International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
NCICCT' 14 Conference Proceedings

Modified Modular Exponentiation for a Faster Implementation of RSA algorithm on FPGA

M.R. GAUTHAMA RAMAN
Dept. of Electronics and Communication Engineering
B.S. Abdur Rahaman University, India
gauthamaraman_mr@yahoo.com

Dr. S. Kaja Mohideen
Prof & Dean (ECS)
Dept. of Electronics and Communication Engineering
B.S. Abdur Rahaman University, India
dean_secs@bsauniv.ac.in

Abstract - **RSA algorithm is a well known, commonly used public key cryptography for the secured data communication. In this paper RSA cryptosystem is used to achieve the secured communication between the multiple FPGAs using RS 232 link. Currently FPGAs are used for the implementing the various embedded applications since it provides the chance for the reconfiguration and also for the change in resources used. The Verilog modelling of this RSA algorithm uses the repeated addition and subtractions for the faster implementation and uses the less area in the FPGA. First encryption and decryption is done on a single FPGA system and later it is extended to two where each running the separate encryption and decryption algorithm.**

I. Introduction

Field Programmable Gate Arrays (FPGAs) were first introduced almost two and a half decades ago. Since then they have seen a rapid growth and have become a popular implementation media for digital circuits. The advancement in process technology has greatly enhanced the logic capacity of FPGAs and has in turn made them a viable implementation alternative for larger and complex designs. Further, programmable nature of their logic and routing resources has a dramatic effect on the quality of final device's area, speed, and power consumption. Now a days the application of the FPGAs are extended to embedded fields and various research works are been carried out which uses the FPGA as a basic platform of the embedded systems. If we consider the field of the data communication, providing the data security is the major challenge. Currently various researches focus on the development of new hardware security modules which provide the trusted communication and also concentrates of the power consumption and also the processing speed. Among the existing, RSA algorithm is been chosen since it has major advantage in the parallel computing as compared with the AES.

II. RSA Algorithm

RSA Algorithm was developed at MIT by Ron Rivest, Adi Shamir, and Len Adleman during the year of 1977. It is a public key cryptographic type in which two different and related keys used for the encryption and decryption process. The task of the RSA algorithm is to compute the reminder of the exponential term. The encryption of the plain text(M) to the cipher text (C) is obtained by

$$C=M^e \bmod n$$

The plain text (M) is been again obtained by

$$M=C^d \bmod n$$

The Steps involved in RSA Algorithm is

- Choose p,q (both should be a prime number & p,q are not same)
- Compute $n = p * q$
- Compute $\varphi(n) = (p - 1) * (q - 1)$
- Choose e such that $1 < e < \varphi(n)$ and e and $\varphi(n)$ are coprime.
- Compute a value for d such that $(d * e) \% \varphi(n) = 1$.
- Public key is (e, n)
- Private key is (d, n)
- The encryption process $C=M^e \bmod n$
- The decryption process $M=C^d \bmod n$

III. Basic Operation

In order to implement the RSA Algorithm for the large integer values, we must concentrate on the calculation of the reminder value for the exponential function both in case of encryption and decryption. The first rule of modular exponentiation is that we do not compute $C = M^e \pmod n$ by first exponentiating M^e and then performing a division to obtain the remainder $C = (M^e) \% n$. The temporary results must be reduced modulo n at each step of the exponentiation. This is because the space requirement of the binary number M^e is enormous. Assuming, M and e have 256 bits each, we need bits in order to store M^e.

$$\log_2(M^e) = e.\log_2(M) \approx 2\text{^}256.256 = 2\text{^}264 \approx 10\text{^}80$$

This number is approximately equal to the number of particles in the universe. we have no way of storing it. In order to compute the bit capacity of all computers in the world, we can make a generous assumption that there are 512 million computers, each of which has 512 MBytes of memory. Thus, the total number of bits available would be which is only enough to store Me when M and e are 55 bits.

$$512 \cdot 2^{20} \cdot 512 \cdot 2^{20} \cdot 8 = 2^{61} \approx 10^{18}$$

Let us find, How many modular multiplications are needed to compute M^e mod n. A naive way of computing C = M^e (mod n) is to start with C = M(mod n) and keep performing the modular multiplication operations C = C . M (mod n) until C = M^e (mod n) is obtained.

The naive method requires e - 1 modular multiplications to compute C = M^e (mod n), which would be prohibitive for large e. For example, if we need to compute M^15 (mod n), this method computes all powers of M until 15:

$$M \rightarrow M^2 \rightarrow M^3 \rightarrow M^4 \rightarrow M^5 \rightarrow M^6 \rightarrow M^7 \rightarrow \ldots\ldots\ldots \rightarrow M^{15}$$

which requires 14 multiplications. However, not all powers of M need to be computed in order to obtain M^{15}. Here is a faster method of computing M^{15}

$$M \rightarrow M^2 \rightarrow M^3 \rightarrow M^6 \rightarrow M^7 \rightarrow M^{14} \rightarrow M^{15}$$

which requires 6 multiplications. The method by which M^{15} is computed is not specific for certain exponents; it can be used to compute M^e for any e. The algorithm is called the square and multiply method. By using this the entire operation can be performed faster and simpler.

The proposed algorithm is the modified form of RSA Algorithm with handles the exponent terms and also the mod function in a most efficient way.

In this algorithm bits of e are scanned two at a time, the possible digit values are (00) = 0, (01) = 1, (10) = 2, and (11) = 3. The multiplication step (Step 4b) may require the values M0, M1, M2, and M3. Thus, we need to perform some pre-processing to obtain M2 and M3. As an example, let e = 250 and partition the bits of e in groups of two bits as
e = 250 = 11 11 10 10

Here, we have s = 4 (the number of groups s = k=r = 8=2 = 4). During the pre-processing step, we compute:

BITS	w	M^w
00	0	1
01	1	M
10	2	$M.M = M^2$
11	3	$M^2.M=M^3$

The method then assigns C = M^{F3} = M^3 (mod n), and proceeds to compute M^{250} (mod n) as follows:

i	F_i	Step 4a	Step 4b
2	11	$(M^3)^4=M^{12}$	$M^{12}.M^3=M^{15}$
1	10	$(M^3)^4=M^{12}$	$M^{12}.M^3=M^{15}$
0	10	$(M^3)^4=M^{12}$	$M^{12}.M^3=M^{15}$

The number of modular multiplications required by this method for computing M^250 (mod n) is found as 2 + 6 + 3 = 11.

A. *Steps for Encryption:*

```
STEP 1: Decompose e
STEP 2: C=(M^Fs-1) mod n
STEP 3: for i=s-2 to 0
            C=C^2r mod n
            if Fi!=0 then C=C*(M^Fi)mod n
            return C
```

B. *Steps for Decryption:*

```
STEP 1: Decompose d
STEP 2: M=(C^Fs-1) mod n
STEP 3: for i=s-2 to 0
            M=M^2r mod n
            if Fi!=0 then M=M*(C^Fi)mod n
            return M
```

III. ARCHITECTURE DESIGN

A. *Xilinx Platform Studio:*

The Xilinx Platform Studio (XPS) is the development environment or GUI used for designing the hardware portion of your embedded processor system.

B. *Embedded Development Kit :*

Xilinx Embedded Development Kit (EDK) is an integrated software tool suite for developing embedded systems with Xilinx Micro Blaze and PowerPC CPUs. EDK includes a variety of tools and applications to assist the designer to develop an embedded system right from the hardware creation to final implementation of the system on an FPGA. System design consists of the creation of the hardware and software components of the embedded processor system and the creation of a verification component is optional. A typical embedded system design project involves: hardware platform creation, hardware platform verification (simulation), software platform creation, software application creation, and software verification. Base System Builder is the wizard that is used to automatically generate a hardware platform according to the user specifications that is defied by the MHS (Microprocessor Hardware Specification) file. The MHS file defines the system architecture, peripherals and embedded processors]. The Platform Generation tool creates the hardware platform using the MHS file as input. The software platform is defied by MSS (Microprocessor Software Specification) file which defines driver and library customization parameters for peripherals, processor customization parameters, standard 110 devices, interrupt handler routines, and other software related routines. The MSS file is an input to the Library Generator tool for customization of drivers, libraries and interrupts handlers.

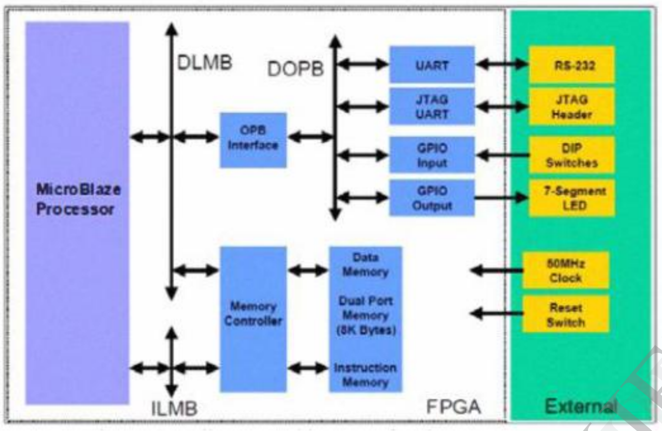


Fig. 1.1 FPGA Architecture

The creation of the verification platform is optional and is based on the hardware platform. The MHS file is taken as an input by the Simgen tool to create simulation files for a specific simulator. Three types of simulation models can be generated by the Simgen tool: behavioural, structural and timing models. Some other useful tools available in EDK are Platform Studio which provides the GUI for creating the MHS and MSS files. Create / Import IP Wizard which allows the creation of the designer's own peripheral and import them into EDK projects. Platform Generator customizes and generates the processor system in the form of hardware net lists.

Library Generator tool configures libraries, device drivers, file systems and interrupt handlers for embedded processor system. Bit stream Initialize tool initializes the instruction memory of processors on the FPGA. The Architecture of FPGA is shown in the Fig 1.1. GNU Compiler tools are used for compiling and linking application executables for each processor in the system . There are two options available for debugging the application created using EDK namely: Xilinx Microprocessor Debug (XMD) for debugging the application software using a Microprocessor Debug Module (MDM) in the embedded processor system, and Software Debugger that invokes the software debugger corresponding to the compiler being used for the processor.

C. *Software Development Kit:*

Xilinx Platform Studio Software Development Kit (SDK) is an integrated development environment, complimentary to XPS, that is used for C/C++ embedded software application creation and verification. SDK is built on the Eclipse opensource framework. Soft Development Kit (SDK) is a suite of tools that enables you to design a software application for selected Soft IP Cores in the Xilinx Embedded Development Kit (EDK).The software application can be written in a "C or C++" then the complete embedded processor system for user application will be completed, else debug & download the bit file into FPGA. Then FPGA behaves like processor implemented on it in a Xilinx Field Programmable Gate Array (FPGA) device.

D. *Serial Communication:*

The system that is used for establishing the serial communication between the multiple FPGA systems is UART (Universal Asynchronous Receiver Transmitter). The Block diagram of UART is shown in Fig 1.2. Here "BRG" stands for "Baud Rate Generator" which controls the speed of the data communication in RS232 channel. Both receiver and sender side must work in the same band ratio otherwise data will be lost. BRG control the received data store initially at received FIFO and the transmit Data FIFO transfer the data through the transmitter Module (TX Module).

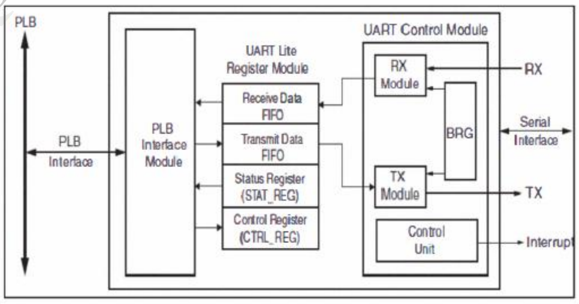
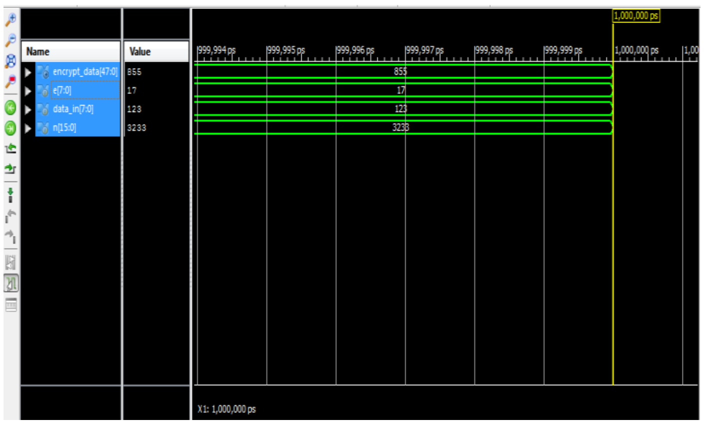


Fig. 1.2 Universal asynchronous receiver transmitter systems

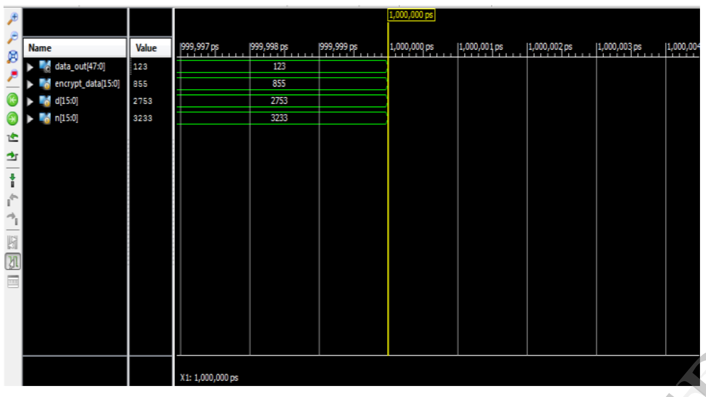
V. RESULTS

The public key is chosen to be (n=3233, e=17) and private key is chosen to be (n=3233d=2753). The data that has to be transferred is 123. The encrypted value of the data is 855. This encryption process is done using the public key similarly 855 is decrypted to 123 using private key. The simulation results for both encryption and decryption is shown in Fig.1.3 and Fig.1.4



Name	Value
encrypt_data[47:0]	855
e[7:0]	17
data_in[7:0]	123
n[15:0]	3233

1.3 Simulation result for encryption



Name	Value
data_out[47:0]	123
encrypt_data[15:0]	855
d[15:0]	2753
n[15:0]	3233

1.4 Simulation result for decryption

A. Comparison with Existing Work:

Since the proposed algorithm uses less number of multiplications, the process speed is high as compared with the existing systems and from the table 1.1, we can find that this architecture uses less number of slices used compared with existing system.

Resource Used	Proposed System	Existing System
4 Input LUT's	1403	2871

Table 1.1 Comparison between proposed and existing system

VI. Conclusion

A new architecture is been proposed in this paper which uses less number of modular multiplication so that high speed of data encryption and decryption is been achieved compared with the existing system. This way of approach can also be implemented in difference cryptographic algorithm which provides the higher level of security in the data transmission.

References

[1]Rourab Paull,Sangeet Saha,Suman Sau,Amlan Chakrabarti. Real time communication between multiple FPGA systems in multitasking environment using RTOS . Processding of the International Conference on Devices, Circuits and Systems (ICDCS), 2012 ISBN:978-1-4577-1545-7

[2]Zutter J. And Martin Klein, Acceleration of RSA Cryptographic Operations using FPGA Technology. Proceeding of the 20th International Workshop, Database and Expert Systems Application, 2009. DEXA '09 ISSN : 1529-4188

[3] Garg R. And Vig, R. An Efficient Montgomery Multiplication Algorithm and RSA Cryptographic Processor. Proceeding of the Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on (Volume:2) ISBN: 0-7695-3050-8

[4]A Mazzeo, L. Romano, G. P. Saggese and N. Mazzocca. 2003. FPGABased Implementation of a Serial RSA Processor. Design. Proceedings of the conference on Design, Automation and Test in Europe - Volume I. ISBN:O- 7695- 1870-2.

[5] Montgomery Algorithm for Modular Multiplication Professor Dr. D. J. Guan ,August 25, 2003.

[6] C. D. Walter. August 1999. Montgomery's Multiplication Technique: How to Make It Smaller and Faster. Cryptographic Hardware and Embedded.

[7] Cryptography & Network Security By Behrouz AForouzan.