International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 3 Issue: 11
ISSN: 2454-4248
227 – 240

Maximizing the Efficiency using Montgomery Multipliers on FPGA in RSA Cryptography for Wireless Sensor Networks

Leelavathi G, Shaila K,
Department of Electronics and Communication Engineering,
VTU-Research Centre, Vivekananda Institute of Technology,
Bengaluru, India
nisargamodini@gmail.com

Venugopal K R,
IEEE Fellow, Principal,
University Visvesvaraya College of Engineering,
Bengaluru, India

Abstract- The architecture and modeling of RSA public key encryption/decryption systems are presented in this work. Two different architectures are proposed, mMMM42 (modified Montgomery Modular Multiplier 4 to 2 Carry Save Architecture) and RSACIPHER128 to check the suitability for implementation in Wireless Sensor Nodes to utilize the same in Wireless Sensor Networks. It can easily be fitting into systems that require different levels of security by changing the key size. The processing time is increased and space utilization is reduced in FPGA due to its reusability. VHDL code is synthesized and simulated using Xilinx-ISE for both the architectures. Architectures are compared in terms of area and time. It is verified that this architecture support for a key size of 128bits. The implementation of RSA encryption/decryption algorithm on FPGA using 128 bits data and key size with RSACIPHER128 gives good result with 50% less utilization of hardware. This design is also implemented for ASIC using Mentor Graphics.

Keywords— *FPGA, Modular Multiplication, Modified Montgomery algorithm, Modular Multiplication, RSA cryptosystem, VHDL*

*****

I. INTRODUCTION

Wireless Sensor Network (WSN) denotes a heterogeneous system, comprised of autonomous devices called sensor nodes. It has limited data transmission and low computational power that leads to a challenging environment to provide security. A WSN is a self-organizing low-cost, low-power, wireless nodes installed to capture information and used to monitor the environmental conditions, like climatic measurements, in health area which includes measurement of vital signs, temperature, home automation etc [1].

The Objective of any Cryptographic systems is to provide the information security: authentication, confidentiality, data integrity. In disparity to Private Key Cryptosystems (Symmetric key), Public-Key Cryptosystems (PKC)(Asymmetric key) are capable of fulfilling all the objectives. All security solutions considered for conventional computer networks cannot be implemented directly in WSN due to limitations of WSNs . It was understood for a long time, that the PKC was not suitable for WSNs because it requires elevated processing power. But in the course of studies of encryption algorithms based on curves it proved the feasibility of that technique in WSN [2]. However, in order to achieve fast and better feasibility in the applications, public key cryptographic schemes have to be implemented in hardware.

Rivest–Shamir–Adleman (RSA) is the utmost widely used public-key cryptosystem, based on the idea originally presented in 1976 by Diffie and Hellman. The consequence of high security and faster implementations covered the way for RSA crypto-accelerators, hardware implementations of the RSA algorithm [3] [4] [5].

With the heightened emphasis on security in realm of computers and computer networks, the RSA encryption algorithm is used as an effective method to encrypt and protect data. This key-based algorithm relies on integer multiplication to accomplish the data encryption or decryption, with the speed of the multiplication algorithm contributing more to the throughput performance of the RSA encryption algorithm. Furthermore, Wireless Sensor Networks have an added vulnerability because nodes are often placed in a hostile or dangerous environment where they are not physically protected. Security mechanisms need a definite quantity of data memory, code space and energy to power the sensor. These resources are very inadequate in tiny wireless sensor nodes.

RSA operation is a modular exponentiation and its security depends on its incapability to factorize large integers. Application-specified integrated circuit (ASIC) solutions have the disadvantage of reduced flexibility and high NRE cost. The implementation of cryptographic algorithms on reconfigurable devices like Field Programmable Gate Arrays gives the solution, which adds up high flexibility with speed and physical security of traditional hardware. According to McIvor et al., [6] the Montgomery multiplication algorithm Modified Montgomery Multiplication 5 to 2 Carry Save Adder (CSA) architecture (MMM52) is used implementations of RSA. This proposed work investigated the performance of RSA encryption algorithm using Montgomery modular multiplication in both hardware and software. The device utilization is more.

Motivation: RSA is a cryptographic technology and relies severely on complex large-number mathematics to provide its security services. Use of dedicated ASIC or Application Specific Instruction Processors to speed up the mathematics, are frequently expensive and inflexible. The combined cost and performance problem can be addressed by taking into account an FPGA based implementation. To achieve practical hardware implementations for RSA, the complex mathematics involved, utilizes a technique known as Montgomery Multiplication. Montgomery's techniques are very proficient

 ISSN: 2454-4248
227 – 240

implementations of RSA-based cryptography systems. Computations involved with Montgomery are concentrated in the region of the cyclic re-use of additions and the challenges encountered with FPGA implementations.

The computation costs associated with public key cryptosystems is restricted due to the limited resources. Crypto-accelerators are encouraging as they characteristically attain enhanced power efficiency and better performance than a software implementation on a generic processor. The proposed work aims to design arithmetic architectures for RSA Cryptosystems which are optimized for modern FPGAs and ASIC technologies. In these architectures, Montgomery algorithm is utilized to increase the speed of modular multiplication [2]-[9].

Contribution: Key contribution of this work is the Field-Programmable Gate Array implementation of the proposed architectures. Two different architectures RSACIPHER128 and modified Montgomery Modular Multiplication 4 to 2 CSA (Carry Save Adder) architecture (mMMM42) multiplier are implemented in Encryption and Decryption processor. The MMM42 multiplier modified to utilize in our algorithm. Both architectures are implemented with VHDL coding one with structural approach and other with behavioral approach. The performance of the both the approaches are compared.

Organization: In Section II various research works linked to security techniques, public key cryptography and RSA, different multiplication algorithms are described. Background work is discussed in Section III. Problem definition is stated in Section IV and implementation model is described in Section V. Modular multiplication with Montgomery techniques and RSA are given in Section VI. Algorithm is described in Section VII. Implementation and Performance Evaluation details are explained in Section VIII and Section IX correspondingly. Conclusions are discussed in Section X.

II. Literature Survey

The popular cryptographic algorithm, RSA include extensive modular exponentiation of long integers. On a general-purpose computer, its operation is a very slow since; current typical operands have larger bits. Repeated modular multiplications are carried out to achieve the modular exponentiation. An proficient Modular Multiplication (MM) algorithm for the calculation of A^BmodM was established by P L Montgomery [2].

Abdullah et al., [3] discusses to protect the data in unsecure networks. It is highlighted the essentiality of strong cryptography is WSNs, with improvement over time efficiency and reduction in power consumption. The Key length considered is 1024 bits and designed for specific applications.

Alan Daly et al., [8] review existing architectures of Montgomery Modular Multiplication and Exponentiation implementation on FPGA Xilinx Virtex V1000FG680-6. The new architecture exploits the maximum carry chain length feature of the FPGA that is used to implement modular exponentiation operation. It is essential for encryption and decryption.Based on different multiplier size for the pipelined modular multiplier and RSA encryptor / decryptor speed and area are provided.

Sutter et al., [10] discussed Montgomery's multiplication, suggested different architectures to accomplish the LSB first and the MSB first algorithms. Conversion of the CSA representation of intermediate multiplication with carry-skip addition and digit serial method for Montgomery multiplication is used. Results are presented in Virtex 5 and in 0.18-m ASIC technologies' that reveals the better delay performance and area-time complexity.

Hong et al., [11-14] reduces the number of partial products in the multiplication algorithm by the radix-4 Booths algorithm. He proposes a radix-4 modular multiplication and exponentiation algorithm, which is good choice for modular multiplier in terms of area-time product.

Kauther et al., [15] presents a new algorithm of Radix-4 MSB modular multiplier using 4-2 compressor. Fournaris et al., [16] defines and analyzes the Montgomery multiplication algorithm presenting two scalable systolic Montgomery MM architectures and implementations with high speed and accomplished improved results in terms of area and speed. The architectures proposed uses redundant Carry-Save arithmetic.

Mentens et al.,[17] presents a pipelined architecture of a modular Montgomery multiplier, which is appropriate for public key coprocessors and have derived a more compact pipelined version. This outperforms earlier designed Montgomery multipliers.

Pourmina et al., [18] modified radix-4 modular multiplication based on Booths multiplication technique. To avoid carry propagation CSA (Carry Save Adder), fast algorithm was developed and employed for computing the modular reduction. It is revealed that the processor can perform RSA operation of 1024-bit in less than 15ms at 54.6MHz and and in 50ms at 16.1MHz on VirtexII and XC4000 FPGA.

Vincent et al., [19] describes the scheme of an effective RSA cryptosystem with a modified Montgomery algorithm. A Kogge-Stone Adder which is very fast parallel prefix adder, employed to reduce the critical path.

New scalable systolic hardware architecture is presented by Tamer Gudu et al., [20] Their design enables making area-time tradeoff with different precision of inputs. The add-shift Montgomery algorithm and R-L binary Montgomery exponentiation algorithm are utilized to implement in Virtex-5 FPGA chips for different radixes. Highest performance per area is obtained with the Radix-216 design. Suitability requires more hardware for resource constrained devices.

Drutarovsk'y et al., [21] gives a comparison of possible methods for an efficient implementation of Multiple-word radix-2 Montgomery Modular Multiplication on FPGAs. An embedded soft-core processor Altera NIOS is used for purely software implementation. The combined hardware-software

 ISSN: 2454-4248
227 – 240

designs on Altera FPGAs, speed and logic requirements comparisons are accomplished.

Kamala et al., [22] proposed Montgomery modular multiplication technique to perform long-integer arithmetic that uses multi-bit shifting and carry-save addition. The resultant hardware realization is best in terms of delay and offers high data throughput. It occupies a little more area. The proposal has been assessed on Virtex2 series for practical bit lengths of 512, 1024 and 2048 bit.

Shian-Rong et al., [23] projected architecture with less energy consumption and higher throughput. The whole design manipulates the MM52 and MM42 Algorithm for Montgomery Multiplication.

Chen et al., [24-25] developed a unified Montgomery modular multiplication algorithm which is useful to realize either the conventional modular multiplication or squaring operation in carry-save procedure to attain area-efficient scheme of modular exponentiation. In this work, the number of input operands is condensed for carry-save addition by mathematical exploitation to reduce the resultant critical path delay, least hardware complexity and area-time complexity.

Shieh et al., [26-27] investigates how to unwind the data dependency that occurs between multiplication, quotient determination and modular reduction in the conventional Montgomery modular multiplication algorithm. A new modular multiplication algorithm for high-speed hardware design is proposed. The speed upgrading is accomplished by reducing the critical path delay from the 4-to-2 to 3-to-2 carry-save addition. Investigational results display that the developed modular multiplication can function at speeds higher than those of related work.

Kuang et al., [28] presented energy-efficient algorithm architecture to decrease the energy consumption and to improve the throughput of Montgomery modular multipliers. The architecture offered is equipped to bypassing the superfluous carry-save addition and register write operations, leading to less energy consumption and higher throughput. Experimental outcomes demonstrate that the intended approaches can accomplish 60% energy saving and 24.6% throughput enhancement for 1024-bit Montgomery multiplier.

Table 1.Comparison of Our Work with Related Work

AUTHOR	ALGORITHM	ADVANTAGES	DISADVANTAGES	PERFORMANCE
Alan Daly et al[ ].,	Pipelined and non pipelined MMM	Input can be built into n-bit words	Occupies more area	At higher bitlengths the speed improved significantly
Shiann-Rong et al[ ].,	Carry Save Addition MMM	Energy consumption of CSAs and registers are reduced	Implementation with backend tools, not on a FPGA	Less energy consumption and higher throughput
John Fry et al[ ].,	RSA Calculation Architecture	RSA-based Cryptography systems	Target FPGA are Cyclone or Stratix family from altera	Less Silicon area and Cost effective
Abdullah Said et al[ ].,	Usage of hardware security	Public key cryptography for WSNs security	The work is Concentrated on timer and power analysis for higher keylengths	More effective in time and power consumption with hardware
Our work	Two methods 1.use of MMM42 2.Without MMM42, direct RSA implementation	Second Method Suitable for WSNs	more area and device usage on Spartan FPGA	With Artix, Kintex and Virtex good performance w.r.t area and time.

Table 1 depicts the survey on different methods of implementations of scalar multiplication and cryptographic process using various techniques. Most of the works are implemented using FPGAs, few are with ASIC design. In our work, input is split into 64bits with use of MMM42. Input of 128bits is used without MMM42. Modified modular multiplication algorithm MMM42 is implemented in our design with little modification.

III. BACKGROUND

Alan et al.,[8] proposed pipelined multipliers to implement a full modular exponentiation for RSA encryption and decryption. By choosing Xilinx target device Virtex-6, a high speed Montgomery multipliers are implemented. The work mainly concentrates on speed and area results for pipelined modular multiplier. Pipelined architectures consume more hardware, so suitability for WSNs requires reduction in hardware.

Shian-Rong et al., [23] presented architecture with less energy consumption and higher throughput. The whole design manipulates the MM52 and MM42 Algorithm for MM, in turn modifying modular exponentiation algorithm. These features motivated to select this MMM42 architecture and modify to implement on 128-bit with Xilinx FPGA. The synthesis of MMM42 multiplier is performed using the Synopsys Design Complier. Only multiplier designs are taken up in [23], whereas in our work we have implemented complete RSA cryptosystem i.e., encryption and decryption, with modified MMM42 Multiplier.

IV. PBOBLEM DEFINITION

In WSNs the security problem is challenging concerning the limitations of sensors and essential to balance data integrity, confidentiality and availability. Several architectures are not area proficient and result in higher cost after implemented in silicon.

we overcome the area utilization problem in our proposed system. Area and throughput are prudently trading off to make it appropriate for wireless sensor nodes, where emphasis is on the speed and on area of implementation. Since the demand for higher levels of security it becomes significant to find the ways of implementing the RSA PKC in more efficient and faster architectures. The FPGA is the target implementation podium for the presented architectures.

 ISSN: 2454-4248
227 – 240

A. *Objectives*

The main objective is to improve the security by:

- Designing the Hardware Crypto engine to increase the Encrypted communication.
- Implementing asymmetric cryptographic algorithms on Sensor node and is extended with an FPGA module for increasing the efficiency of the system.

B. *Assumptions*

- The prime numbers p, q are selected with $p \neq q$.
- The sender's public key is published and is made available to the receiver.
- The sender has public key e, and only the receiver knows the private key d. Thus, a *public key (e, n) and secret key (d, n)* is distributed to transmitter and receiver separately.

V. RSA CRYPTOSYSTEM MODEL

Figure 1 shows the model of RSA cryptosystem in which 128 bit input data and key are used to generate cipher. In the model public key is employed for encryption and private key for decryption since utilization of private key for decryption provides more security.

Our proposed work depends on the hardware acceleration for security algorithms which significantly improve the performance and power of the system, even though, additional hardware is required thus increasing the design complexity. In our proposed approach, the computation time, energy and hardware implementation of cryptographic application are considered to increase the average network lifetime.

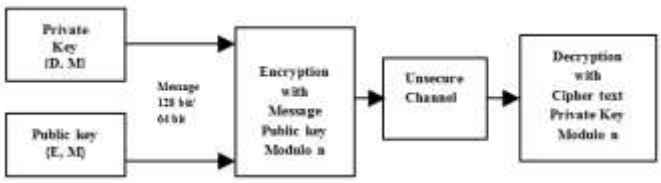

Fig.1 Block Diagram of RSA Cryptosystem Model

VI. RSA CRYPTOSYSTEM

Equations for encryption and decryption of information are shown in figure 2. Plaintext P, Exponent (public key) E and Modulus M are represented as indata, inExp and inMod respectively.

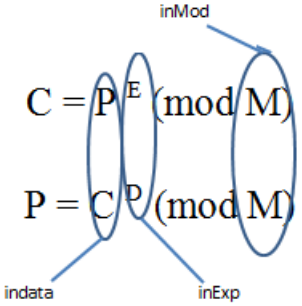

Fig. 2 Description of Equation (1) and (2)

A. *Montgomery ModularMultiplication*

In the design, for implementation of modular multiplication, we have used Montgomery multiplication algorithm. RSA encryption and decryption with and without MMM42 algorithm is executed and performance is compared to check suitability for WSNs in terms of execution speed and hardware utilization.

The Montgomery's algorithm[7] for modular multiplication shown is in Algorithm 1. This algorithm calculates the product of two integers modulo and third one without execution of division by M yielding the condensed product using series additions. This algorithm is modified with carry save representation of the data A, B and M in [23]. The same algorithm is utilized and modified to compute the modular multiplication. In the modified algorithm the data is represented individually as 64bits to perform RSA-128bit operations.

The following Algorithm 1 calculates the Montgomery Multiplication.

Algorithm 1: Basic Montgomery Algorithm

```
MontProd (A,B,M)
{
        S-1=0;
        For i=0 to n-1 do
        Ci= (Si-1+BiA) Mod 2
        Si= (Si-1+CiM+BiA)/2
        End For
        Return Sn-1
}
```

McIvor et al[7]., presented Algorithm MM52 and Algorithm MM42, two efficient algorithms termed as to fast compute the Montgomery modular product. In these algorithms the carry-save arithmetic is employed to evade the ripple-carry propagation. I/O operands A, B and S are represented as *(A1, A2), (B1, B2)* and *(S1, S2)* in the carry-save representation respectively. The time-consuming ripple carry propagation is detached, used only in the last step for attaining the result of modular exponentiation. The following pseudo code gives the modified MMM42 multiplier to perform RSA encryption and decryption.

Function1: Pseudo Code for the Implementation of RSA Cryptosystem Without Modified MMM42 Multiplier

```
Entity Main
{Inputs: clk,rst, Publickey, Privatekey, modin, data_in,
  Outputs: cyphertext, original_msg

Component RSACypher
{Generic (KEYSIZE:integer :=32);
        Inputs: indata, inExp, inMod, clk, ds, reset;
```

 ISSN: 2454-4248
227 – 240

Outputs: Cypher, ready;
Encryption: RSACypher generic map (keysize=>128)
Portmap**(**data_in, public key, modin, enc_msg, clk, ds1, rst, rdy1**);**
Decryption: RSACypher generic map (keysize=>128)
Portmap**(**enc_msg, private key, modin, original_msg, clk, ds2, rst, rdy2**);**

Pseudo Code of Modified MMM42_Multiplier

```
Entity MMM42_multiplier
{
Port(clk,rst,lda,A1,A2,B1,B2,N,S_1,S_2)
Component RSACypher
Generic(KEYSIZE:integer:=128);
Port(indata, inExp, inMod, Cypherout);
Component CSA
Port(X1,X2,X3,y1,y2);
Component mux_4_1
Port(a,b,c,d,sel,y1,out);
Component lu_unit
Port(a1,a2,S1,S2,S3,q_not,bypass,a_not);
Component mbrfa
Port(clk,rst,bypass,A1,A2,ai1,ai2,out);
}
```

The schematic diagram of the entity for MMM42 is shown in Figure 3 and Figure 4. The 64bit inputs *(A1, A2)* =*A*, *(B1, B2)* = *B* and *N* and output *(S-1, S-2)* = *S* are observed in the Figure 3 and Figure 4. Clk, lda, rst and done are control signals.

B. Modular Exponentiation

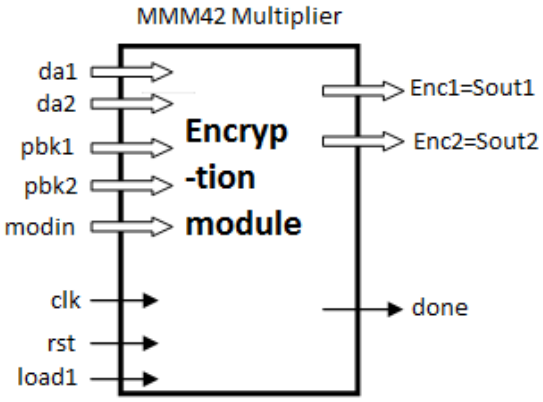

Fig 3. Schematic of MMM42 Multiplier for Encryption

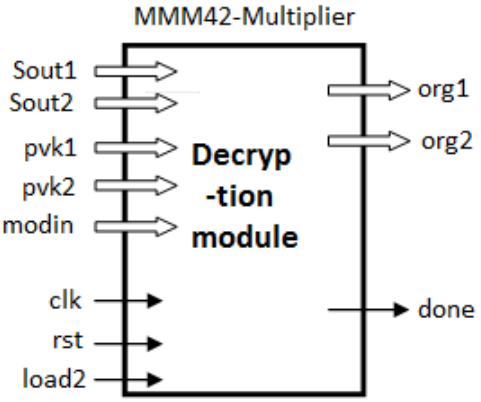

Fig 4. Schematic of MMM42 Multiplier for Decryption

In this work *L* Algorithm or Right to Left algorithm, is selected because square and multiply are independent operations and can be executed in parallel. 50% of clock cycles are required to accomplish the modular exponentiation. But, two physical multipliers are compulsory to achieve acceleration of the algorithm. The algorithm 2 is shown below.

Algorithm 2: Modular Exponentiation using MMM42 multiplier

```
RSACypher (P, E, M)
{
        C=2 2n Mod M;
        P= Modmult (C, E, M);
        R=Modmult(C, 1, M);
        For i=0 to k-1 do
        If (E(i)=1) then
        R=Modmult(R, P, M)
        End if
        P=ModSqr(P, P, M)
        End for
        R=Modmult(1, R, M)
        Retrun R;
}
```

Figure 5 shows the block diagram of modular exponentiation in which MPNAND, MPLIER and Modulus are of 128 bit data. The Multiplicand and Multiplier should be less than the modulus value *M*.

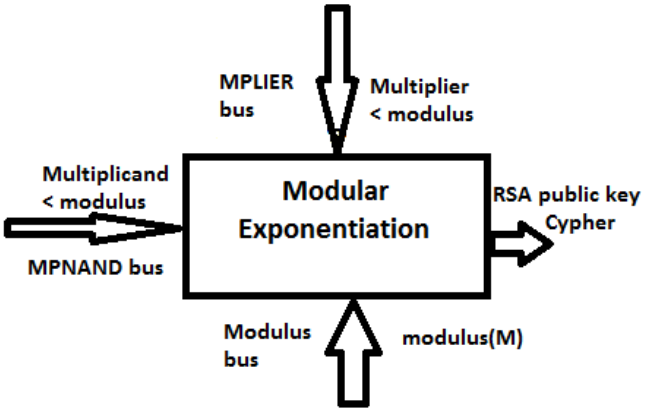

Fig.5. Block Diagram of Modular Exponentiation

 ISSN: 2454-4248
227 – 240

As shown in Equation (1) and Equation (2) both encryption and decryption involve an algorithm for computing a modular exponentiation.

Modular exponentiation operation is series of modular multiplication and squaring operation involves *square and multiply algorithm*, which is based on scanning the bit of the exponent from the left (MSB) to the right (LSB). In each iteration, *i.e.*, for every exponent bit, the current outcome is squared, If and only if the currently scanned exponent bit has the value 1, a multiplication of the current result by *M* is performed following the squaring. The algorithm is signified in Function 3 in the form of pseudo code and its implementation in Figure 6.

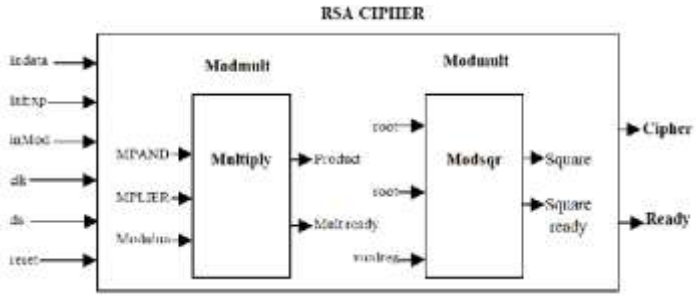

Fig 6. Implementation of Modular Exponentiation

Function 3: Pseudo code for RSACipher

```
RSACipher
{
Generic (KEYSIZE:integer:=128);
Port(indata:in std_logic_vector(KEYSIZE-1 downto 0);
        inExp: in std_logic_vector(KEYSIZE-1 downto 0);
        inMod: in std_logic_vector(KEYSIZE-1 downto 0);
        cipher: out  std_logic_vector(KEYSIZE-1 downto 0);
        clk,ds,reset : in std_logic  ready: out std_logic);
Component modmult
Generic (KEYSIZE:integer:=128)
Port (MPNAND, MPLIER, Modulus, Product);
}
```

To implement RSA cryptosystem the MMM42 algorithm is utilized and modified [Cheng]. The structural VHDL coding of the RSA includes two MMM42, one for encryption and another for decryption. The RSA-128 is carried out to compare the performance.

Function 4: Pseudo code for the Implementation of RSA Cryptosystem with MMM42 Multiplier

```
RSA_Encryption_Decryption
{
Encryption: MMM42_Multiplier port map (clk,rst, lda1, da1,
da2, pbk1,pbk2,mod in, sout1, sout2, done)
Decryption: MMM42_multiplier port map (clk, rst, lda2,
sout1,sout2, pvk1,pvk2, mod in, org1,org2,done2)
        Entity MMM42_multiplier
        {
                Port(clk,rst,lda,A1,A2,B1,B2,N,S_1,S_2)
                Component RSACypher
                Generic (KEYSIZE:integer:=128);
                Port (indata, inExp, inMod, Cypherout);
                Component CSA
                Port(X1, X2,X3,y1,y2);
                Component mux_4_1
                Port(a,b,c,d,sel,y1,out);
                Component lu_unit
                Port(a1,a2,S1,S2,S3,q_not,bypass,a_not);
                Component mbrfa
                Port(clk,rst,bypass,A1,A2,ai1,ai2,out);
        }
}
```

Figure 7 and Figure 8 shows Schematic Diagram of Main Unit which consists of both encryption and decryption of implemented VHDL code using Xilinx on FPGA. The operation of modules is described in sections B and C. The input data size for the system is 128 bit. The inputs to the module are *(da1, da2)=da* , the *da* is data which is a plaintext input to the system is represented in individual 64 bits. Similarly *(pbk1,pbk2)=pbk* ,*(pvk1,pvk2)=pvk*, and *modin* are the 64bit data given to cryptosystem. The *(enc1,enc2)=enc*,*(org1,org2)=org* are the output data.

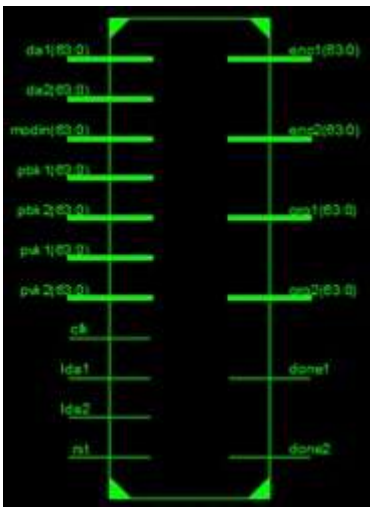

Fig 8. Entity of Main Unit

As shown in Figure 8 for encryption and decryption, the RSA module consists of two Modified MMM42 multiplier. The *(da1, da2)=da, (pbk1,pbk2)=pbk* and *modin* are the inputs to the encryption unit. The encrypted data output *(enc1,enc2)=enc* is obtained. This data is fed to the Decryption unit as *(Sout1,Sout2)=Sout* along with *(pvk1,pvk2)=pvk* and *modin*. The decrypted original plaintext *(org1,org2)=org* is obtained from decryption unit. The separated multiplier diagrams are shown in Figure 3 and Figure 4.

 ISSN: 2454-4248
227 – 240

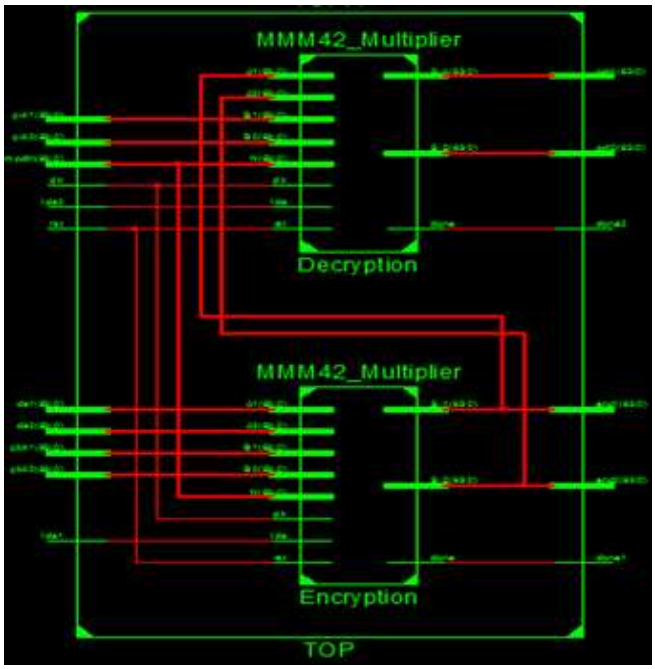

Fig 7. Complete RSA module

VII. Performance Analysis

A. *Simulation and Results with modified MMM42 Multiplier*

Spartan 3 XC3S400-5pq208 is chosen for software and hardware implementation. VHDL code is simulated with ISE simulator and waveforms are observed with Modelsim. The corresponding RTL schematic and timing diagrams are obtained.

In Figure 9 the data input (A1, A2), message bits (B1,B2), exponent (public key) and modulus (N) of 64 bits are applied and output (S_1,S_2) is obtained when done signal goes high. The entity and timing diagram for the encryption unit with modified MMM42 is shown in Figure 9 and Figure 10. The inputs considered are *((0011223344556677), (8899AABBCCDDEEFF))=(a1,a2)=A, (0000000000000000) ,(0000000000010001)=(b1,b2)=public key.* It is observed from the waveform that transmitter uses the public key for the encryption process which is shared between the sender and receiver. The following output is obtained *((125DB969FF426764),(FD832F8B30971598)=(s_1,S_2)=cy pher output, Clk, rst, lda and done are control signals.*

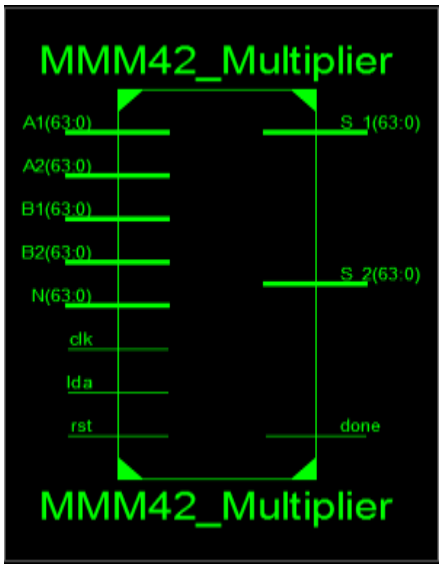

Fig 9. Entity of modified MMM42 multiplier

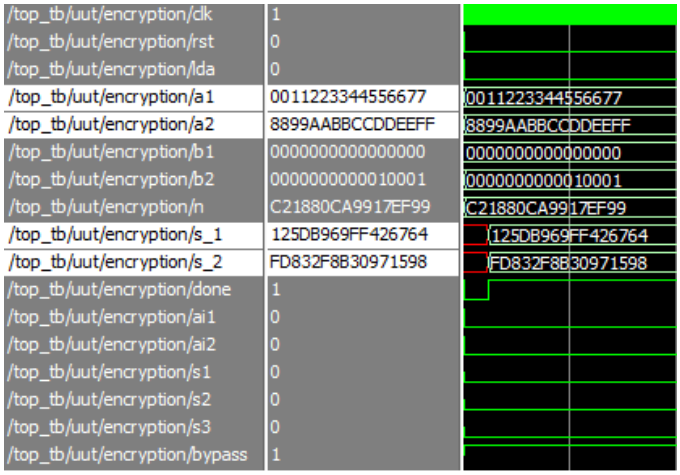

Fig 10. Timing Diagram of Encryption Unit

Table 3. Device Utilization of mMMM42 Multiplier

Device utilization Summary			
Logic utilization	***Used***	***Available***	***Utilization***
Number (No.) of slices	2271	5472	41%
No. of Slice Flip Flops	2035	10944	18%
No. of 4 input LUTs	4329	10944	39%
No. of bonded IOBs	452	320	141%
No. of GCLKs	1	32	3%

Table 2. Timing Summary of MMM42 Multiplier

Timing analysis of the RSA Cryptosystem with MMM42	
Speed Grade	12
Minimum (Min) period	9.847 nano seconds
Maximum (Max)frequency	101.554MHz
Min. input arrival time before clock	6.631 nano seconds
Max. output required time after clock	7.517 nano seconds
Max. combination path delay	9.547 nano seconds

 ISSN: 2454-4248
227 – 240

Figure 11 gives the Decryption output for modified MMM42 multilpier. Table 2 and Table 3 gives device utilized for the MMM42 multiplier in terms of Slices, Flip-flops, LUTs and timing summary with time and frequency respectively. It is observed from the results the utilization of hardware for each MMM42 multiplier is more, when compared with the other architecture, the details are given in Section B.

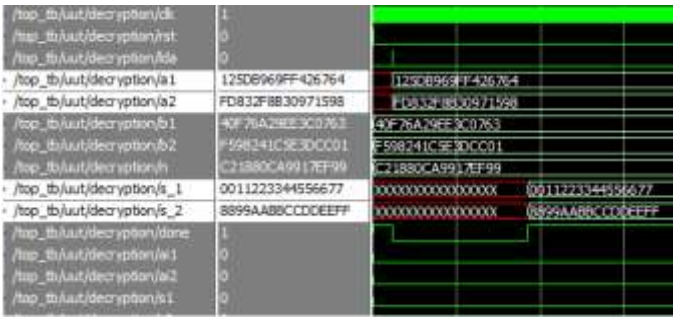

Fig 11. Timing Diagram of Decryption Unit

The entity and timing diagram for the decryption unit with modified MMM42 is shown in Figure 6 and Figure 8. The inputs considered are *((125DB969FF426764), (FD832F8B30971598))=(a1,a2)=A, ((40F76A229EE3C0763) ,(F598241C5E3DCC01)=(b1,b2)=private key*. It is observed from the waveform that transmitter uses the private key for the encryption process which is not shared between the sender and receiver, *only receiver knows about private key.* Table 4 gives the details about device utilized in FPGA and the timing details for input and output along the frequency are provided in the Table 5. From the device utilization, it is almost two times of the device utilization of MMM42 multiplier, as complete RSA system is constituted by two MMM42 multipliers. The same results are compared with other architecture given in section B.

Figure 12 shows the timing diagram obtained for implementation of Figure 6, which takes the 64 bit data and gives the result for both encryption and decryption. It is observed from the timing diagram that *da1, da2* which is plaintext input is obtained back through decryption as *org1, org2*.

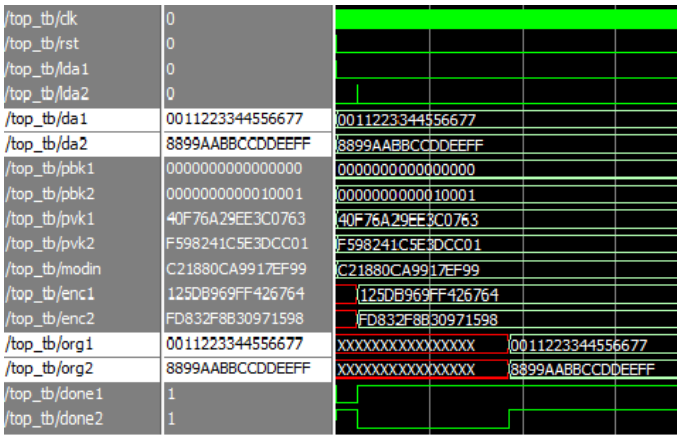

Fig 12. Timing Diagram of Encryption and Decryption

Table 4. Device Utilization of Complete RSA System

Device utilization Summary			
Logic utilization	***Used***	***Available***	***Utilization***
No. of slices	4645	5472	84%
No. of Slice Flip Flops	4068	10944	37%
No. of 4 input LUTs	8920	10944	81%
No. of bonded IOBs	710	320	221%
No. of GCLKs	1	32	3%

Table 5. Timing Summary of Complete RSA System

Timing analysis of the RSA Cryptosystem with MMM42	
Speed Grade	12
Min. period	9.847 nano seconds
Max. frequency	101.554MHz
Min. input arrival time before clock	7.003 nano seconds
Max. output required time after clock	7.545 nano seconds
Max. combination path delay	9.555 nano seconds

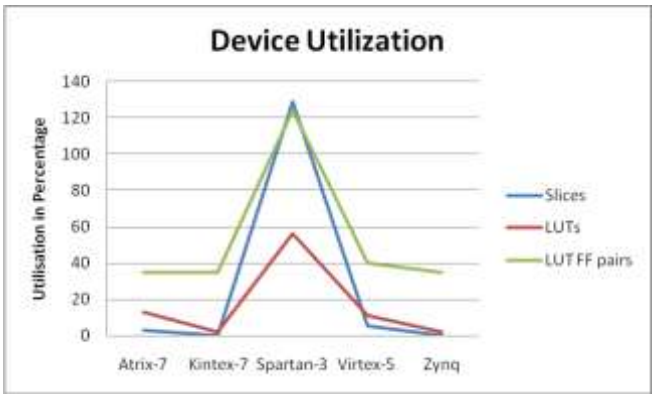

Fig 13. Device utilization with mMMM42 Multiplier

Figure 13 describes the usage of Slices, Look Up Tables (LUT), LUT Flip Flop pairs with the algorithm executed using the MMM42 multiplier with reference to Table 7. This is compared with the algorithm executed without MMM42. In Spartan-3 (Xc3s400-5pq208) utilization is more and number of LUTs used are beyond 100%, so it is not used for hardware implementation. Atrix-7 (xc7a100t-3csg324) is utilized for implementation and encryption and decryption time are also measured.

B. Simulation and Results without MMM42 Multiplier

 ISSN: 2454-4248
227 – 240

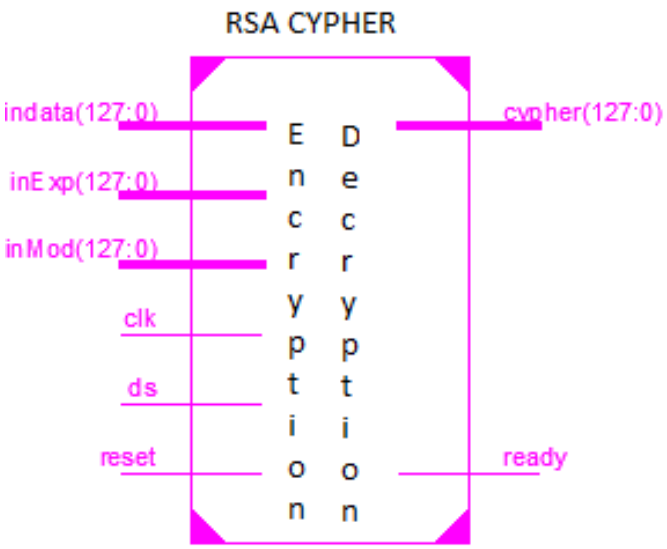

Fig.14 Entity Module of RSACypher_128

Figure 14 provides the module to perform encryption and decryption which is utilized to implement complete RSA Cryptosystem. The RTL Schematic of the same is shown in Figure 15.

Fig.15 RTL Schematic of RSACipher_128

The inputs to the **RSA CYPHER-128** Encryption includes Public key, (modulus M) modin and data_in (plain text message) as inputs of 128 bits and Cipher as output. Decryption unit takes Cipher, Private key and Modulus(M) as inputs to produce the plaintext message output i.e., original_msg. Figure 16 gives the corresponding timing diagram of **RSA CYPHER_128** shown in Figure 15 and Static power utilization details in Figure 13. The device utilisation for the complete RSA system is very less that is below 50% compared to the earlier architecture discussed in section A. It is also observed from the execution and timing diagrams the time required for encryption and decryption is around 50%. Through comparison, the second architecture is best suitable for WSNs as it satisfies the constraints and limitations in terms of hardware usage.

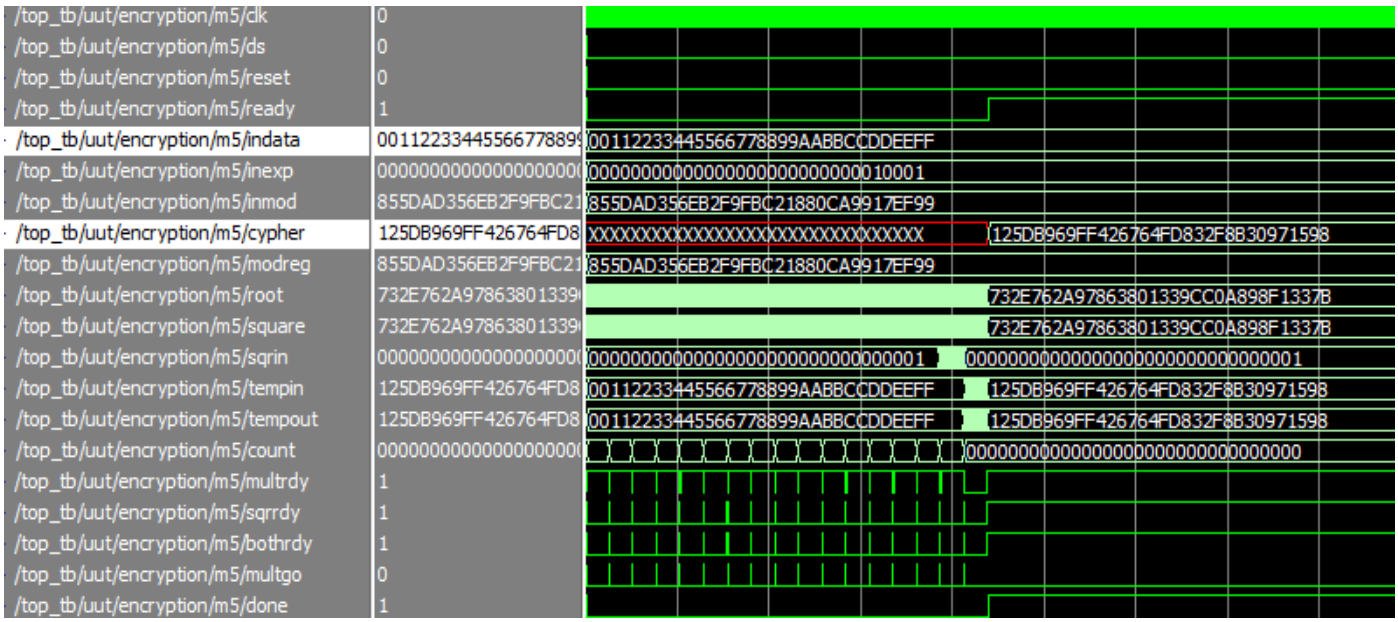

Fig 16. Timing Waveform RSA System Without Modified MMM42 Multiplier(RSACIPHER 128 Cryptosystem)

 ISSN: 2454-4248
227 – 240

Table 6. Device Utilization of Complete RSA System without mMMM42 Multiplier

Device utilization Summary			
Logic utilization	***Used***	***Available***	***Utilization***
No. of slices	1889	5472	34%
No. of Slice Flip Flops	1807	10944	16%
No.r of 4 input LUTs	3474	10944	31%
No. of bonded IOBs	516	320	161%
No. of GCLKs	1	32	3%

Figure 17 describes the usage of Slices, Look Up Tables (LUT), LUT Flip Flop pairs with the algorithm executed without using the MMM42 multiplier, with reference to Table 8. This is compared with the algorithm executed with MMM42. In Spartan-3 (Xc3s400-5pq208) utilization is more and numbers of LUTs used are beyond 100%, so it is not used for hardware implementation.

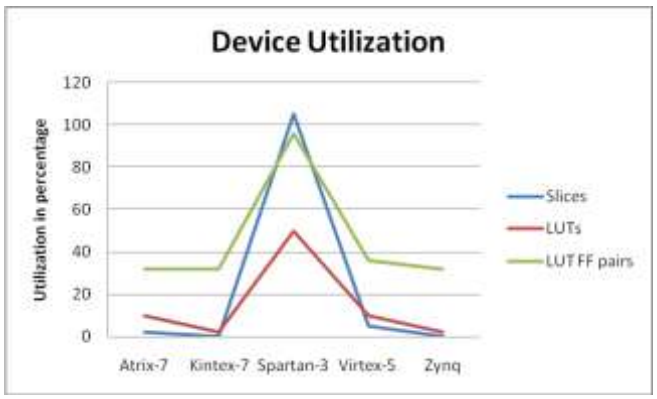

Fig 17. Device utilization without MMM42 Mulltiplier (RSACIPHER 128 Cryptosystem)

Fig 18. Static Power Utilization

C. Hardware Results with MMM42 Multiplier

The VHDL code is synthesized using the Spartan 3E FPGA on Chip Scope Pro with Virtual input and output as Spartan 3 E cannot provide complete 64-bit data on LEDs. Figure 14 shows the encryption of 64-bit data individually and the same data bits are utilized in software implementation. Figure 15 gives the decryption of data. The Virtual inputs and outputs are chosen only for encryption and decryption *i.e.*, plaintext and cipher text. Other inputs and outputs are not shown in Figures 19 and Figure 20.

As the device utilization is less in the architecture RSA CIPHER-128 (Figure 11) , the same code is implemented using Mentor Graphics for ASIC design. Figure 17 gives the Area report of Figure 11 *i.e.*, RTL Schematic of RSACipher_128. It describes the number of ports used for implementation as 772 and number of instances as 2, that shows minimum use of hardware with the second architecture implementation. Figure 18 shows very less delay time and the time required for data arrival is 0 and for output arrival is 10.0.

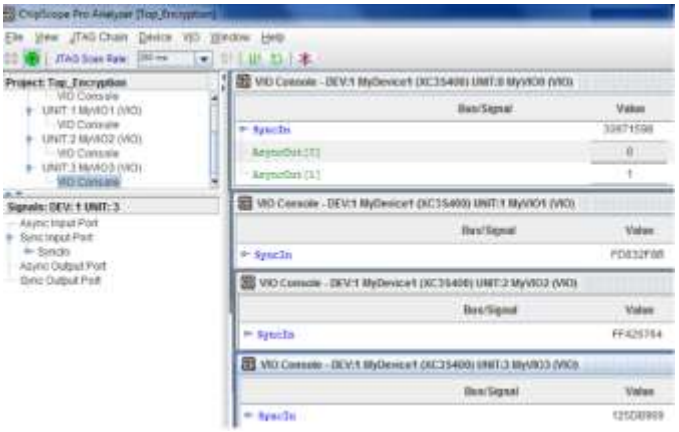

Fig19. Encryption of Data

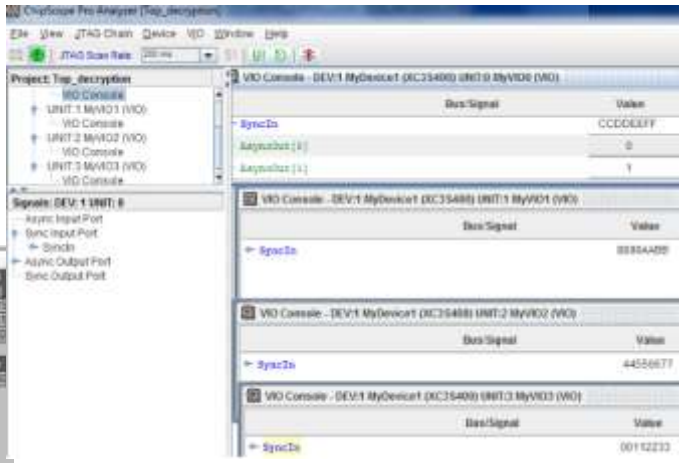

Fig 20.Decryption of Data

Two different architectures are designed and implemented to compare the performance. RSA CIPHER-128 multiplier gives the good performance with respect to speed and area. It is observed from the results shown in Figure 21 and Figure 22 that implementation with modified MMM42 multiplier is not suitable for WSN nodes as it consumes 50% more hardware in FPGA.

The design RSA CIPHER-128 is selected and also implemented using Mentor Graphics for ASIC design. It is also observed from the Figures 17 and 18 , with device Spartan-3 for both architectures *i.e.*, with and without MMM42, the utilisation of hardware is more, that leads to select the device Atrix-7 for Hardware implementation.

 ISSN: 2454-4248
227 – 240

```
Cell: TOP     View: Behavioral     Library: work

Cell: TOP     View: Behavioral     Library: work

*******************************************************

*******************************************************


 Cell              Library  References      Total Area


 Cell              Library  References      Total Area

RSACypher_128      work      2 x       1       2 RSACypher_128


 Number of ports :                        772
 Number of nets :                         772
 Number of instances :                      2
 Number of references to this view :        0

Total accumulated area :

 Black Box RSACypher_128 :                  2

 Number of accumulated instances :          2
Info, Command 'report_area' finished successfully
```

Fig 16. Area Report

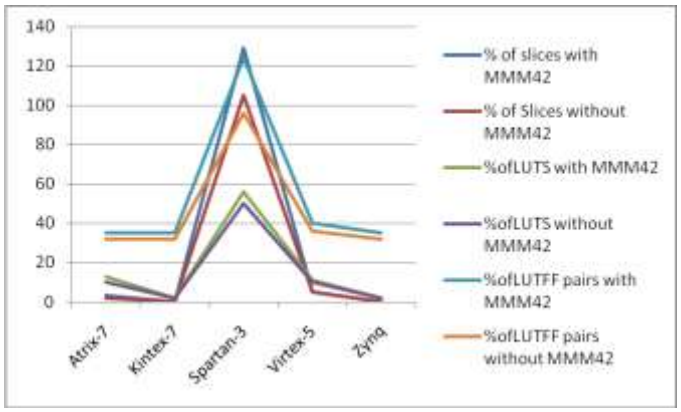

Fig. 21 Comparison of Device Utilization

Device	Atrix-7	Kintex-7	Spartan-3	Virtex-5	Zynq
% of slices with MMM42	3	0	129	5	0
% of Slices without MMM42	2	0	105	5	0
%ofLUTS with MMM42	13	2	56	11	2
%ofLUTS without MMM42	10	2	50	10	2
%ofLUTFF pairs with MMM42	35	35	124	40	35
%ofLUTFF pairs without MMM42	32	32	96	36	32

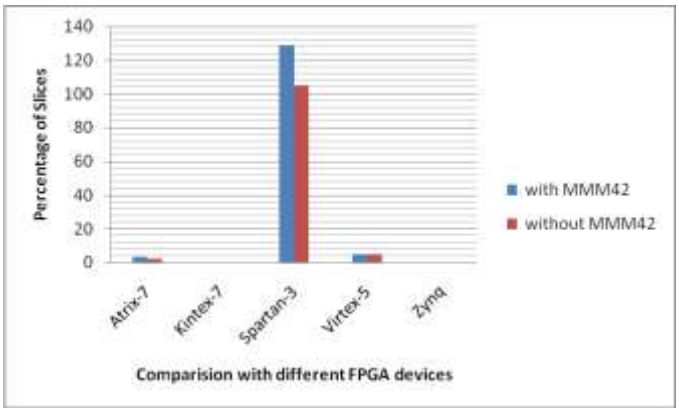

Fig. 22. Comparision of Slices Utilized with and without MMM42 Multiplier

```
                        Critical Path Report

Critical path #1, (path slack = 10.0):


Critical path #1, (path slack = 10.0):

NAME                                        GATE              ARRIVAL            LOAD
------------------------------------------------------------------------------------
encryption/cypher(0)                        GENERIC_BLACK_BOX  0.00  0.00 up          0.07
cyphertext(0)/                                                 0.00  0.00 up    0.00
data arrival time                                                    0.00

data required time                                                   10.00
------------------------------------------------------------------------------------
data required time                                                   10.00
data arrival time                                                    0.00
                                                                  ---------
slack                                                              10.00
------------------------------------------------------------------------------------


Info, Command 'report_delay' finished successfully
```

Fig 23. Critical Path Report

 ISSN: 2454-4248
227 – 240

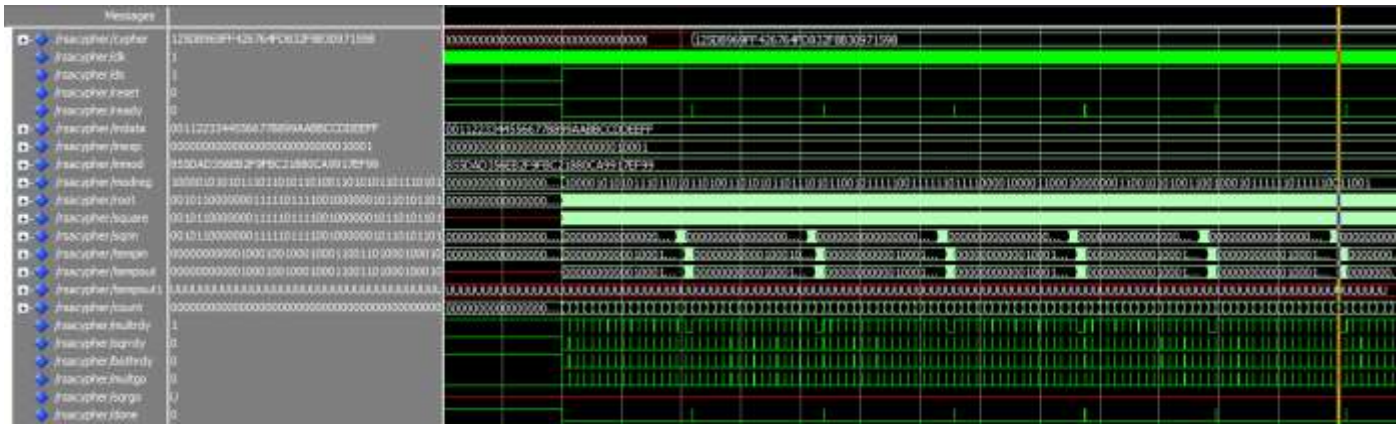

Fig 24. Timing diagram - RTL Schematic of RSACipher_128

Table 7. Device Utilization of Complete RSA System with mMMM42 Multiplier

Device	Clock	Time	Available Slices	Used	Usage	Available LUTs	Used	Usage	Available LUT FF pairs	Used	Usage
Atrix-7 xc7a100t-3csg324	148.534 MHz	6.732 Nano sec	126800	4062	3%	63400	8294	13%	9110	3246	35%
Kintex-7 Xc7k480t-3ffg1156	222.074 MHz	4.503 Nano sec	597200	4062	0%	298600	8294	2%	9110	3246	35%
Spartan-3 Xc3s400-5pq208	46.677 MHz	21.424 Nano sec	3584	4639	129%	7158	4068	56%	7168	8908	124%
Virtex-5 Xc5vls110t-3ff1136	144.52 MHz	6.919 Nano sec	69120	4062	5%	69120	8202	11%	8750	3514	40%
Zynq Xc7z100-2ffg1156	199.931 MHz	5.002 Nano sec	554800	4062	0%	277400	8294	2%	9112	3244	35%

Table 8. Device Utilization of Complete RSA System RSACIPHER-128 Multiplier

Device	Clock	Time	Available Slices	Used	Usage	Available LUTs	Used	Usage	Available LUT FF pairs	Used	Usage
Atrix-7 xc7a100t-3csg324	148.534 MHz	6.732 Nano sec	126800	3606	2%	63400	6850	10%	7865	2591	32%
Kintex-7 Xc7k480t-3ffg1156	222.074 MHz	4.503 Nano sec	597200	3606	0%	298600	6850	2%	7865	2591	32%
Spartan-3 Xc3s400-5pq208	46.677 MHz	21.424 Nano sec	3584	3764	105%	7158	3606	50%	7168	6928	96%
Virtex-5 Xc5vls110t-3ff1136	142.77 MHz	7.664 Nano sec	69120	3606	5%	69120	7108	10%	7864	2850	36%
Zynq Xc7z100-2ffg1156	199.931 MHz	5.002 Nano sec	554800	3606	0%	277400	6850	2%	7865	2591	32%

 ISSN: 2454-4248
227 – 240

Table 9. Execution Speed Performance Analysis with mMMM42 Multiplier Cryptosytem

Device	Clock (MHz)	Time (∏Sec)	Area (slices)	Usage (%)	Encryption time (ns)	Decryption time (ns)
Atrix-7	148.534	6.732	3606	2%	50.012	190285
Kintex-7	142.776	7.004	3606	5%	50.012	190285
Spartan-3	49.679	20.129	3764	105%	50.012	190285
Virtex-5	142.776	7.004	3606	5%	50.012	190285
Zynq	199.931	5.002	3606	0%	50.012	190285

Table 10. Execution Speed Performance Analysis RSACIPHER-128 Cryptosystem

Device	Clock (MHz)	Time (∏Sec)	Area (slices)	Usage (%)	Encryption time (ns)	Decryption time (ns)
Atrix-7	148.534	6.732	4062	3	30.002	185764
Kintex-7	222.074	4.503	3606	0	30.002	185764
Spartan-3	46.677	21.424	4639	129	30.002	185764
Virtex-5	144.526	6.919	4062	5	30.002	185764
Zynq	199.931	5.002	4062	0	30.002	185764

The modules discussed in this section are designed and implemented in VHDL code and tested on ISE 13.2 software using XILINX FPGA Artix-7 xc7a100t-2csg324 device and simulated with Modelsim Simulator. This FPGA is advised for implementation of Public Key Cryptography as it supports different degrees of security and high-speed execution for GF computations. The static power consumption is also very less compare to other FPGAs [2].

Table 11. Execution Speed Performance Analysis on FPGA Artix-7 xc7a100t-2csg324 with mMMM42 Cryptosystem

Device	Clock (MHz)	Time (∏Sec)	Area (slices)	Usage (%)	Encryption time (ns)	Decryption time (ns)
ARTIX-7	148.534 MHz	6.732	4062	3	20.198	125064

Table 12. Execution Speed Performance Analysis on FPGA Artix-7 xc7a100t-2csg324 RSACIPHER-128 Cryptosystem

Device	Clock (MHz)	Time (∏Sec)	Area (slices)	Usage (%)	Encryption time (ns)	Decryption time (ns)
ARTIX-7	148.534	6.732	3606	2	33.670	128108

VIII. CONCLUSIONS

To achieve better security and improve the speed constraints a high degree of flexibility with respect to the cryptographic algorithms is desirable in WSNs. RSACIPHER-128 cryptosystem gives good performance in terms of speed and area. It is observed from the results that the implementation with mMMM42 multiplier is not suitable for WSN nodes as it consumes 50% more hardware in FPGA. The design RSACIPHER-128 is selected and also implemented using Mentor Graphics for ASIC design. It is planned to take up Vedic multipliers to perform modular multiplications in future work to speed up the multiplication operation and to reduce the hardware usage.

REFERENCES

[1] 1. Ian F Akylidiz, Weilian Su, Yogesh Sankara subramani --am and E Cayirci, "Wireless Sensor Network : A Sur vey on Sensor Networks," *in IEEE Communication Magazine,* ISSN:0163-6804, vol. 40, no. 8, pp. 102-114 , 2002.

[2] Antonio de la Piedra, An Braeken, and Abdellah Touhafi. 2012. Sensor Systems Based on FPGAs and Their Applications: A Survey. Sensors (2012), 12, 12235-12264; DOI:10.3390/s 120912235.

[3] Abdullah Said Alkalbani, Teddy Mantoro, Abu Osman Md Tap, "Comparison between RSA Hardware and Software Implementation for WSNs Security Schemes", in Proceedings of Third International Conference on ICT4M, pp. E84-E89, 2010.

[4] P L Montgomery, " Modular Multiplication Without Trial Division," Mathematics Computations, vol.44, no. 170, pp. 519-521, 1985.

[5] Sushanta Kumar Sahu Manoranjan Pradhan, "Implementation of Modular Multiplication for RSA Algorithm," in IEEE Conference on Communication Systems and Network Technologies, pp. 112-114, 2011.

[6] Jainath Nasreen P, Denila N, " A Novel Architecture for VLSI Implementation of RSA Cryptosystem," in IEEE International conference on ICCEET, pp. 606-609, 2012.

[7] C. Mclvor, M. McLoone, J. V., McCanny: Fast Montgomery Modular Multiplication and RSA Cryptographic Processor Architectures. In: proceedings of 37th Asilomar Conference on Signals, Systems, Computations, vol.1, pp. 379-384, 2003.

[8] Alan Daly, William Marnane: Efficient Architectures for Implementing Montgomery Modular Multiplication and RSA Modular Exponentiation on Reconfigurable Logic. In: Proceedings of the 2002 ACM/SIGDA tenth International Symposium on FPGAs, pp. 40-49, ACM 1-58113-452-5/02/2002, Monterey, California, USA

[9] Ridha Ghayoula, ElAmjed Hajlaoui, Talel Korkobi, Mbarek Traii, Hichem Trabelsi, " FPGA Implementation of RSA Cryptosystem," in World Academy of Science, Engineering and Technology, vol. 20, no. 3, pp. 1005-1009, 2008.

[10] Gustavo D Sutter, Deschamps, Jean-Pierre and Imana, Jos´e Luis, "Modular Multiplication and Exponentiation Architectures for Fast RSA Cryptosystem Based on Digit Serial Computation.," IEEE Transactions on Industrial Electronics, vol. 58, no.7 pp. 3101-3109, 2011.

[11] A Miyamoto, N Homma, T Aoki and A Satoh, "Systematic Design of RSA Processors Based on High-Radix Montgomery Multipliers," in IEEE Transactions on VLSI, pp. 1-11, 2011.

[12] T Blum, and C Paar, "High Radix Montgomery Multiplication on Reconfigurable Hardware," in IEEE Transactions on Computers , vol. 50, no.7, pp. 759-764, 2001.

[13] E Michalski and D A Buell, "A Scalable Architecture For RSA Cryptography on Large FPGAs," in Proceedings of International Conference on Field Programmable Logic and Applications, 2006.

[14] Jin-Hua Hong, Cheng-Wen Wu Cellular-Array Modular Multiplier for Fast RSA Public-Key Cryptosystem Based on Modified Booths Algorithm", in IEEE Transactions on Very Large Scale Integration(VLSI) Systems, vol. 11, no. 3, pp. 474-484, June 2003.

[15] Kauther M Amer, Sami M Sharif, Ahmed S Ashur, " Enhancement of Hardware Modular Multiplier Radix-4 Algorithm for Fast RSA Cryptosystem," in Proceedings of the 10th Conference on computing , Electrical and Electronic engineering, pp. 692-697, 2013.

[16] A P Fournaris and O Koufopavlou, "Montgomery Modular Multiplier Architectures and Hardware Implementations for an RSA Cryptosystem," pp. 778-793.

[17] Mentens, Nele and Sakiyama, Kazuo and Preneel, Bart and Verbauwhede, Ingrid, "Efficient Pipelining for Modular Multiplication Architectures in Prime Fields," in Proceedings of the 17th ACM Great Lakes Symposium on VLSI, pp. 534-539, 2007.

[18] Mentens S S Ghoreishi, M A Pourmina, H Bozorgi, M Dousti, "High Speed RSA Implementation Based on Modified Booths Technique and Montgomery's Multiplication for FPGA Platform," in Proceedings of Second International Conference on Advances in Circuits, Electronics and Micro-Electronics, pp. 534-539, 2009.

[19] Desiree Juby Vincent, " Fast and Area Efficient RSA Cryptosystem Design Using Modified Montgomery Multiplication for FPGA Applications," in International Journal of Scientific Engineering Research, vol.4, no. 7, pp. 2221-2228, 2013.

[20] Tamer Gudu, " A new scalable hardware architecture for RSA algorithm," in Proceedings of the International conference on Field Programmable Logic and Applications, pp. 670-674, 2007.

[21] Milo Drutarovsk, Martin Imka, Viktor Fischer, "Comparison of Scalable Montgomery Modular Multiplication Implementations Embedded in Reconfigurable Hardware," in Acta Electrotechnica et Informatica, Versita, vol. 6, no. 2, pp. 37-45, 2008.

[22] Kamala, Ramachandruni Venkata and Srinivas, M B, " High-Throughput Montgomery Modular Multiplication," in proceedings of International Conference on Very Large Scale Integration, pp. 58-62, 2006.

[23] Kuang, Shiann-Rong and Wang, Jiun-Ping and Chang, Kai-Cheng and Hsu, Huan-Wei, " Energy-Efficient High-Throughput Montgomery modular multipliers for RSA Cryptosystems," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.21, no.11, pp. 1999-2009, 2013.

[24] Chen, Jun-Hong and Wu, Haw-Shiuan and Shieh, Ming-Der and Lin,Wen-Ching, "A New Montgomery Modular Multiplication algorithm and its VLSI Design for RSA Cryptosystem," in IEEE International Symposium on Circuits and Systems, pp. 3780-3783, 2007.

[25] Chen, Jun-Hong and Wu, Haw-Shiuan and Shieh, Ming-Der and Lin, Wen-Ching, "An efficient montgomery Multiplication Algorithm and RSA Cryptographic Processor," in International Conference on Computational Intelligence and Multimedia Applications, vol.2, pp.188-195, 2007.

[26] Shieh, Ming-Der and Chen, Jun-Hong and Lin, Wen-Ching and Wu, Hao-Hsuan, "A New Algorithm for High-Speed Modular Multiplication Design," in IEEE Transactions on Circuits and Systems , vol.56, no.9, 2009.

[27] Shieh, Ming-Der and Chen, Jun-Hong and Wu, Hao-Hsuan and Lin, Wen-Ching, "A New Modular Exponentiation Architecture for Efficient Design of RSA Cryptosystem," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.16, no.9, pp. 1151-1161, 2008.

[28] Shiann-Rong Kuang, Kun-Yi Wu, Ren-Yao Lu: Low Cost High Performance VLSI Architectures for Montgomery Modular Multiplication, In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.24, no.2, pp. 434-443, 2016.