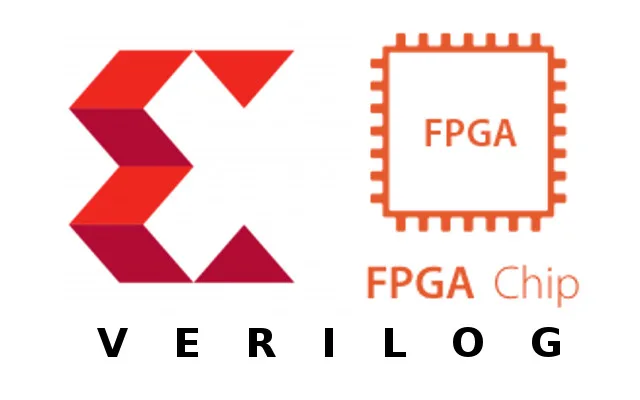
RSA Encryption/Decryption Implementation in an FPGA Using Verilog HDL



**Department of Electrical and Computer Engineering**

**King Abdulaziz University, Jeddah, KSA**

**Problem definition:**

The goal here is to implement the RSA Algorithm on an FPGA, where ultimately a Plain Text is Encrypted, and a Cipher Text is Decrypted, provided that the “Public key” or “Private key” and “n” are given.

**Cryptography:**

Diagram

Description automatically generatedCryptography, or cryptology (from Ancient Greek: κρυπτός, romanized: kryptós "hidden, secret"; and γράφειν graphein, "to write", or -λογία -logia, "study", respectively), is the practice and study of techniques for secure communication in the presence of adversarial behavior. More generally, cryptography is about constructing and analyzing protocols that prevent third parties or the public from reading private messages; various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation. are central to modern cryptography. Modern cryptography exists at the intersection of the disciplines of mathematics, computer science, electrical engineering, communication science, and physics.

The first known evidence of cryptography can betraced to the use of 'hieroglyph'. Some 4000 years ago, the Egyptians used to communicate by messages written in hieroglyph. This code was the secret known only to the scribes who used to transmit messages on behalf of the kings.

Figure 1: General encryption model

**Cryptography applications:**

Applications of cryptography include electronic commerce, chip-based payment cards, digital currencies, computer passwords, and military communications.

**RSA:**

RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem that is widely used for secure data transmission. It is also one of the oldest. The acronym RSA comes from the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described the algorithm in 1977.

RSA relies on asymmetric key cryptography. Asymmetric Encryption uses two distinct, yet related keys. One key, the Public Key, is used for encryption and the other, the Private Key, is for decryption. As implied in the name, the Private Key is intended to be private so that only the authenticated recipient can decrypt the message.

In a public-key cryptosystem, the encryption key is public and distinct from the decryption key, which is kept secret (private). An RSA user creates and publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers are kept secret. Messages can be encrypted by anyone, via the public key, but can only be decoded by someone who knows the prime numbers.

The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers, the "factoring problem". Breaking RSA encryption is known as the RSA problem. Whether it is as difficult as the factoring problem is an open question. There are no published methods to defeat the system if a large enough key is used.

RSA is a relatively slow algorithm. Because of this, it is not commonly used to directly encrypt user data. More often, RSA is used to transmit shared keys for symmetric key cryptography, which are then used for bulk encryption-decryption.

**General steps of RSA:**

1- Key generation, 2- Key distribution, 3- Encryption, 4- Decryption

**Graphical user interface, application

Description automatically generated**

Figure 2: RSA process model

**RSA Algorithm steps: [[[1]](#endnote-1)]**

**1. keys generation**

* 1. Select two large prime numbers, X and Y.

Note: The prime numbers need to be large so that they will be difficult for someone to figure out.

* 1. Calculate n such that n = XY.
  2. Calculate the totient function; ϕ(n) = (x−1)(y−1).
  3. Select an integer e, such that e is co-prime to ϕ(n) and 1 < e < ϕ(n).

The pair of numbers (n, e) make up the public key.

Note: Two integers are co-prime if the only positive integer that divides them is 1.

* 1. Calculate d such that d = 1 mod ϕ(n)/e.
  2. d can be found using the extended Euclidean algorithm. The pair (n, d) make up the private key.

**2. Encryption**

Given a plaintext P, represented as a number, the ciphertext C is calculated as:

**3. Decryption**

Using the private key (n, d), the plaintext can be found using:

**Important Note:**

Since, RSA is considered an Asymmetric key procedure, its security relies partially on the fact that it's easy to choose two random prime numbers, but it's very hard to discover what they are when just given the product of them.

Therefore, typical key sizes are 1,024 or 2,048 or 4,096 bits, unlike Symmetric key which are much smaller. The reason RSA is that large that's because there are only so many prime numbers of that size and below. The RSA scheme can only use pairs of prime numbers, whereas the symmetric schemes can use any number at all the same size [[[2]](#endnote-2)]. But for sake of simplicity and practicality we used a small key length in the code.

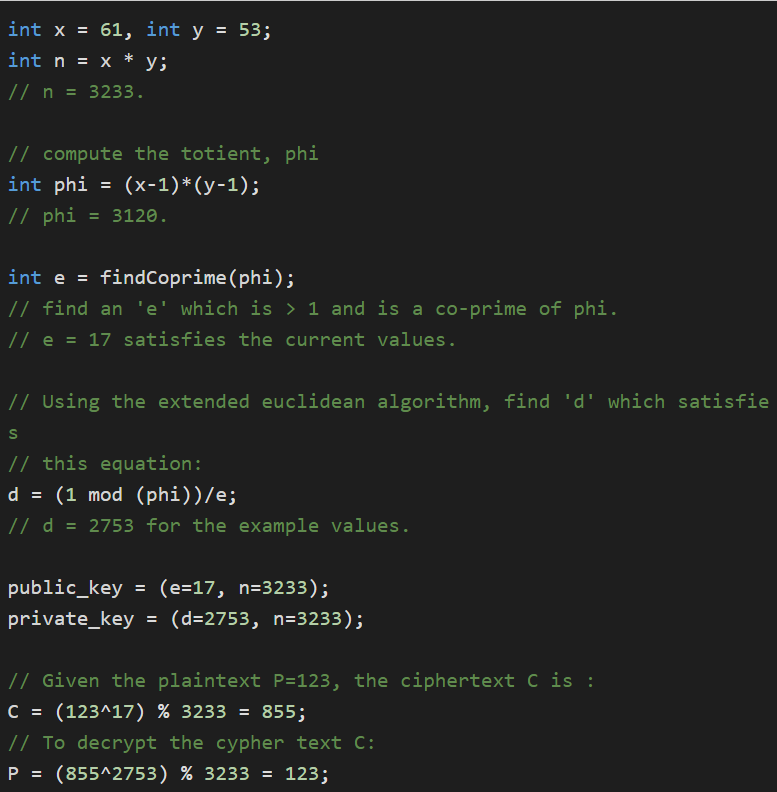
**RSA Pseudocode:**

Figure 3: RSA Pseudocode

**Digital Design:**

To design the Encryption and Decryption Algorithm will utilize Verilog HDL

**Black Box Design:**

Diagram

Description automatically generated

Figure 4: General Black Box Design

The basic I/O of our design will utilize the two control signals “go & done” and a reset signal to abort operations, and of course a clock.

The main I/O here are the mode, key, Input, and Output.

**Mode:** will control which state the digital circuit should be in, either Encryption state or Decryption state.

**Key:** is basically the key pair, it depends on whether Decryption/Encryption mode is chosen.

**Input:** is going to be either Plain Text or Cipher Text, it depends on whether Encryption / Decryption mode is chosen.

**Output:** is going to be either Cipher Text or Plain Text, it depends on whether Encryption / Decryption mode is chosen.

Graphical user interface, application

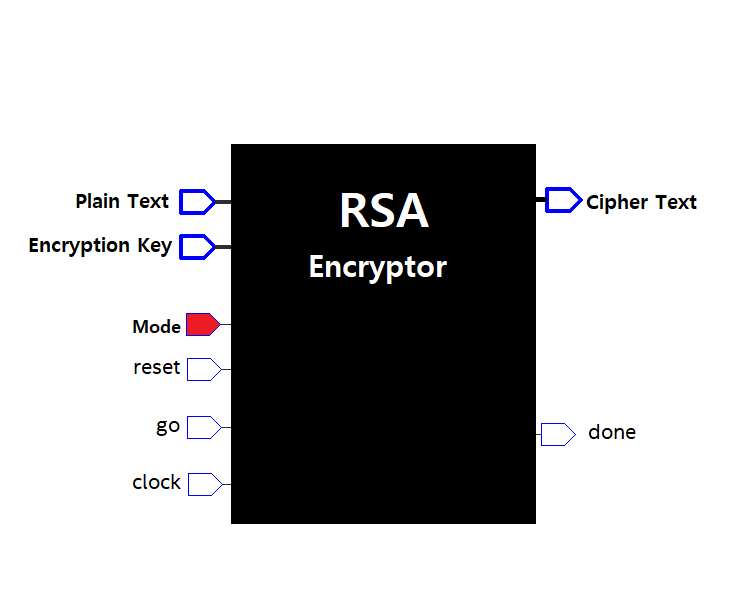
Description automatically generated

Figure 5: Black Box Design in Encryption State

Figure 6: Black Box Design in Decryption State

**Top Level Design:**

The Control Unit “CU” receives the “go” signal, and from it generates the control signals required in the Data Path “DP” to output the desired result.

The Data Path “DP” is where the processing takes place.

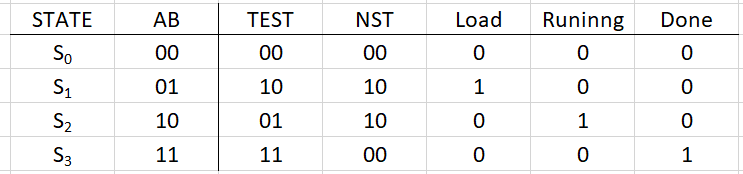
**Control Unit SM chart:**

Diagram

Description automatically generated

As seen our control unit has three different states: S0”Idle”, S1”Running”, S2”Done”

**Control Unit One Address ROM table:**



1. []: <https://www.educative.io/answers/what-is-the-rsa-algorithm> [↑](#endnote-ref-1)
2. []: <https://blog.cloudflare.com/why-are-some-keys-small/>

   [↑](#endnote-ref-2)