BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
Advanced Computer Organization (CS C)
LAB TEST (BATCH 1)

Date: 22/04/2012 **Duration: 1 Hr 30 Mins**
Marks: 20M **Time : 3:00 pm to 4:30 pm**

ID NO:	NAME:

INSTRUCTIONS: (1) Create a folder, Rename it to your IDNO.

(2) Rename the project file with your IDNO.

(3) Write IDNO and Name in .v file. Write all modules in single .v file.

Messages are encrypted before passing them on the network for the purpose of security. At the receiving end they are decrypted to obtain the original message. For this purpose there are different encryption and decryption algorithms. Many of them can be implemented in hardware to accelerate the process.

Design a system which takes a 256-bit binary input message, encrypts it with the help of the encryption algorithm explained below and produces the cipher text (i.e. encrypted message). Then this cipher text is given as input to the decryption algorithm which produces the original message. Each algorithm makes use of two binary keys.

Implement these algorithms in verilog HDL. Module prototypes are given below. Add necessary parameters to the modules if required.

Following are the details of the encryption/decryption algorithm:

Message Size: User can input a message of size ranging from 1 bit to 248 bits. Rest of the 8 bits are used to store the length of the input message.

Steps-by-step description of the algorithm is as follows:

Padding(Refer Figure 1): If user message is less than 248 bits then extra zeros are appended to make it 248 bit. Then the length of message is stored in other 8 bits and appended to the 248 bit message to get 256 bit message. This is given as an input to the *Encryption* algorithm.

Key Generation Algorithm (Refer Figure 2):

Encryption and *Decryption* algorithms require two 64-bit keys. Key Generation algorithm takes as input a 64-bit key and produces two 64-bit keys. Steps are as follows:

1. Divide 64 bit key into two halves i.e. left and right
2. Using 64 bit ALU, add left and right to generate 64 bit key1
3. Using 64 bit ALU, subtract left and right to generate 64 bit key2

Encryption Algorithm (Refer Figure 3): It takes 256-bit message and 64-bit key as input and produces 256-bit Encrypted message (a.k.a Cipher Text). Steps are as follows:

i. **Intra-block rotation** : Divide 256-bit message into four 64-bit blocks. For each block, perform 1 bit right circular shift.

ii. **Inter-block rotation** :
   a. For block no. 0 to 2 :
      i. Place **i^{th}** block at **$((i+1) \bmod N)^{th}$** block position where N is the number of blocks.
   b. For block no. 3,
      i. Perform XOR with key1,
      ii. Then place at **$((i+1) \bmod N)^{th}$** block position

iii. XOR each block with key2

Decryption Algorithm: (Reverse of Encryption Algorithm**)**

Input: 256 bit cipher text and a 64 bit key (same key which was passed to encryption module)
Output: 256 bit original message. Steps are as follows:

i. XOR each block with key2

ii. **Inter-block rotation** :
   a. For block no. 0:
      i. Perform XOR with key1,
      ii. Then place at **$(i-1 \bmod N)^{th}$** block position
   b. For block no 1 to 3 :
      i. Place **i^{th}** block at **$((i-1) \bmod N)^{th}$** block position

iii. **Intra-block rotation** : For each block, perform 1 bit left circular shift

Module Description:

1. **module AlgoTest**: Stimulus Module for Algorithm
2. **module Encrypt(cipherText, plainText , key):**Design Module for Encryption
3. **module Padd(paddedText,plainText):** Design Module for Padding
4. **module Keygen(key1,key2,key):** Design Module for Key generation
5. **module ALU64bit(key,left,right,select):** Design Module for ALU for add/sub

module Decrypt(decipherText,cipherText,key): Design Module for Decryption

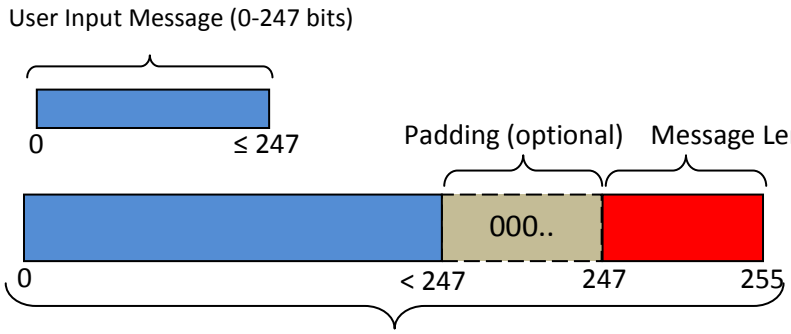

Figure 1. Padding

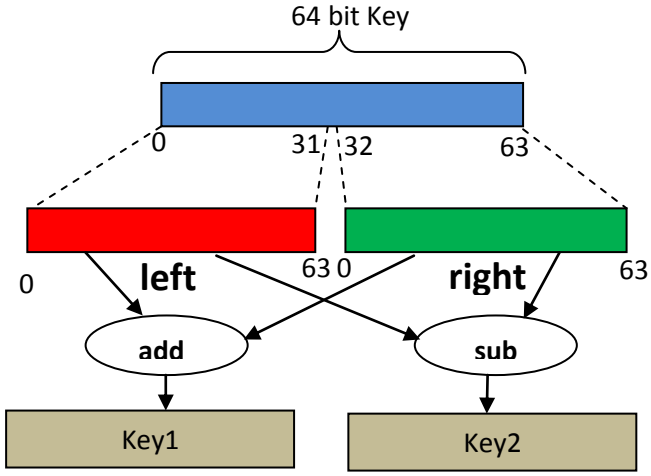

Figure 2. Key generation

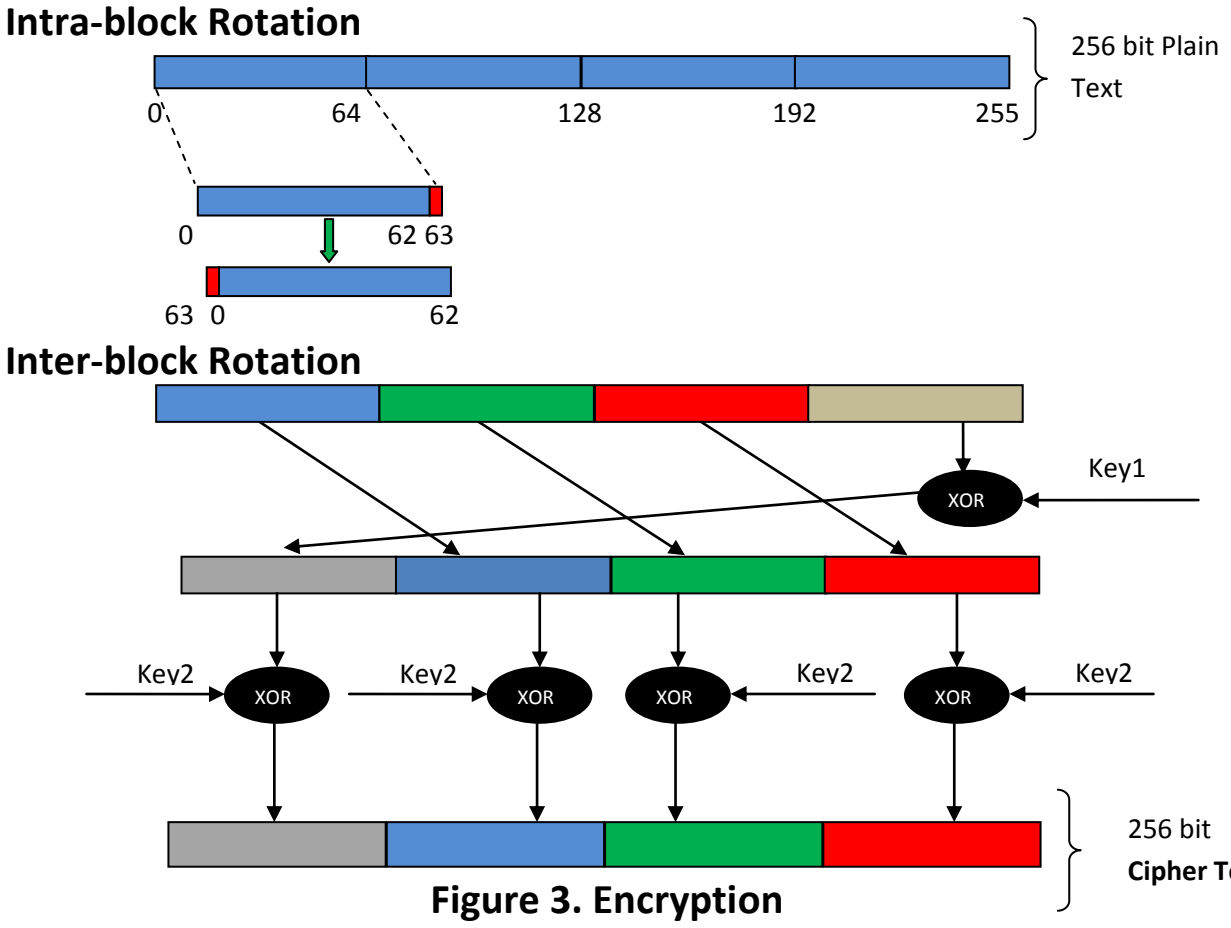

Figure 3. Encryption