Encryption using RC4 Stream Cipher on FPGA Board

Goal:

To implement RC4 stream cypher encryption algorithm to encrypt text using a key entered by the user.

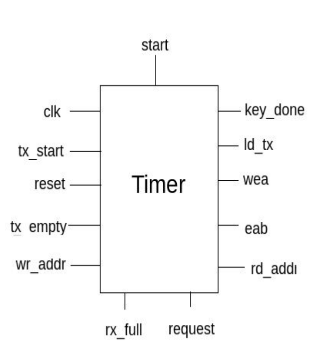
Components Used:

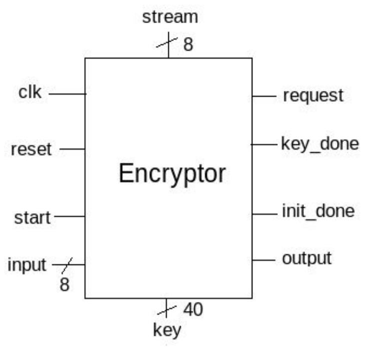
- Basys-3 FPGA board
- Keypad Peripheral Module

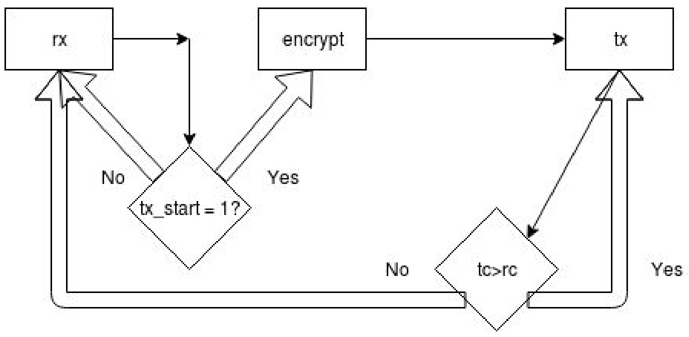
Design:

We require an input of 9 hex keys to form a 40 bit key. This is done using PmodKYPD. All the text sent by the user is stored in the memory. On transmission, we encrypt each characters one-by-one and encrypt it before transmission. Following this, we cannot reuse the keystream due to its changed state.

Flow of Logic:







RC4 Algorithm:

RC4 is a stream cipher algorithm and hence uses the XOR operation to encode each bit of the text to be encrypted. This algorithm can be divided into two steps:

1. Generation of the value with each character is encrypted
2. Encryption of the character

We perform the following process:

1.
    a. Consider the table **S** of all the 8 bit(1 byte) binary sequences in increasing order. Hence we can conclude that **S[i] = i** for ≤ *i* ≤ 255 0
    b. Consider **K** a 10 digit long hexadecimal key which in turn can be converted to a 40-bit binary key
    c. In **S** we perform the following process:

```
i.   j := 0
ii.  for i from 0 to 255
         1. j := (j + S[i] + K[i mod 40]) mod 256
         2. swap(S[i], S[j])
```

2. Now, we have a randomised form of table which is totally dependent on the initial key **K.** Next, for encrypting each 8-bit character of the input, we generate another 8-bit number with which it is xorred. For this, we perform the following process:

```
a. i := 0, j := 0
b. while moreKeysRequired:
       i.   i := (i + 1) mod 256
       ii.  j := (j + S[i]) mod 256
       iii. swap(S[i], S[j])
       iv.  output(S[(S[i] + S[j]) mod 256]
```

3. Let the output of this process be an 8-bit vector called **V**. Let the plaintext/ciphertext character be **C.** Then, the character **C**⊕**V** is its ciphertext/plaintext dual.

Testing:

Various keys and matched the result with a C++ code written to perform the same operation and the generated keystream and 8 bit arrays came out to be in agreement.

We also simulated our implementation for various text files and different keys. In the first part we sent a text file for a particular key and save the encrypted data to another file. Then we send this file for encryption and recorded the final output. As we are performing symmetric key cryptography, the initial file and the final output should be the same as the 8 bit array generated for each character only depends on the shared key and position of character in the whole text.

Observations:

For the test text as :

Anitej and Daman are doing COL215 together. They’re doing well.

We observed the following results for various runs:

- Anitej and ¢°¶07 are doing COL215 together. They're î̂ªº well.
- Anitej and Daman are do4-däÈ&15 together. They're doing well
- Anitej and Daman are doing COL2Ódœépether. They¹2²·ing well
- Anitej072Daman are doing COL215 together. They're doing well
- Anitej and Daman are doing COL215 °·³2:her. Theyð×=+îoing well
- ·4tej and Daman are doing COL215 top}p;r. They're doing well

This leads to the conclusion that there are possible transmission errors. This could be due to a mismatch in timings or random errors due to faulty equipment. It is confirmed that there is no error in reception since an error there would be propagated in all the following characters as well.