ΥΛΟΠΟΙΗΣΗ CRC

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ 2023

createMessage(k)

- Η συνάρτηση createMessage(k) δέχεται έναν αριθμό k και δημιουργεί έναν τυχαίο δυαδικό μήνυμα μεγέθους k. Για να δημιουργηθούν τα τυχαία bits χρησιμοποιείται η συνάρτηση random.ranrange(0,2). Το μήνυμα αποθηκεύεται σε μια λίστα.

```
# Function that creates a k-bit random binary message.
def createMessage(k):
    message = []  # Initialize list to store the message
    for i in range(0, k):  # Fill the message with k bits
        num = random.randrange(0, 2)  # Generate a random binary number
        message.append(num)  # Append the generated number in the message
    return message  # Return the k bit random message
```

xor(A,B)

- Η συνάρτηση xor(A, B) δέχεται δύο λίστες με το ίδιο μέγεθος και επιστρέφει τη λειτουργία XOR μεταξύ τους. Η συνάρτηση υπολογίζει και επιστρέφει το λογικό xor για κάθε θέση bit των δύο αριθμών αγνοώντας το πρώτο σημαντικό bit.

```
# Function that calculates the XOR of list A and B (lists are the same length).
def xor(A, B):
    result = []  # Initialize a list that will store the XOR of A and B
    # Calculate the XOR for each position of A and B lists. Ignore the most significant bit.
    for i in range(1, len(B)):
        if A[i] == B[i]:  # Definition of XOR
            result.append(0)
        else:
            result.append(1)
    # Return the result
    return result
```

division(A,B)

- Αυτη η συναρτηση υλοποιεί την διαίρεση modulo-2.

1. Ελέγχει το πρώτο (αριστερότερο) bit του ενδιάμεσου αποτελέσματος.
2. Αν αυτό το bit είναι 1, τότε κάνει XOR με τον διαιρέτη B.
3. Αν αυτό το bit είναι 0, τότε κάνει XOR με έναν αριθμό που αποτελείται μόνο από μηδενικά.
4. Προσθέτει το επόμενο bit του διαιρετέου A στο ενδιάμεσο αποτέλεσμα.
5. Επαναλαμβάνει τη διαδικασία για το επόμενο bit.

- Μετά το τέλος των bits του διαιρετέου A, κάνει μία τελευταία επανάληψη για να εξασφαλίσει ότι το αποτέλεσμα είναι σωστό. Στο τέλος, επιστρέφει το ενδιάμεσο αποτέλεσμα, το οποίο είναι ο υπόλοιπος της διαίρεσης του A με το B σε modulo-2.

```python
# Function that calculates the modulo-2 division of two numbers A and B. A is the dividend and B is the divisor.
def division(A, B):
    counter = len(B)  # initialize counter that stores the size of bits that have to be XORed each time.
    result = A[0:counter].copy()  # Initialize the result-dummy list that will store the individual remainder.
    # Loop that puts 1 bit down each time and XORs with the divisor.
    while counter < len(A):
        # If leftmost bit is 1 then XOR with the divisor.
        if result[0] == 1:
            result = xor(result, B)
        elif result[0] == 0:  # If leftmost bit is 0 then drop down zeros.
            dummy = [0 for i in range(0, len(B))]  # Create a divisor representing 0 with counter-bits.
            result = xor(result, dummy)  # XOR with 0.

        # Drop down the next bit.
        result.append(A[counter])

        # Increase by one the size counter.
        counter = counter + 1
    # Make one last loop for the nth bit.
    if result[0] == 1:
        result = xor(result, B)
    else:
        dummy = [0 for i in range(0, len(B))]
        result = xor(result, dummy)
    # Return the final result.
    return result
```

constructFCS(D, P, n)

- Αυτή η συνάρτηση υλοποιεί τη δημιουργία ενός Frame Check Sequence (FCS) για ένα δεδομένο μήνυμα D, με κλειδί P και συνολικά n bits. Το FCS χρησιμοποιείται για τον έλεγχο σφαλμάτων σε μεταδεδομένα.
- Πρακτικά, η συνάρτηση υπολογίζει ένα check sequence, ώστε όταν προστεθεί στο αρχικό δυαδικό μήνυμα, το συνολικό μήνυμα (μήνυμα + FCS) να είναι διαιρετό με το κλειδί P.

```python
# Function that constructs the FCS for a message D, key P and n total bits.
def constructFCS(D, P, n):
    D2nk = []  # Initialize a list to store the 2^(n-k)D number.
    D2nk = D.copy()  # Fill the most significant bits with the number D.
    # Fill the rest number with zeros.
    k = len(D)
    for i in range(0, n - k):
        D2nk.append(0)
    # Calculate the FCS as the remainder of the mod-2 division of 2^n-k/P.
    FCS = division(D2nk, P)
    # Return the FCS
    return FCS
```

createCRC(D, P, n)

- Η συνάρτηση createCRC δημιουργεί έναν κωδικό CRC (Cyclic Redundancy Check) για ένα μήνυμα, χρησιμοποιώντας ένα δεδομένο κλειδί P. Αυτό επιτρέπει την ανίχνευση σφαλμάτων στα μεταδιδόμενα δεδομένα. Πρώτα υπολογίζει την ακολουθία ελέγχου (FCS) για το μήνυμα, και στη συνέχεια προσθέτει αυτήν την ακολουθία στο τέλος του αρχικού μηνύματος. Το τελικό μήνυμα που επιστρέφει περιλαμβάνει τα αρχικά δεδομένα και την ακολουθία ελέγχου.

```python
# Function that creates Cyclic Redundancy Check for a message total of n bits, with data message D and key P.
def createCRC(D, P, n):
    FCS = constructFCS(D, P, n)  # Calculate the FCS.
    T = D.copy()  # initialize the total message T with data message D.
    # Fill the rest of n bits with the FCS.
    for i in range(0, len(FCS)):
        T.append(FCS[i])
    # Return the final message(Data + FCS).
    return T
```

createError(message, BER)

- Η συνάρτηση createError δημιουργεί σφάλματα σε ένα μήνυμα, με βάση μια δεδομένη τιμή Bit Error Rate (BER). Η BER αναπαριστά την πιθανότητα ένα bit να αλλοιωθεί κατά τη μετάδοση.
- Η συνάρτηση περνάει μέσα από κάθε bit του μηνύματος και, με τη χρήση τυχαίων αριθμών, αποφασίζει αν θα αλλάξει την τιμή του bit ή όχι, βάσει της τιμής BER. Αν ο τυχαίος αριθμός είναι μικρότερος από το BER, τότε το bit αλλοιώνεται.

```python
# Function that creates Error in a message based on a BER value.
def createError(message, BER):
    transmitted = message.copy()  # initialize a list to store the transmitted message.
    for i in range(0, len(transmitted)):  # For every bit in the message to be transmitted
        num = random.random()  # Generate a random float number in range 0 to 1.
        if num < BER:  # If the number is smaller than the BER
            # then create an error in the specific bit.
            if transmitted[i] == 0:
                transmitted[i] = 1
            else:
                transmitted[i] = 0
    # Return the transmitted message.
    return transmitted
```

checkMessage(message, P)

- Η συνάρτηση checkMessage ελέγχει αν ένα μήνυμα μεταδόθηκε με σφάλματα χρησιμοποιώντας τη μέθοδο Cyclic Redundancy Check (CRC).
- Πρώτα, υπολογίζει το υπόλοιπο της διαίρεσης modulo-2 του μηνύματος με το κλειδί P. Αν το υπόλοιπο δεν είναι μηδενικό, τότε υπάρχει σφάλμα και η συνάρτηση επιστρέφει False. Αν το υπόλοιπο είναι μηδενικό, δεν υπάρχει σφάλμα και η συνάρτηση επιστρέφει True.

```python
# Function to check if a message has been transmitted with an error using the CRC method.
def checkMessage(message, P):
    # Calculate the remainder of mod-2 division message/p.
    rem = division(message, P)
    # If remainder is not zero then there is an error. Function returns false.
    for i in range(0, len(rem)):
        if rem[i] != 0:
            return False
    # If remainder is zero then there is no error. Function returns true.
    return True
```

Εκτέλεση του προγράμματος

- β) Για k=20, P=110101 και BER=10^{-3}, να υπολογίσετε:
  - Το ποσοστό των μηνυμάτων που φθάνουν με σφάλμα (στο block δεδομένων ή στο CRC) στον αποδέκτη.
  - Το ποσοστό των μηνυμάτων που ανιχνεύονται ως εσφαλμένα από το CRC.
  - Το ποσοστό των μηνυμάτων που φθάνουν με σφάλμα στο αποδέκτη και δεν ανιχνεύονται από το CRC.

Αποτελέσματα εκφώνησης

```
PS C:\Users\tsami> python -u "c:\Users\tsami\OneDrive - Αριστοτέλειο Π
ανεπιστήμιο Θεσσαλονίκης\Επιφάνεια εργασίας\CRC-main\main.py"
Give number of bits of message:20
Give divider P in binary with spaces :1 1 0 1 0 1
Give Bit Error Rate (BER):0.001
n-bit message created:  00001101111010001000
CRC created for n-bit message:  00001101111010001000010100
Message transmitted:  00001101111010001000010100
No error occurred!
Give length of data messages:1000
Give divider P in binary with spaces :1 1 0 1 0 1
Give Bit Error Rate (BER):0.001
Give number of transmitted messages:1000
Total of messages transmitted: 1000
Number of errors: 654 with error percentage: 65.4 %
Number of detected errors from CRC: 641 with detected error percentage: 64.1 %
```

Ευχαριστώ για των χρόνο σας

Στοιχεία επικοινωνίας

Όνομα: Kristi

Επώνυμο: Cami

Email: tsamikristi@csd.auth.gr

ΑΕΜ: 3882