World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:11, 2009

Fully Parameterizable FPGA based Crypto-Accelerator

Iqbalur Rahman, Miftahur Rahman, *Member, IEEE,* Abul L Haque, *Member, IEEE,* and Mostafizur Rahman,

Open Science Index, Computer and Information Engineering Vol:3, No:11, 2009 publications.waset.org/6829.pdf

Abstract—In this paper, RSA encryption algorithm and its hardware implementation in Xilinx's Virtex Field Programmable Gate Arrays (FPGA) is analyzed. The issues of scalability, flexible performance, and silicon efficiency for the hardware acceleration of public key crypto systems are being explored in the present work. Using techniques based on the interleaved math for exponentiation, the proposed RSA calculation architecture is compared to existing FPGA-based solutions for speed, FPGA utilization, and scalability. The paper covers the RSA encryption algorithm, interleaved multiplication, Miller Rabin algorithm for primality test, extended Euclidean math, basic FPGA technology, and the implementation details of the proposed RSA calculation architecture. Performance of several alternative hardware architectures is discussed and compared. Finally, conclusion is drawn, highlighting the advantages of a fully flexible & parameterized design.

Keywords—Crypto Accelerator, FPGA, Public Key Cryptography, RSA.

I. Introduction

SECURED data transmission through the internet or in the networks is of utmost importance to many researchers across the world. The advent of cryptography provides us a layer of security during data transfer in the net. The RSA (Rivest, Shamir, Adleman) algorithm is a secure, high quality, public key algorithm. However, the RSA algorithm is very computationally intensive, operating on very large (typically thousands of bits long) integers. One way to address this problem is to apply cryptographic hardware. A RSA accelerator which works like coprocessor can provide means of performing the computationally expensive workload that usually accompanies various algorithms and protocols.

RSA cryptographic accelerator can provide usefulness on two fronts. First and most noticeable is increased speed, which is particularly important to e-commerce companies that interact with a considerable number of customers daily. The second benefit is a spin-off of the first one: by reducing the workload on the system's CPU, accelerators allow the system to be used more efficiently for other tasks.

In RSA, a longer key size means better security. Improvements in the factorization algorithm may inadvertently require that the size of the key be continually and appropriately recommended. The flexibility to change key length or modify the embedded algorithm to respond to design flaws or changes in standards or data formats, requires hardware reconfigurability. Reconfigurable hardware applies to a device that can be configured, at run-time, to implement a function as a hardware circuit. Commercially available reconfigurable devices include Field Programmable Gate Arrays (FPGA) and Complex Programmable Logic Devices(CPLD).

This paper presents an efficient design and implementation technique of an RSA accelerator on FPGAs. The following chronology is being followed in presenting the paper. In section II, RSA algorithm is given in general terms. Then, in section III, other fundamental algorithms that were used in the designed are presented. Section IV deal with design architecture. A top level view of the design and the hierarchy is shown. Then in section V, we discussed implementation strategies. Detail of design architecture and hardware blocks are shown. We explain how parallel computation can generate faster results, efficient methods of implementing those hardware blocks and the data path. Further on, in section VI, implementation results are presented and conclusion is drawn.

II. RSA Basics

RSA is still the most popular cryptosystem in use today. It is widely used to suffice the growing demands for security in communication and computer systems. So, there has been lot of research for faster and more powerful RSA implementation in software and hardware end. Hardware solution provides the best results and maximum throughput compared to software one's by far. Its security is closely associated with the difficulty of factorizing large numbers. This paper describes how to use the RSA cryptosystem. Especially key generation, the encryption, and the decryption are treated in detail.

III. Fundamental Algorithms

In this section, brief explanation of the algorithms that were used are presented.

A. *Exponential calculation[1]*

For fast RSA encryption and decryption, efficient modular exponentiation is crucial. This can be achieved by the "Square and Multiply" algorithm. The square and multiply method is the most popular and effective algorithm for computing

Mr. Iqbalur Rahman Rokon is with the Department of Electrical Engineering and Computer Science, North South University, Dhaka, 1213 Bangladesh e-mail: irahman@northsouth.edu.

Dr. Miftahur Rahman is the Chairman at Department of Electrical Engineering and Computer Science, North South University, Dhaka, 1213 Bangladesh e-mail: mrahman@northsouth.edu.

Dr. Abul L Haque is the Dean of Department of Electrical Engineering and Computer Science, North South University, Dhaka, 1213 Bangladesh e-mail: ahaque@northsouth.edu.

M. Mostafizur Rahman is graduate student from Department of Electrical Engineering and Computer Science, North South University, Dhaka, 1213 Bangladesh e-mail: mz.rahman@inbox.com

modular exponentiation. The idea is based on the binary representation of the exponent. The technique reduces the problem to a series of modular multiplications and squaring steps The square-and-multiply procedure starts from the position of first one in 'e', therefore lot of unwanted calculations are saved.

B. Interleaved Modular Multiplication[2]

The basic idea of this algorithm is to interleave multiplication and reduction such that the intermediate results are kept as short as possible.

C. Division[3]

A simple and widely implemented class of division algorithm is digit recurrence. The most common implementation of digit recurrence division in modern microprocessors is *SRT* division and non SRT division. SRT or restoring division is named after Sweeney, Robertson and Tocher, each of whom developed it independently at around the same time. Restoring division operates on fixed-point fractional numbers.

D. Primality tester[4][5]

Prime number test is the critical part of RSA key generation. We implemented the algorithm developed by Miller and Rabin. It exploits Fermat's theorem that states $(a^{n-1}$ mod n) = 1, if n is a prime. Miller and Rabin's primality test algorithm is: for efficient hardware implementation, the random number n is counted as a prime only if TEST returns 10 successive "inconclusive". According to the distribution of primes, on average every 142 trials $(0.4 ln(2^{512})$ will find a prime.

E. GCD[6]

The Euclidean algorithm is used to determine the greatest common divisor (GCD) of two elements of any Euclidean domain. The extended Euclidean algorithm is an extension to the Euclidean algorithm for finding the greatest common divisor (GCD) of integers a and b: it also finds the integers x and y in Bzout's identity: ax+by=gcd (a,b).

The extended Euclidean algorithm is particularly useful when a and b are co-prime, since x is the modular multiplicative inverse of (a modulo b). The algorithm is simplified by removing unnecessary variables and computations and made it more suitable for hardware implementation.

Simplified extended Euclidean algorithm keeps trying new b until it finds gcd(m,b) = 1, then it returns b and it multiplicative inverse modulo m.

IV. Design Architecture

The system architecture of the design is shown in Figure 1. A random number generator generates pseudo random numbers and stores them in the rand FIFO. Once the FIFO is full, the random number generator stops working until a number is pulled out by the primality tester. The primality tester takes a random number as input and tests if it is a prime. Confirmed primes are put in the prime FIFO. Similar to the random number generator, primality tester starts new test only when prime FIFO is not full. A lot of power is saved by using the two FIFOs because computation is performed only when needed. When new key pair is required, the down stream component pulls out two primes from the prime FIFO, and calculates n and φ(n). N is stored in a register. φ(n) is sent to the GCD block, where public exponent e is selected such that gcd(φ(n),e) = 1, and private exponent d is obtained by inverting e modulo φ(n). E, d and n is then stored in a bigger FIFO which holds 48 bit. Once n, d, and e are generated, RSA encryption/decryption is simply a modular exponentiation operation.

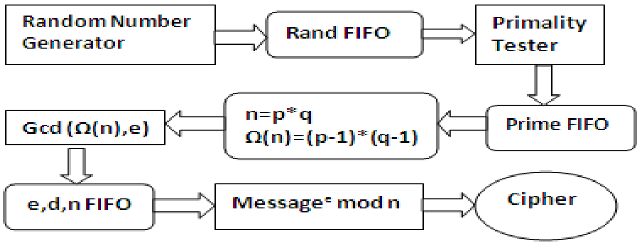

Fig. 1. System Architecture of RSA Accelerator

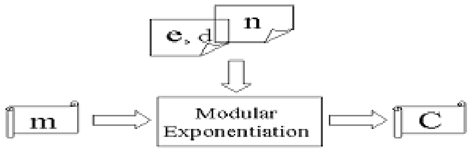

Fig. 2. RSA Encryption

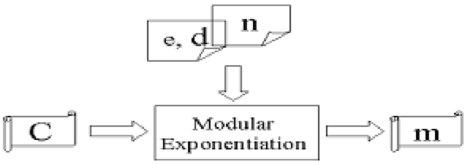

Fig. 3. RSA Decryption

The core of the RSA implementation is how efficient the modular arithmetic operations are, which include modular addition, modular subtraction, modular multiplication and modular exponentiation. The RSA also involves some regular arithmetic operations, such as regular addition, subtraction and multiplication used to calculate n and φ(n), and regular division used in GCD operation. This thesis shows the detail work of those units. The design hierarchy is show below-

V. Implementation Details of Basic Blocks

All the blocks in this design are dependent on new different operand for functionality, that is- they start working only when any of the input operands are different otherwise not. This has a great impact on overall design throughput and allows maximum utilization in minimum time. 8-bit version of this

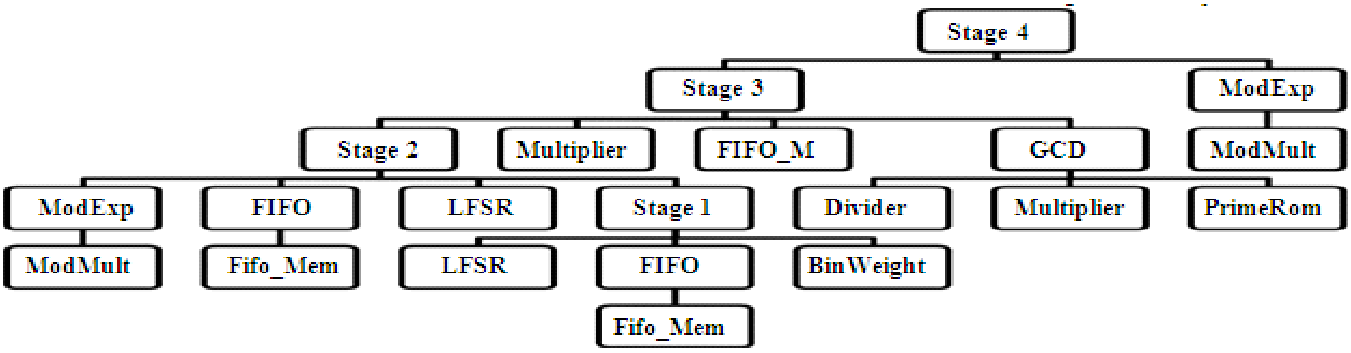

Fig. 4. Design Hierarchy for RSA crypto-accelerator.

parameterized design is shown here for simplicity, which can be easily extended for 1024 bit or 2048 bit real life usage. Basic blocks for this design are – FIFO, lfsr, binary weight calculator, modular multiplier, multiplier, divider, modular exponent calculator, rom.

A. FIFO

A specialized FIFO is used for this particular design. The FIFO is 14 bits and avoids consecutive duplicate numbers and numbers less then 3 as input. It's another Feature is, it operates as circular queue, so- it can be used for long period without initialization. The FIFO is full when there is only one input left, Thus, helping to avoid erroneous outputs

B. LFSR

Linear Feedback Shift Register (LFSR) is used to generate pseudo random numbers. In particular, Fibonacci LFSR was implemented because it is more suitable for hardware implementation than Galois LFSR. In theory, an n-bit linear feedback shift register can generate a $(2^n - 1)$ bit long pseudo random sequence before repeating. However, an LFSR with a maximal period must satisfy the following property: the polynomial formed from a tap sequence plus the constant 1 must be a primitive polynomial modulo 2. We implemented simple 8 bit LFSR with 4 tap-in, it can be easily modified to work with 1024 bit numbers with specific tap placement as described in [7].

C. Binary Weight Calculator

Binary weight calculator calculates number of 1's in a binary number; also it gives the position of first one as output, which is very helpful in modular multiplication and modular exponentiation unit to avoid unnecessary calculation.

D. Modular Multiplication

Using ripple carry adders, modular multiplication using shift-add multiplication algorithm is constructed. Interleaved modular multiplication described in algorithm 2 is used for modular multiplication, which is particularly suitable for hardware implementation, the advantages are—

1) No additional modular addition or Subtraction.
2) Direct results are obtained.
3) Relatively simple operation only involves shift and add.
4) 2 subtraction operations can be reduced to one only.

For a 8-bit modular multiplier, inputs opA and opB are both 8 bits. However, opB might be a small number with a lot of leading 0s. In the hardware implementation, before getting into the shift-add iterations, we search for the position of the first leading 1 in opB, and set (k - 1) to be this position, a separate block called 'BinaryWeight' is implemented to perform the operation. By doing this, unnecessary shift and modular operations are avoided, making the multiplication faster when opB is small. A 1024-bit modular multiplier can also be implemented in a similar fashion.

E. Modular Exponent Calculator

The modular exponentiation operation is simply an exponentiation operation where modular multiplication is intensively performed. We implemented the 8-bit and 16-bit modular exponentiation components using RL binary method, where RL stands for the right-to-leftt scanning direction of the exponent. Algorithm 1 shows detail of RL optimized algorithm. Similar to modular multiplication, we search for the position of the first leading 1 in exponent B and set (k - 1) to be the position. This avoids unnecessary modular squaring operations. For small exponent such as the public exponent e, the modular exponentiation is much faster than big exponent such as the private exponent d.

F. Divider

Non Restoring Division algorithm was chosen for Divider unit, it is particularly suitable for hardware design because of its simplicity and less computation. It requires only n steps and gives quotient and remainder in relatively small time. The algorithm is described in algorithm 3. The divider accepts big dividend and small divisor, and returns a big quotient and a small remainder.

G. Rom

A Rom is implemented which holds only 32 words, prime numbers from 11- 255 are stored in ROM to assist in GCD calculation, for 1024 bit implementation, a ROM with 2048 words will be required.

VI. Implementation Details of Building Blocks

Figure 5 shows, a simplified datapath of the whole design, the clock signal is synchronous and reset is asynchronous.

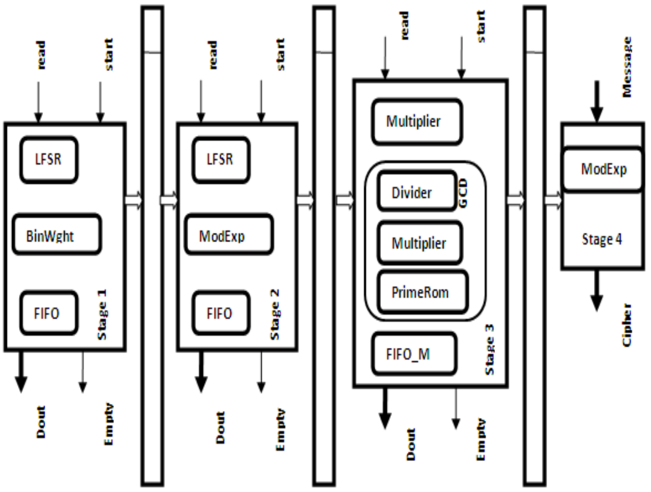

Fig. 5. RSA Datapath

Initially, the reset signal has to be high for few pulses to initialize all system signals. Additionally a start signal is used as synchronous input in building blocks to control execution of the block and therefore helps in reducing power consumption.

A. Stage 1

Stage1 module consist of 3 basic building blocks BinWeight, FIFO and LFSR. The module itself has control signals such as 'clk', 'reset', 'start', 'read' as input and 14 bit data with the name 'dout' as output with two other control signals 'emptyp' and 'fullp'.

In stage 1, only odd numbers that are greater then 2 are selected to store in FIFO, this check for odd number removes unnecessary computation for even prime numbers. The numbers are then placed into BinWght module. It has three outputs. First output is the odd number which is 8 bit, second is the position, where the first 1 was found, its is 3 bit and the third output is single bit signal named 'done' to indicate the operation is complete. The 8 bit random number with its binary weight and position of first one then placed into FIFO which holds 8 words 14 bit each. The block continues to execute until the FIFO is full, when FIFO is full, it enters IDLE state. Thus, reduces the power consumption and unnecessary logic execution. Figure 6 shows block diagram of stage 1.

B. Stage 2

Stage 2 executes miller, rabin's prime tester for prime check. Separate LFSR block is used for random number generation with different initial value, which allows the sequence to be altered and complete random checking for primes. Special care is taken to avoid random numbers less then or equal to 3, and to avoid duplicate random numbers for prime test. ModExp and InterleavedMult unit does the modular multiplication and modular exponent computations that are required.

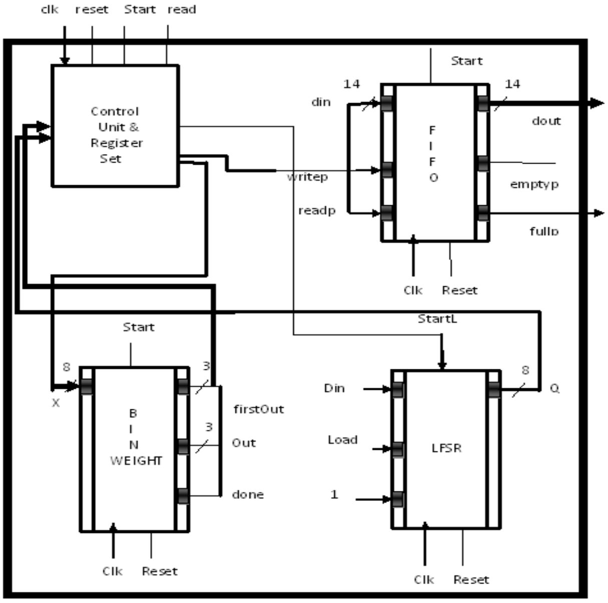

Fig. 6. Block diagram for Random number generation unit.

The 14 bit FIFO stores 8 bit prime number along with its binary weight and the position of first one which are 3 bits each for faster multiplication and exponentiation. The block continues to execute until the FIFO is full, which stores maximum 8 words. Whenever a word is pulled by upper level block- stage3, stage2 starts functioning, to find another prime number to fill that void. Figure 7 shows detail of stage 2.

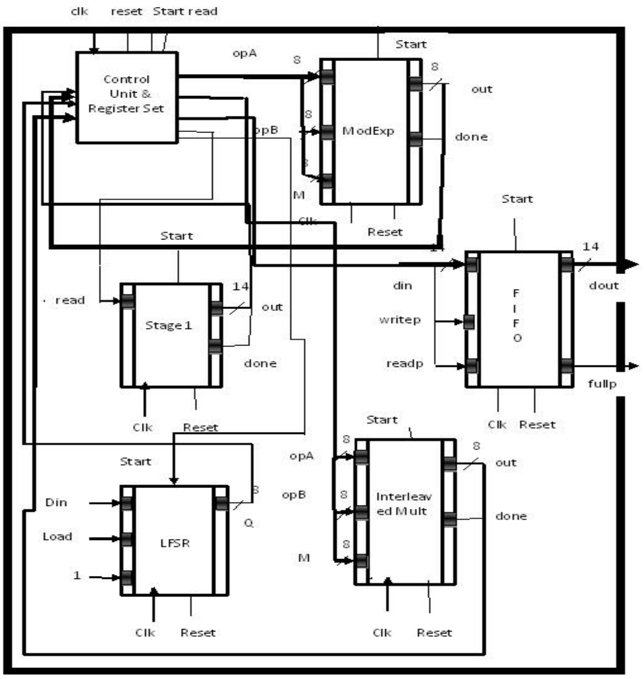

Fig. 7. Block diagram of Prime number Checker unit.

C. GCD

GCD calculator calculates gcd of input and a prime number from ROM and finds suitable multiplicative inverse. Sub blocks of this module are Divider, primeRom, Multiplier. For 8 bit implementation only 32 prime numbers are sufficient, but for 1024 bit implementation, To get a valid b faster, pre-computation of all primes less than 2000 excluding 2 is required. Because the product of all these primes is greater than 2^{1024}, which means that any m less than 2^{1024} is prime to at least one of these primes. These primes are hard-coded in the on-chip block memory called primeRom.

Extended Euclidean requires a regular multiplier to compute Q*B2. Normal add-shift multiplication algorithm was used for that. The interface of GCD block is show below-

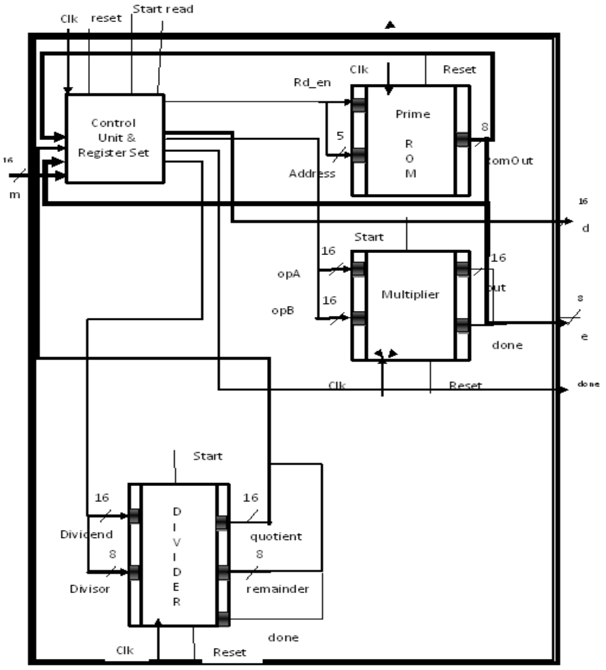

Fig. 8. Block diagram for GCD calculator

D. Stage 3

Stage 3 is the key generation stage. In this stage both private keys (d, n) and public keys (e, n) are generated. Sub blocks that do the computations are Multiplier, GCD, and FIFO.

At first two different prime number p, q are fetched from Stage2 FIFO which holds 8 prime number always, result of (p-1)x(q-1) is then put into GCD module for e, d calculation.

GCD calculator then returns a suitable prime number e with the condition gcd((n),e)=1 and the multiplicative modulo inverse d of e using (n). The multiplier and GCD calculator works in parallel, as the multiplier calculates (pxq) while gcd calculator works with (n). The result e, d, n are 16 bit each, and a larger FIFO that holds 48 bit word is required to put the result. Thus the FIFO holds both private and public keys at the end of execution. The block enters idle state when the FIFO with 8 word depth is full. Figure 9 shows the block diagram.

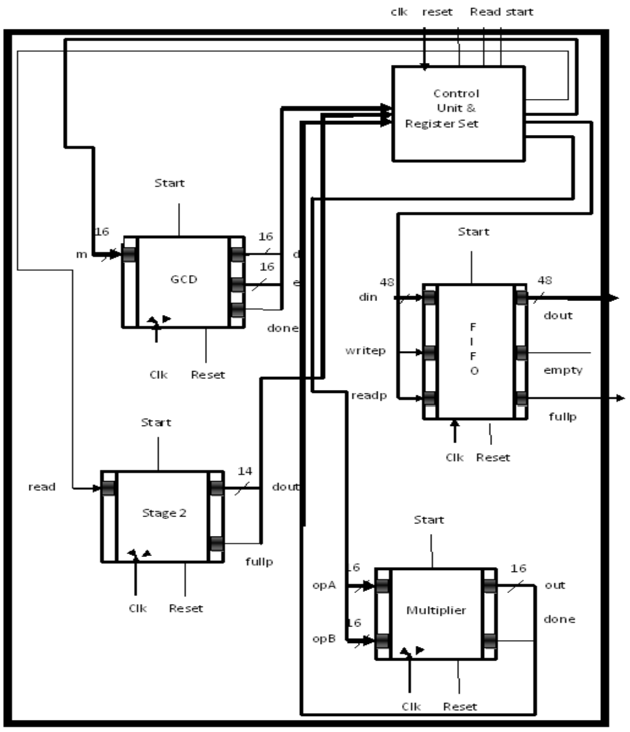

Fig. 9. Block diagram for key generation unit

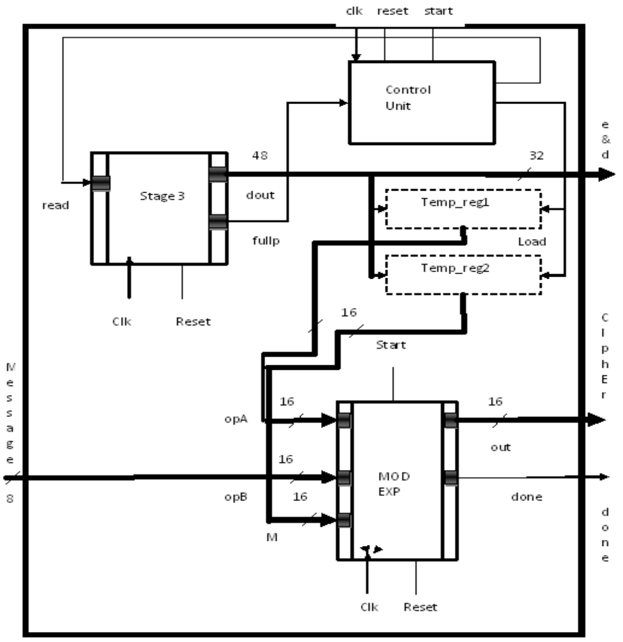

Fig. 10. Block diagram for Encryption unit. (stage 4)

E. Stage 4

Stage 4 is the higher block in the hierarchy, it encapsulates all the lower level blocks. This block takes 8 bit message (M) as input and gives 16 bit cipher text as output. A 16 bit modular exponential unit does the M^e (mod n) computation

F. Conclusion

In this paper, a detail implementation technique for 8-bit RSA circuit is shown. It is a full-featured parameterized design of RSA circuit including key generation and data encryption, which can be extended for 1024/2048 bit implementation of RSA cryptography. Both RSA key generation and data encryption component for this 8 bit RSA is suitable to fit into a single Xilinx Virtex II pro FPGA. The test was conducted in real hardware. Each sub-component was simulated in XST and implemented in FPGA and is functionally correct. According to the synthesis statistics, critical parts like modular multiplication and modular exponentiation takes 4.2 s, 10s and 45s and 94s for 8 and 16 bits respectively. Turnout time for 8 bit RSA implementation is 1.2 ms for 100 Mhz clock.

Following figures show the webworms of major blocks-

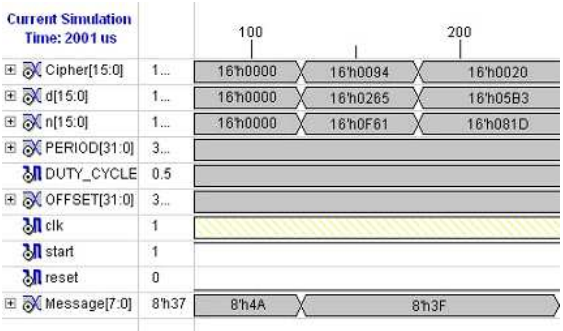

Fig. 11. 'Stage4' input output signals in webform

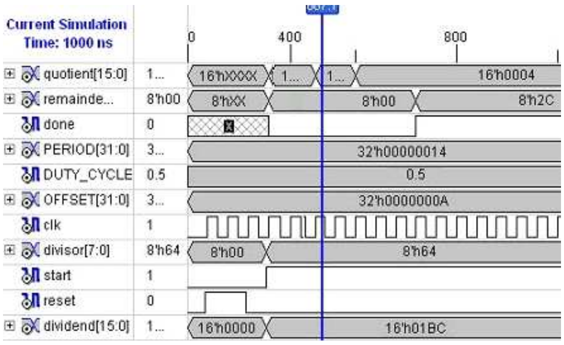

Fig. 12. Figure12: 'Divider' input output signals in webform

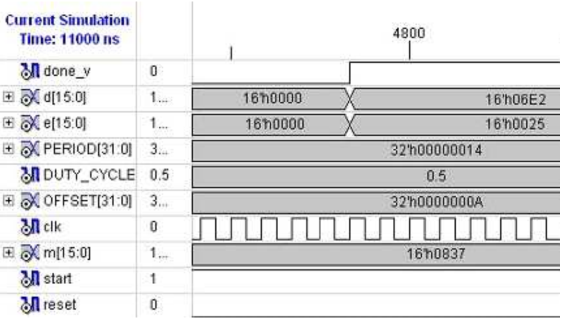

Fig. 13. 'GCD' input output signals in webform

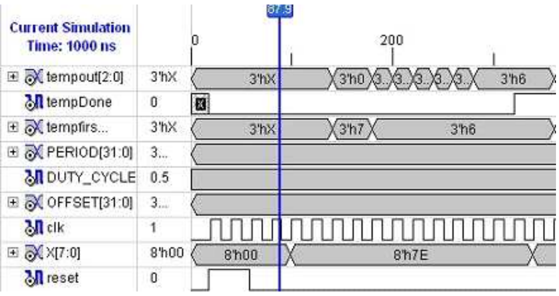

Fig. 14. 'BinWeight' input output signals in webform

References

[1] Lu, Jing, and Qian Wan, . Implementing a 1024 Bit RSA on FPGA. 03 May 2003. Dept. of CSE., Washington U. 21 Jan. 2008 <www.arl.wustl.edu/˜jl1/education/cs502/doc/report.doc>.

[2] Amanor, David Narh, comp. Efficient Hardware Architectures For Modular Multiplication On FPGAs. 19 Feb. 2005. The University of Applied Sciences CityplaceOffenburg. 07 Dec. 2007 <www.crypto.rub.de/imperia/md/content/texte/theses/dnamanorthesis.pdf>.

[3] "Division (Digital)." Wikipedia. dateMonth4Day17Year200717 Apr. 2007. Wikipedia. dateMonth12Day12Year200712 Dec. 2007 <http://en.wikipedia.org/wiki/Restoring_division#Restoring_division>.

[4] G. Miller, Riemann's Hypothesis and Tests for Primality. Proceeding sof the 7th Annual ACM Symposium on the Theory of Computing, May 1975.

[5] M. Rabin, Probabilistic Algorithms for Primality Testing. Journal of Number Theory, Dec. 1980.

[6] Wu, C.-L., Lou, D.-C., Chang, T.-J.," Fast Binary Multiplication Method forModular Exponentiation." Tanet, 2005. 22 Nov. 2007. <tanet2005.nchu.edu.tw/session/TANet2005Sessiondetail.pdf>.

[7] "DS257",Linear Feedback Shift Register v3.0. V-1, dateMonth3Day28Year200328 MAR 2003. Xilinx. 04 Jan. 2008.

[8] Wockinger, thomas. High-Speed RSA Implementation.07 Jan 2005, Institute of applied information., Graz U. 27 Nov 2007 <www.iaik.tugraz.at/teaching/11_diplomarbeiten/archive/woeckinger.pdf>.

Mr. Iqbalur Rahman Rokon received his BSc degree in Electrical and Electronic Engineering from Bangladesh Institute of Technology, Rajshahi, Bangladesh 1991 and MS degree in Electrical and Computer Engineering from California University, Northridge, USA in 1997. During his MS study in US, he also worked as System Engineer at National Telecom, Santa Ana, California, USA from 1994 to 1997. Then he started his career at High-Tech Chip Companies in USA and served as Design Engineer, Chip (ASIC/FPGA/CPLD) Development, R&D, Emulex Corporation, Costa Mesa, USA from 1997 to 2003. Then he joined North South University, Dhaka and has been teaching there as a Faculty Member of Electrical Engineering and Computer Science Department since 2004. He also served as Proctor of the University from 2007 to 2009. In addition to his current teaching profession at NSU, Iqbalur Rahman Rokon is also continuing his high-tech involvement with industry and serving as consultant of ASIC/FPGA Development at Power IC Limited – a Chip Design Company at Dhaka, Bangladesh.

Dr. Miftahur Rahman (M'04–to-date) received his BS and MSc in Physics from the University of Dhaka, Bangladesh, in 1976 and 1977 respectively. He started his teaching career as a Lecturer in the Department of Physics, at the Bangladesh University of Engineering and Technology (BUET) in 1980. In 1983 he received his MS in Solid State Physics from Marquette University, Milwaukee, Wisconsin. Later in 1988 he received his Ph.D. in Solid State Physics and Optics from the University of Massachusetts Lowell. He worked as a Technical Consultant at Arthur D. Little, Inc., Cambridge, MA, USA, from 1988 to 1989. He served as a Research Associate and Consultant at the Institute for Plastics Innovation (IPI), UMASS-Lowell, from 1990 to 1995.

He served as a Technical Manager at Grameen Shakti, Dhaka, Bangladesh from 1995 to 1997. He served as a Professor and Proctor at East West University, Dhaka, Bangladesh from 1997 to January 2002. In spring 2002 he joined the Department of Computer Science and Engineering, at North South University (NSU), Dhaka, Bangladesh. He served as the Chairman of the Department from 2003 to 2007 and is currently serving as the Chairman of the department, which has been renamed as the Department of Electrical Engineering and Computer Science (EECS). He is also the coordinator of the MS in Electronics and Telecommunications program and has been teaching MS telecom courses and supervising MS theses and projects in the area of wireless and mobile technology. He is also serving as the Director of the Center for Information and Communication Technology (CICT). During his first tenure as the Department Chair, he organized an International Conference on the Next Generation Wireless Systems in 2006 in Dhaka, Bangladesh, with the sponsorship of IEEE Communications Society in which many internationally reputed scientists and scholars attended the conference and contributed their technical papers. He had the privilege to act as the Editor of the ICNEWS'06 proceedings in which 71 technical papers were internationally reviewed and had been included for publication. Currently he is a member of the Technical Program Committee of the ISIEA09 (IEEE Symposium on Industrial Electronics and Applications). He has published papers in refereed journals, and has about 25 international conference papers, about 25 industrially funded research reports and supervised about 25 BS and MS projects.

Dr.Abul L. Haque earned his BSc and MSc degrees in Applied Mathematics from Dhaka University and his MSc degree in Computer Science from the University of Western Ontario and his Ph.D. degree in Computer Science from the University of Oklahoma. He served in a various teaching positions at Dhaka University, Yarmouk University, University of Bahrain before joining as an Assistant Professor position at North South University in 1994. Currently he is a Professor of Electrical Engineering and Computer Science and the Dean of School of Engineering and Applied Sciences. He has published many papers in the area of Neural Networks. His current research interests are in the areas of Algorithms and Computer Architectures.

Mostafizur Rahman is a graduate from the Department of Electrical Engineering and Computer Science at North South University. He graduated in April 08 with the distinction of Magna Cum Laude. His research interests are in Reconfigurable Computing, Computer Architecture and SoC design.