ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E

DIGITAL CIRCUITS LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	8
EXPERIMENT DATE	:	21.05.2021
LAB SESSION	:	FRIDAY - 14.00
GROUP NO	:	G14

GROUP MEMBERS:

150180112 : ÖMER MALİK KALEMBAŞI

150190014 : FEYZA ÖZEN

150190108 : EKİN TAŞYÜREK

SPRING 2021

Contents

1 INTRODUCTION [10 points]

In this experiment, we tried to implement three cryptography applications. First, we designed Caesar Cipher which is one of the first examples of cryptography in history. Secondly we implemented the technique Vigen‘ere Cipher which is used by the Confederate Army. Finally, we tried to implement the Enigma machine which is used by the German Army in the World War 2 and a sample communication environment will be realized.

2 MATERIALS AND METHODS [40 points]

2.1 PART 1

In this part, we designed 4 helper modules.

- **CharDecoder Module**: It transforms ASCII codes of to decoded binary character. It takes 8-bit input which is the ASCII code, gives 26-bit output as decoded char.
- **CharEncoder Module**: It transforms the binary decoded version of the char to ASCII code. It takes 26-bit input, gives 8-bit output.
- **CircularRightShift Module**: It takes 26-bit input data and 5-bit shiftAmount input, gives 26-bit output. The output is data shifted to the right ‘shiftAmount’ times.
- **CircularLeftShift Module**: It takes 26-bit input data and 5-bit shiftAmount input, gives 26-bit output. The output is data shifted to the left ‘shiftAmount’ times.

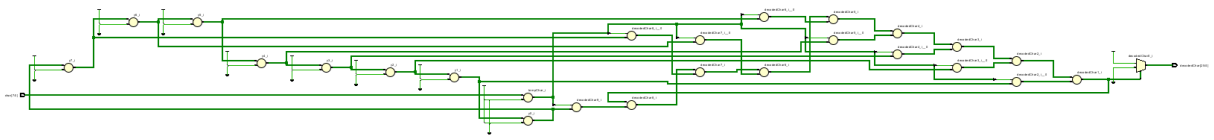

Figure 1: Char Decoder RTL Schematic

2.2 PART 2

In this part, we implemented Caesar Cipher Encryption and Decryption modules. Then we created a Caesar Module to show encrypted and decrypted messages. We created 3 modules here.

- **CaesarEncryption Module**: It works like a shifter and encrypt the char using Caesar Cipher technique.
- **CaesarDecryption Module**: It works like a shifter and decrypt the encrypted char using Caesar Cipher technique.
- **CaesarEnvironment Module**: It is a module that contains CaesarEncryption and CaesarDecryption modules inside.

2.3 PART 3

In this part, we implemented the Vigen‘ere Cipher Encryption and Decryption modules. Then, we created a Vigen‘ere Module to show encrypted and decrypted messages.

$C_i = (P_i + K_i) mod 26$ is the encryption formula of this method. If Key char is shorter than Plain char, we restart i indexes of Key.

$D_i = (C_i K_i) mod 26$ is the decryption formula of this method.

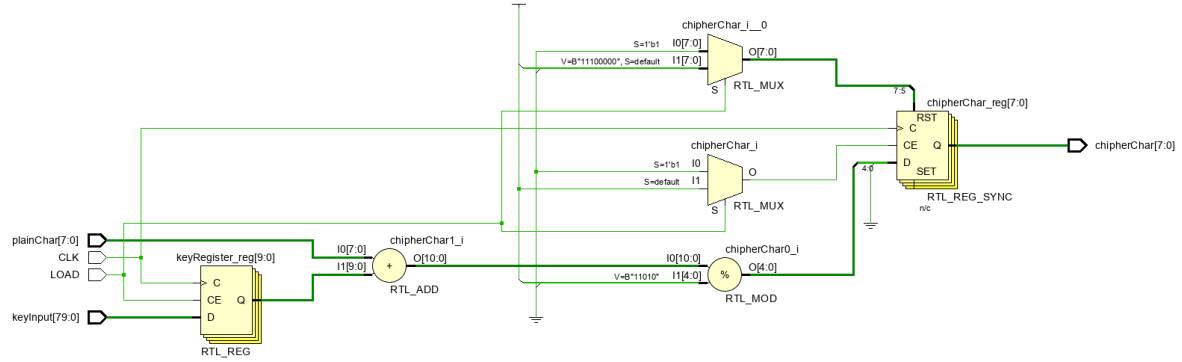

Figure 2: Vigen‘ere Cipher Encryption RTL Schematic

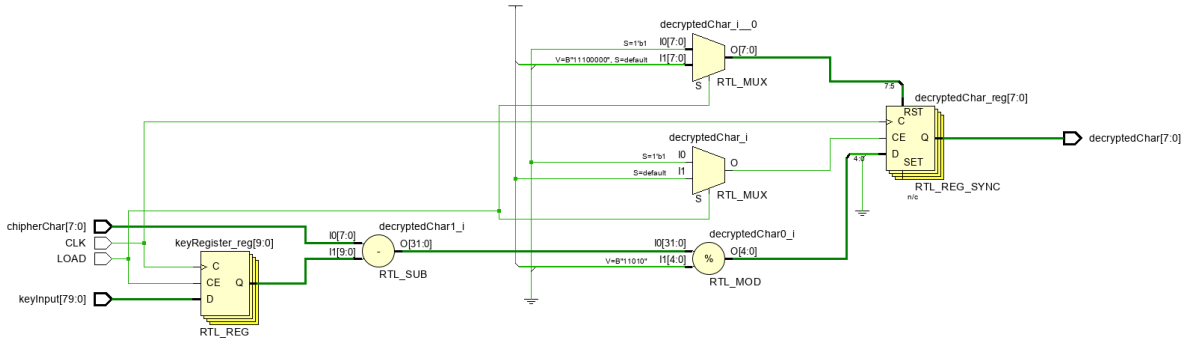

Figure 3: Vigen‘ere Cipher Decryption RTL Schematic

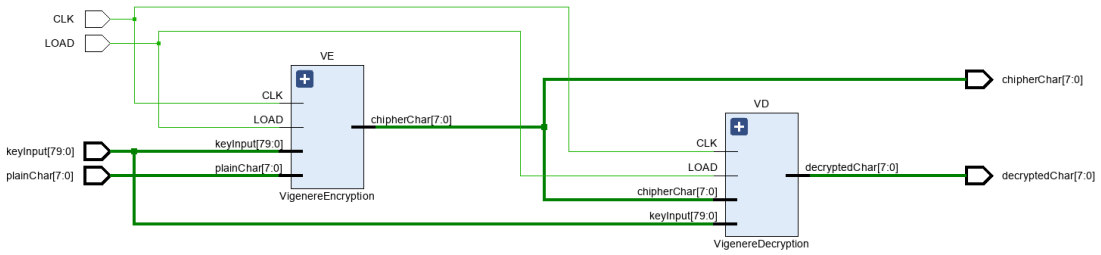

Figure 4: Vigen‘ere Cipher Environment RTL Schematic

- **VigenereEncryption Module**: It encrypts the char using Vigen‘ere Cipher technique. It has 2 char input, one of them is plain char which is going to be encrypted. The other one is key char. It has 1 bit clock and load input that allows registers to work. It gives 8-bit encrypted char output.

- **VigenereDecryption Module**: It decrypts the char using Vigen‘ere Cipher technique. It has 2 char input, one of them is plain char which is encrypted. The other one is key char. It has 1 bit clock and load input that allows registers to work. It gives 8-bit decrypted char output.

- **VigenereEnvironment Module**: It is a module that contains VigenereEncryption and VigenereDecryption modules inside.

3 RESULTS [15 points]

3.1 PART 1

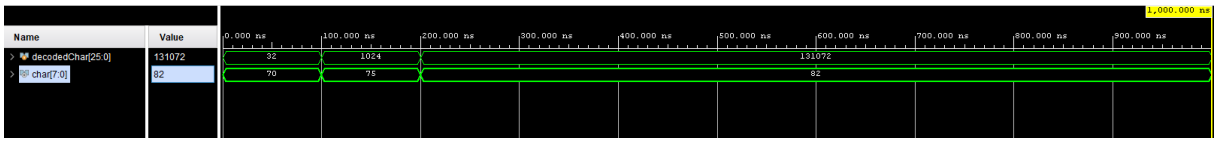

Figure 5: Char Encoder Simulation

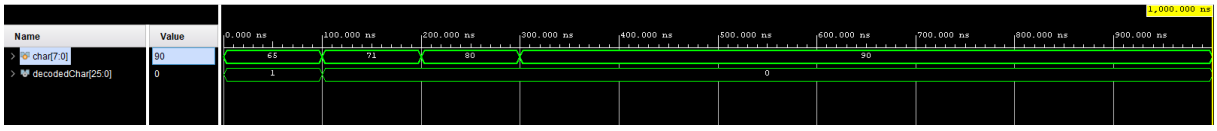

Figure 6: Char Decoder Simulation

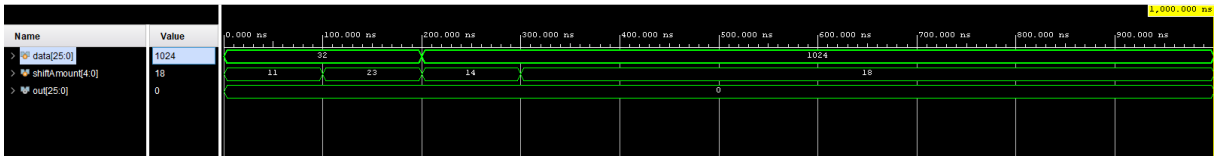

Figure 7: Circular Left Shift Simulation

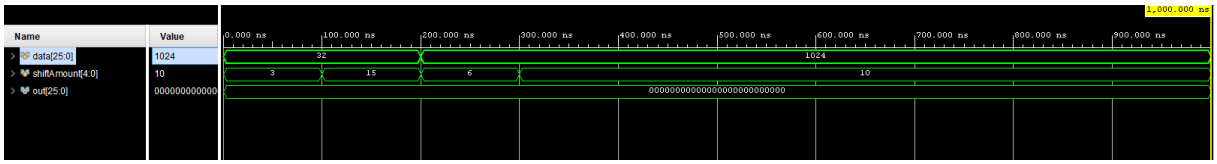

Figure 8: Circular Right Shift Simulation

3.2 PART 2

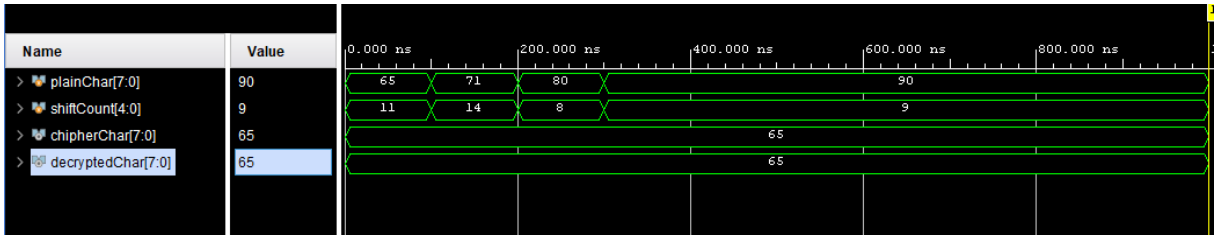

Figure 9: Ceaser Environment Simulation

3.3 PART 3

4 DISCUSSION [25 points]

In Part 1, we implemented 4 modules. These are the codes of the modules that we designed in this part. Circular Left Shift and Right Shift are similar.

```
    always @(*) begin
        tempDecoded=26'd0;

        for (k=0; k<8; k=k+1)begin
            x=x+tempChar[k]*y;
            y=y*2;
        end
        tempDecoded[x]=1;
    end

    assign decodedChar=tempDecoded;
```

Listing 1: CharDecoder

We used an always block and transformend char to decoded char here. It transforms ASCII codes of to decoded binary character.

```
always@(*)begin
        k=0;
        tempChar=8'd0;

        for(k=0; k<26; k=k+1)begin
            if(decodedChar[k]==1)begin
                x=k;
            end
        end
        k=0;

        while(x!=0)begin
            if(x%2==1)begin
                tempChar[k]=1;
            end

            k=k+1;
            x=x/2;
        end
    end

    assign char=tempChar;
```

Listing 2: CharEncoder

We used an always block and transformend decoded char to encoded char. It transforms the binary decoded version of the char to ASCII code.

```
CharEncoder CE(data, tempData);

    always @(*) begin
        k=tempData;
        k=k-shiftAmount;

        while(k<65)begin
            k=k+1;
        end
    end

    CharDecoder CD(k,out);
```

Listing 3: CircularRightShift

This code does shifting by taking shiftAmount. 65 is the difference between ASCII code of a char and its index.

In Part 2, we implemented 3 modules: CaesarEncryption, CaesarDecryption and CaesarEnvironment. In CaesarEncryption module, we have 2 inputs -plainChar and shiftCount- and 1 output -chipherChar-. First, we use CharDecoder module and decode plainChar. Then, we use CircularRightShift module and shift our decoded char to right. Lastly, we encode our decoded char with CharEncoder module. CaesarDecryption module does almost the same thing as the CaesarEncryption module; but instead of shifting right, it shifts left using CircularLeftShift module. In CaesarEnvironment module, we have 2 inputs -plainChar and shiftCount- and 2 outputs -chipherChar and decryptedChar-. We use both CaesarEncryption and CaesarDecryption in this module. The output of the CaesarEncryption is chipherChar and the output of CaesarDecryption is decryptedChar.

In Part 3, we implemented 3 modules: VigenereEncryption, VigenereDecryption and VigenereEnvironment. In VigenereEncryption module, we have 4 inputs -plainChar, keyInput, LOAD and CLK- and 1 output -chipherChar-. We use "Ci = (Pi + Ki) mod 26" equation to calculate chipherChar if LOAD is 0. If LOAD is 1, it loads the keyInput into keyRegister. VigenereDecryption module does almost the same thing as the VigenereEncryption; but it uses a different equation to calculate decryptedChar ,"Di = (Ci Ki) mod 26". In VigenereEnvironment module we have 2 inputs, the output of the VigenereEncryption is chipherChar and the output of VigenereDecryption is decryptedChar.

5 CONCLUSION [10 points]

It was more fun to do this homework compared to previous assignments. In this experience, we learned about different types of cryptography applications: Caesar Cipher and Vigenere Cipher. We also learned how to use for and while modules. Unfortunately, we didn't have enough time to complete Part 4. In some parts, we could not get RTL schematics and simulation results. So we could not put them in report.