Architectural Analysis of RSA Cryptosystem on FPGA

Vibhor Garg
School of Electronics Engineering
VIT University
Vellore, India

V. Arunachalam
School of Electronics Engineering
VIT University
Vellore, India

ABSTRACT
This paper presents different architectures in FPGA based implementations of a public key crypto algorithm - RSA algorithm. A hardware-based cryptographic system is preferred as it provides - better security, integrity and is resistant to power analysis attacks [1]. After the complete cryptosystem is simulated in VERILOG [8] and synthesized for specific XILINX FPGAs, the architecture of the cryptosystem is evolved by performing scheduling in the Data Flow Graph. This way there are two types of architectures realized: – one with high concurrency (which takes lesser number of clock cycles) and the other with maximum sequential operations. Subsequently the size of the key is extended and its effects on the architecture, with respect to area and power consumed, are observed. Finally trade-off analysis of the various implementations is done.

KEYWORDS
RSA Encryption, Public Key Cryptography, FPGA, security, Verilog, Data Flow Graph.

1. INTRODUCTION
With the ever increasing popularity of Electronic Communication and the Internet, data security has become of more concern now days. In 1978, Rivest, Shamir and Adleman proposed the RSA public-key cryptosystem which has become the most widely used public-key cryptosystem due to the fact that it can be used for both data encryption and authentication [9]. In RSA cryptosystem we require two keys, the public and private key. The public key is advertised to the world and the private key is kept secret. Therefore an anonymous person will not be able do decrypt the encrypted message if he does not have the private key. The safety depends upon the length of the key, longer the key-length much safer is the data.

The RSA cryptosystem is briefly described as follows [9]:

- Select two random prime numbers p and q
- Calculate $n = p \times q$
- Calculate $ø(n) = (p – 1) \times (q – 1)$
- Select integer e such that $gcd\ (ø(n),e) = 1;\ 1<e< ø(n);$ where e & $ø(n)$ are relatively prime [11], [12]
- Calculate $d = e^{-1} \bmod ø(n)$
- Encryption is done as $C = M^e \bmod n$
- Decryption is done as $M = C^d \bmod n$

The encryption and decryption contain the modular exponentiation operation which is the most critical in RSA. The implementation of various architectures would provide for flexibility in the end users design. This would help him design an architecture based solely upon his requirements and the application he would be using. For example, the application used for online transactions would require higher security and faster execution and hence the user may opt for a higher bit security system that operates concurrently.

This paper is organized as follows. Section 2 of this paper describes the previous work done in this field. In section 3 the DFG synthesis is described. The DFG scheduling is described in section 4. The simulation results are described in section 5 followed by the conclusion in section 6.

2. PREVIOUS WORK
During the past years, numerous papers dealing with RSA public key cryptography have been published [2], [3], [4], [10]. However, the design presented in this paper differs from previous work in two important aspects: (1) We pay special attention not only to any individual module but to the system as a whole. (2) We work on both the arithmetic and the functional level with which we provide flexible design specifications to the user making his cryptosystem suitable for a large range of applications.

Firstly, the various papers that have been published focus on any particular module of the RSA cryptosystem and to be more specific majorly on the modular exponentiation (encryption/decryption) as it is the core operation of RSA. Our design in this paper focuses on the complete cryptosystem and also the scheduling of functional units at the arithmetic level of particular modules, which provides different architectures with flexibility in the performance specifications of the system.

A second point in which our design differs from previous work is scalability, i.e. the ability given to the user to select his required modules and integrate them together to target efficiently his application and the availability of resources. This provides for the user to have a very flexible design based upon his requirements.

3. DATA FLOW GRAPH SYNTHESIS
A Data Flow Graph (DFG) is a graphical representation of the "flow" of data through an information system and can also be used for the visualization of data processing [6]. After the identification of all the various modules required in the different processes of the RSA algorithm, data flow graphs of individual modules were first constructed and then connected together.

The identified modules which have been implemented in the RSA algorithm are described as below:

- Public Key Generation
- Encryption
- Private Key Generation
- Decryption

3.1 Public Key Generation
Initially ‘p’ and ‘q’ are randomly generated prime numbers.

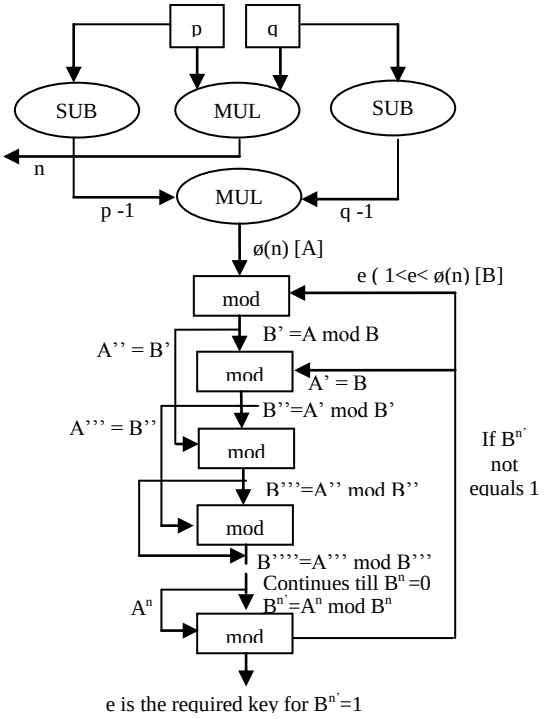

Fig 1: DFG for Public Key Generation using Euclid's Algorithm

3.2 Encryption

For Encryption we have to calculate '$c = m^e \bmod n$'

$n = p \times q$ and ($e = \sum_{i=0}^{m-1} 2^i \times e_i$).

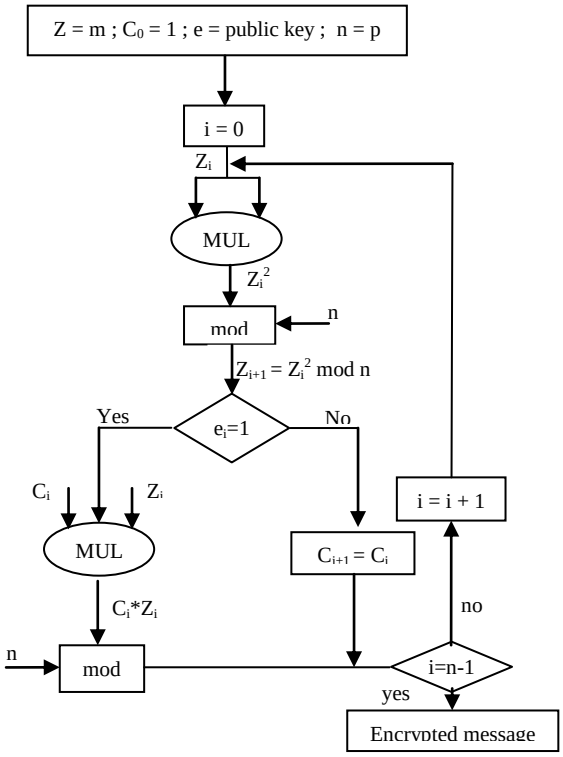

Fig 2: DFG for Encryption using Square and Multiply Algorithm

3.3 Private Key Generation

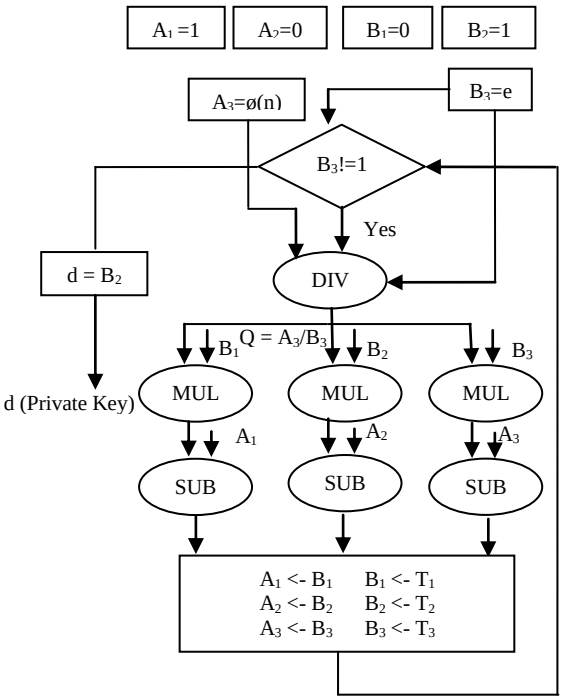

Fig 3: DFG for Private Key Generation using Extended Euclid's Algorithm

3.4 Decryption

For Decryption we have to calculate '$M = c^d \bmod n$'

$n = p \times q$ and ($d = \sum_{i=0}^{n-1} 2^i \times d_i$)

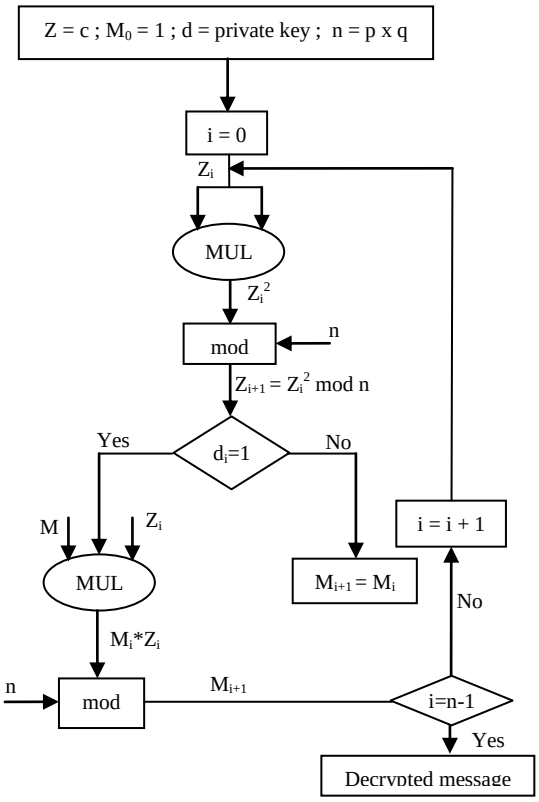

Fig 4: DFG for Decryption using Square and Multiply Algorithm

4. DFG SCHEDULING

Scheduling is done on any process due to limited resources or to meet performance standards. After the formation of the Data Flow Graph (DFG), certain scheduling has to be done in order to perform a trade off analysis between Time, Area and Power. DFG scheduling is done on two levels:

- Arithmetic Level
- Functional Level

At the arithmetic level we work on the arithmetic operators within a module. We make necessary changes in order to optimize the performance (in regards to time, area and power). In arithmetic level scheduling we design various modules of the RSA system in two different modes namely: sequential and parallel. In this regard we make changes in the Extended Euclid's algorithm by changing the three parallel multipliers being used in to a sequential logic thus reducing the number of multipliers but increasing the critical path length.

In the functional level scheduling we deal with the implementation of different algorithms for the modules forming the cryptosystem and executing entire modules in parallel if they are not data dependent on each other. We choose algorithms based on the criterion of efficiency with respect to critical path, area and power consumed. At the functional level scheduling we perform the modification of the encryption and decryption modules with different algorithms. This is done as modular exponentiation, which is used in both encryption and decryption, is hugely responsible for the efficiency of the entire cryptosystem. The algorithms applied in relation to the aforementioned are namely: Square and Multiply algorithm [3] and Montgomery algorithm [7], [5].

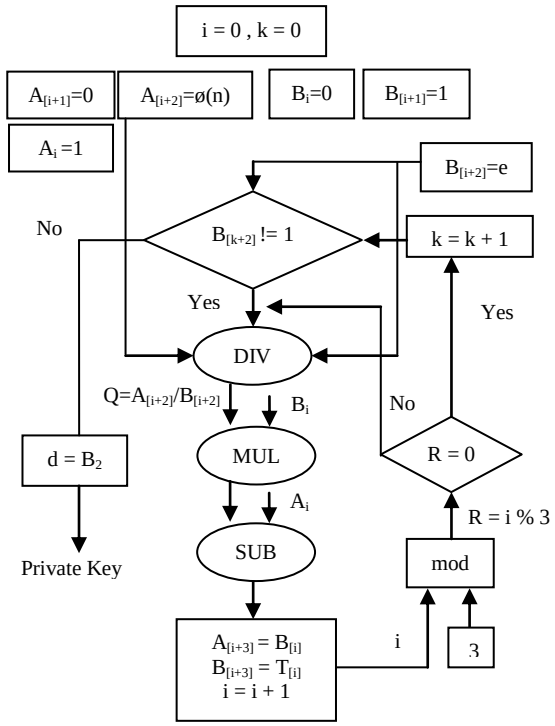

Fig 5: Arithmetic level scheduling in Private Key Generation (Extended Euclid's Algorithm)

Another area where functional level scheduling is applied is the parallel execution of the private key generation and encryption modules. This is possible due to the non-existence of data dependency between them.

The following figures explain the scheduling done in the Data Flow Graphs.

This is where the three parallel multipliers in the Extended Euclid algorithm are replaced by only one sequential multiplier.

The next scheduling is done at the functional level where the entire module of Square and Multiply algorithm is replaced by Montgomery algorithm.

To calculate $A \times B \bmod n$ where $B = \sum_{i=0}^{n-1} 2^i \times B$

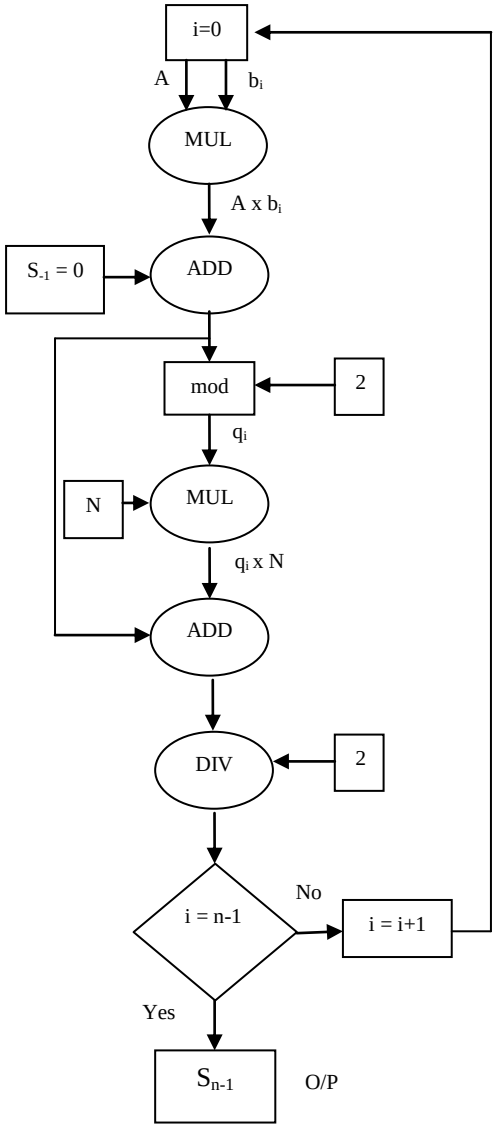

Fig 6: Montgomery multiplication, a pre-requirement for Montgomery exponentiation

To calculate $C^e \bmod n$ where $n = p \times q$

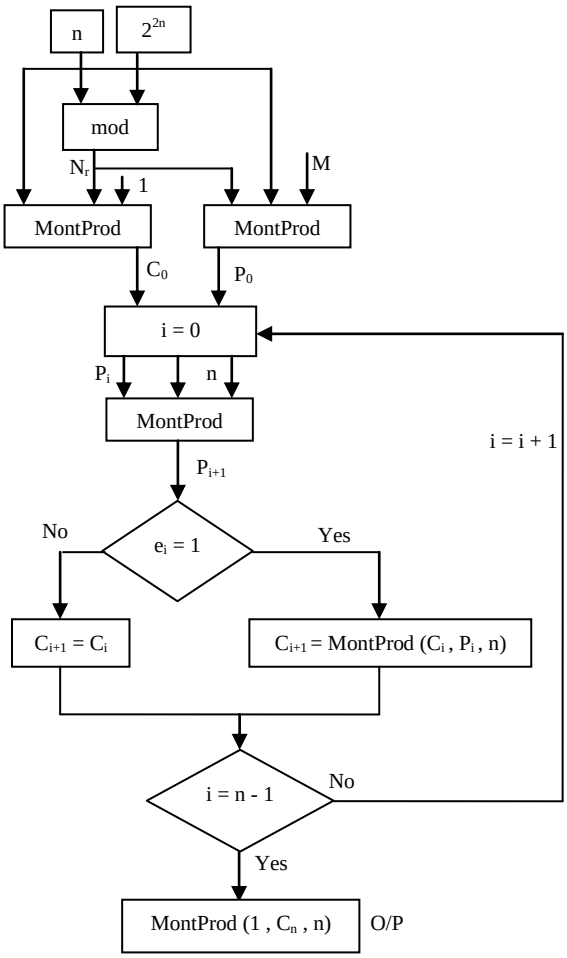

Fig 6: Functional level scheduling in Encryption/Decryption (using Montgomery Exponentiation)

5. RESULTS

We now compare the results for a transceiver of different feasible key sizes.

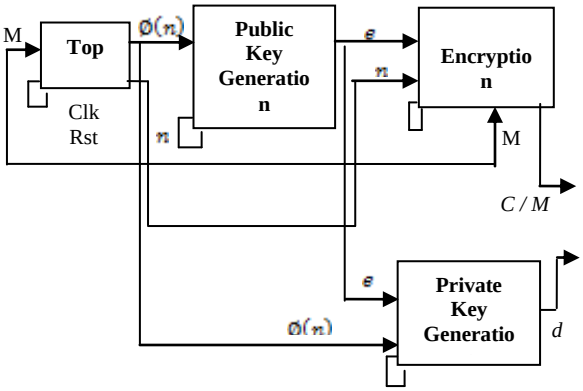

Fig 7: Transceiver

The various key sizes applied are of: 8-bit, 16-bit and 32-bit key lengths. The FPGA board used is Virtex-5 -- xc5vlx50t-2ff1136. The various cases applied for the FPGA implementation of the cryptosystem are as shown below:

- Euclid's Algorithm is used in all the cases for calculation of GCD which is used for the generation of the Public-Key [9].
- The prime numbers p and q are randomly generated using the principle of Linear Feedback Shift Register (LFSR) [9].
- Extended Euclid's Algorithm with two variations as below is used for the generation of the Private-Key [9].
- 'Square and Multiply' and 'Montgomery' are the two algorithms used for either encryption or decryption [3].

TABLE 1
VARIOUS CASES IMPLEMENTED BY SCHEDULING

Encrypt / Decrypt (Functional Scheduling)	Extended Euclid (Arithmetic Scheduling)	Parallel use of Multipliers	Sequential use of Multipliers
Square and Multiply		Case 1	Case 2
Montgomery		Case 3	Case 4

The following results are for a 8-bit key Cryptosystem (Results are generated on Virtex-5 -- xc5vlx50t-2ff1136):

TABLE 2
8-BIT CRYPTOSYSTEM ON VIRTEX5 – XC5VLX50T-2FF1136

Data Compared	Case1	Case2	Case3	Case4
Multipliers	7	5	5	3
Adders/ Subtractors	225	251	605	631
Area (No. of Slice LUTs)	1882 6%	2779 9%	6584 22%	7597 26%
Power(mW)	455	457	449	454
Maximum Frequency (MHz)	46.512	46.511	261.7	88.046
Period(ns)	21.500	21.500	3.821	11.358

The following results are for a 16-bit key Cryptosystem (Results are generated on Virtex-5 -- xc5vlx50t-2ff1136):

TABLE 3
16-BIT CRYPTOSYSTEM ON VIRTEX5 – XC5VLX50T-2FF1136

Data Compared	Case1	Case2	Case3	Case4
Multipliers	7	5	5	3
Adders/ Subtractors	593	635	2125	2167
Area (No. of Slice LUTs)	7195 24%	6991 24%	42733 148%	42448 147%
Power(mW)	462	461	*	*
Maximum Frequency(MHz)	24.271	24.271	247.838	247.838
Period(ns)	41.201	41.201	4.5	4.035

The following results are for a 32-bit key Cryptosystem (Results are generated on Virtex-5 -- xc5vlx50t-2ff1136):

TABLE 4
32-BIT CRYPTOSYSTEM ON VIRTEX5 – XC5VLX50T-2FF1136

Data Compared	Case1	Case2	Case3	Case4
Multipliers	7	5	**	**
Adders/ Subtractors	729	803	**	**
Area (No. of Slice LUTs)	28045 97%	33760 117%	**	**
Power(mW)	*	*	**	**
Maximum Frequency (MHz)	8.947	8.947	**	**
Period(ns)	111.76	111.76	**	**

* The power report could not be generated in this case due to insufficient LUTs in the FPGA.
** The synthesis report have not been generated for this case as it is evident from the lesser key size that the area would surely be insufficient and hence the power will not be calculated.

6. CONCLUSION

We have successfully synthesized the different FPGA implementations of RSA cryptographic system in software. The various architectures evolved from the DFG scheduling were implemented on the FPGA platform and a comparative study is performed between them. Virtex-5 -- xc5vlx50t-2ff1136 FPGA board is chosen as the number of LUTs is very large. As the key size was increased in all the cases we observed an increase in the number of LUTs used in each case. The number of LUTs used showed a tremendous increase in both the cases where Montgomery algorithm was used for encryption as compared to the cases where Square and Multiply algorithm was used. In cases 1, 2 and 3 the increase in maximum frequency is observed to be proportional to the increase in key size. While in case 4 the increase in maximum frequency was inversely proportional to the increase in key size. We encountered limitation of LUTs on the FPGA boards used in the 32-bit case for Square and Multiply encryption algorithm and for 16-bit and 32-bit cases of Montgomery encryption algorithm (inferred from tables – 2, 3 and 4). In future, the work done so far can be extended to even higher bit sizes by modification of the encryption/decryption algorithm namely Montgomery algorithm. A new architecture can further be implemented by using the subtraction algorithm to find out the public key. An actual scheduling algorithm can be applied to further improve the performance constraints of the RSA cryptosystem. After an analysis of all the systems a final architecture can then be implemented on a physical FPGA board.

This work is part of project "FPGA Implementation of RSA Cryptographic System" by Vibhor Garg under guidance of Prof.V.Arunachalam for the award of BACHLOR OF TECHNOLOGY DEGREE at VIT University, Vellore.

7. REFERENCES

[1] Hagai Bar-El (Hagai.Bar-El@Discretix.com), "Security implications of Hardware vs. Software cryptographic modules," Advanced Embedded Security, Discretix Technologies Ltd., White Paper, October 2002.

[2] S.S. Ghoreishi, M.A. Pormina, H. Bozorgi, and M. Dousti, "High Speed RSA Implementation Based on Modified Booth's Technique and Montgomery's Multiplication for FPGA Platform," in Second International Conference on Advances in Circuits, 2009.

[3] T. Blum and C. Paar, "High-Radix Montgomery Modular Exponentiation on Reconfigurable Hardware," in IEEE Transactions on Computer, Vol. 50, no. 7, pp. 759-764, July 2001, ISSN: 0018-9340.

[4] Yachao Zhou and Xiaojun Wang, "An improved implementation of Montgomery Algorithm using efficient pipelining and structured parallelism techniques," in IET Irish Signals and Systems Conference (ISSC 2010), pp. 7 – 11, June 2010.

[5] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, "Efficient Algorithms," Handbook of Applied Cryptography, pp. 606-609,613-620, June 1996.

[6] Gang Quan, "Data Flow Graph Intro," Maseeh College of Engineering and Computer Science, Portland State University, http://web.cecs.pdx.edu/~mperkows/temp/JULY/data-flow-graph.pdf .

[7] John Fry and Martin Langhammer, "RSA and Public Key Cryptography in FPGAs," Altera Corporation, Tech. Rep., 2005, pp.1-3.

[8] Samir Palnitkar, Verilog HDL: A Guide to Digital Design and Synthesis, SunSoft Press, 1996.

[9] William Stallings, Cryptography and Network Security Principals and Practices, 4th edition, Pearson Education, Inc., 2006, ISBN 81-7758-774-9.

[10] Omar Nibouche, Mokhtar Nibouche, Ahmed Bouridane, and Ammar Belatreche, "Fast architectures for FPGA-based implementation of RSA encryption algorithm," in IEEE International Conference on Field-Programmable Technology, 2004.

[11] T. Jebelean, "Comparing several GCD algorithms," in 11th Symposium on Computer Arithmetic, 1993, Proceedings, pp. 180 – 185, 1993, ISBN: 0-8186-3862-1.

[12] John D. Carpinelli, Computer Systems Organization and Architecture, Pearson Education, Inc., 2001, pp. 353-357, ISBN 81-7808-268-3.