Microelectronics Journal 40 (2009) 1032–1040

Contents lists available at ScienceDirect

Microelectronics Journal

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

IDEA and AES, two cryptographic algorithms implemented using partial and dynamic reconfiguration

José M. Granado *, Miguel A. Vega-Rodríguez, Juan M. Sánchez-Pérez, Juan A. Gómez-Pulido

Department Technologies of Computers and Communications, University of Extremadura, Spain

ARTICLE INFO

Article history:
Received 12 July 2007
Received in revised form
5 November 2008
Accepted 12 November 2008
Available online 9 January 2009

Keywords:
IDEA
AES
Cryptography
FPGA
VHDL
Handel-C

ABSTRACT

In this work, we present our experience in implementing two different cryptographic algorithms in an FPGA: IDEA and AES. Both implementations have been done by means of mixing Handel-C and VHDL and using partial and dynamic reconfiguration in order to reach a very high performance. In both cases, we have obtained very satisfactory results, achieving 27.948 Gb/s in the IDEA algorithm and 24.922 Gb/s in the AES algorithm.

1. Introduction

We all have listened to the sentence "the information is power" more than once and, when we speak about digital information, this statement takes a special importance. We live in a digital world and a lot of important information goes through insecure networks. The wireless networks are especially insecure because the unauthorised access to these networks is very easy; we only need to scan the air near a company to capture important information of this company. To solve this problem we can employ cryptographic algorithms, which transform the raw information to an unintelligible sequence of bytes.

But the new network standards and new technologies (like the Internet private TV) demand that these algorithms must be very fast. One solution to reach a high performance is to employ FPGAs to implement the algorithms. Besides, these devices allow taking advantage of the parallel parts that these algorithms have.

In this work, we have implemented two different algorithms: the international data encryption algorithm (IDEA), one of the most secure cryptographic algorithms, and the advanced encryption standard (AES), the one used in wireless networks. In both cases, we have employed pipelining, and dynamic and partial reconfiguration.

We can find other implementations of the IDEA algorithm, like [1], where a pipelining implementation using partial and dynamic reconfiguration is done to reach a very high performance (8.3 Gb/s). In our case, we have developed a new reconfigurable element, concretely the constant coefficient adders (KCAs) and we use Handel-C [2] and VHDL [3] to make the implementation (VHDL to implement the reconfigurable elements and Handel-C to implement the non-reconfigurable elements). Besides, we duplicate the data path, encrypting two data blocks at the same time (or to encrypt and decrypt at the same time). These improvements give us a better performance (27.948 Gb/s). In addition, Handel-C language allows us to decrease the development time because it is a high-level language closer to the traditional programmer. The pipelining is also a well-known technique and we can find implementations which use it, like [4], where a seven stages of implementation are carried out, unlike in our implementation, where we have 182 stages. Finally, as we will see, multipliers are the critical elements of this algorithm, and so, to optimize them is a very important task. In [4] partial product generation and a three-stage diminished-one adder are used to calculate the multiplication modulo $2^{16}+1$. We can find another multiplier implementation technique in [5,6], where parallel-serial multipliers are used. However, our dynamically and partially reconfigurable implementation uses constant coefficient multipliers (KCMs) [7], which are very fast because they employ look-up tables (LUTs) to store part of the result of the multiplication.

Taking all this into account, we achieve to improve the results obtained by these authors, among whom the best result is 8.3 Gb/s [1], and our result is 27.948 Gb/s.

* Corresponding author. Tel.: +34 616 557 935; fax: +34 927 257 202.
E-mail address: granado@unex.es (J.M. Granado).

0026-2692/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mejo.2008.11.044

On the other hand, we have the AES algorithm. We have implemented a 128-bit version of this algorithm using pipelining, and partial and dynamic reconfiguration. The most complex element of this algorithm is the multiplication modulo the irreducible polynomial $m(x) = x^8+x^4+x^3+x+1$. We find one possible implementation of this element in [8], where the *xtime* function is implemented to do the multiplication. In our case, we have implemented this element simply calculating the XORs operations needed to make the multiplication. Another important element is the KeyExpansion phase. It is very usual to implement this phase and we can find several implementations of it, like [9], where a hierarchical simultaneous key generation methodology is used. In our case, we use dynamic and partial reconfiguration to modify the LUTs which contain the sub-keys and so, we do not re-calculate each sub-key and we do not use resources to implement this method. Besides, we employ Handel-C and VHDL in the same way as we do in the IDEA algorithm, i.e., VHDL to implement the reconfigurable elements and Handel-C to implement the non-reconfigurable elements, obtaining the same advantages.

Taking all this into account, we achieve a very good result (24.922 Gb/s), only surpass by [9] (29.8 Gb/s), but we have a better performance/area, occupation and latency values.

In conclusion, there are many implementations of cryptographic algorithms using FPGAs. All the FPGA-based implementations found in the literature employ some of the characteristics we mix in this work, but there are no papers that employ our methodology completely (pipelining, replication, partial and dynamic reconfiguration, and the mix of three hardware languages—Handel-C, VHDL and JBits). We will see how using our methodology we improve the results obtained by other FPGA-based implementations of IDEA and AES. In fact, our results surpass the best results published in the literature.

This paper is structured as follows. Section 2 reviews the different techniques and approaches used in all the FPGA-based implementations of both cryptographic algorithms that we have found in the literature. In Section 3, we describe briefly the IDEA and AES algorithms. We explain the used methodology in Section 4, focusing on the union between VHDL and Handel-C, and the partial and dynamic reconfiguration was performed. Later, we describe the exact implementation of both algorithms. Finally, in Section 6, we analyze the results, and we give the conclusions in the last section.

2. Related work

This section gives a background review, detailing different techniques and approaches used by other FPGA-based implementations. This allows distinguishing our work from others.

Table 1 shows all the papers we have found in the literature about FPGA-based implementations of the IDEA algorithm. For every work we detail the reference, the publication year, and if this work uses the following techniques and approaches: partial and dynamic reconfiguration (PDR), pipelining (Pi), replication/parallelism (R/Pa), constant coefficient multipliers and constant coefficient adders. In this table, column "Mult." indicates the technique used for the implementation of the multipliers modulo $2^{16}+1$: PDR (implemented by means of partial and dynamic reconfiguration, i.e., by using KCMs), three-state adder tree (TAT) and modified Lyon's parallel-serial multiplier (MLPM). Finally, we also state the hardware description language used in each work (VHDL or Handel-C—H-C). In this table, "??" means that those data were not found in the corresponding reference.

Table 1
Summary of techniques used in the FPGA-based implementations of IDEA.

Ref.	Year	Techniques							
		PDR	Pi	R/Pa	KCM	KCA	Mult.	VHDL	H-C
This work	2008	X	X	X	X	X	PDR	X	X
[1]	2003	X	X		X		PDR	X	
[4]	2002		X				TAT	X	
[6]	2001	X		X			MLPM	??	??
[10]	2002	??	??	??	??	??	??	??	??
[11]	2004		X		??		??	X	
[5]	2000	X		X			MLPM	??	??
[12]	2002	X	X		X		PDR	X	

Table 2
Summary of techniques used in the FPGA-based implementations of AES.

Ref.	Year	Techniques						
		PDR	Pi	BRAM	MCP	KEP	VHDL	H-C
This work	2008	X	X		*xtime*	PDR	X	X
[13]	2001			X	LUTs	??	??	??
[14]	2003		X	X	*xtime*	??	??	??
[15]	2004		X	X	??	??	??	??
[9]	2005		X	X	??	HSKG	??	??
[16]	2005		X	X	TBOX	SCD	X	
[17]	2003		X	X	??	OKG	X	
[18]	2004		X		??	OKG	X	
[19]	2004		X		*xtime*	OKG	??	??
[20]	2003		X		$GF2^4$	$GF2^4$	X	
[21]	2001		X	??	??	??	X	

Observing Table 1, we can conclude that our work is the only one that combines all the techniques shown in this table. For example, any other work uses the Handel-C language (or its combination with VHDL) or KCAs. In the same way, although KCMs are interesting, few works use them.

Table 2 lists all the papers we have found in the literature about FPGA-based implementations of the AES algorithm. For every work we detail the reference, the publication year, and if this work uses PDR, pipelining (Pi) and/or BRAM. In this table, column MCP indicates the MixColumns phase implementation technique used *xtime* (implemented by means of the *xtime* function), LUTs (precalculated tables which store all the possible multiplying results), TBOX (combining the MixColumn and SubByte operations) and $GF2^4$ (implemented by using the $GF(2^4)$ instead of $GF(2^8)$). In the same way, column KEP shows the KeyExpansion phase implementation technique used PDR (implemented by means of Partial and Dynamic Reconfiguration), hierarchical simultaneous key generation (HSKG), Separated Clock Domain implementation (SCD), online key generation (OKG) and $GF2^4$ (implemented by using the $GF(2^4)$ instead of $GF(2^8)$). Finally, we also provide the hardware description language used in each work (VHDL or Handel-C—H-C). In this table, "??" means that those data were not found in the corresponding reference.

From Table 2 we obtain conclusions similar to those indicated for Table 1 (IDEA algorithm): our work is the only one that combines all the techniques shown in this table (except the use of BRAM, in fact, the half of the references does not use BRAM). We have to highlight that no other work uses the Handel-C language (or its combination with VHDL) or PDR (we use this important technique in order to implement the KeyExpansion phase).

3. The cryptographic algorithms used

In this section, we give a brief overview about the two cryptographic algorithms implemented in this work: IDEA and AES. Among the possible secure modes, for both algorithms we