Journal of Systems Architecture 56 (2010) 116–123

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

FPGA schemes for minimizing the power-throughput trade-off in executing the Advanced Encryption Standard algorithm

Jason Van Dyken, José G. Delgado-Frias *

School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, USA

ARTICLE INFO

Article history:
Received 23 April 2009
Received in revised form 16 October 2009
Accepted 3 December 2009
Available online 16 December 2009

Keywords:
FPGA
Advanced Encryption Standard (AES)
Power
Performance

ABSTRACT

Today most research involving the execution of the Advanced Encryption Standard (AES) algorithm falls into three areas: ultra-high-speed encryption, very low power consumption, and algorithmic integrity. This study's focus is on how to lower the power consumption of an FPGA-based encryption scheme with minimum effect on performance. Three novel FPGA schemes are introduced and evaluated. These schemes are compared in terms of architectural and performance differences, as well as the power consumption rates. The results show that the proposed schemes are able to reduce the logic and signal power by 60% and 27%, respectively on a Virtex 2 Pro FPGA while maintaining a high level of throughput.

1. Introduction

Communication among a large number and diverse set of devices has increased dramatically in recent years. This in turn has led not only to the need for connecting a large number of devices, but means to ensure that communication is secure. Today the most commonly used and accepted standard for private key encryption is the Advanced Encryption Standard (AES). Since its inception AES has been the subject of in-depth research that has typically focused on three areas. The first is algorithmic integrity, which ensures there is no feasible way to obtain the original data in a timely manner [1]. The second area is high throughput schemes, which seek to carry out encryption as quickly as possible [2,3]. The final area of emphasis has been on low power systems, which seek to minimize power consumption at all costs [4,5].

Recently a combination of the latter two areas of research have become of great importance, allowing new implementations to maintain a high throughput while using as little hardware and power as possible. This area is growing increasingly important as devices are not only communicating in higher frequency, but in larger quantities and need to translate the secured data into a usable state as efficiently as possible. It is this research emphasis that this study explores. More specifically this study focuses on the implementations pertaining to FPGAs and reconfigurable hardware for the implementation of the AES algorithm. The primary reason for this is that as reconfigurable hardware technologies continue to progress it is becoming more viable for devices to use a single chip for multiple tasks and simply reconfigure the chip for its current need. This is particularly useful in the field of sensor networks where hardware and power is limited, but secure communications may be critical.

This paper is organized as follows: Section 2 provides a brief overview of the AES algorithm. In Section 3 a brief description of current schemes is presented. Section 4 introduces the proposed schemes. In Section 5 a comparative analysis of the schemes is presented, the first half focuses on how the decisions made in a given scheme affected their performance and hardware requirements. The second half focuses on the power consumption of the schemes available for testing and what can be learned about how to control power consumption. Some concluding remarks are included in Section 6.

2. Advanced Encryption Standard (AES)

AES was developed to replace the aging Data Encryption Standard (DES), which after twenty years of use had become susceptible to brute force attacks. AES was introduced in 2001 [6] after an extensive vetting process were many algorithms were proposed for evaluation and in the end the Rijndael algorithm was adopted as the AES algorithm. The core of the algorithm is made up of four basic components, which work on 8 bit data blocks. The whole 128 bit input to the algorithm is organized into a 4×4 matrix termed a state, to obtain the 8 bit blocks. The component blocks then carry out their transformations on the input data repeatedly to generate the cyphertext. The overall architecture of the round

* Corresponding author. Tel.: +1 509 335 1156; fax: +1 509 335 3818.
E-mail address: jdelgado@eecs.wsu.edu (J.G. Delgado-Frias).

1383-7621/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2009.12.001

block is shown in Fig. 1. Each of the basic transformations are briefly described below, along with a fifth block that performs the transformation on the encryption key that produces each round key. A more detailed description of the standard can be found in [7].

Byte substitution. The byte substitution block (SBOX) is based on taking a non-linear byte substitution. This is the most mathematically complex of the components and as a result is typically implemented by a look up table. Each of the schemes contained in this paper chose to use the look up table implementation for the SBOX.
Shift rows. The shift rows component takes the data in the state matrix and circularly shifts each data block left by its row index.
Mix columns. The mix Column transformation generates a new value for each of the 8 bit data blocks by using a weighted sum of all of the data blocks within a given column of the state matrix. This is the only transformation that cannot be simplified to a single operation working on a single state matrix data block, but requires the 32 bits in a given column.
Add round key. The add round key component takes in a unique round key form the key expansion component and simply performs a bit by bit addition with each of the bits in the state matrix.
Key expansion. The key expansion component takes in the initial key or previous round key and organizes the data into a state matrix, then circularly shifts each of the data blocks in the final column up one position and combines the topmost block with a round constant. The final column then undergoes the substitution transformation and finally the columns are added together to generate a new 128 bit round key.

The standard architecture for any AES device consists of the series connection of the SBOX, shift row, mix column, and add round key transformations. The key expansion block is used to generate the required number of round keys, 10 for 128 bit initial key sizes, and providing them for the Add Round Key block. For standard 128 bit key encryption the data is fed through the transformations 10 times with the Mix Column transformation being omitted in the final round.

It is in the assembly of these components that many of the major design decisions arose. The choices that differentiate the schemes are explained during the scheme introductions. This included whether to use full 128 bit busses, which require a larger amount of hardware. Using full bus widths has two contrasting results, using more component blocks and requiring larger FPGAs or using more complex control logic with storage and buffer registers. For all the proposed and reference schemes full 128 bit data paths are used.

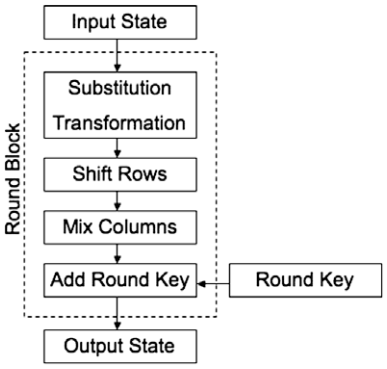

Fig. 1. Round block organization.

3. Current schemes

In this section existing FPGA-based schemes are presented and ordered chronologically from oldest to most recent. This description includes both the overall structure of the scheme, and the purpose for which each scheme was designed. As mentioned earlier the schemes use commercially available FPGAs rather than an application specific integrated circuit.

3.1. NIST scheme

The primary purpose of the National Institute of Standards and Technology (NIST) scheme was to demonstrate how to build an AES device that is fully compatible with all the AES encryption key sizes [8]. Since this scheme did not intend to be a demonstration of the fastest possible implementation or the most efficient it was chosen because it was representative of an average non-optimized implementation, which was concerned more about total standard compliance than speed or power consumption. As a result this scheme requires the largest FPGA of the considered schemes. There are full 128 bit data input and output busses, along with a 256 bit key input bus. In this scheme the key can be saved for multiple encryptions or stored concurrently with the data to allow for maximum flexibility in key handling.

Unlike the forthcoming proposed schemes the NIST scheme's round implementation is accomplished through Hardware Definition Language (HDL) function calls from a specially designed package file rather than through explicit round descriptions. This means that the physical layout is very dependant on the synthesizing, mapping, and routing functions of the design software, rather than the designer explicitly defining the organization of components. Once the encryption process is started the scheme takes 12 cycles until the ciphertext is written to the data output bus.

3.2. Gaj scheme

The Gaj et al. scheme [9] was built for evaluating the final candidates of the AES algorithm and how well they could be adapted for use on FPGAs. The reason for this evaluation was that the NIST evaluations were focusing predominately on ASIC implementations as well as Software implementation, but not on reconfigurable hardware applications. More specifically the evaluation focused of the comparison of hardware size and total throughput, which meant that while hardware size was important hardware-performance trade-offs were not fully considered to achieve the highest throughput possible.

The architecture of the scheme is a very simple repeated round block, which was built using the exact specifications of the Rijndael algorithm, and is shown in Fig. 2. This means that there were no optimizations to any components in the block. Additionally the SBOX component was made to output both the encryption

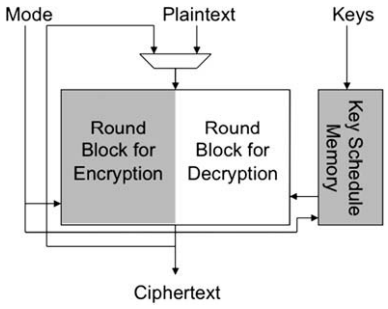

Fig. 2. Block diagram of the Gaj scheme.