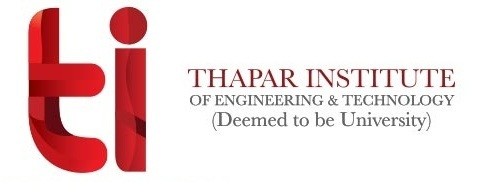
**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**



PROJECT REPORT

ON

**IMPLEMENTATION OF ADVANCED ENCRYPTION STANDARD (AES) ALGORITHM IN VERILOG**

**Submitted by**

**PRATIBHA SINGH**

**602162015**

**M.Tech (VLSI Design)**

**ADVANCED ENCRYPTION STANDARD**

**Aim:**

To implement the Advanced Encryption Standard (AES) algorithm using Verilog and to give optimum circuit with clock frequency, path delay, time required to generate keys and decoding the data.

**Technology used:** Xilinx ISE software

**Code:**

// AES for verilog

module AES(key,inputText,enable,operation,outputText);

    input [15:0] key;

    input [15:0] inputText;

    input enable;

    input operation;

    output [15:0] outputText;

    wire [15:0] encryptedText,decryptedText,result;

    Encryption e(key,inputText,encryptedText);

    Decryption d(key,inputText,decryptedText);

    MUX\_16Bit\_2x1 mux1(decryptedText,encryptedText,operation,result);

    MUX\_16Bit\_2x1 mux2(result,inputText,enable,outputText);

Endmodule

//KeySchedule

module KeySchedule(key,k1,k2);

    input [15:0] key;

    output [15:0] k1,k2;

    wire [7:0] w2,w3,w4,w5;

    wire [3:0] w1a,w1b,w3a,w3b;

    SBoxEncrypt box1(key[7:4],w1a);

    SBoxEncrypt box2(key[3:0],w1b);

    SBoxEncrypt box3(w3[7:4],w3a);

    SBoxEncrypt box4(w3[3:0],w3b);

    assign w2 = key[15:8] ^ {w1b,w1a} ^ 8'b10000000;

    assign w3 = key[7:0] ^ w2;

    assign w4 = w2 ^ {w3b,w3a} ^ 8'b00110000;

    assign w5 = w3 ^ w4;

    assign k1 = {w2,w3};

    assign k2 = {w4,w5};

endmodule

//Encryption

module Encryption(key,plainText,cipherText);

    input [15:0] key;

    input [15:0] plainText;

    output [15:0] cipherText;

    wire [15:0] k1,k2,p1;

    wire [3:0] a,b,c,d,a\_out,b\_out,c\_out,d\_out,

                a1,b1,c1,d1,a1\_out,b1\_out,c1\_out,d1\_out;

    // Generate Round Keys

    KeySchedule keys(key,k1,k2);

    // Add Round Key

    assign {a,b,c,d} = plainText ^ key;

     // Substitute Nibbles

    SBoxEncrypt box1(a,a\_out);

    SBoxEncrypt box2(b,b\_out);

    SBoxEncrypt box3(c,c\_out);

    SBoxEncrypt box4(d,d\_out);

     // Shift Rows and Mix Columns

    assign p1 = {

        (a\_out[3] ^ d\_out[1]), (a\_out[2] ^ d\_out[3] ^ d\_out[0]), (a\_out[1] ^ d\_out[3] ^ d\_out[2]), (a\_out[0] ^ d\_out[2]),

        (a\_out[1] ^ d\_out[3]), (a\_out[3] ^ a\_out[0] ^ d\_out[2]), (a\_out[3] ^ a\_out[2] ^ d\_out[1]), (a\_out[2] ^ d\_out[0]),

          (c\_out[3] ^ b\_out[1]), (c\_out[2] ^ b\_out[3] ^ b\_out[0]), (c\_out[1] ^ b\_out[3] ^ b\_out[2]), (c\_out[0] ^ b\_out[2]),

        (c\_out[1] ^ b\_out[3]), (c\_out[3] ^ c\_out[0] ^ b\_out[2]), (c\_out[3] ^ c\_out[2] ^ b\_out[1]), (c\_out[2] ^ b\_out[0])

    };

     // Add Round Key

    assign {a1,b1,c1,d1} = p1 ^ k1;

    // Substitute Nibbles

    SBoxEncrypt box5(a1,a1\_out);

    SBoxEncrypt box6(b1,b1\_out);

    SBoxEncrypt box7(c1,c1\_out);

    SBoxEncrypt box8(d1,d1\_out);

     // Shift Rows and Add Round Key

    assign cipherText = {a1\_out,d1\_out,c1\_out,b1\_out} ^ k2;

endmodule

//Decryption

module Decryption(key,cipherText,plainText);

    input [15:0] key;

    input [15:0] cipherText;

    output [15:0] plainText;

    wire [15:0] k1,k2;

     wire [3:0] a,b,c,d,a\_out,b\_out,c\_out,d\_out,

                a1,b1,c1,d1,a2,b2,c2,d2,a1\_out,b1\_out,c1\_out,d1\_out;

    // Generate Round Keys

    KeySchedule keys(key,k1,k2);

     // Add Round Key

    assign {a,b,c,d} = cipherText ^ k2;

    // Inverse Shift Rows and Substitute Nibbles

    SBoxDecrypt box1(a,a\_out);

    SBoxDecrypt box2(d,b\_out);

     SBoxDecrypt box3(c,c\_out);

    SBoxDecrypt box4(b,d\_out);

    // Add Round Key

    assign {a1\_out,b1\_out,c1\_out,d1\_out} = {a\_out,b\_out,c\_out,d\_out} ^ k1;

         // Inverse Mix Columns and Shift Rows

    assign {a1,b1,c1,d1} = {

        (a1\_out[0] ^ b1\_out[2]), (a1\_out[3] ^ b1\_out[1]), (a1\_out[2] ^ b1\_out[3] ^ b1\_out[0]), (a1\_out[1] ^ a1\_out[0] ^ b1\_out[3]),

        (c1\_out[2] ^ d1\_out[0]), (c1\_out[1] ^ d1\_out[3]), (c1\_out[3] ^ c1\_out[0] ^ d1\_out[2]), (c1\_out[3] ^ d1\_out[1] ^ d1\_out[0]),

          (c1\_out[0] ^ d1\_out[2]), (c1\_out[3] ^ d1\_out[1]), (c1\_out[2] ^ d1\_out[3] ^ d1\_out[0]), (c1\_out[1] ^ c1\_out[0] ^ d1\_out[3]),

        (a1\_out[2] ^ b1\_out[0]), (a1\_out[1] ^ b1\_out[3]), (a1\_out[3] ^ a1\_out[0] ^ b1\_out[2]), (a1\_out[3] ^ b1\_out[1] ^ b1\_out[0])

    };

      // Inverse Substitute Nibbles

    SBoxDecrypt box5(a1,a2);

    SBoxDecrypt box6(b1,b2);

    SBoxDecrypt box7(c1,c2);

    SBoxDecrypt box8(d1,d2);

    // Add Round Key

    assign plainText = {a2,b2,c2,d2} ^ key;

endmodule

//SBoxDecrypt

module SBoxDecrypt(s1\_in,s1\_out);

    input [3:0] s1\_in;

    output [3:0] s1\_out;

    reg [3:0] s1\_out;

    always@(s1\_in)

    begin

        case(s1\_in)

            4'b0000: s1\_out = 4'hA;

            4'b0001: s1\_out = 4'h5;

            4'b0010: s1\_out = 4'h9;

            4'b0011: s1\_out = 4'hB;

                4'b0100: s1\_out = 4'h1;

            4'b0101: s1\_out = 4'h7;

            4'b0110: s1\_out = 4'h8;

            4'b0111: s1\_out = 4'hF;

            4'b1000: s1\_out = 4'h6;

            4'b1001: s1\_out = 4'h0;

            4'b1010: s1\_out = 4'h2;

            4'b1011: s1\_out = 4'h3;

            4'b1100: s1\_out = 4'hC;

            4'b1101: s1\_out = 4'h4;

            4'b1110: s1\_out = 4'hD;

            4'b1111: s1\_out = 4'hE;

        endcase

    end

endmodule

// SBoxEncrypt

module SBoxEncrypt(s0\_in,s0\_out);

    input [3:0] s0\_in;

    output [3:0] s0\_out;

    reg [3:0] s0\_out;

    always@(s0\_in)

    begin

        case(s0\_in)

            4'b0000: s0\_out = 4'h9;

            4'b0001: s0\_out = 4'h4;

                4'b0010: s0\_out = 4'hA;

            4'b0011: s0\_out = 4'hB;

            4'b0100: s0\_out = 4'hD;

            4'b0101: s0\_out = 4'h1;

            4'b0110: s0\_out = 4'h8;

            4'b0111: s0\_out = 4'h5;

            4'b1000: s0\_out = 4'h6;

            4'b1001: s0\_out = 4'h2;

            4'b1010: s0\_out = 4'h0;

                4'b1011: s0\_out = 4'h3;

            4'b1100: s0\_out = 4'hC;

            4'b1101: s0\_out = 4'hE;

            4'b1110: s0\_out = 4'hF;

            4'b1111: s0\_out = 4'h7;

        endcase

    end

endmodule

//MUX 16X1

module MUX\_2x1(I0,I1,S0,Result);

    input I0,I1,S0;

    output Result;

    and(w1,I0,S0);

    and(w2,I1,~S0);

    or(Result,w1,w2);

endmodule

module MUX\_16Bit\_2x1(IO,I1,SO,Result);

    input [15:0] IO,I1;

    input SO;

    output [15:0] Result;

    MUX\_2x1 b0(IO[0],I1[0],SO,Result[0]);

    MUX\_2x1 b1(IO[1],I1[1],SO,Result[1]);

    MUX\_2x1 b2(IO[2],I1[2],SO,Result[2]);

    MUX\_2x1 b3(IO[3],I1[3],SO,Result[3]);

    MUX\_2x1 b4(IO[4],I1[4],SO,Result[4]);

    MUX\_2x1 b5(IO[5],I1[5],SO,Result[5]);

    MUX\_2x1 b6(IO[6],I1[6],SO,Result[6]);

    MUX\_2x1 b7(IO[7],I1[7],SO,Result[7]);

    MUX\_2x1 b8(IO[8],I1[8],SO,Result[8]);

    MUX\_2x1 b9(IO[9],I1[9],SO,Result[9]);

    MUX\_2x1 b10(IO[10],I1[10],SO,Result[10]);

    MUX\_2x1 b11(IO[11],I1[11],SO,Result[11]);

    MUX\_2x1 b12(IO[12],I1[12],SO,Result[12]);

    MUX\_2x1 b13(IO[13],I1[13],SO,Result[13]);

    MUX\_2x1 b14(IO[14],I1[14],SO,Result[14]);

    MUX\_2x1 b15(IO[15],I1[15],SO,Result[15]);

endmodule

//tb

`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date:   09:29:35 03/10/2022

// Design Name:   AES

// Module Name:   /home/ise/file\_xilinx/Advanced-Encryption-Standard/tb.v

// Project Name:  Advanced-Encryption-Standard

// Target Device:

// Tool versions:

// Description:

//

// Verilog Test Fixture created by ISE for module: AES

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

////////////////////////////////////////////////////////////////////////////////

module tb;

         reg [15:0] key;

    reg [15:0] inputText;

    reg enable;

    reg operation;

    wire [15:0] outputText;

    AES a1(key,inputText,enable,operation,outputText);

         initial begin

        inputText = 16'b0000\_0111\_0011\_1000;

        key = 16'b1010\_0111\_0011\_1011;

        enable = 1;

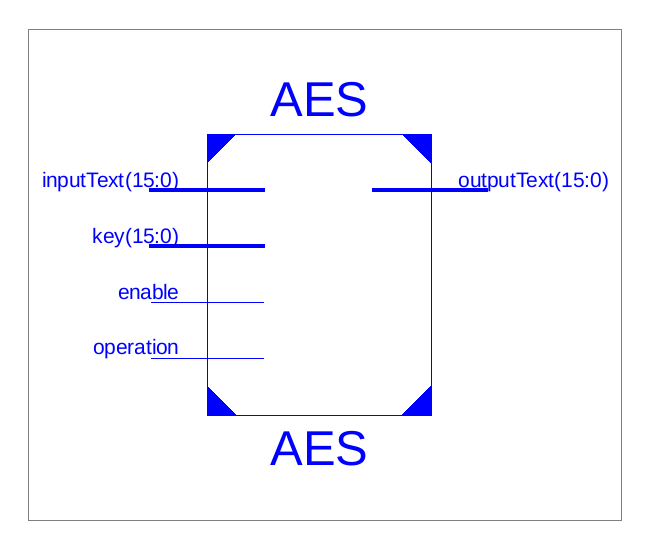
        operation = 1;

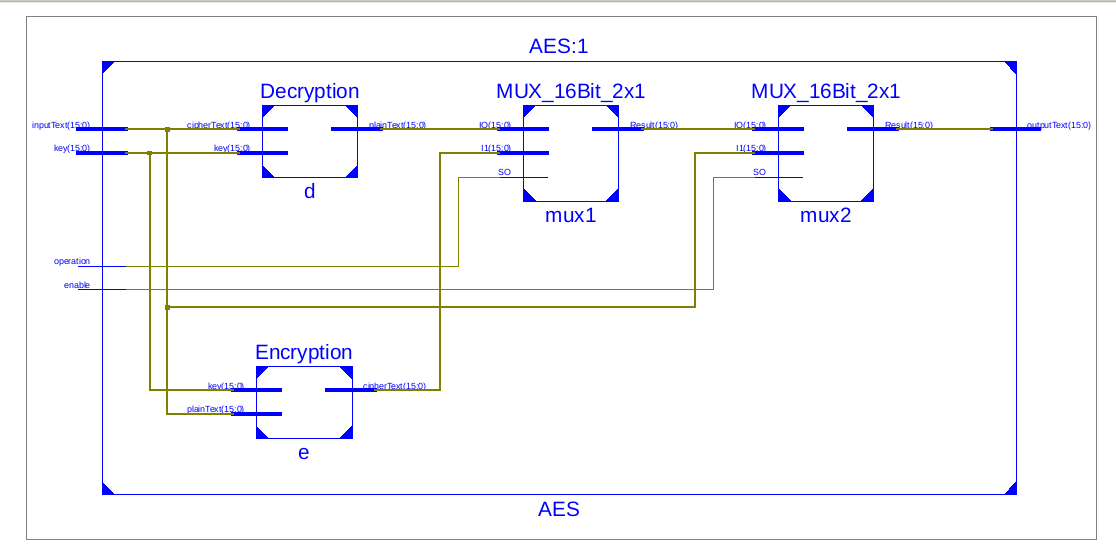
        $monitor ("inputText= %b    outputText= %b", inputText, outputText);

        end

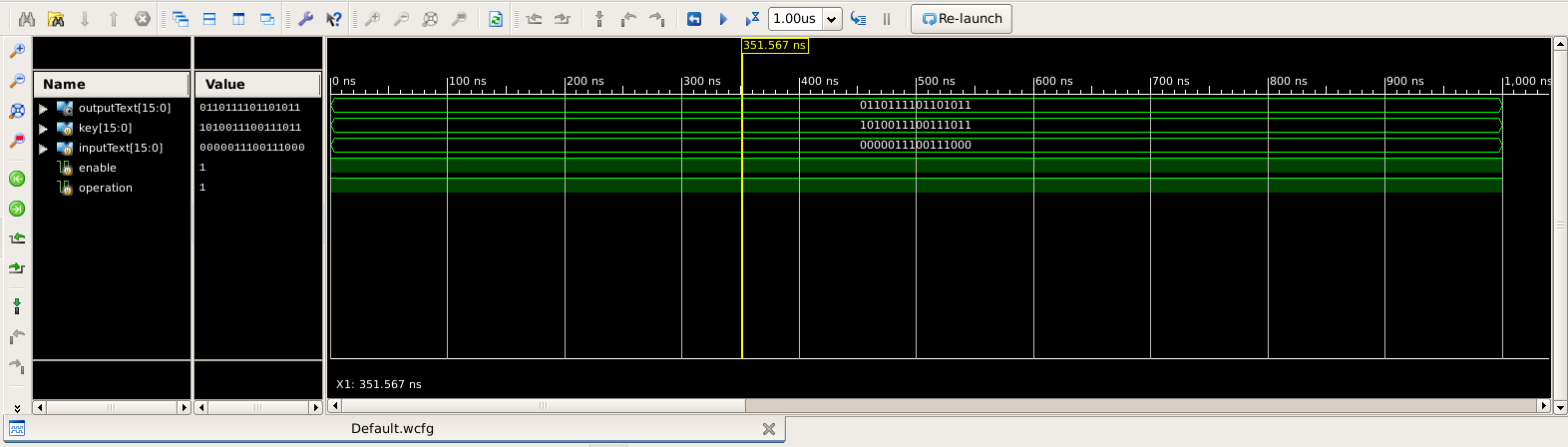
endmodule

**RTL Schematic:**





**Simulation Results:**



**Output Screen:**

