**9. WAP to Implement Error Detection: CRC**

**Steps to Implement CRC**

1. **Input Data**:
   * Binary message string (e.g., "11010011101100").
2. **Generator Polynomial**:
   * A fixed binary polynomial, e.g., 1011.
3. **Division**:
   * Perform binary division of the data appended with zeros (equal to the degree of the generator polynomial) using XOR operations.
4. **Append Remainder**:
   * Append the calculated remainder to the data.
5. **Validation**:
   * Receiver divides the received data (original message + checksum) by the generator polynomial. If the remainder is zero, the message is error-free.

**Implementation in C**

c

Copy code

#include <stdio.h>

#include <string.h>

// Function to perform XOR operation

void xorOperation(char \*dividend, const char \*divisor, int length) {

for (int i = 0; i < length; i++) {

dividend[i] = (dividend[i] == divisor[i]) ? '0' : '1';

}

}

// Function to perform CRC calculation

void calculateCRC(const char \*data, const char \*generator, char \*crc) {

int dataLen = strlen(data);

int genLen = strlen(generator);

// Append zeros to the data (equal to generator length - 1)

char dividend[100];

strcpy(dividend, data);

for (int i = 0; i < genLen - 1; i++) {

dividend[dataLen + i] = '0';

}

dividend[dataLen + genLen - 1] = '\0';

// Perform binary division

char temp[100];

strncpy(temp, dividend, genLen);

temp[genLen] = '\0';

for (int i = 0; i < dataLen; i++) {

if (temp[0] == '1') {

xorOperation(temp, generator, genLen);

}

memmove(temp, temp + 1, genLen - 1);

temp[genLen - 1] = dividend[i + genLen];

}

strncpy(crc, temp, genLen - 1);

crc[genLen - 1] = '\0';

}

// Function to validate CRC

int validateCRC(const char \*data, const char \*generator) {

int dataLen = strlen(data);

int genLen = strlen(generator);

// Perform binary division

char temp[100];

strncpy(temp, data, genLen);

temp[genLen] = '\0';

for (int i = 0; i < dataLen - genLen + 1; i++) {

if (temp[0] == '1') {

xorOperation(temp, generator, genLen);

}

memmove(temp, temp + 1, genLen - 1);

temp[genLen - 1] = data[i + genLen];

}

// Check if remainder is zero

for (int i = 0; i < genLen - 1; i++) {

if (temp[i] != '0') {

return 0; // Error detected

}

}

return 1; // No error detected

}

int main() {

char data[100], generator[100], transmittedData[100], crc[100];

// Input data and generator polynomial

printf("Enter the data bits: ");

scanf("%s", data);

printf("Enter the generator polynomial: ");

scanf("%s", generator);

// Calculate CRC

calculateCRC(data, generator, crc);

// Append CRC to the data

strcpy(transmittedData, data);

strcat(transmittedData, crc);

printf("Transmitted Data (Data + CRC): %s\n", transmittedData);

// Validate at receiver

if (validateCRC(transmittedData, generator)) {

printf("CRC Check Passed: No Errors Detected.\n");

} else {

printf("CRC Check Failed: Errors Detected.\n");

}

return 0;

}

**How It Works**

**Sender Side:**

1. **Input**:
   * Data bits: 11010011101100
   * Generator polynomial: 1011
2. **Appending Zeros**:
   * Append three zeros (degree of generator polynomial - 1): 11010011101100000
3. **Division**:
   * Perform binary division using XOR.
4. **CRC Calculation**:
   * Append the remainder to the original data.
   * Transmitted data: 11010011101100110.

**Receiver Side:**

1. Perform the same binary division.
2. If the remainder is 0, no error is detected; otherwise, there is an error.

**Compilation and Execution**

**Compilation:**

bash

Copy code

gcc -o crc crc.c

**Execution:**

bash

Copy code

./crc

**Sample Input and Output**

**Input:**

plaintext

Copy code

Enter the data bits: 11010011101100

Enter the generator polynomial: 1011

**Output:**

plaintext

Copy code

Transmitted Data (Data + CRC): 11010011101100110

CRC Check Passed: No Errors Detected.

**Key Points**

1. **Error Detection**:
   * CRC can detect single-bit errors and burst errors effectively.
   * It cannot correct errors, only detect them.
2. **Generator Polynomial**:
   * The choice of the generator polynomial affects error-detection capability.
   * Common choices: 1011, 1101, etc.