..- . ..--- ..--- . -.-. ..--- ....--- .-

CRYPTOGRAPHY USING SYSTEMVERILOG LEVERAGING AES

- .... . / . -. .. --. -- .-

by: Arun I

Objective

- **Problem Statement:** Cryptography is one of the key elements in data transmission it prevents third parties from accessing data. encryption methods such as AES,DOS are 3DOS are time taking processes and are mostly preffered for large amount of data.When it comes down to smaller systems implementation of these encryption might not be necessary due to its complexity and requirements.So providing a adaptable and scalable tool is nessecery for smaller devices which do not require a very complex encryption. This brings us the need for simple and quick cipher for short data which can also be scaled for large data by replication.
- **Points of focus:** Secure Data Encryption and Decryption,Flexibility and Adaptability, ensuring encryption standards and high utilization and efficieny.

-.-. .- -.. -..

About the Tool

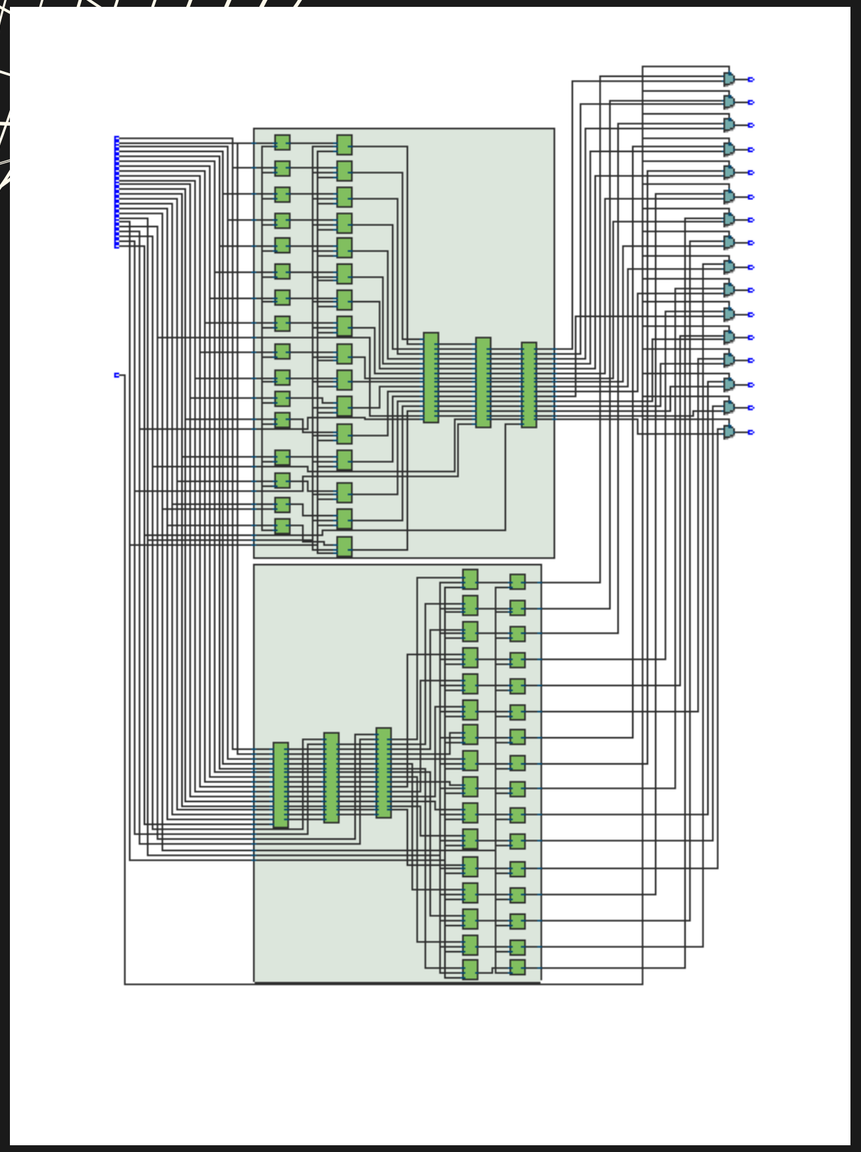
- Our tool operates on 128-bit data using an 8-bit key, enabling secure encryption and decryption processes.
- **Input:** 128bit data,Key,mode.
- **Output**: Encrypted Data.
- **Key Combinations:** With a total of 256 possible combinations for the 8-bit key, our tool ensures a wide range of encryption possibilities.
- **User-Controlled Mode:** The tool's functionality is dictated by a user-input mode, providing flexibility for encryption or decryption operations.
- The Tool uses 5 modules with multiple components to cipher data.

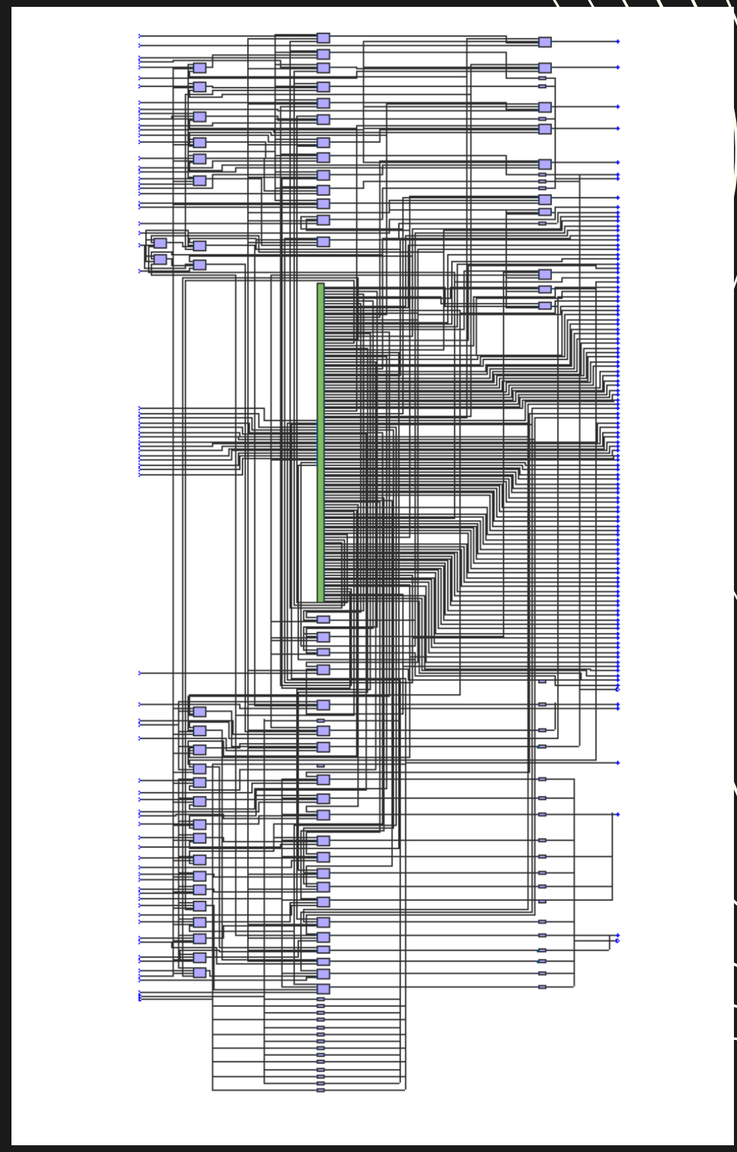
Compilation Report

- The tool uses 2166 Logic Elements & 265 I/O pins.
- Compilation time: 3 mins
- Additionally, it consists of 20 module files.

Top-level Entity Name	Enigma
Family	MAX 10
Device	10M50DAF672C7G
Timing Models	Final
Total logic elements	2,166 / 49,760 (4 %)
Total registers	0
Total pins	265 / 500 (53 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

RTL & Technology Schematic





- .... . / . -. .. --. -- .-

Code Snippets

Main Module:

```
module Enigma(input logic [7:0]a0,a1,a2,a3,b0,b1,b2,b3,c0,c1,c2,c3,d0,d1,d2,
 input logic s0,s1,s2,s3,s4,s5,s6,s7,
 output logic [7:0]w0,w1,w2,w3,x0,x1,x2,x3,y0,y1,y2,y3,z0,z1,z2,z3,
 input logic mode
 );
 logic [7:0]n0,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15;
 logic [7:0]m0,m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15;

 Encryption En1(a0,a1,a2,a3,b0,b1,b2,b3,c0,c1,c2,c3,d0,d1,d2,d3,s0,s1,s2,s3,s

 Decryption De1(a0,a1,a2,a3,b0,b1,b2,b3,c0,c1,c2,c3,d0,d1,d2,d3,s0,s1,s2,s3,s

 always@(mode)
 begin
 if (mode ==0)
    begin
    w0 = n0;w1 = n1;w2 = n2;w3 = n3;
    x0 = n4;x1 = n5;x2 = n6;x3 = n7;
    y0 = n8;y1 = n9;y2 = n10;y3 = n11;
    z0 = n12;z1 = n13;z2 = n14;z3 = n15;
    end
 else if(mode ==1)
    begin
    w0 = m0;w1 = m1;w2 = m2;w3 = m3;
    x0 = m4;x1 = m5;x2 = m6;x3 = m7;
    y0 = m8;y1 = m9;y2 = m10;y3 = m11;
    z0 = m12;z1 = m13;z2 = m14;z3 = m15;
    end
 end
 endmodule
```

Encryption Module:

```
module Encryption(
input logic [7:0]a0,a1,a2,a3,b0,b1,b2,b3,c0,c1,c2,c3,d0,d1,d2,d3,
input logic s0,s1,s2,s3,s4,s5,s6,s7,
output logic [7:0]w0,w1,w2,w3,x0,x1,x2,x3,y0,y1,y2,y3,z0,z1,z2,z3
);
logic [7:0]l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15,l16;

DataInv data0(a0,s0,l1);
DataInv data1(a1,s0,l2);
DataInv data2(a2,s0,l3);
DataInv data3(a3,s0,l4);
DataInv data4(b0,s0,l5);
DataInv data5(b1,s0,l6);
DataInv data6(b2,s0,l7);
DataInv data7(b3,s0,l8);
DataInv data8(c0,s0,l9);
DataInv data9(c1,s0,l10);
DataInv data10(c2,s0,l11);
DataInv data11(c3,s0,l12);
DataInv data12(d0,s0,l13);
DataInv data13(d1,s0,l14);
DataInv data14(d2,s0,l15);
DataInv data15(d3,s0,l16);

logic [7:0]m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16;

barrel bar0(l1,s1,s2,m1);
barrel bar1(l2,s1,s2,m2);
barrel bar2(l3,s1,s2,m3);
barrel bar3(l4,s1,s2,m4);
barrel bar4(l5,s1,s2,m5);
barrel bar5(l6,s1,s2,m6);
barrel bar6(l7,s1,s2,m7);
barrel bar7(l8,s1,s2,m8);
barrel bar8(l9,s1,s2,m9);
barrel bar9(l10,s1,s2,m10);
barrel bar10(l11,s1,s2,m11);
barrel bar11(l12,s1,s2,m12);
barrel bar12(l13,s1,s2,m13);
barrel bar13(l14,s1,s2,m14);
barrel bar14(l15,s1,s2,m15);
barrel bar15(l16,s1,s2,m16);

logic [7:0]n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16;

Column col1(m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16,s3,s4,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,

logic [7:0]o1,o2,o3,o4,o5,o6,o7,o8,o9,o10,o11,o12,o13,o14,o15,o16;

Rows r1(n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16,s5,s6,o1,o2,o3,o4,o5,o6,o7,o8,o9,o10,o11,o12,o13,o14,o15,

Rotate rr1(o1,o2,o3,o4,o5,o6,o7,o8,o9,o10,o11,o12,o13,o14,o15,o16,s7,w0,w1,w2,w3,x0,x1,x2,x3,y0,y1,y2,y3,z0,z1,z2,z3);

endmodule
```

Code Snippets

Decryption:

```
module Decryption(
input logic [7:0]a0,a1,a2,a3,b0,b1,b2,b3,c0,c1,c2,c3,d0,d1,d2,d3,
input logic s0,s1,s2,s3,s4,s5,s6,s7,
output logic [7:0]w0,w1,w2,w3,x0,x1,x2,x3,y0,y1,y2,y3,z0,z1,z2,z3
);

logic [7:0]l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15,l16;

ReRotate rel(a0,a1,a2,a3,b0,b1,b2,b3,c0,c1,c2,c3,d0,d1,d2,d3,s7,l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15,l16);

logic [7:0]m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16;

ReRows rrl(l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15,l16,s5,s6,m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16);

logic [7:0]n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16;

ReColumn rcl(m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16,s3,s4,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16);

logic [7:0]o1,o2,o3,o4,o5,o6,o7,o8,o9,o10,o11,o12,o13,o14,o15,o16;

Rebarrel bar0(n1,s1,s2,o1);
Rebarrel bar1(n2,s1,s2,o2);
Rebarrel bar2(n3,s1,s2,o3);
Rebarrel bar3(n4,s1,s2,o4);
Rebarrel bar4(n5,s1,s2,o5);
Rebarrel bar5(n6,s1,s2,o6);
Rebarrel bar6(n7,s1,s2,o7);
Rebarrel bar7(n8,s1,s2,o8);
Rebarrel bar8(n9,s1,s2,o9);
Rebarrel bar9(n10,s1,s2,o10);
Rebarrel bar10(n11,s1,s2,o11);
Rebarrel bar11(n12,s1,s2,o12);
Rebarrel bar12(n13,s1,s2,o13);
Rebarrel bar13(n14,s1,s2,o14);
Rebarrel bar14(n15,s1,s2,o15);
Rebarrel bar15(n16,s1,s2,o16);

ReDataInv data0(o1,s0,w0);
ReDataInv data1(o2,s0,w1);
ReDataInv data2(o3,s0,w2);
ReDataInv data3(o4,s0,w3);
ReDataInv data4(o5,s0,x0);
ReDataInv data5(o6,s0,x1);
ReDataInv data6(o7,s0,x2);
ReDataInv data7(o8,s0,x3);
ReDataInv data8(o9,s0,y0);
ReDataInv data9(o10,s0,y1);
ReDataInv data10(o11,s0,y2);
ReDataInv data11(o12,s0,y3);
ReDataInv data12(o13,s0,z0);
ReDataInv data13(o14,s0,z1);
ReDataInv data14(o15,s0,z2);
ReDataInv data15(o16,s0,z3);


endmodule
```

Data Encryption1:

```
module DataInv(
    input logic [7:0] a,
    input logic s0,
    output logic [7:0] w
);

    logic [7:0] out;
    Subtractor s1(a, out);

    always @(s0)
    begin
        if (s0 == 0)
        begin
            w <= ~a;
        end
        else if (s0 == 1)
        begin
            w <= out;
        end
    end

endmodule
```

Code Snippets

Data Encryption 2:

```
module barrel(
input logic [0:7] a,
input logic s1,s2,
output [0:7] d
);

mux4(a[6],a[2],a[2],a[4],s1,s2,d[0]);
mux4(a[7],a[3],a[3],a[5],s1,s2,d[1]);
mux4(a[0],a[4],a[0],a[6],s1,s2,d[2]);
mux4(a[1],a[5],a[1],a[7],s1,s2,d[3]);
mux4(a[2],a[6],a[6],a[0],s1,s2,d[4]);
mux4(a[3],a[7],a[7],a[1],s1,s2,d[5]);
mux4(a[4],a[0],a[4],a[2],s1,s2,d[6]);
mux4(a[5],a[1],a[5],a[3],s1,s2,d[7]);

endmodule
```

Level 1: Rearragement Module:

```
module Column(
input logic [7:0] a0,a1,a2,a3,b0,b1,b2,b3,c0,c1,c2,c3,d0,d1,d2,d3,
input logic s3,s4,
output logic [7:0]w0,w1,w2,w3,x0,x1,x2,x3,y0,y1,y2,y3,z0,z1,z2,z3
);

always @(s3 or s4)
   begin
   if(s3 == 0)
      begin
      if(s4 == 0)
         begin
         w0 = a2;
         w1 = a1;
         w2 = a0;
         w3 = a3;

         x0 = b2;
         x1 = b1;
         x3 = b0;
         x2 = b3;

         y0 = c2;
         y1 = c1;
         y2 = c0;
         y3 = c3;

         z0 = d2;
         z1 = d1;
         z2 = d0;
         z3 = d3;
         end
      else if(s4 == 1)
         begin
         w0 = a1;
         w1 = a0;
```

Code Snippets

Level 2: Rearragement Module:

```
module Rows(
input logic [7:0] a0,a1,a2,a3,b0,b1,b2,b3,c0,c1,c2,c3,d0,d1,d2,d3,
input logic s3,s4,
output logic [7:0] w0,w1,w2,w3,x0,x1,x2,x3,y0,y1,y2,y3,z0,z1,z2,z3
);

always @(s3 or s4)
   begin
   if(s3 == 0)
      begin
      if(s4 == 0)
         begin
         w0 = c0;
         x0 = b0;
         y0 = a0;
         z0 = d0;

         w1 = c1;
         x1 = b1;
         y1 = a1;
         z1 = d1;

         w2 = c2;
         x2 = b2;
         y2 = a2;
         z2 = d2;

         w3 = c3;
         x3 = b3;
         y3 = a3;
         z3 = d3;
         end
      else if(s4 == 1)
         begin
         w0 = a0;
         x0 = d0;
         y0 = c0;
         z0 = b0;

         w1 = a1;
         x1 = d1;
         y1 = c1;
         z1 = b1;

         w2 = a2;
         x2 = d2;
```

Level 3: Rearragement Module:

```
input logic [7:0]a0,a1,a2,a3,b0,b1,b2,b3,c0,c1,c2,c3,d0,d1,d2,d3,
input logic s7,
output logic [7:0]w0,w1,w2,w3,x0,x1,x2,x3,y0,y1,y2,y3,z0,z1,z2,z3
);

always @(s7)
   begin
   if(s7 == 0)
      begin
      w0 = a3;
      w1 = b3;
      w2 = c3;
      w3 = d3;

      x0 = a2;
      x1 = b1;
      x2 = b2;
      x3 = d2;

      y0 = a1;
      y1 = c1;
      y2 = c2;
      y3 = d1;

      z0 = a0;
      z1 = b0;
      z2 = c0;
      z3 = d0;
      end
      else if (s7 == 1)
      begin//rotate right
         w0 = d0;
         w1 = c0;
         w2 = b0;
         w3 = a0;

         x0 = d1;
         x1 = b1;
         x2 = b2;
         x3 = a1;

         y0 = d2;
         y1 = c1;
         y2 = c2;
         y3 = a2;

         z0 = d3;
         z1 = c3;
         z2 = b3;
         z3 = a3;
       end
      end
endmodule
```

Encryption

CLASSIFIED

Data Encryption: The Data consist of 16 bytes.Where each byte carries a character. Based on the first bit of key the given 8 bit data is either inverted or decremented.

Data Shifter: The 8 bit data after inverting or decrementing is rotated to the right or left to further encrypt the data based on 2nd & 3rd bit of key.

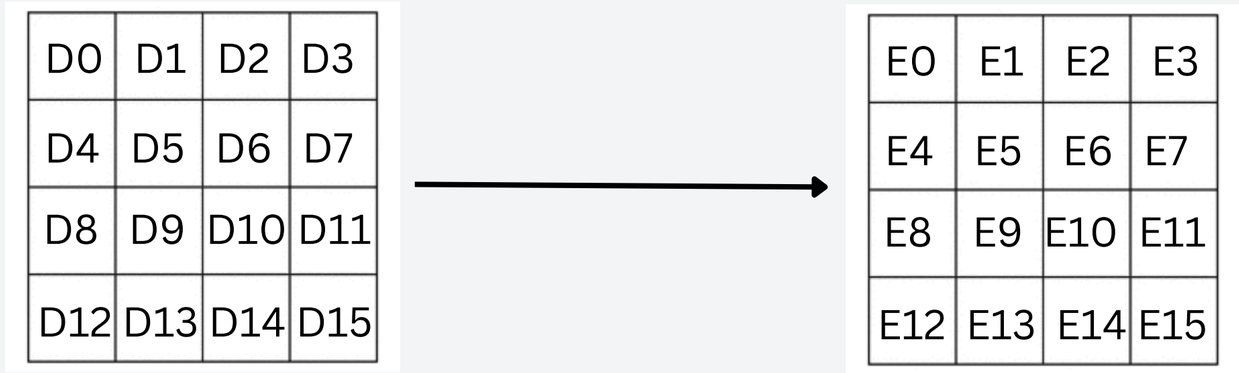
Data Visualization: Given 16 byte data is visualized in the form of 4x4 matrix for shuffling the order.

D0	D1	D2	D3
D4	D5	D6	D7
D8	D9	D10	D11
D12	D13	D14	D15

Rearrangement

CLASSIFIED

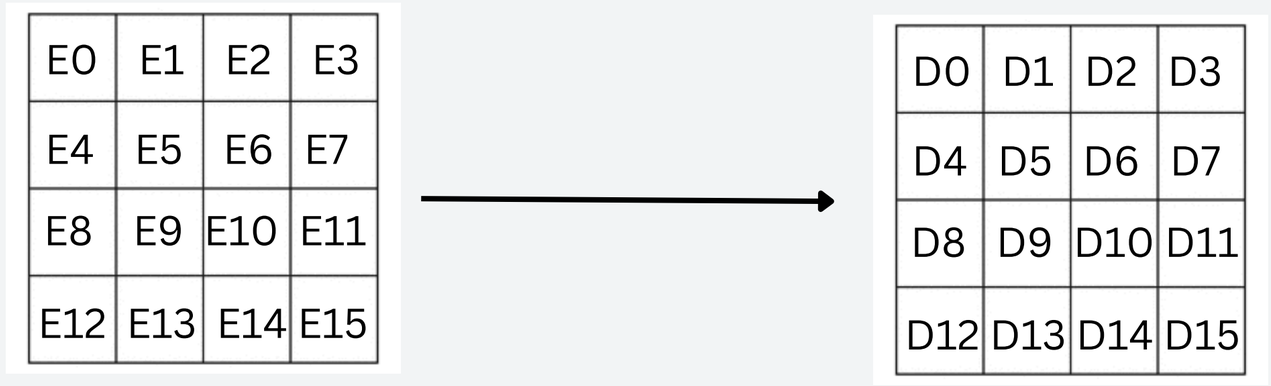
- Level 1: In 4x4 Visualization,the columns of the matrix are rearranged in multiple patterns based on 4rd and 5th bit of key.
- Level 2: The Rows of the matrix are rearranged. based on 6th and 7th bit of the key.
- Level 3: The center 4 elements hold their position and the outermost layer is rotated to the right or left based on 8th bit of the key.
- The Data after Level 3 is given to output.
- The Encryption Process contains 5 Levels including Data Manipulation.



Decryption

CLASSIFIED

- Decryption Procress is the Inversion of Encryption Process.
- The Decryption Process starts with the Level 3 which is Rotating the matrix.
- Then Data’s rows are shifted to the initial state.
- Then Columns are shifted based on the same key sequence.
- The 8-bit element is rotated as an inverse operation of encryption.
- Then as final Level the byte is either inverted again or incremented based on the 1st bit key.
- After final bit operation the Decrypted data is fed as output



Project Dynamics: Encryption

Testbench Input data “CRYPTOGRAPHICSYS”

```
23      begin
24          #10//CRYPTOGRAPHICSYS
25              A0  = 8'b01000011;
26              A1  = 8'b01010010;
27              A2  = 8'b01011001;
28              A3  = 8'b01010000;
29              A4  = 8'b01010100;
30              A5  = 8'b01001111;
31              A6  = 8'b01000111;
32              A7  = 8'b01010010;
33              A8  = 8'b01000001;
34              A9  = 8'b01010000;
35              A10 = 8'b01001000;
36              A11 = 8'b01001001;
37              A12 = 8'b01000011;
38              A13 = 8'b01010011;
39              A14 = 8'b01011001;
40              A15 = 8'b01010011;
```

```
41              mode = 1'b0;
42              S0 =  1'b1;
43              S1 =  1'b1;
44              S2 =  1'b1;
45              S3 =  1'b1;
46              S4 =  1'b0;
47              S5 =  1'b0;
48              S6 =  1'b0;
49              S7 =  1'b0;
```

Output: $$5%d□t%

```
GIVEN DATA : 01000011 01010010 01011001 01010000 01010100 01001111 01000111 01010010 01000001 01010000 01001000 01001001 01000011 01010011 01011001 01010011
ENCRYPTED DATA : 00100100 00100100 00110101 00000100 00100101 00010101 01100100 10000100 10000101 11110100 10000101 01110100 00100101 00010101 11100100 11110100
```

Project Dynamics: Decryption

Testbench Input “$$5%d□t%”

```
54          A0 = D0;
55          A1 = D1;
56          A2 = D2;
57          A3 = D3;
58          A4 = D4;
59          A5 = D5;
60          A6 = D6;
61          A7 = D7;
62          A8 = D8;
63          A9 = D9;
64          A10 = D10;
65          A11 = D11;
66          A12 = D12;
67          A13 = D13;
68          A14 = D14;
69          A15 = D15;
```

```
70          mode = 1'b1;
```

Output: “CRYPTOGRAPHICSYS”

```
# GIVEN DATA : 00100100 00100100 00110101 00000100 00100101 00010101 01100100 10000100 10000101 11110100 10000101 01110100 00100101 00010101 11100100 11110100
# DECRYPTED DATA : 01000011 01010010 01011001 01010000 01010100 01001111 01000111 01010010 01000001 01010000 01001000 01001001 01000011 01010011 01011001 01010011
```

Project Dynamics: Encryption

Testbench Input data “CRYPTOGRAPHICSYS”

```
24      #10//CRYPTOGRAPHICSYS
25          A0  = 8'b01000011;
26          A1  = 8'b01010010;
27          A2  = 8'b01011001;
28          A3  = 8'b01010000;
29          A4  = 8'b01010100;
30          A5  = 8'b01001111;
31          A6  = 8'b01000111;
32          A7  = 8'b01010010;
33          A8  = 8'b01000001;
34          A9  = 8'b01010000;
35          A10 = 8'b01001000;
36          A11 = 8'b01001001;
37          A12 = 8'b01000011;
38          A13 = 8'b01010011;
39          A14 = 8'b01011001;
40          A15 = 8'b01010011;
```

```
92          mode = 1'b0;
93          S0 =  1'b0;
94          S1 =  1'b1;
95          S2 =  1'b1;
96          S3 =  1'b1;
97          S4 =  1'b0;
98          S5 =  1'b0;
99          S6 =  1'b0;
100         S7 =  1'b0;
```

Output: ËËºëÊÚ;:ú:{

```
# GIVEN DATA : 01000011 01010010 01011001 01010000 01010100 01001111 01000111 01010010 01000001 01010000 01001000 01001001 01000011 01010011 01011001 01010011
# ENCRYPTED DATA : 11001011 11001011 10111010 11101011 11001010 11011010 10001011 01101011 01101010 11111010 01101010 01111011 11001010 11011010 00001011 11111010
```

Project Dynamics: Decryption

Testbench Input “ËËºëÊÚ;:ú:{”

```
54          A0 = D0;
55          A1 = D1;
56          A2 = D2;
57          A3 = D3;
58          A4 = D4;
59          A5 = D5;
60          A6 = D6;
61          A7 = D7;
62          A8 = D8;
63          A9 = D9;
64          A10 = D10;
65          A11 = D11;
66          A12 = D12;
67          A13 = D13;
68          A14 = D14;
69          A15 = D15;
```

```
70          mode = 1'b1;
```

Output: “CRYPTOGRAPHICSYS”

```
# GIVEN DATA : 11001011 11001011 10111010 11101011 11001010 11011010 10001011 01101011 01101010 11111010 01101010 01111011 11001010 11011010 00001011 11111010
# DECRYPTED DATA : 01000011 01010010 01011001 01010000 01010100 01001111 01000111 01010010 01000001 01010000 01001000 01001001 01000011 01010011 01011001 01010011
```

Conclusion

- **Challenges Faced:** Debugging was more complex compared to building the module.Decrypted and Encrypted data are not in a fixed state,therefore making it hard to find the exact point of error. Therefore to debug we had to manually go through each module and verify.
- **Future Expansion:** Looking ahead, we will be focusing on better efficieny an utilization of the system.We are trying to use the same module for encryption and decryption of data.which would bring the logic elements below 1000.
- **Learnings & Growth:** The project journey provided invaluable insights into cryptography, SystemVerilog, and project management.

.- .-. ..- -. / ..

THANK YOU

Transforming vulnerability into invincibility