International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.2, Issue No.6, pp : 488-491 1 June 2013

A Novel Architecture for an efficient data encryption system

Bhaskar.R[1], Ramakrishna.A[2], Venkateshwarlu.K[3], Haribabu.M[4]

[1,2,3,4] Department of Electrical & Electronics Engineering, Vardhaman College of Engineering, Shamshabad, Hyderabad, India.

[1] bhaskar767@gmail.com [2] adiraju2662@gmail.com, [3] koppolu.06@gmail.com, [4] haribabu3e@gmail.com

Abstract — The standard techniques for providing privacy and security in data networks include encryption/decryption algorithms such as Advanced Encryption System (AES) (private-key) and RSA (public-key). RSA is one of the safest standard algorithms, based on public-key, for providing security in networks. Even though the RSA Algorithm is an old and simple encryption technique, there is a scope to improve its performance.

One of the most time consuming processes in RSA encryption/ decryption algorithm is the computation of a^b mod n where “a” is the text, (b, n) is the key. Generally the prime number used for RSA Encryption system will around 100 to 150 decimal digits. The computations involved are tedious and time consuming. Also the hardware is quite complex. To increase the computation speed, the multiplication principle of Vedic mathematics is used and also an improvement is made in the conventional restoring algorithm which does the modulus operation.

“Urdhva-tiryakbhyam” is the sutra (principle) which used to compute the multiplication. It literally means vertical and crosswise manipulation. The significance of this technique is that it computes the partial products in one step and avoids the shifting operation which saves both time and hardware. Also an improvement is made in the restoring algorithm by avoiding unnecessary restorations when they are not required.

The Verilog HDL code is simulated and synthesized in ModelSim 6.5 and Xilinx ISE 12.1 version tools respectively.

I. INTRODUCTION

The RSA Algorithm is the most popular and proven asymmetric key cryptographic algorithm. In 1977, Ron Rivest, Adi Shamir and Len Adleman at MIT developed the first major asymmetric key cryptography system. Hence it is called as RSA Algorithm Nowadays, more and more reconfigurable hardware devices are used in network applications due to their low cost, high performance and flexibility. Such applications include extensible network routers, firewalls and Internet-enable sensors, etc [1]. These reconfigurable hardware devices are usually distributed in a large geographic area and operated over public networks, making on-site configuration inconvenient or infeasible.

Therefore, robust security mechanisms for remote control and configuration are highly needed. The RSA algorithm is a secure, high quality, public key algorithm. It can be used in these applications as a method of exchanging secret information such as keys and producing digital signatures. However, the RSA algorithm is very computationally intensive, operating on very large (typically thousands of bits long) integers. The RSA algorithm has been adopted by many commercial software products and is built into current operating systems by Microsoft, Apple, Sun, and Novell. Commercial Application Specific Standard Products (ASSPs) like the security processors offered by several vendors have a much higher RSA performance than software implementation. However, their solution is inflexible and expensive. With the exponential increase in FPGA size over time (Moore's law), it is possible to implement a relatively high performance, user parameterizable RSA at low cost on FPGA [2].

II.VEDIC MULTIPLICATION

Vedic mathematics is mainly based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc. The multiplier is based on an algorithm Urdhva Tiryakbhyam (Vertical & Crosswise) of ancient Indian Vedic Mathematics. Urdhva Tiryakbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. It literally means “Vertically and crosswise”. It is based on a novel concept through which the generation of all partial products can be done with the concurrent addition of these partial products [2]. Vedic multiplier is faster than array multiplier and Booth multiplier. As the number of bits increases from 8x8 bits to 16x16 bits, the timing delay is greatly reduced for Vedic multiplier as compared to other multipliers. Vedic multiplier has the greatest advantage as compared to other multipliers over gate delays and regularity of structures. Delay in Vedic multiplier for 16 x 16 bit number is 19 ns while the delay in Booth and Array multiplier are 37 ns and 43 ns respectively. Thus this multiplier shows the highest speed among conventional multipliers. It has this advantage than others to prefer a best multiplier [3].

III. IMPROVED RESTORING ALGORITHM

The number of restorations performed in the standard restoring division algorithm can be reduced by the following technique which is called as “Needy restoring division algorithm”. The needless restorations are removed by the use of an extra register. This removal decreases the time for the division. Current restoring division algorithm checks restoration at each iteration after shifting the register. There are some needless restorations which are not required to be checked: 1. Checking restoration until the first set bit of the dividend is achieved. 2. Checking restoration each time until the value of the register (here, the left half of the Remainder register) equals or becomes greater than the Divisor. These needless restorations are removed by the use of an extra register. Therefore, three registers are used here. One is for storing Divisor, second is the Remainder register and third one is M register (the extra register). Divisor register is a 32 bits register. M is also a 32 bits register of which only that bit is on, which is the first set bit of the Divisor from the left, e.g. if Divisor is 00001010 then M will be 00001000 or if Divisor is 00111001 then M is 00100000. [4]. The Remainder register is a 64 bits register which initially contains the dividend. The right half of the remainder register gives the quotient and left half gives the remainder. The first type of needless restorations (i.e. until the first set bit of dividend is achieved) are left unchecked by first shifting the Remainder register left continuously till the left half of the Remainder register is zero. e.g. Considering an eight bit ALU model, let, if the dividend is 00001001 then remainder register will be 00000000 00001001 and left half of Remainder register will be 1 or non zero after five times shifting the Remainder, so there is no need of restorations for doing this. It should be done by first shifting the Remainder left continually till the left half is zero and then move further. The second needless restorations are removed by using the register M. It is done by the checking the result for logical AND of Left half of Remainder register and the M register. It is just like that as if we are checking and counting how many bits are required to make the left half of Remainder register equal or greater than the
Divisor and then shifting the Remainder register that much times. There is no need of restorations for doing this by using the M register. E.g. Considering an eight bit ALU model, if the Divisor is 00000110 then M is 00000100. The M register can get its required value while shifting the Remainder register left continuously till the left half of the Remainder register is zero. This can be done independently and in a parallel manner. Suppose Remainder is 00000000 00001001, then we can continually shift the Remainder register left till (Left half of Remainder & M) is zero. There is no need of restoring the left half of Remainder register during this work. After this the Remainder will be 00000100 10000000. The left half of the Remainder register is still less than the Divisor, so now the subtraction will be performed and the restoration will be done. But the special point is that only one or nil restoration will be needed in each case after removing the second needless restorations. In this way the current restoring division algorithm is improved as there are less numbers of restorations for doing the division. The efficiency is best improved when the dividend is very small or divisor is very large. The algorithm steps for the 8 bit dividend can be summarized as follows:
Remainder register initially contains dividend. and initialize N=0. M is a register whose only that bit is on, which is the first set bit of Divisor from the left, e.g. Divisor is 00001010 then M will be 00001000. Shift the Remainder register left 1 bit, and increment N. If left half of remainder=0, then go to step2; otherwise step4.
If the “and” operation of left half of the remainder and M=0, then go to step5; otherwise step6. Shift the remainder register by 1 bit and increment N. If N=9, go to step8. Subtract the Divisor register from the left half of the Remainder register and place the result in the left half of the Remainder register. If remainder>=0 then shift the Reminder register to the left, setting the new rightmost bit to 1, and increment the value of N ; otherwise Restore the original value by adding the Divisor register to the left half of the Reminder register and place the sum in the left half of the Remainder register. Then go to step4. Continue the iterations till N=9. Shift the left half of the remainder to the right by 1 bit to obtain the final remainder.

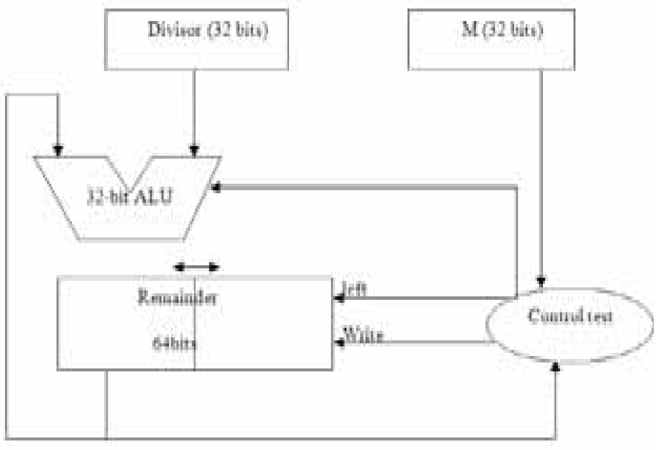


Fig. 1 Division Hardware with register M

IV. RSA ENCRYPTION SYSTEM

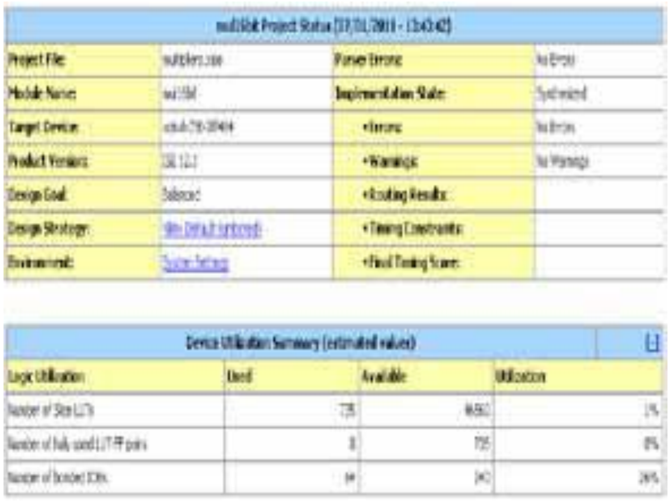
The RSA Algorithm is based on the mathematical fact that it is easy to find and multiply the large prime numbers together, but it is extremely difficult to factor their product [5]. The public and private keys in RSA are based on very large prime numbers. The algorithm is simple but the complexity lies in the selection and generation of public and private keys. The algorithm steps are as follows:

- Choose two large prime numbers P and Q.
- Calculate N=PxQ.
- Select the private key (i.e., the encryption key) E such that the GCD of E and the product of
- (p-1), (q-1) is equal to 1.
- Calculate the cipher text CT from the plain text PT as follows: CT=PTE mod N.
- Send CT as the cipher text to the receiver.
- Select the private key (i.e., the decryption key) D such that following equation is true:
- (DXE) mod (P-1) (Q-1) =1.
- For decryption, calculate the plain text PT from the cipher text CT as follows:
- PT=CT Dmod N. [6]

V. SIMULATION RESULTS

The fig 2 shows the simulation results of the 16 bit Vedic multiplier obtained using the Model Sim 6.5 and the fig.3 shows the corresponding design summary obtained using Xilinx 12.1 ISE tool.

Fig.2 Simulation results of Vedic multiplication



mul16bit Project Status (17/01/2013 - 13:43:42)			
Project File:	multipliers.xise	Parser Errors:	No Errors
Module Name:	mul16bit	Implementation State:	Synthesized
Target Device:	[illegible]	•Errors:	No Errors
Product Version:	ISE 12.1	•Warnings:	No Warnings
Design Goal:	Balanced	•Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	•Timing Constraints:	
Environment:	System Settings	•Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	775	4656	16%
Number of fully used LUT-FF pairs	0	775	0%
Number of bonded IOBs	64	240	26%

Fig.3 Design summary of Vedic multiplication

The Vedic multiplier is fast when compared to the standard multipliers. The comparison of different multipliers is shown in the table 1. [7]

Name of Multiplier	Array Multiplier		Booth Multiplier		Vedic Multiplier	
	8x8 Bit	16x16 Bit	8x8 Bit	16x16 Bit	8x8 Bit	16x16 Bit
Delay (ns)	47	92	36	69	11	19

Table1: Comparison of different multipliers

The data is encrypted using RSA algorithm. Instead of using a conventional multiplication, the Vedic multiplication algorithm is used to increase the computation speed. Different 8 bit messages are taken and encrypted using the RSA algorithm. The simulation results are shown in fig 4. The maximum combinational path delay is 69.686 ns. All the 8 bit messages that are encrypted using the RSA encryption algorithm are obtained by finding the exact value of the private key using the appropriate equation. The fig 5 shows the simulation results of the decryption. The maximum combinational path delay obtained after synthesizing is equal to 71.23 ns. The table 2 shows the comparison of RSA Encryption system using the Vedic multiplier, improved restoring division algorithm and without using improved restoring division algorithm.

Fig.4 Simulation results of RSA Encryption

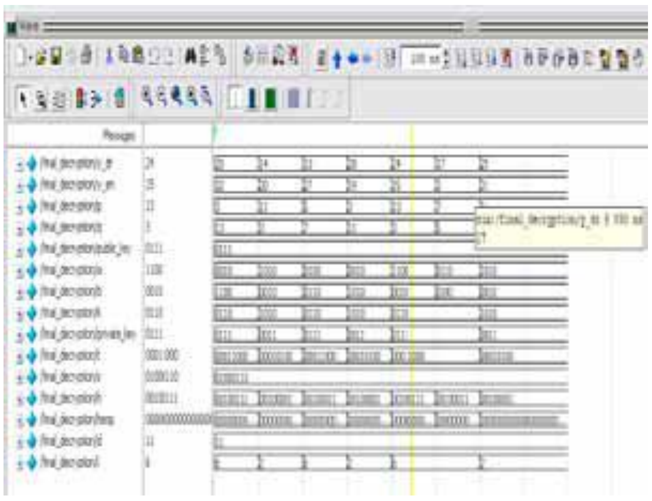


Fig.5 Simulation results of RSA Decryption

RSA Encryption System employing Vedic Multiplier	Delay in (ns)	
without improved restoring division algorithm	69.68	71.23
with improved restoring division algorithm	58.89	61.20

Table 2: Timing simulation results of RSA architecture

VI .CONCLUSION

The RSA encryption system is implemented using the Vedic Multiplier and the improved restoring division algorithm to increase its computation speed. The advantage of the Vedic multiplier is that it calculates the partial products in one single step and there are no shift operations which saves the time and the hardware. In case of the improved restoring division algorithm the unnecessary computations are avoided by comparing the dividend and the divisor. Therefore these architectures are included in the 8 bit RSA encryption system to evaluate its performance and it is found that it is efficient in terms of speed and hardware. As the number of message bits increases the gate delay as well as the area increase slowly. Hence it can be used effectively in all the cryptographic applications

ACKNOWLEDGMENT

The authors thank the management of Vardhaman College of Engineering for providing the necessary facilities to undertake the above work.

REFERENCES

[1] Himanshu Thapliyal and M.B Srinivas, “VLSI Implementation of RSA Encryption System Using Ancient Indian Vedic Mathematics”, Center for VLSI and Embedded System Technologies, International Institute of Information Technology Hyderabad-500019, India.

[2] Shamim Akhter, “VHDL Implementation of Fast NXN Multiplier Based on Vedic Mathematics”, Jaypee Institute of Information Technology University, Noida, 201307 UP, INDIA, 2007 IEEE.

[3] Himanshu Thapliyal and M.B Srinivas “High speed Efficient Hierarchial Overlay multiplier based on Ancient Indian Vedic mathematics”, proceedings of International Conference on Signal Processing, ICSP 2004, Turkey 2004.

[4] Nitish Aggarwal, Kartik Asooja, Saurabh Shekhar Verma, Sapna Negi,” An Improvement in the Restoring Division Algorithm”, IEEE 2009.

[5] R. Rivest, A. Shamir and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, Communications of the ACM, 21 (2), pp. 120-126, February 1978.

[6] Behrouz A Forouzan, Debdeep Mukhopadhyay, ”Cryptography and Network Security”, second edition, Tata McGraw Hill Education Pvt Ltd, 2008.

[7] Sumit Vaidya, Deepak Dandekar, “Delay-Power Performance Comparison of multipliers in VLSI circuit design”, International Journal of Computer Networks & Communications (IJCNC), Vol.2, No.4, July 2010.