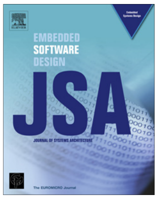
Journal of Systems Architecture 60 (2014) 440–451

Contents lists available at ScienceDirect

Journal of Systems Architecture

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



# Improving the computational efficiency of modular operations for embedded systems

Ismail San *, Nuray At

*Department of Electrical and Electronics Engineering, Anadolu University, 26470 Eskişehir, Turkey*

ARTICLE INFO

*Article history:*
Received 31 January 2013
Received in revised form 8 September 2013
Accepted 18 October 2013
Available online 31 October 2013

*Keywords:*
Public-key cryptography
Modular arithmetic
Acceleration
FPGA
System-on-chip

ABSTRACT

Security protocols such as IPSec, SSL and VPNs used in many communication systems employ various cryptographic algorithms in order to protect the data from malicious attacks. Thanks to public-key cryptography, a public channel which is exposed to security risks can be used for secure communication in such protocols without needing to agree on a shared key at the beginning of the communication. Public-key cryptosystems such as RSA, Rabin and ElGamal cryptosystems are used for various security services such as key exchange and key distribution between communicating nodes and many authentication protocols. Such public-key cryptosystems usually depend on modular arithmetic operations including modular multiplication and exponentiation. These mathematical operations are computationally intensive and fundamental arithmetic operations which are intensively used in many fields including cryptography, number theory, finite field arithmetic, and so on. This paper is devoted to the analysis of modular arithmetic operations and the improvement of the computation of modular multiplication and exponentiation from hardware design perspective based on FPGA. Two of the well-known algorithms namely Montgomery modular multiplication and Karatsuba algorithms are exploited together within our high-speed pipelined hardware architecture. Our proposed design presents an efficient solution for a range of applications where area and performance are both important. The proposed coprocessor offers scalability which means that it supports different security levels with a cost of performance. We also build a system-on-chip design using Xilinx's latest Zynq-7000 family extensible processing platform to show how our proposed design improve the processing time of modular arithmetic operations for embedded systems.

## 1. Introduction

With the advancement of communication technology and information systems, networking and data streaming applications call for higher data rates as well as security at the same time. There are many such emerging applications especially for embedded systems that need to communicate, store or manipulate confidential data. This makes security is a primary concern in the design of embedded systems for certain applications. Security is provided by the set of cryptographic algorithms which are usually computationally intensive algorithms. Hence, embedded security is a constantly developing field along with the research on efficient hardware design of cryptographic algorithms. To meet security needs of such applications, there are several security protocols such as IPSec, SSL and VPNs which are used between a pair of communicating hosts called sender and recipient. Further, with the introduction of Public-Key Cryptography (PKC), a public channel can be used for secure communication. PKC have also many benefits in such protocols such as key-distribution and various authentication protocols due to the fact that PKC does not require a secure initial key exchange between the sender and the recipient.

Most of the public-key cryptosystems use modular arithmetic operations, specifically, modular multiplication and exponentiation. These two operations are also widely used in other fields. Hence, their efficient computation is quite important for the security of embedded systems. There are many different algorithms and computation models in the literature to perform modular multiplication such as Montgomery, Booths, Karatsuba, etc. Moreover, repetitive use of modular multiplication is needed in the modular exponentiation operation. Therefore, efficient design of modular multiplication has a great importance in the computational efficiency of modular exponentiation.

In this study, we focus on high-speed pipelined hardware implementation of the modular multiplication and exponentiation operations on FPGAs. In [1], San and At designed a compact coprocessor which efficiently exploits intrinsic properties of Karatsuba algorithm on Xilinx Virtex-5 FPGA devices. Here, in particular, the design strategy in [1] is improved in terms of latency. Improved pipelined version of Karatsuba coprocessors employed in the

* Corresponding author. Tel.: +90 222 321 3550/6469.
*E-mail addresses:* isan@anadolu.edu.tr (I. San), nat@anadolu.edu.tr (N. At).

1383-7621/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2013.10.013

proposed architecture in this study. Hence, a pipelined hardware implementation for the modular multiplication and exponentiation operations using both Karatsuba and Montgomery algorithms is proposed.

Today's FPGAs have enhanced hard cores such as DSP and Block RAM components yielding very high operating frequencies. As emphasized by Güneysu, "the use of device specific components lead to considerably higher system performance" [2]. With this approach in mind, we have adapted Karatsuba and high-radix Montgomery modular multiplication algorithms to efficiently compute the modular multiplication on FPGA using device specific components such as DSP and shift register components. In order to extend arithmetic precision to be used in larger modulus and make modular multiplication algorithm more compact for larger bits, Karatsuba and Montgomery algorithms can be used by exploiting the embedding multiplier modules existing in almost all FPGAs in pipelined mode. In this article, we first efficiently combine very high-radix Montgomery and Karatsuba algorithms for performing modular multiplication algorithm using these embedded hard cores on latest high-speed FPGAs. This gives us very powerful results in those cases where the parallel and pipelined processing is available. Then, we implement compact modular exponentiation architecture using the proposed high-speed modular multiplication blocks. We provide hardware performance figures of our methods on FPGA in terms of latency, area, and frequency. We achieve efficient high-radix modular multiplications by means of pipelined Karatsuba coprocessor. The proposed hardware uses efficiently Karatsuba algorithm to multiply large numbers in a compact manner with saving multiplier blocks. We also analyze the effects of radix changes on our hardware architecture. The comparison of our hardware performance results with the results of other reported hardware architectures for modular multiplication and exponentiation is also given. The results show that the architectures proposed in this study for modular multiplication and exponentiation yield good performance with a reasonable area in hardware for embedded systems where FPGA comes into play.

The main contribution of this paper is to present a high-speed hardware architecture for modular multiplication and exponentiation using both Karatsuba and Montgomery modular multiplication algorithms with the presented design methods using embedded building blocks on the latest Xilinx FPGA. The proposed method efficiently exploit embedded multipliers and adder units on FPGA. The presented architecture aims to provide an efficient architecture to be used where modular arithmetic is definitely required such as Coding theory, Cryptography, especially PKC, DSP and so on. There are many extensive studies in the literature related to increase the efficiency of multiplication [3–7], especially in PKC. From our point of view, the main advantage of this method over other existing methods is that iterative utilization of hardware resources with pipelining and tight scheduling brings better performance with smaller logic area. Our method also attains very high frequency. The other advantage of our architecture is its scalability which respect to the operand size.

Two levels of scalability are considered:

- The design should require hardware resources as small as possible.
- The design allows multiplication on larger size on the same architecture with small modifications.

In this study, we also propose a system-on-chip (SoC) design which uses our efficient modular arithmetic hardware architecture as a coprocessor with Xilinx's latest Zynq-7000 family extensible processing platform. We adapt Hardware/Software codesign approach in order to exploit the advantages of both methodologies. The design rationale of the proposed architecture is to accelerate the performance of computational intensive tasks required by the modular arithmetic operations. Furthermore, the low latency presented by the coprocessor empowers to operate in higher frequencies within the SoC platforms.

The paper is outlined as follows. Section 2 briefly overviews the literature on the algorithms for modular multiplication and exponentiation operations and their realizations on FPGA devices. After we give the key preliminaries for this study in Section 3, we present our modular arithmetic coprocessor including all hardware aspects in Section 4. We then illustrate our SoC design, which uses our modular arithmetic coprocessor, including the details for the Zynq-7000 based embedded system and its communication mechanism with the coprocessor. In Section 6, we show the performance results of the proposed SoC architecture and discuss our design from all perspectives. Finally, some conclusions are drawn in Section 7.

## 2. Previous work

There are many studies about efficient implementations of modular arithmetic operations included multiplication and exponentiation using various techniques to decrease the computation of these operations, especially in public key cryptography. Montgomery [8] and Karatsuba [9,10] multiplication algorithms are the most efficient and popular algorithms.

High-radix algorithms [7,11] have been proposed for improving the performance. However, as the radix increases, the design complexity and the length of clock cycle also increase dramatically due to requiring the use of larger digit multipliers. These high-radix designs generally consumes huge amounts of hardware area. So, previously, low-radix designs are more attractive for hardware implementation due to the lack of larger digit multipliers. However, dedicated 17-bit embedded multipliers can easily be found in almost all FPGAs today and implementing larger than 17-bit multiplier is also possible with Karatsuba algorithm. By efficiently exploiting Karatsuba algorithm, one can achieve a larger bit multiplier by using less multiplier units. By using Karatsuba algorithm, doubling the bit-width of the multiplier is achieved in less amount of multiplier compared to classical multiplication. Employing Karatsuba algorithm in a pipelined mode allows to perform high-radix Montgomery modular multiplication based on these Karatsuba coprocessors within much less clock cycles. Our aim is to find a compact trade-off between the computation time and the required hardware area by efficiently exploiting Montgomery and Karatsuba algorithms together. Analysis of the design trade-offs for high-radix modular multipliers is found in [12].

Tenca and Koç proposed a radix-2 scalable Montgomery multiplication architecture [13] which multiplicand is scanned word-by-word and the other operand, multiplier, is scanned bit-by-bit. This multiple word radix-2 Montgomery multiplication allows efficient scalable hardware implementation. Parallelism among instructions of the algorithm in different scanning bits of multiplier is possible. The main difference and advantage of the architecture compared to the other algorithms in the literature is its scalability to any operand size which in fact enables to find good design trade-offs for different application needs. Tenca et al. [14] describe a scalable Montgomery multiplier which is adjustable to constrained areas and capable to work on any given precision of the operands. The algorithm proposed in [14] uses Booth encoding technique and a radix-8 scalable multiplier is implemented to show the performance of the design. In contrast to [14], our proposed hardware architectures operate with very high-radix values to reveal the computational efficieny of very high-radix changes in the modular multiplication and exponentiation.