```
return (e,d)
extended_euclidian_loop(a,b)
(y,y_prev) := (1,0)
While (b!= 0)
Begin
(y,y_prev) := (y_prev – a/b*y,y)
(a,b) := (b,a mod b)
end
return (a,y_prev) (gcd(p-1)(q-1),e) is a , inverse is y_prev
```

4.3 Encryption/Decryption

Encryption is the process of converting plain text in such a way that eavesdroppers or hackers cannot read it, it is called as cipher text. Decryption is the inverse process by which cipher text is converted back into the form that is readable namely plain text. After generation of the keys, RSA encryption and decryption is done using the mathematical operation $C = M^e \pmod n$ and $M = C^d \pmod n$ respectively. Hence encryption/decryption is just a modular exponentiation operation. It involves few modular operations like modular addition, modular subtraction and modular multiplication.

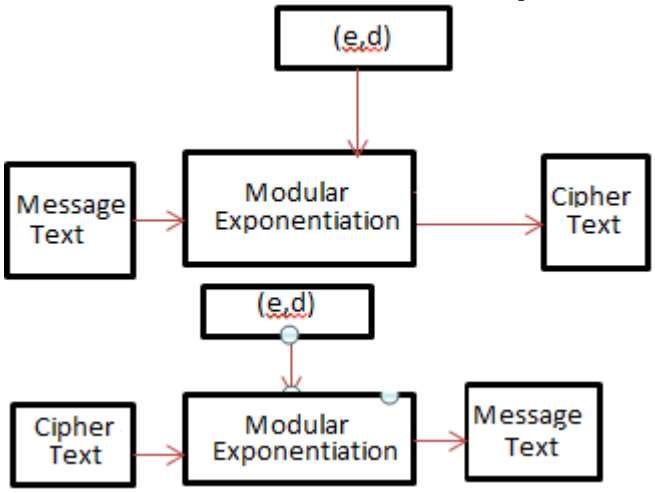


Figure 3.4: Encryption and Decryption

5. Hardware Implementation

For hardware design and implementation, the RSA Cryptosystem is divided into 4 modules.
i. Initial module
ii. Modular exponentiation
iii. Core algorithm
iv. Top module

5.1 Initial Module

This module consists of a Prime number generation module for generating prime numbers for the algorithm, Then these numbers are used for public key and private key generation. The exponentiation part of this algorithm is done by using the right to left binary algorithm implemented for encryption/decryption.

5.2 Modular Exponentiation

The most important and time consuming part of RSA algorithm is calculating the modular exponentiation of a number. For this purpose we implement the Square and Multiply algorithm by using the right-to-left-binary approach. It speeds up the exponent calculation and limits the number of cycles needed. The exponent function is also required in miller and Rabin tester so this module can be called there for processing and calculating, saving both space and time.

5.3 Core Algorithm

Here we implement the basic functions and steps of RSA algorithm. This is further divided into two steps:
1. Key generation
2. Cryptography

When a new used comes to the system this module takes two numbers as input. n and Φ(n) are calculated by inserting them into the multiplier, Φ(n) is then used to find the encryption and decryption keys. For this purpose the Euclidean Algorithm is used which calculates the GCD of Φ(n) and an odd number. If the GCD is found to be 1 this proves that the modular inverse of the odd number exists and the number is co-prime. This number is then saved in a register and will be used as the encryption key. The Extended Euclidean Algorithm is then used to find the decryption key which is the modular inverse of the encryption key. Hence the key generation process is completed for this user and he is allotted with a public and private key that user will use to encrypt and decrypt the data. Once the keys are created they can be used for encrypting and decrypting the message. The Modular Exponentiation is used for this purpose.

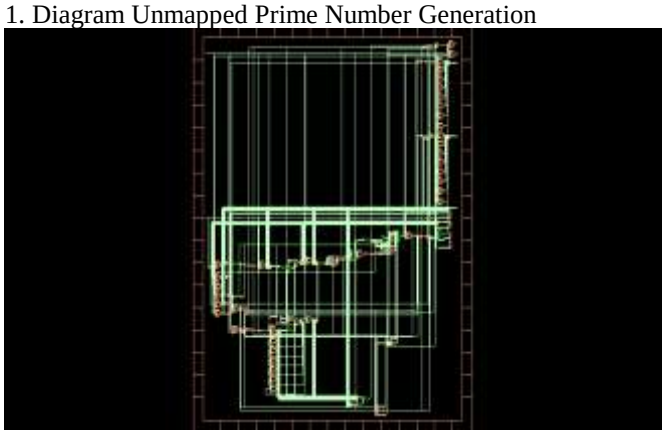
5.4 Top Module

The top module controls the functions of the other modules and interconnects them so as the RSA Algorithms flow is maintained. This module implements a controller with multiple checks so as to get the desired results.

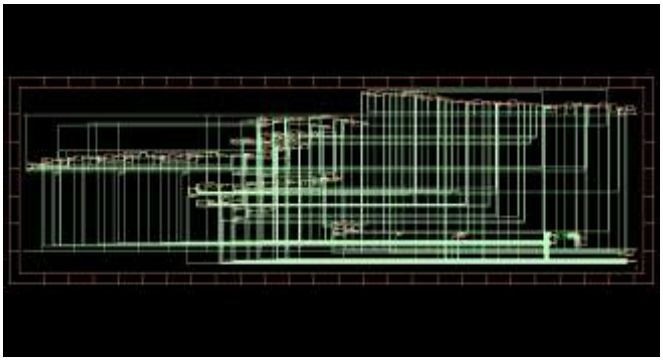
Synthesis of RSA Cryptosystem

6.1 Synthesis is carried out in Cadence RC compiler

1. Diagram Unmapped Prime Number Generation



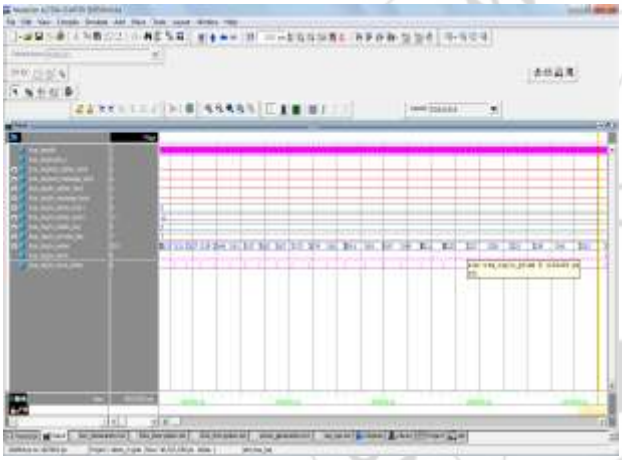
2. Diagram for Mapped Synthesis Prime Number Generation.



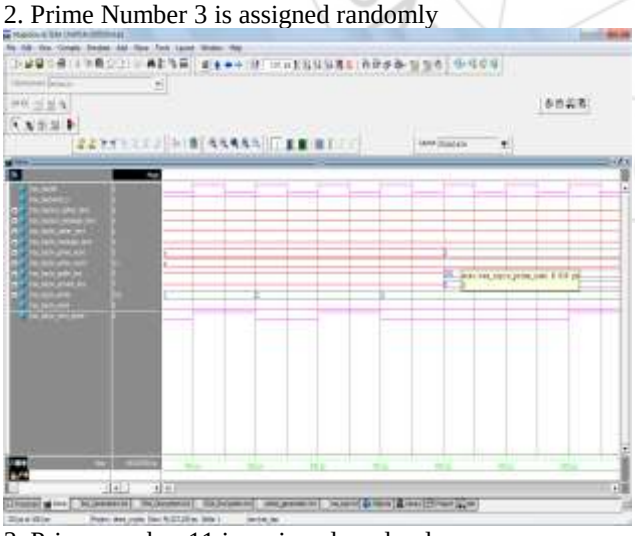
6. Simulation Results

Prime number generation module Extended Euclidean and modular exponentiation have been successfully written and tested on Xilinx ISE 13.2 and Modelsim. The simulation shows the desired results of these algorithms. The following section shows the simulation results of these algorithms.

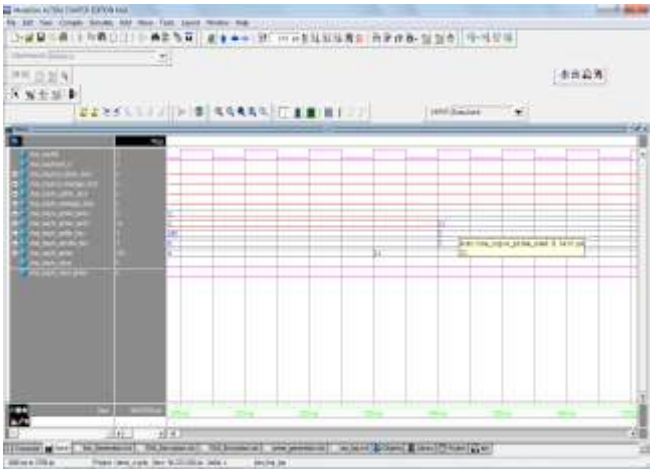
1. Prime Number generation till 255 in figure prime numbers are shown form 127 to 251.



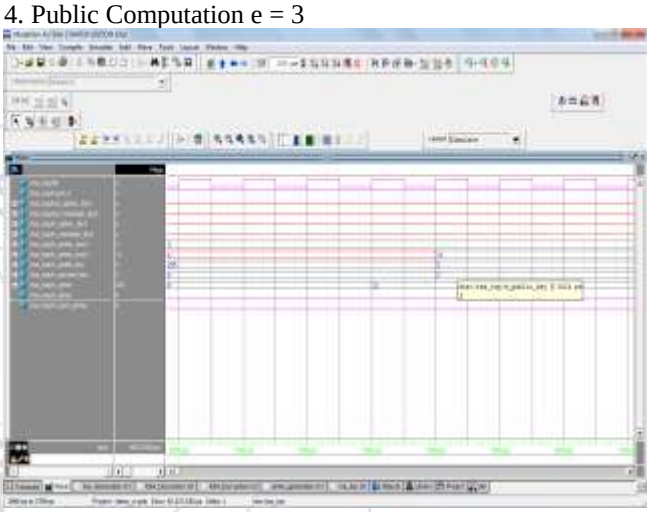
2. Prime Number 3 is assigned randomly



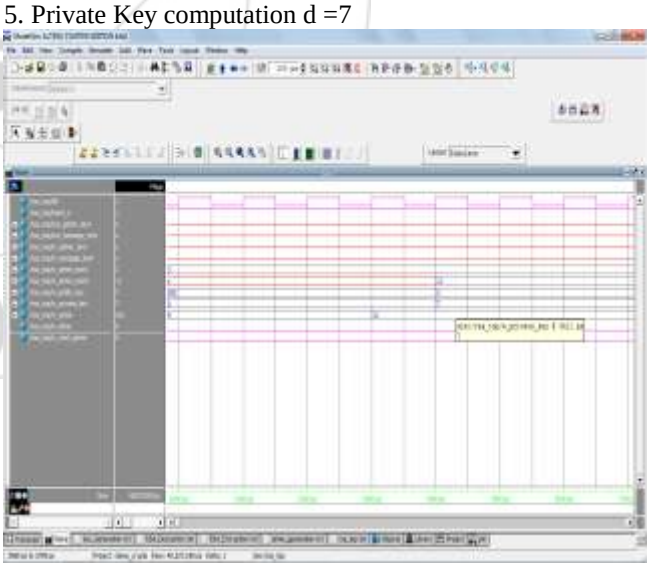
3. Prime number 11 is assigned randomly



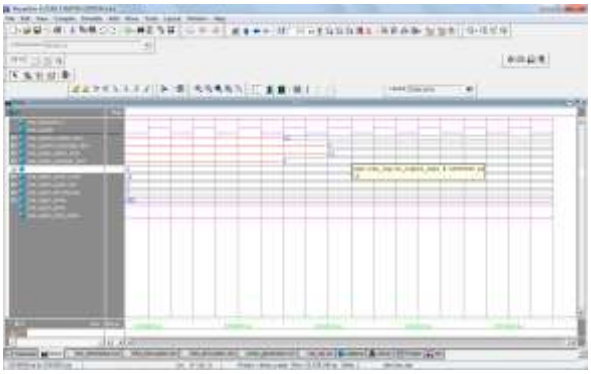
4. Public Computation e = 3



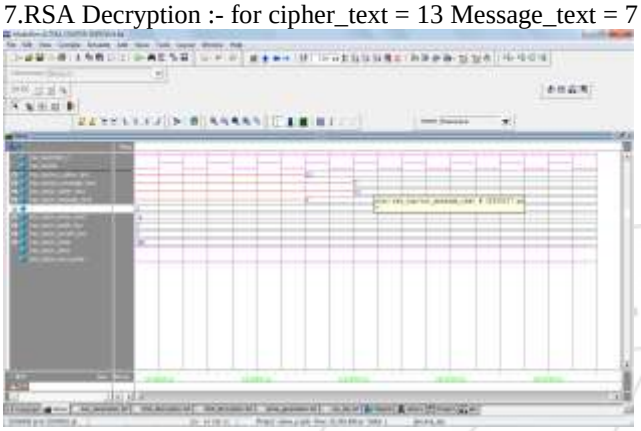
5. Private Key computation d =7



6. RSA Encryption: - Message_text = 7 cipher_text = 13



7.RSA Decryption :- for cipher_text = 13 Message_text = 7



7. Conclusion

Here we implemented a 64-bit RSA circuit in Verilog. It is a full-featured and efficient RSA circuit this includes Prime generation, key generation, data encryption and data decryption. We have implemented random prime number generator using Prime number generation module, GCD and modular inverse algorithm using extended Euclidean algorithm and Encryption and Decryption using Modular multiplication and modular exponentiation algorithms (R-L binary algorithms). Each sub-component and top module of RSA was simulated in Xilinx, Modelsim and proved functionally correct. This can easily scale up to large bits such as 512 or 1024 or even longer.

8. Acknowledgment

This research is supported by the BMS College of Engineering, Bangalore. The authors wish to thank BMS college of Engineering for supporting this work by encouraging and supplying the necessary tools.

References

[1] Moore, Gordon E. (1965): Cramming more components onto integrated circuits Electronics, Volume 38, Number 8
[2] Benedikt Gierlichs: Hardware-Software Co-Design: A Case Study on an accelerated Implementation of RSA
[3] Menezes, van Oorshot, Vanstone (1997) Handbook of applied cryptography, CRC Press
[4] M.K. Hani, T.S. Lin, N. Shaikh-Husin: FPGA Implementation of RSA Public-Key Cryptographic Coprocessor Proceedings of TENCON, vol. 3, pp. 6-11, Kuala Lumpur, Malaysia, 2000
[5] R.L. Rivest, A. Shamir, and L. Adleman (1978): A Method for Obtaining Digital Signatures and Public-Key Cryptosystems Communications of the ACM 21,2, pp. 120-126
[6] Ankit Anand, Pushkar Praveen: Implementation of RSA Algorithm on FPGA Centre for Development of Advanced Computing, (CDAC) Noida, India

Author Profile

Rafeek Alas received B.E degree from S D M College of Engineering and Technology, Dharwad, Karnataka, India in 2011. he is currently pursuing her MTech degree in the field of Electronics, department of Electronics and Communication in BMS College of Engineering, Bangalore, India. Her fields of interest in research are VLSI design and VLSI circuits..

Dr. Kiran Bailey received her B.E degree from Dayananda College of Engineering of Bangalore University, Bangalore in the year 1997. She got her M.Tech degree from B.M.S. College of Engineering of Visvesvaraya Technological University, Bangalore in 2001. She joined BMSCE in 1998 and has since been teaching Electronics related subjects. Her areas of interest are solid state devices, VLSI design, Low power VLSI circuits. Presently she is an Associate professor in the dept. of E&C, BMSCE, Bangalore.