American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-4, Issue-6, pp-144-151
www.ajer.org
Research Paper Open Access

Implementation of RSA Encryption Algorithm on FPGA

Amit Thobbi[1] ShriniwasDhage[2] Pritesh Jadhav[3] Akshay Chandrachood[4]
[1234]*Department Of Electronics & Telecommunication, PVG's college of Engineering & Technology, Pune, India.*

ABSTRACT: *This paper presents a scheme for implementation of RSA encryption algorithm on FPGA. A 64 bit cipher text is accepted and using 128 bit public key RSA encryption technique, a 64 bit encrypted message is generated. Each block is coded using VHDL and the code is synthesized and simulated using Xilinx ISE Design Suite 14.7.Unlike previous approaches we have systematically provided timing, area and power measures for Spartan 3 and Virtex 6 FPGA using Pre and Post synthesis simulations. The design is optimized for either speed or power and a tradeoff is presented between speed, power and space. If the design is optimized for power then fewer resources are consumed but the maximum usable frequency is also reduced. Spartan 3 FPGAs are best suited for low power designs. As a major practical result we show that it is possible to implement RSA algorithm at secure bit lengths on a single commercially available FPGA.*

KEYWORDS: *RSA Encryption, Virtex6 FPGA, Spartan3 FPGA, Montgomery Algorithm*

I. INTRODUCTION

The art of keeping messages secure is Cryptography. Cryptography plays an important role in the security of data. It enables us to store sensitive information or transmit it across insecure networks so that unauthorized persons cannot read it. [5]

The RSA algorithm was introduced in the year 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman. It is a public key cryptography algorithm. It is a secure, high quality algorithm. [5] It can be used as a method of exchanging secret information such as keys and producing digital signatures. However, the RSA algorithm is computationally intensive, operating on very large (typically thousands of bits long) integers. [2]

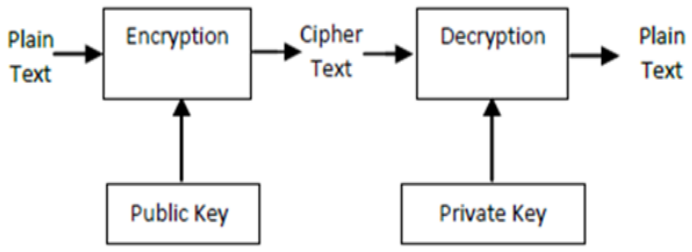

Figure 1: Public key cryptography

The rising growth of data communication and electronic transactions has made system security to become the most important issue. To provide modern security features Public key cryptosystems are used; one such cryptosystem is the RSA algorithm. A public key stored in the hardware, itself improves security greatly.

II. IMPLEMENTATION METHOD

a) Block Diagram of RSA

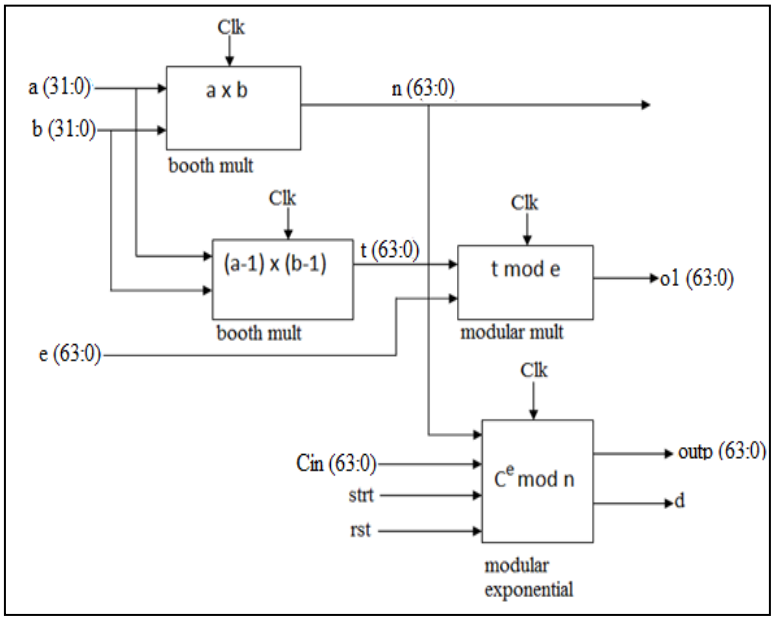

Figure 2: Implementation of RSA

The blocks shown in figure 2 are implemented in Xilinx ISE 14.7 using VHDL. The Top module of the implemented code is as shown in figure 3.

```
begin

u1:booths port map("00000000000000000000000000001101","00000000000000000000000000010001",clk,n); --13 & 17
u2:booths port map("00000000000000000000000000001100","00000000000000000000000000010000",clk,t); --12 & 16

--choose public key exponent 17 which is not a factor of 221

-- u3 to check public key validity
u3:modmult port map("00000000000000000000000000000000000000000000000000000000000000001",t,
                    "00000000000000000000000000000000000000000000000000000000000010001",o1,clk,'1',rst,rd);

-- add public key exponent 17  to u4
u4:Montgomery_exponentiator_msb port map("00000000000000000000000000000000000000000000000000000000000010001",
                                         cin,clk,rst,strt,outp,d);


end Structural;
```

Figure 3: Top block of the implemented code

The steps involved in the implementation procedure, as shown in figure 2 are

1. Selection of the prime nos. ‘a’ and ‘b’
   The prime no’s ‘a’ and ‘b’ are 32 bit each. These are the system inputs.
   ‘13’ and ‘17’ are the two numbers taken for explanation purpose

2. Generation of the Public Key modulus (n)
n is generated by multiplying the two 32 bit prime nos.' a' and 'b' resulting into the 64 bit modulus 'n'. This multiplication is performed by the Booth's algorithm.
This is the U1 component. '13' and '17' are the inputs given and output generated is 221

3. Generation of the totient (t).
The totient is generated by multiplying (a-1) and (b-1), which gives a 64 bit totient. This multiplication is accomplished by Booth's algorithm.
This is accomplished by the U2 component.'12' and '16' are inputs and '192' is output generated.

4. Generation of public key exponential (e).
'e' is a 64 bit prime number which is co prime to 't'. The modular block checks this condition. Modular multiplication is carried out between e and t i.e. (e mod t) resulting into a 64 bit output denoted as 'o1'. The public key exponential is chosen by the user. Number '17' is chosen as the public key exponent. This is given to the U3 component and the public key condition is checked.

5. Encryption of the data.
In this stage the encryption of user data is carried out by modular and exponential operations
i.e. outp = Cin^e mod n.
This step results into a 64 bit encrypted message. The U4 component performs this operation. The public key exponent and modulus is given to the component and Cin is the user message. 'Outp' is the output generated. The public key exponent taken is '17 'and public key modulus taken is '221'.

b) Functional verification of RSA Algorithm (Behavioral Simulation)

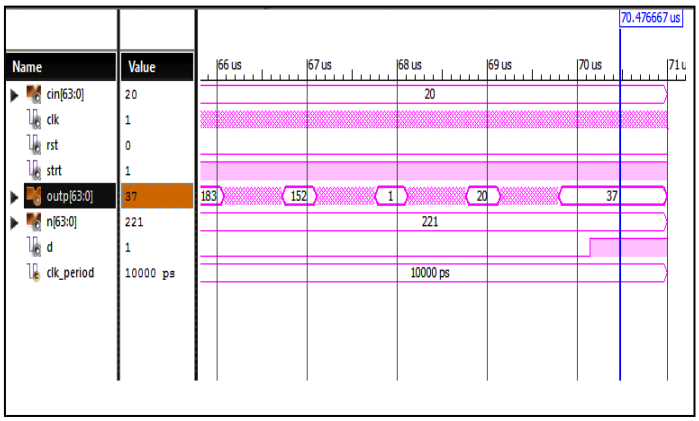

Figure 4: Behavioral simulation result of RSA Algorithm

Figure 4 shows the simulation results. Table I shows the parameters used for checking the correctness of the design i.e. behavioral simulation. The result is verified using theoretical calculation.

Parameter	Description	Value
cin	User input message	20
n	Public key modulus	221
e	Public key exponential	17
outp	Encrypted message output	37

Table I : Input Parameters for behavioral Simulation

Operation performed is outp = cin^e mod n
Substituting the values
outp = 20^{17} mod 221
outp=37

Figure 5: Result verified using Wolfram Alpha

III. TEST CONDITIONS

Device used : Spartan3 XC3s400fg320-4, Virtex6 XC6vlx75tff484-1
Design and Analysis Tool : Xilinx PlanAhead 14.7

Results were achieved for the following synthesis and implementation settings shown in Tables II and III.

Parameter	Spartan 3		Virtex 6	
	Power	Speed	Power	Speed
Opt mode	area	speed	area	speed
Opt level	1	2	1	2
FSM Encoding	gray	Speed1	gray	Speed1
lc	N.A.	N.A.	area	auto
Netlist hierarchy	As_optimised	Rebuilt	As_optimised	Rebuilt

Table II. Synthesis Settings

Parameter	Spartan 3		Virtex 6	
	Power	Speed	Power	Speed
pr	none	b	none	b
ir	off	off	off	off
cm	area	speed	N.A	N.A
Logic_opt	on	off	on	off
Global_opt	on	off	on	off
lc	on	off	on	off
Ol(map)	none	none	none	none
Mt(map)	N.A	N.A	on	on
Power(map)	on	off	on	off
Ol(par)	std	none	std	none
Power(par)	on	off	on	off

Table III: Implementation Settings

IV. RESULTS

a) Synthesis results

```
Selected Device : 3s400fg320-4

 Number of Slices:                  420  out of   3584   11%
 Number of Slice Flip Flops:        425  out of   7168    5%
 Number of 4 input LUTs:            714  out of   7168    9%
    Number used as logic:           710
    Number used as Shift registers:   4
 Number of IOs:                     196
 Number of bonded IOBs:             196  out of    221   88%
 Number of GCLKs:                     1  out of      8   12%
```

Figure 6: synthesis result for spartan3 power optimized

```
Selected Device : 3s400fg320-4

 Number of Slices:                  625  out of   3584   17%
 Number of Slice Flip Flops:        563  out of   7168    7%
 Number of 4 input LUTs:            851  out of   7168   11%
    Number used as logic:           847
    Number used as Shift registers:   4
 Number of IOs:                     196
 Number of bonded IOBs:             196  out of    221   88%
 Number of GCLKs:                     1  out of      8   12%
```

Figure 7: synthesis result for spartan3 speed optimized

```
Selected Device : 6vlx75tff484-1


Slice Logic Utilization:
 Number of Slice Registers:         187  out of  93120    0%
 Number of Slice LUTs:              241  out of  46560    0%
    Number used as Logic:           239  out of  46560    0%
    Number used as Memory:            2  out of  16720    0%
       Number used as SRL:            2
```

Figure 8: synthesis result for Virtex6 power optimized

```
Selected Device : 6vlx75tff484-1


Slice Logic Utilization:
 Number of Slice Registers:         205  out of  93120    0%
 Number of Slice LUTs:              253  out of  46560    0%
    Number used as Logic:           251  out of  46560    0%
    Number used as Memory:            2  out of  16720    0%
       Number used as SRL:            2
```

Figure 9: synthesis result for Virtex6 speed optimized

b) Maximum Frequency (PAR result)

The below figures show the PAR result i.e. the maximum usable frequency. A Timing score of zero indicates design has met all constraints. [6]

Name	Part	Constraints	Strategy	Status	Progress	Start	Elapsed	Util (%)	FMax (MHz)	Timing Score
synth_4	xc6vlx75tff484-1	constrs_1	PlanAhead Defaults (XS...	XST Complete!	100%	5/29/15 6:43 PM	00:00:28	0.000	228.595	
synth_5 (active)	**xc6vlx75tff48...**	**constrs_1**	**PlanAhead Defaults ...**	**XST Complete!**	**100%**	**5/29/15 4:13 PM**	**00:00:25**	**0.000**	**228.595**	
impl_2 (active)	**xc6vlx75tff48...**	**constrs_1**	**ISE Defaults (ISE 14)**	**Implementation Out-of-date**	**100%**	**5/29/15 4:27 PM**	**00:01:52**	**1.000**	**224.568**	**0**
synth_6	xc6vlx75tff484-1	constrs_1	implaxsat (XST 14)	XST Complete!	100%	5/30/15 1:23 PM	00:00:38	0.000	228.595	
synth_7	xc6vlx75tff484-1	constrs_1	implaxsat (XST 14)	XST Complete!	100%	5/31/15 2:10 PM	00:00:29	0.000	228.595	
impl_6	xc6vlx75tff484-1	constrs_1	impl4 (ISE 14)	PAR Complete!	100%	5/31/15 2:18 PM	00:02:05	1.000	203.046	0

Figure 10: PAR Result for Virtex6 speed optimized and power optimized design

Name	Part	Constraints	Strategy	Status	Progress	Start	Elapsed	Util (%)	FMax (MHz)	Timing Score
synth_4	xc3s400fg320-4	constrs_1	PlanAhead Defaults (XS...	Synthesis Out-of-date	100%	4/11/15 6:27 PM	00:00:25	9.000	71.355	
impl_4	xc3s400fg320-4	constrs_1	ISE Defaults (ISE 14)	Implementation Out-of-date	100%	5/28/15 5:47 PM	00:01:08	9.000	66.854	0
synth_5	xc3s400fg320-4	constrs_1	TimingWith1OBPacking (...	Synthesis Out-of-date	100%	5/28/15 7:24 PM	00:00:32	11.000	86.730	
impl_5	xc3s400fg320-4	constrs_1	ParHighEffort (ISE 14)	Implementation Out-of-date	100%	5/28/15 7:57 PM	00:00:44	11.000	78.920	0
synth_6 (active)	xc3s400fg320...	constrs_1	synth4spartan (XST ...	XST Complete!	100%	5/30/15 11:21 PM	00:00:28	9.000	71.355	
impl_6 (active)	xc3s400fg320...	constrs_1	impl4sprtan (ISE 14)	PAR Complete!	100%	5/31/15 9:14 AM	00:01:18	9.000	67.696	0

Figure 11: PAR Result for Spartan3 speed optimized and power optimized design

c) Timing result

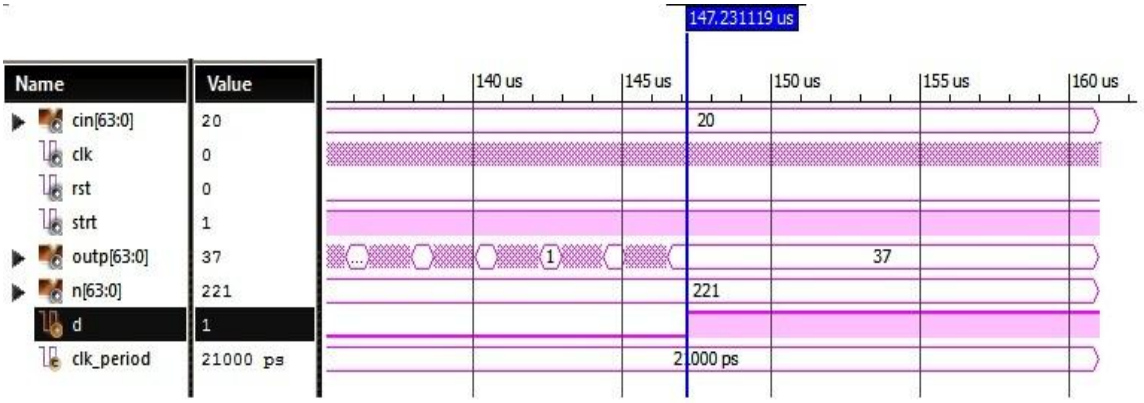

Figure 12: Timing result for Spartan3 power optimized design

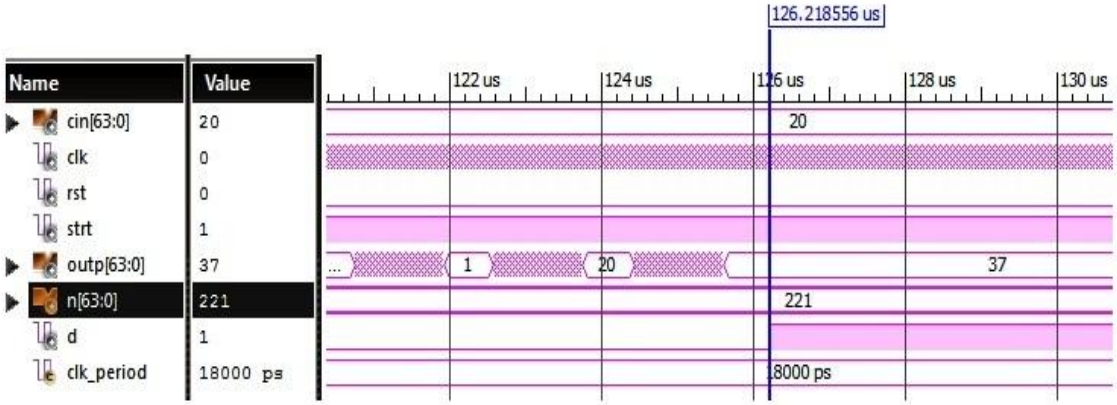

Figure 13: Timing result for Spartan3 speed optimized design

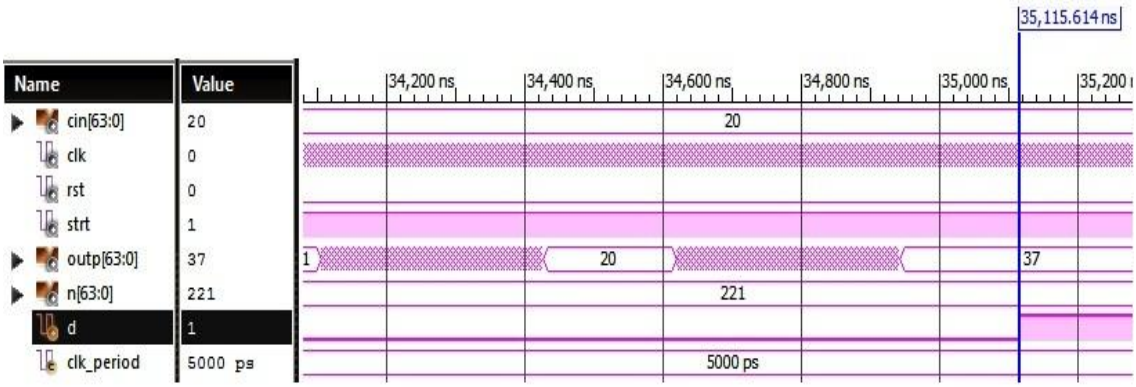

Figure 14: Timing result for Virtex6 power optimized design

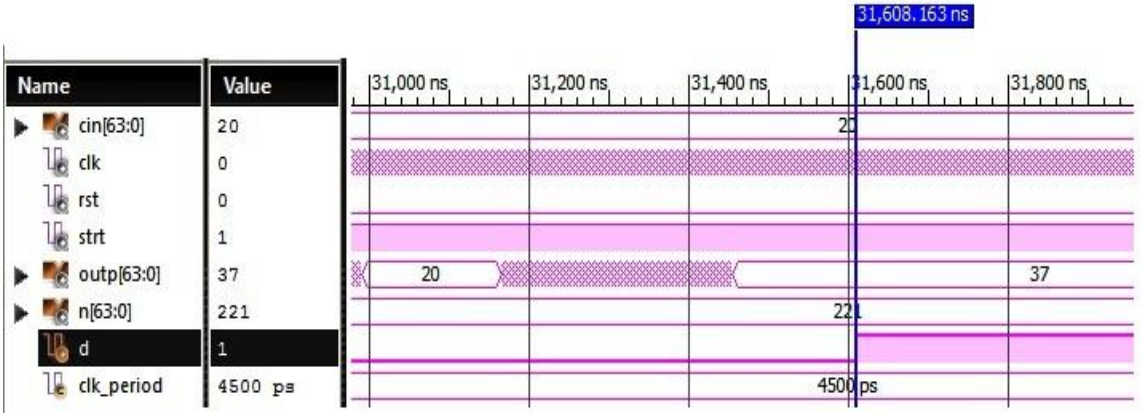

Figure 15: Timing result for Virtex6 speed optimized design

d) Power analysis results

Device	
Family	Spartan3
Part	xc3s400
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-4

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	774	7168	11
Signals	0.000	974	---	---
IOs	0.026	196	221	89
Leakage	0.060			
Total	0.086			

Figure 16: Power result forSpartan3 power optimized design

Device	
Family	Spartan3
Part	xc3s400
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-4

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	848	7168	12
Signals	0.000	1247	---	---
IOs	0.037	196	221	89
Leakage	0.060			
Total	0.097			

Figure 17: Power result forSpartan3 speed optimized design

Device	
Family	Virtex6
Part	xc6vlx75t
Package	ff484
Temp Grade	Commercial
Process	Typical
Speed Grade	-1

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.012	1	---	---
Logic	0.001	208	46560	0
Signals	0.002	338	---	---
IOs	0.040	196	240	82
Leakage	1.294			
Total	1.349			

Figure 18: Power result for Virtex6 power optimized design

Device	
Family	Virtex6
Part	xc6vlx75t
Package	ff484
Temp Grade	Commercial
Process	Typical
Speed Grade	-1

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.013	1	---	---
Logic	0.001	208	46560	0
Signals	0.002	338	---	---
IOs	0.044	196	240	82
Leakage	1.294			
Total	1.355			

Figure 19: Power result for Virtex6 speed optimized design

e) Tabular description and comparison of results

Parameter	Spartan 3		Virtex 6	
	Power	Speed	Power	Speed
Number of slices used / Total Available	420/3584	625/3584	187/93120	201/93120
Number of LUTs used / Total Available	714/7168	851/7168	241/46560	251/46560
Utilization	9 %	11 %	0.51 %	0.54 %
Max Frequency	67.696MHz	78.920 MHz	203.046 MHz	224.568 MHz
Timing Simulation Result	147.2311 µs	126.285 µs	35.1156 µs	31.6081 µs
Power Consumption	0.086 Watts	0.097 Watts	1.349 Watts	1.355 Watts

Table IV: Results Summary

The synthesis reports show that when optimized for speed the number of slices and number of LUTs is increased. Virtex 6 FPGA has more resources compared to Spartan 3 and hence Virtex 6 should be used for larger designs. The PAR report shows maximum usable frequency [6]. Maximum Frequency has been achieved for Virtex 6 FPGA when optimized for speed.

Timing Simulation Results are obtained using Isim Simulator [6]. It shows the time required to obtain the final output. Virtex 6 is faster due to high clock frequency. Hence Virtex 6 FPGAs should be used to obtain faster results. The power reports are obtained using Xilinx Power Analyzer [6]. The reports indicate that Virtex 6 FPGAs consume more power. The clock frequency also consumes significant power. Increasing clock frequency increases power consumption. Hence for power conscious designs Spartan 3 FPGAs can be used.

V. CONCLUSION

The RSA algorithm can be effectively implemented on FPGA .The arithmetical operations necessary for the RSA algorithm are time consuming as the number of bits used is usually large. However as the length of public key and message increases the utilization of FPGA resources also increases.

The design on FPGA can be optimized for power or speed but not both by changing synthesis & implementation settings. Optimizing design for power uses fewer resources but offers less speed. Optimizing design for speed uses more resources and offers more speed. Thus for large designs power optimized designs should be used.

For better security, FPGAs with more resources should be used. Also a dedicated ASIC prototype can be developed which can be used to encrypt data in Communication equipments, Set Top Boxes, Personal computers.

REFERENCES

[1]. Sushanta Kumar Sahu and Manoranjan Pradhan "FPGA Implementation of RSA Encryption system", International Journal of Computer Applications, Volume 19- No.9,2011

[2]. Perovic N.S; Popovic-Bozovic : "FPGA implementation of RSA crypto algorithm using shift and carry algorithm ",IEEE Telecommunications Forum (TELFOR), 2012 pages (1040-1043)

[3]. Nibouche O. Nibouche M.Bouridane A. Belatreche A. : "Fast architectures for FPGA Architectures of RSA Encryption Algorithm",IEEE Conference Field programmable technology,2004 (pages 271-278)

[4]. C-Chao Yang ; Tian-Sheuan Chang ; Chein-Wei Jen, "A new RSA **C**ryptosystem hardware design based on Montgomery's algorithm"**,** Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on (Volume:45 No 7,2010)

[5]. Cetin Kaya Koc, High speed RSA algorithm, RSA laboratories, version 2, 1994

[6]. Xilinx user guide 612- timing closure user guide 2012

[7]. Xilinx user guide 625- constraints user guide 2012