Efficient Implementation of Modular Multiplication using Carry Look Ahead Adder

[1]Muhammad Aqeel Aslam, [2]Ahmad Bilal

Electrical (Electronics) Department of Swedish College of Engineering & Technology
Rahim Yar Khan

Abstract*— Data security is an important aspect of information transmission and storage in an electronic form. Cryptographic systems are used to encrypt such information to guarantee its security. To retrieve such information, the encrypted form must be first decrypted. One of the most popular cryptographic systems is the RSA public key crypto system. The larger the RSA public modulus size, the stronger will be the RSA cryptosystem. Unfortunately, the RSA is extremely vulnerable to timing attacks which can deduce the private RSA exponent due to regularity of operations in the straight forward implementation of exponentiation using the square and multiply method or its variants. Timing attacks constitute a major threat to the all systems using RSA and hence, implementations must be protected. The work reported here proposes Secure Implementation of RSA algorithm against timing attacks. This implementation is done using Verilog HDL and targeting Xilinx FPGA devices.

Keywords— RSA, FPGA, Modular Multiplication, Ripple Carry Adder, Carry Look Ahead Adder

I. INTRODUCTION

Encryption is a well recognized technique for the protection of data and information. It is used effectively to protect sensitive data. The transfer of data from one form to another form, which is unreadable without the secret key, is called encryption. Several techniques have been used for many years. Cryptographic systems are classified into two main categories secret key cryptosystem and public key cryptosystem. Only one key is involved in Secret key cryptosystem. This key is used for both encryption and decryption. However, public key cryptosystem uses two different keys one for encryption and other one for the decryption.

Day by day the significance of FPGA (Field Programmable Gate Array) is increasing. FPGAs are playing very important role in commercial area and research area as well. The technology of FPGA is more affordable as compared to ASICs. FPGAs are reconfigurable platforms. The choice of reconfigurable platforms for cryptographic algorithm appears to be practical and it provides high speed in applications.

In 1978 RSA algorithm was first developed by Rivest, Shamir and Aldeman. RSA is an example of public key algorithm. If the modulus size is large, the algorithm is secure. This algorithm is computationally intensive and it operates on very large integers. RSA algorithm can be used for encryption / decryption and digital signatures. RSA is the most popular method for the public key cryptosystems. The key size determines the security of RSA algorithm. The larger is the key size, more is the security.

II. MODULAR MULTIPLICATION

Multiplication was done by suing Shift and Add method. In Modular Multiplication this requires an additional module of division. Division is the most complex part of the hardware. We switch on to Booth multiplier and 16 bit multiplication results are shown in the following figure. Booth multiplication also has some limitations. Since we are working on the security of the system, so we have already loss some of the speed. There are two methods which are quite useful in Modular Arithmetic. They are Wallace Tree Reduction and Dadda Tree Reduction. Therefore, we proffered Wallace Tree Reduction Method in order to gain some speed in the RSA Algorithm.

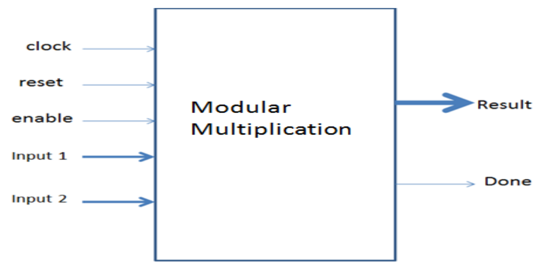

Figure 1.Modular Multiplication

The following diagram shows the complete architecture of the multiplication. In the following figure the multiplicand and the multiplier are of 4 bits each. Partial product array we generated using simply the AND operation of two bits. After generating the partial products array we implemented partial product array reduction by using Wallace tree reduction scheme, instead of dada reduction scheme. After producing partial product array reduction we finally add the result using adder. This addition was done by Carry Look Ahead adder (CLA), as the Carry Look Ahead Adder is the fastest adder. The detail of the implementation of Carry Look Ahead Adder has been shown in the following section.

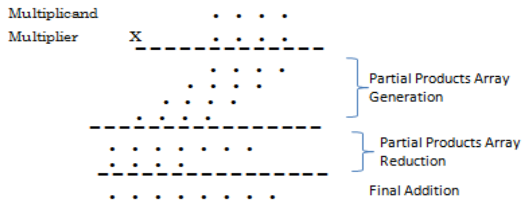
Multiplicand
Multiplier
X
Partial Products Array Generation
Partial Products Array Reduction
Final Addition

Figure 2.Multiplication using reduction scheme

The following diagram shows the simulation results of Shift and Add Multiplication algorithm.

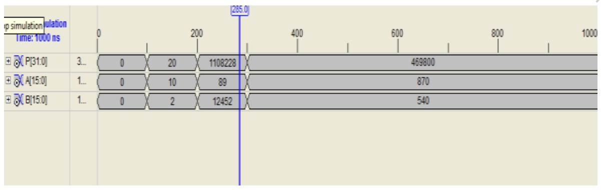

Figure 3.Simulation Result of Shift and Add Multiplication

Figure 4 shows the simulation results of he Montgomery multiplication using Wallace tree reduction scheme.

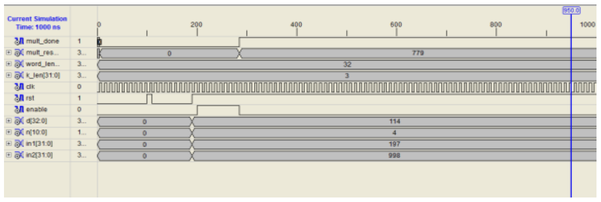

Figure 4.Simulation Result of Multiplication using Wallace Tree Reduction Method

III. Addition

Addition is the basic part of the Arithmetic Logic Unit (ALU), which is used in all other operations. The overall performance depends upon the speed of addition. We started our implementation by using Ripple Carry Adder. After some further literature review it came into notice that Ripple Carry Adder has some drawbacks in terms of delay. Therefore, we switched on to Carry Look Ahead Adder, which has less delay while propagating carry. As already mentioned, that addition operation is the most important and fundamental and it plays a major role in all operations. The delay effect is minimizing by CLA. First, we made a Carry Look Ahead adder of 4 bits, we instantiate it to make a 16 bit adder. We instantiate these 16 bit adder to form the Carry Look Ahead Adder of 64 bits and then these 64 bit adders cascaded to make the required adder of 512 bits.

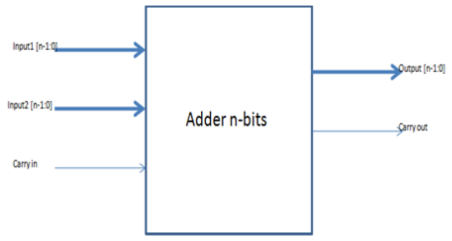

Figure 5.Addition of n bits

The following figure shows the RTL schematic of addition of 64 bits. The results of the Ripple Carry Adder and Carry Look ahead Adder has been shown in the following figure.

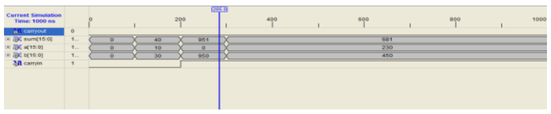

Figure 6.Simulation Result of 64 bit Carry Look Ahead Adder

Figure 7 Simulation Result of 64 bit Ripple Carry Adder

A comparison between Ripple Carry Adder and Carry Look Ahead Adder has shown in the following section.

S. No	*Ripple Carry Adder*	*Carry Look Ahead Adder*
1.	*It is a straight forward way of adding two or more than two bits.*	*Carry Look Ahead Adder calculates the carry bits before the sum.*
2.	*Wait time is much more because carry is calculated alongside sum.*	*It reduces the wait time.*
3.	*In 16 bit Ripple Carry Adder, we find* *CPU : 2.89 / 3.06 s | Elapsed : 3.00 / 3.00 s*	*In 16 bit Carry Look Ahead Adder we get* *CPU : 2.90 / 3.07 s | Elapsed : 3.00 / 3.00 s*
4.	*In Ripple Carry Adder every carry out has to be carry in of the next stage.*	*Carry Look Ahead Adder uses the logic of generating and propagating carries.*

Table 1 Comparison between Adders

IV. Results

In this section we have shown the comparison of different implementations of Modular Multiplication. We have implemented modular multiplication using Carry Look Ahead Adder. The following table is clearly showing that this implementation is far fast as compare to the previous implementations.

V. Conclusion and Future Work

We have implemented the modular multiplication which was the fastest due to Carry Look Ahead adder. We can enhance the speed of the addition operation by implementing the combination of adders. If we use combination of two adders for the addition the speed of the final result will be enhanced and time will be reduced significantly.

VI. References

[1]. Secure Implementation of RSA Algorithm against Timing Attacks, MS Thesis Project, Department of Electronic and Power Engineering, College of Marine Engineering (PNEC), National University of Sciences and Technology, Islamabad

[2]. Implementation of Efficient Modular Exponentiation on Reconfigurable Platforms, August 2008, Kashif Latif, Department of Electronic and Power Engineering, College of Marine Engineering (PNEC), Karachi, National University of Sciences and Technology, Islamabad

[3]. Efficient Hardware Design and Implementation of AES Cryptosystem, Pravin B. Ghewari et al. International Journal of Engineering Science and Technology, Vol. 2(3), 2010, 213-219

[4]. Carry-Save Montgomery Modular Exponentiation on Reconfigurable Hardware, A. Cilardo, A. Mazzeo, L. Romano, G. P. Saggese, Universit`a degli Studi di Napoli Federico II, Italy, Volume 3, IEEE Computer Society Washington, DC, USA @ 2004

[5]. A Review of Modular Multiplication Methods and Respective Hardware Implementations, Nadia Nedjah, Department of

Electronics Engineering and Telecommunications, Engineering Faculty, Informatica **30** (2006) 111–129 **111**

[6]. Montgomery Algorithm for Modular Multiplication, Professor Dr. D. J. Guan, August 25, 2003
[7]. Modular Multiplication using Montgomery method, Haoxin (Larry) Song, ECE 543 Final Project Report