. So in any circumstance, the two advantages can be mixed to acquire streamlined execution. Considering these factors, we have selected a hardware-software co-design strategy for AES performance assessment. Some part of the system is executed using NIOS-II tool and a portion of the segment is actualized in FPGA as custom IP. This co-design is actualized utilizing the Altera platform.

March - April 2020
ISSN: 0193-4120 Page No. 107 - 113

Critical programmable requirements comprise of:

SW, TSBOX or GSBOX: A developer will have a software table *(SW)*, pre-store hardware table *(TSBOX)*, delivering change with the aid of combinational logic to execute *SBOX (GSBOX)*, which is acknowledged by amalgamated field calculation as communicated in the earlier part.

Number of SBOX: A user can select many SBOX to implement: 1, 4, 8 or16 if *TSBOX or GSBOX* used.

Mix-Column: A user can decide whether to realize it using hardware.

ShiftRow+ Add-round-key: A user can decide whether to realize it using hardware.

By using the grouping of the appropriate programmable parameter, 36 combinations can be made [8].

AES is generated and interfaced with NIOS-II as a periphery. This is connected with register logics available in NIOS processor. These registers are used as a memory of 32 bits. The plain text data are fetched in 4 parts using this Four 32-bit registers. The plain text is of 32 bytes. Every plain text is expressed in ASCII format. Association arrangement is utilized for the conversion of a byte to 32-bit register. Key data and Plain text data takes 4 each. The cipher text will be stored using 4 more registers for additional implementation or presentation. AES algorithm and status of the AES algorithm are controlled by two registers. Status of operation completion is checked through AES controls to the AES engine. Time is calculated using this flag.

The identical key is used to implement encryption and decryption. But keys are used in reverse order for decryption process. The key for first round of the decryption is the key used for last round of encryption. The keys are stored in an array and then the array the values are called in the reverse order.

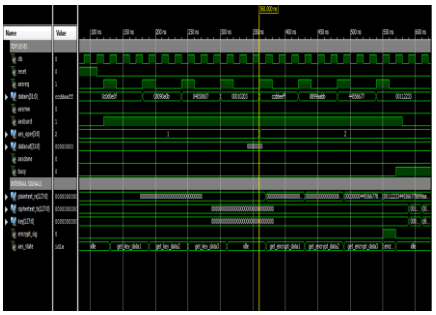

Figure 6(a) Simulation of Encryption

In figure 6(a) & (b), all in-between and ending result of design are demonstrated. Time line of the process is shown by a yellow line.

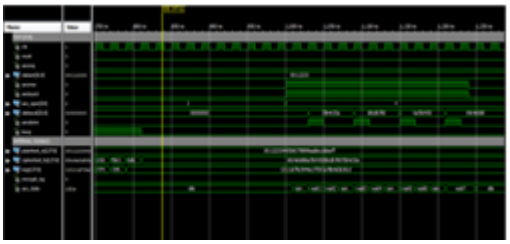

Figure 6(b) Simulation of Encryption

Figure 7 shows performance of Hardware and software designs. AES encryption implemented using Cyclone-II FPGA and NIOS-II from Altera. To evaluate the power, we export the HDL files from Quartus-II to create an EDIF (Electronic Design Interchange Format) netlist.

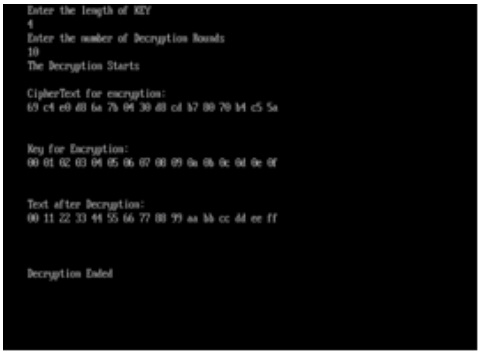

Fig.7: Output of encryption and decryption

March - April 2020
ISSN: 0193-4120 Page No. 107 - 113

Synopsys Design Compiler is used afterward. By the proposed method around 25%-30% power reduction is achieved. Table 1 gives the achievement of proposed method in terms of reduction in dynamic power for the design types with only Hardware and Co-Design.

Table 1 Dynamic Power consumption comparison

System implementation type	**Dynamic consumption (µW)**
AES Hardware only design	20.0146
AES Hardware Software Co design	14.9077

5. Conclusion

Hardware-software co-design seems imperative given the prevailing day design demanding situations of time to market, cost and ever-increasing product capabilities and design complexity. Co-design allows a technique to view the product from various perspectives and optimize product performance. We have seen how co-design offers an option for power management. Partitioning is at the crux of hardware-software co-design. This approach is a effective tool with its own set of dynamism and complexities, which when dealt with diligence and cognizance, yields great results.

References

[1] FIPS PUB 197, Advanced Encryption Standard (AES), National Institute of Standards and Technology, U.S. Department of Commerce, November 2001,(http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf).

[2] Jason G. Tong, Ian D. L. Anderson, and Mohammed A. S. Khalid: Soft-Core Processors for Embedded Systems, the 18th International Conference on Microelectronics (ICM) 2006.

[3] J. Daemen and V. Rijmen," AES Proposal: Rijndael," AES Algorithm Submission, Sept. 1999.

[4] Ernst, R.: "Co-design of embedded systems: status and trends", Proceedings of IEEE Design and Test, April–June 1998, pp.45–54

[5] Y. Li, T. Callahan, E. Darnell, R. Harr, U. Kurkure, and J.Stockwood, "Hardware-Software Codesign of Embedded Reconfigurable Architectures", in Proc. Design Automation Conference, 2000.

[6] Altera Corporation, "Nios Embedded Processor, 32-Bit Programmer's Reference Manual" [Online Document] January 2003

[7] Altera Corporation, "Nios Software Development Tutorial," [Online Document], 2003 July, [Cited2004 March 1], Available HTTP: http://www.altera.com/literature/tt/tt_nios_sw.pdf

[8] Kuan Jen Lin, Chin-Mu Hsiao, and Ching Hung Jhan. (2009). Exploring HW/SW Co-design of AES Algorithm Using Custom Instructions.