EE - 652 PROJECT PRESENTATION

DESIGN AND IMPLEMENTATION OF HARDWARE MODULE THAT PERFORMS DATA ENCRYPTION USING A SYMMETRIC-KEY ALGORITHM

Instructor:

Prof. JOYCEE MEKIE

TA:

Mr. KAILASH PRASAD

Mr. ALOK PRADHAN

Presented By:

Ashiwini Pathak

MTech. Electrical Engineering

IIT Gandhinagar

AES_encryption system verilog implementation

Pre Synthesis using Genus

Full Gds Flow

24 March

27 March

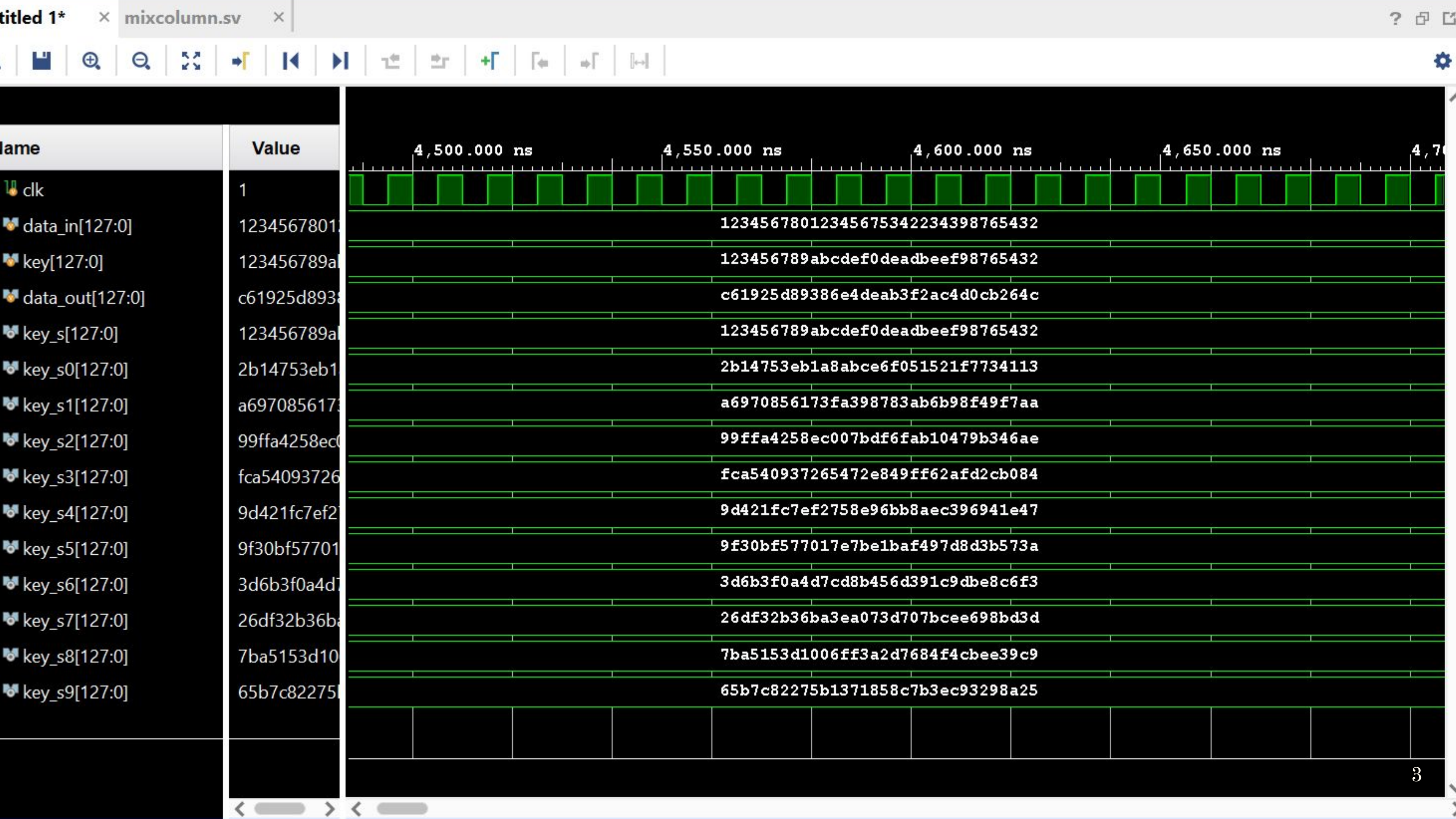
31 March

1 April

8 April

UVM verification using testbench

Post synthesis simulation on NCSIM


titled 1*
mixcolumn.sv
Name
Value
clk
1
data_in[127:0]
key[127:0]
data_out[127:0]
key_s[127:0]
key_s0[127:0]
key_s1[127:0]
key_s2[127:0]
key_s3[127:0]
key_s4[127:0]
key_s5[127:0]
key_s6[127:0]
key_s7[127:0]
key_s8[127:0]
key_s9[127:0]
4,500.000 ns
4,550.000 ns
4,600.000 ns
4,650.000 ns
12345678012345675342234398765432
123456789abcdef0deadbeef98765432
c61925d89386e4deab3f2ac4d0cb264c
123456789abcdef0deadbeef98765432
2b14753eb1a8abce6f051521f7734113
a6970856173fa398783ab6b98f49f7aa
99ffa4258ec007bdf6fab10479b346ae
fca54093726547 2e849ff62afd2cb084
9d421fc7ef2758e96bb8aec396941e47
9f30bf577017e7be1baf497d8d3b573a
3d6b3f0a4d7cd8b456d391c9dbe8c6f3
26df32b36ba3ea073d707bcee698bd3d
7ba5153d1006ff3a2d7684f4cbee39c9
65b7c82275b1371858c7b3ec93298a25

BRIEF INTRODUCTION

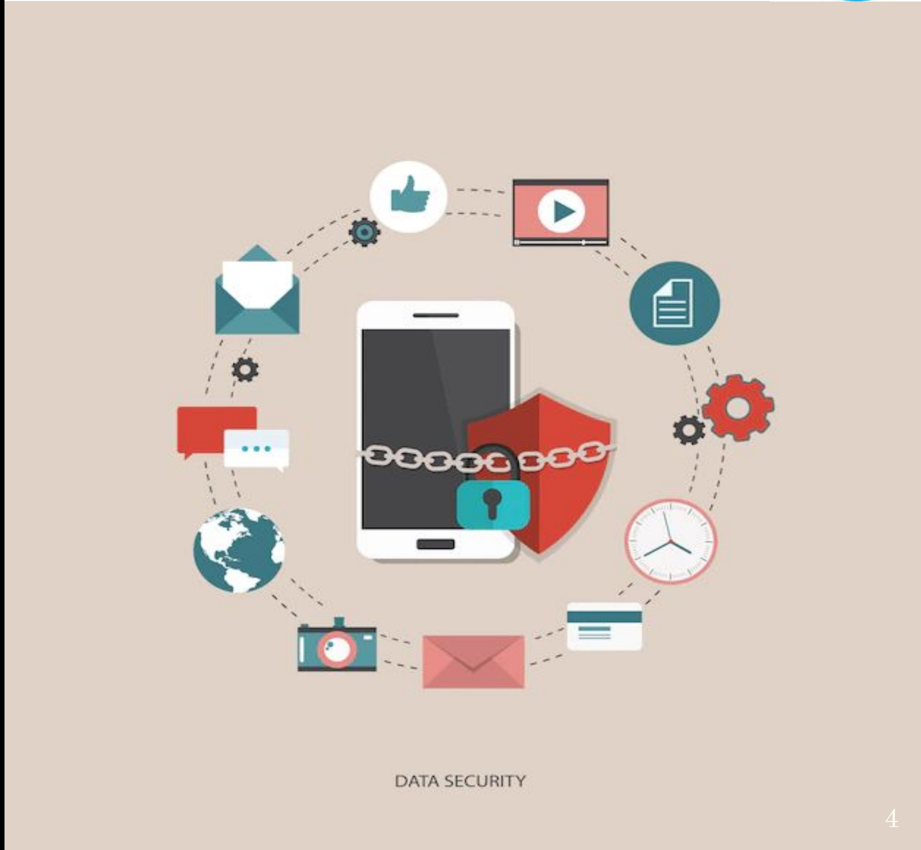
The Advanced Encryption Standard (AES) is a widely used symmetric encryption algorithm designed to provide secure data encryption and decryption. It was developed by the National Institute of Standards and Technology (NIST) in 2001 as a replacement for the Data Encryption Standard (DES).

AES uses a fixed block size of 128 bits and supports key sizes of 128, 192, or 256 bits. The algorithm consists of several rounds of substitution, permutation, and mixing operations that transform the input plaintext into ciphertext

he strength of AES lies in its ability to resist various attacks, including brute-force attacks, differential cryptanalysis, and linear cryptanalysis.

AES has become the industry standard for secure data encryption and is used in various applications, including online banking, secure communications, and file encryption.

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce



ROUNDS

- AES is a non-Feistel cipher that encrypts a data block of 128 bits, it uses 10,12, and 14 rounds.

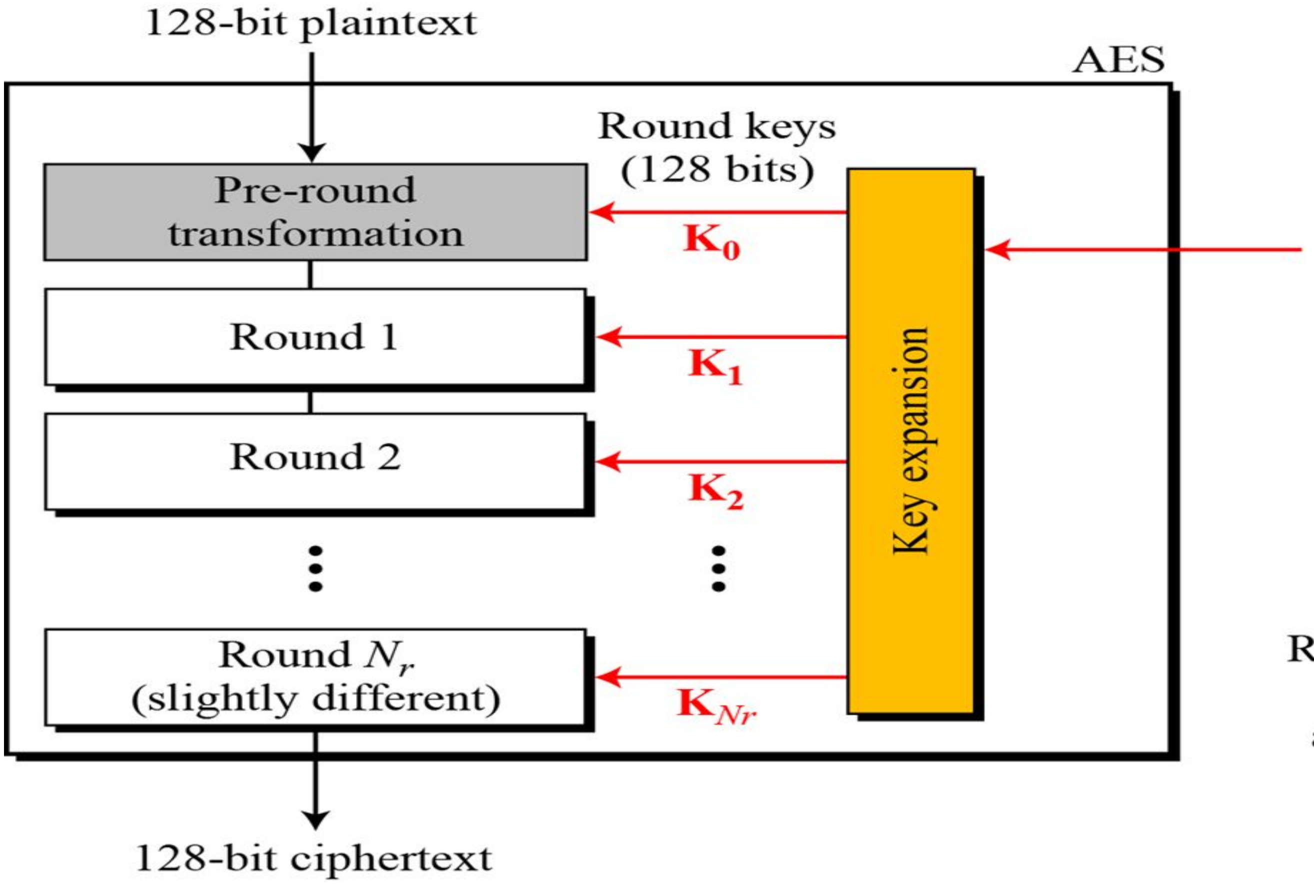
- The key size which can be 128, 192, or 256 bits, depends on the no. of rounds.

Note

AES has three version of 10, 12 and 14 rounds.

Each version uses different cypher key size (128, 192, or 256), but the round keys are always 128 bits.

GENERAL DESIGN OF AES ENCRYPTION CIPHER



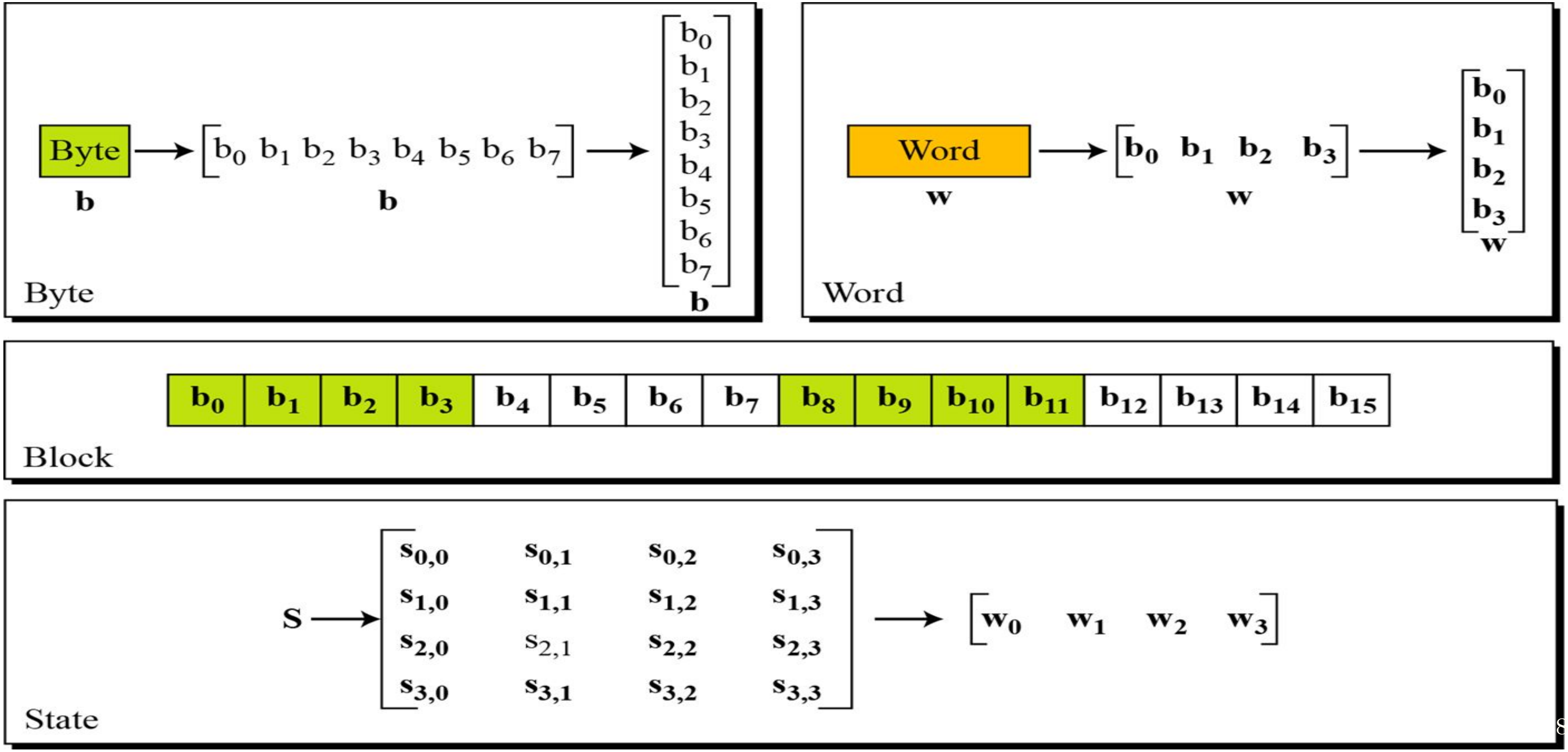
Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

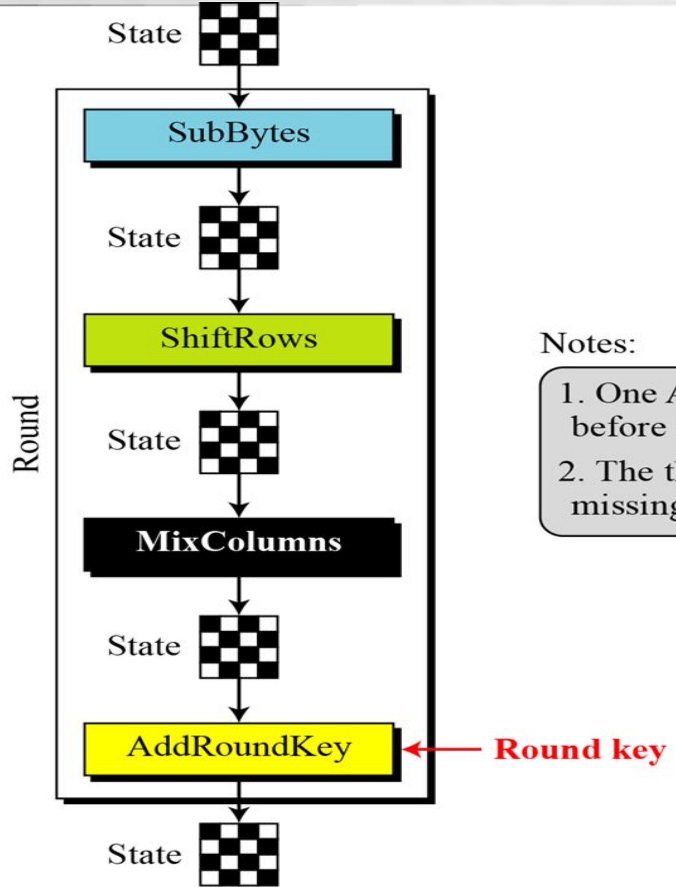
GENERAL DESIGN OF AES ENCRYPTION CIPHER

```
module aes_main(clk,data_in,key,data_out);
input logic clk;
Input logic [127:0] data_in,key;
output logic  [127:0] data_out;

Logic [127:0] key_s,key_s0,key_s1,key_s2,key_s3,key_s4,key_s5,key_s6,key_s7,key_s8,key_s9;
Logic [127:0]r_data_out,r0_data_out,r1_data_out,r2_data_out,r3_data_out,r4_data_out,r5_data_out,r6_data_out,r7_data_out,r8_data_out,r9_data_out;

//ADD_Round_key
assign r_data_out=data_in^key_s;


aes_key_expand_128 a0( clk,key, key_s,key_s0,key_s1,key_s2,key_s3,key_s4,key_s5,key_s6,key_s7,key_s8,key_s9);
round r0(clk,r_data_out,key_s0,r0_data_out);
round r1(clk,r0_data_out,key_s1,r1_data_out);
round r2(clk,r1_data_out,key_s2,r2_data_out);
round r3(clk,r2_data_out,key_s3,r3_data_out);
round r4(clk,r3_data_out,key_s4,r4_data_out);
round r5(clk,r4_data_out,key_s5,r5_data_out);
round r6(clk,r5_data_out,key_s6,r6_data_out);
round r7(clk,r6_data_out,key_s7,r7_data_out);
round r8(clk,r7_data_out,key_s8,r8_data_out);
last_round r9(clk,r8_data_out,key_s9,r9_data_out);

assign data_out=r9_data_out;
endmodule
```

DATA UNITS USED IN AES



STRUCTURE OF EACH ROUND AT ENCRYPTION SITE



Notes:

1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

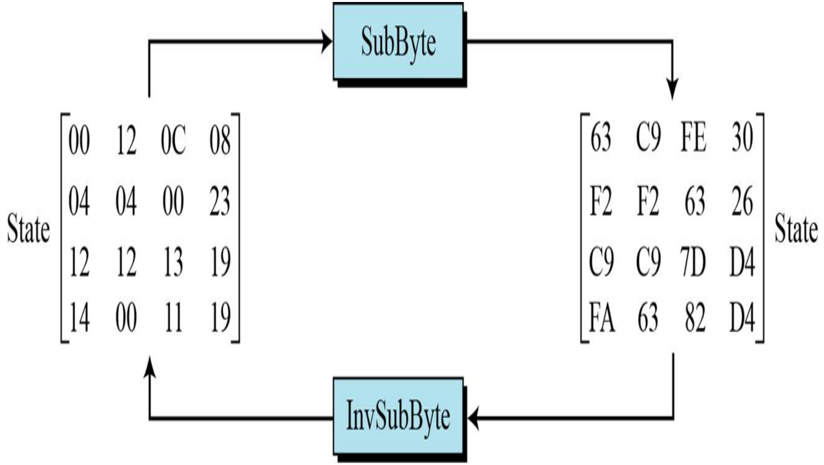
Substitution

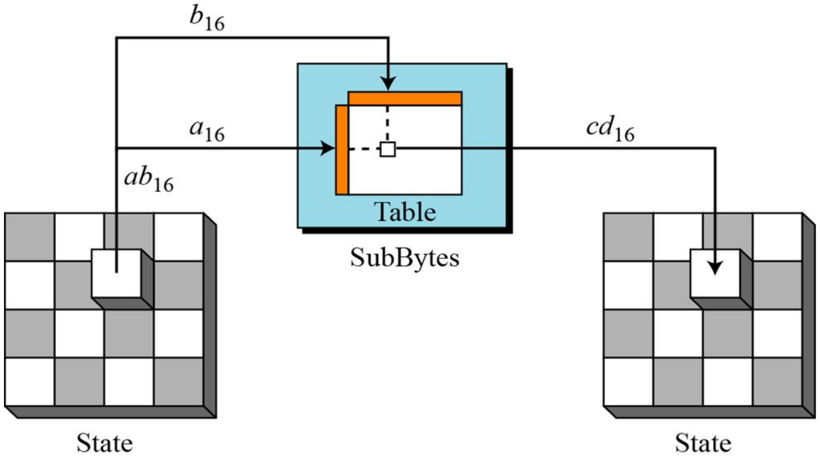
➢ Substitution is done for each byte

➢ Only one table (S-box) is used for transformation bytes.

SubBytes-

The first transformation, SubBytes, is used at the encryption site. To substitute a byte, we interpret the byte as two hexadecimal digits.





SUBBYTES TRANSFORMATION TABLE

Table 7.1 *SubBytes transformation table*

Table 7.1 *SubBytes transformation table (continued)*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	CB	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

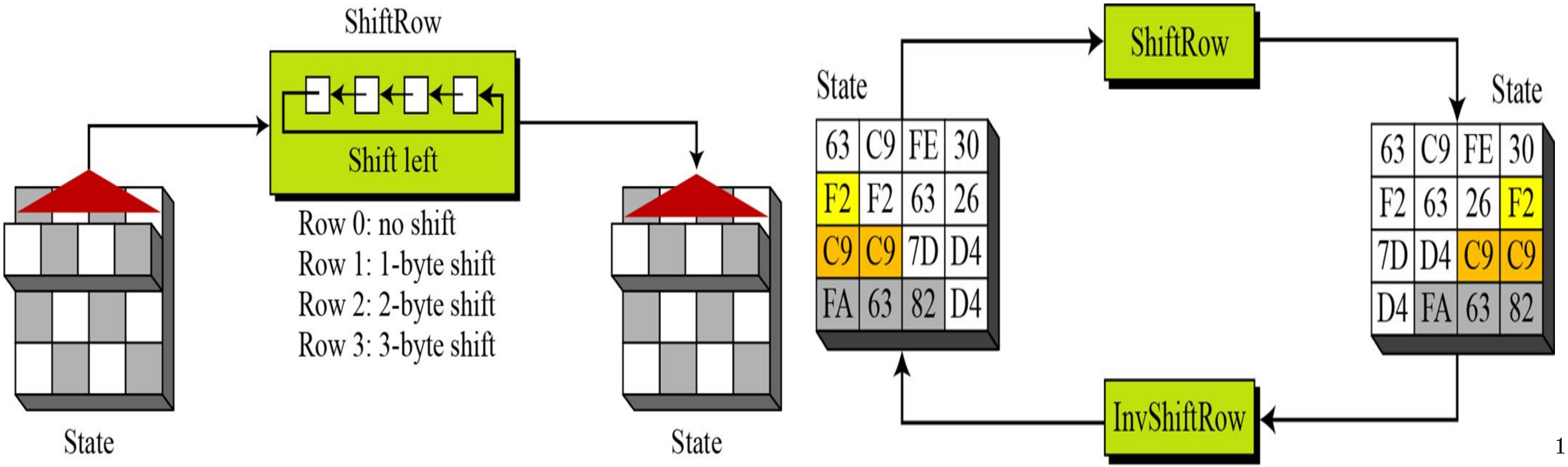
```
module subbytes(clk,data,s_data_out);
        input logic clk;
        input logic [127:0]data;
        output  logic [127:0]s_data_out;

       Logic  [127:0] tmp_out;

    sbox q0(data[127:120],tmp_out[127:120] );
    sbox q1( data[119:112],tmp_out[119:112] );
    sbox q2( data[111:104],tmp_out[111:104] );
    sbox q3( data[103:96],tmp_out[103:96] );

    sbox q4( data[95:88],tmp_out[95:88] );
    sbox q5( data[87:80],tmp_out[87:80] );
    sbox q6( data[79:72],tmp_out[79:72] );
    sbox q7( data[71:64],tmp_out[71:64] );

    sbox q8( data[63:56],tmp_out[63:56] );
    sbox q9( data[55:48],tmp_out[55:48] );
    sbox q10(data[47:40],tmp_out[47:40] );
    sbox q11(data[39:32],tmp_out[39:32] );

    sbox q12(data[31:24],tmp_out[31:24] );
    sbox q13(data[23:16],tmp_out[23:16] );
    sbox q14(data[15:8],tmp_out[15:8] );
    sbox q15(data[7:0],tmp_out[7:0] );

        always@(posedge clk)
        begin

         s_data_out<=tmp_out;
        end

endmodule
```

PERMUTATION

➢ Another transformation found in a round is shifting, which permutes the bytes.

SHIFTROWS

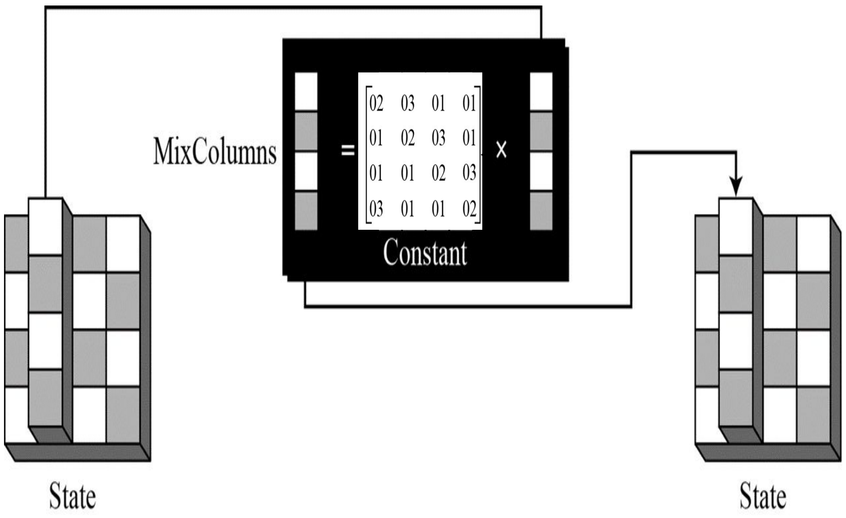
In the encryption, the transformation is called ShiftRows.

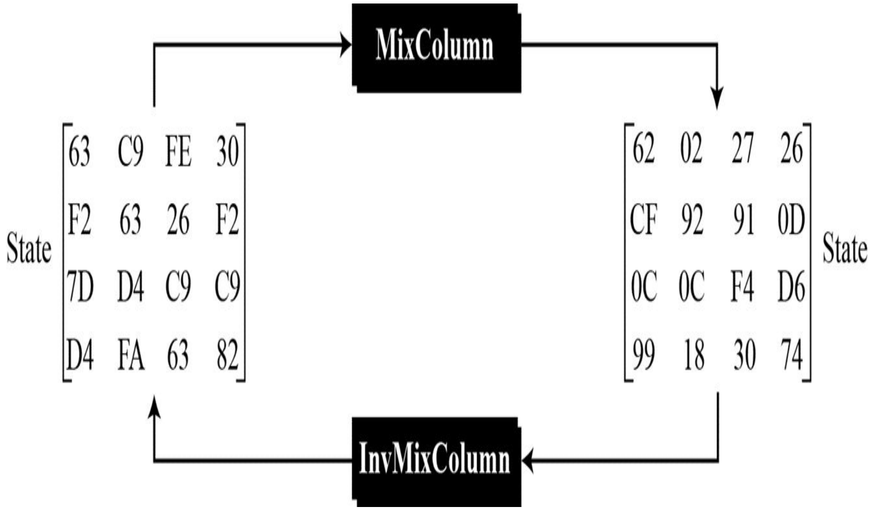


```
module shiftrows(clk,data_in,data_out
  );
        input clk;
        input [127:0]data_in;
        output reg [127:0]data_out;

        always@(posedge clk)
        begin
        data_out[127:120]<=data_in[95:88];
        data_out[119:112]<=data_in[55:48];
        data_out[111:104]<=data_in[15:8];
        data_out[103:96]<= data_in[103:96];


        data_out[95:88]<=data_in[63:56];
        data_out[87:80]<=data_in[23:16];
        data_out[79:72]<=data_in[111:104];
        data_out[71:64]<=data_in[71:64];

 data_out[63:56]<=data_in[31:24];
        data_out[55:48]<=data_in[119:112];
        data_out[47:40]<=data_in[79:72];
        data_out[39:32]<=data_in[39:32];

        data_out[31:24]<=data_in[127:120];
        data_out[23:16]<=data_in[87:80];
        data_out[15:8]<= data_in[47:40];
        data_out[7:0]<=data_in[7:0];
        end
endmodule
```

MIXCOLUMNS

➢ The MixColumns transformation operates at the column level; it transforms each column of the state to a new column.





```
module mixcolumn(clk,data_in,data_out);
input logic clk;
input logic [127:0] data_in;
output  logic [127:0] data_out;

logic  [31:0] n1,n2,n3,n4;
logic  [31:0] n_tmp_out1, n_tmp_out2, n_tmp_out3,
n_tmp_out4;

assign n1 = data_in[127:96];
assign n2=data_in[95:64];
assign n3=data_in[63:32];
assign n4=data_in[31:0];

mul_32 m1 (clk,n1,n_tmp_out1);
mul_32 m2 (clk,n2,n_tmp_out2);
mul_32 m3 (clk,n3,n_tmp_out3);
mul_32 m4 (clk,n4,n_tmp_out4);

assign
data_out={n_tmp_out1,n_tmp_out2,n_tmp_out3,n_tmp_
out4};

endmodule
```

```
//multiplication by 2 in the Galois field GF(2^8),
// which is a key operation in the AES encryption
//algorithm's MixColumns step.
module mul_2(clk,data_in,data_out);
Input logic[7:0] data_in;
input logic clk;
output logic reg [7:0]data_out;

always@(posedge clk)
data_out<={data_in[6:0],1'b0} ^ (8'h1b & {8{data_in[7]}});
Endmodule\

module mul_3(clk,data_in,data_out);
input logic clk;
input logic [7:0]data_in;
Output logic  [7:0] data_out;
logic [7:0]tmp_out;

mul_2  m1(clk,data_in,tmp_out);
assign  data_out=tmp_out^data_in;
endmodule
```

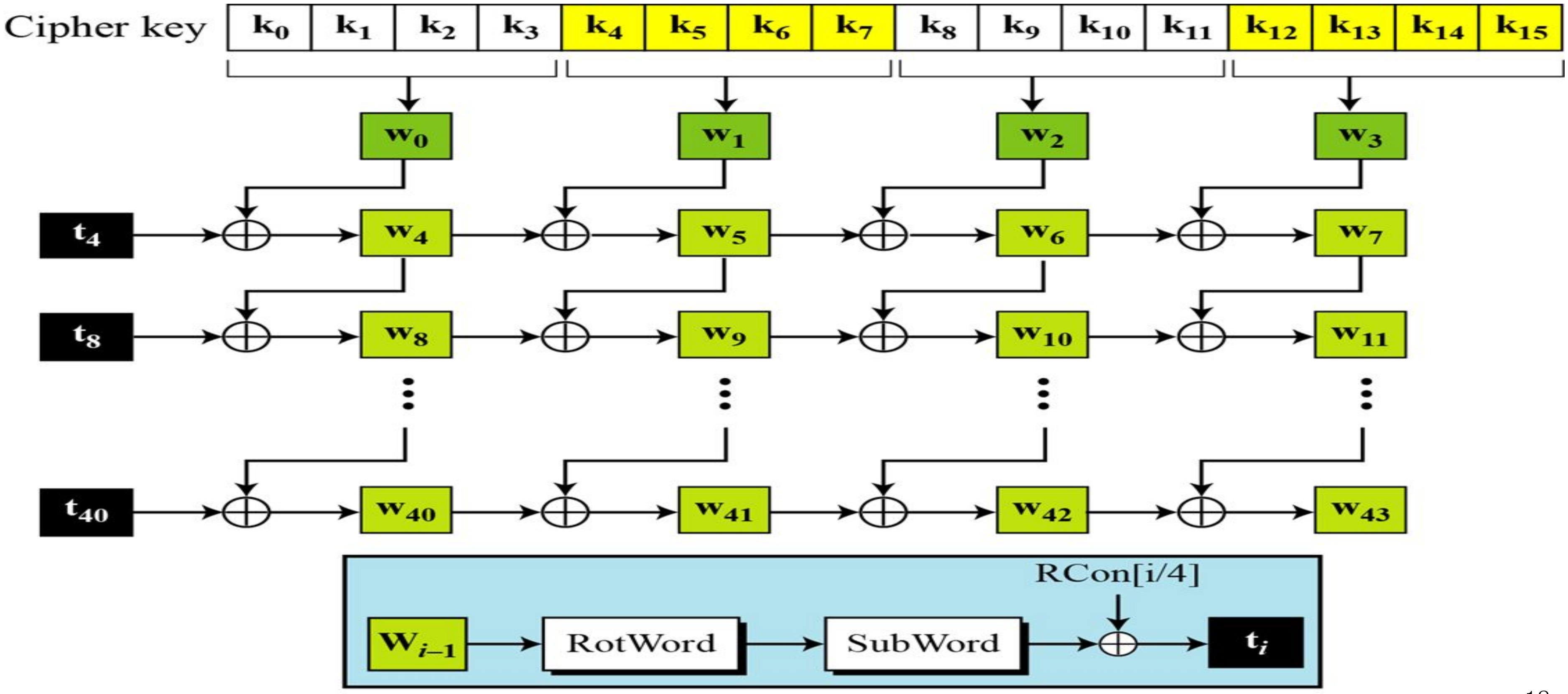
KEY EXPANSION

To create round keys for each round, AES uses a key-expansion process. If the number of rounds is N_r , the key-expansion routine creates $N_r + 1$ 128-bit round keys from one single 128-bit cipher key.

Table 7.3 *Words for each round*

Round	*Words*			
Pre-round	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$
1	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$	$\mathbf{w}_7$
2	$\mathbf{w}_8$	$\mathbf{w}_9$	$\mathbf{w}_{10}$	$\mathbf{w}_{11}$
. . .	. . .			
N_r	$\mathbf{w}_{4N_r}$	$\mathbf{w}_{4N_r+1}$	$\mathbf{w}_{4N_r+2}$	$\mathbf{w}_{4N_r+3}$

KEY EXPANSION

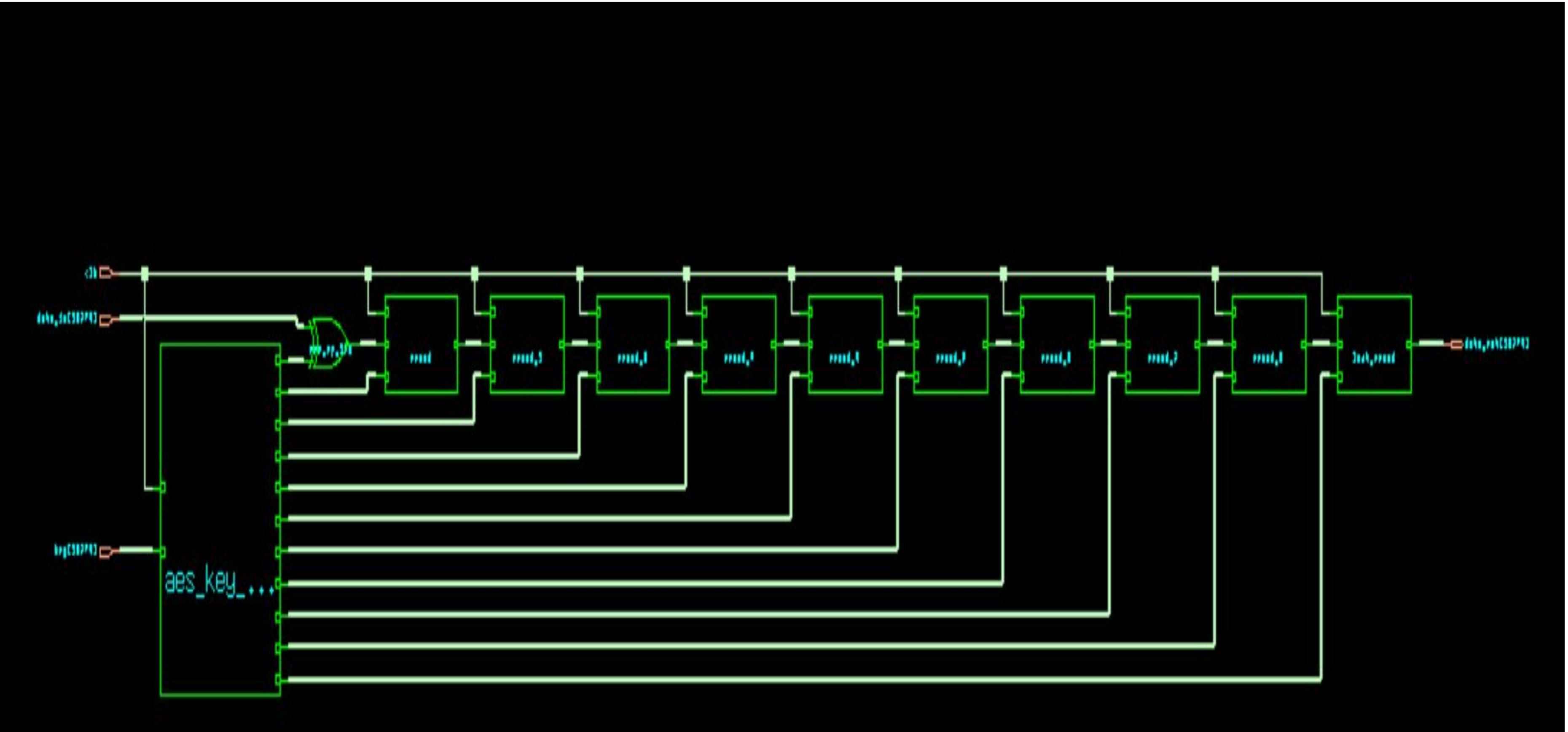


Making of t_i (temporary) words $i = 4\ N_r$.

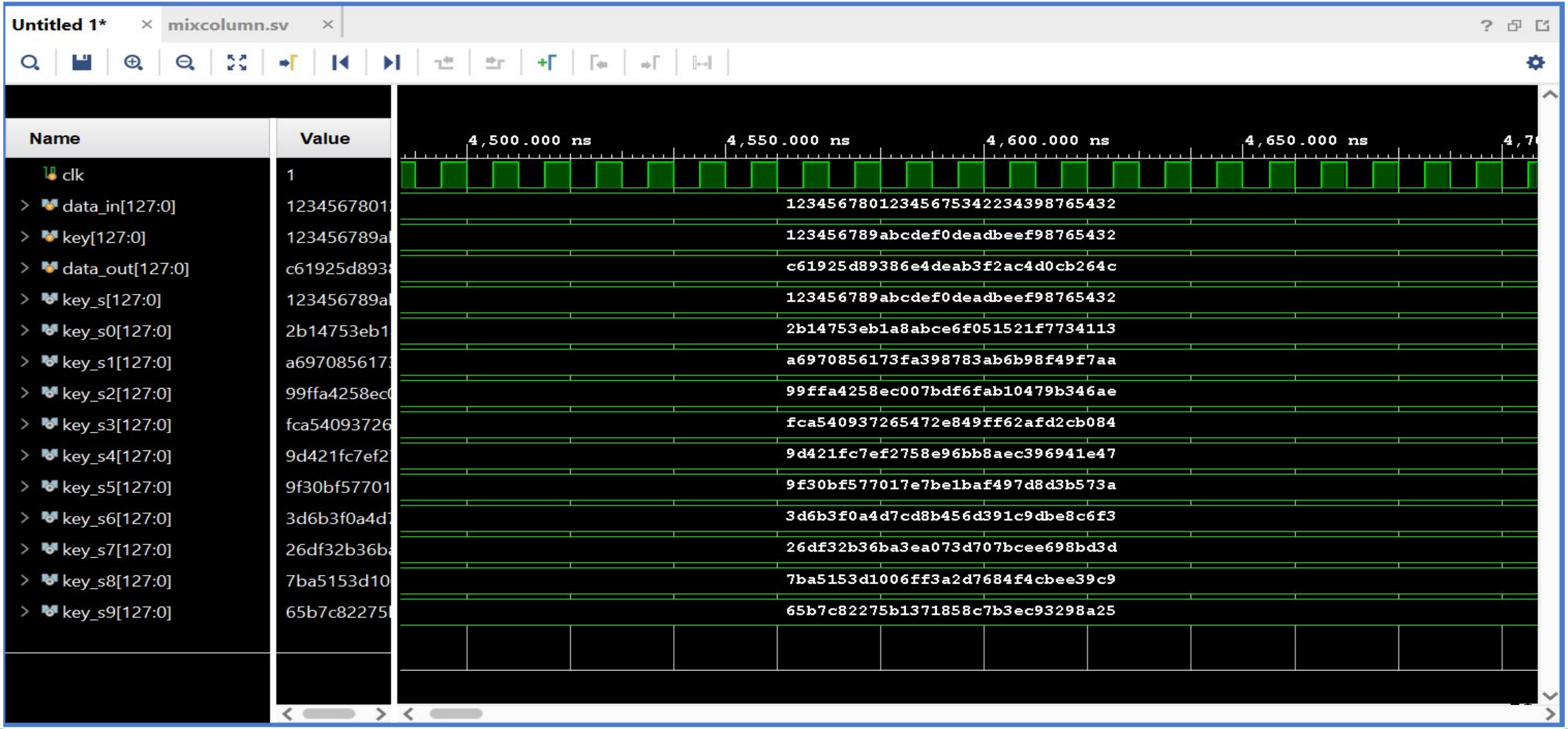
```
module aes_key_expand_128( clk,key,
key_s0,key_s1,key_s2,key_s3,key_s4,key_s5,key_s6,key_s7,
key_s8,key_s9,key_s10);
input logic   [127:0]        key;
input logic clk;
output logic [127:0]
key_s0,key_s1,key_s2,key_s3,key_s4,key_s5,key_s6,key_s7,
key_s8,key_s9,key_s10;

logic  [31:0] w0,w1,w2,w3, w4, w5, w6, w7, w8, w9, w10,
w11, w12, w13, w14, w15, w16, w17,w18, w19, w20, w21,
w22, w23, w24, w25, w26, w27, w28, w29, w30, w31, w32,
w33,w34, w35, w36, w37, w38, w39, w40, w41, w42, w43;
wire   [31:0] subword,
subword2,subword3,subword4,subword5, subword6,
subword7,subword8,subword9,subword10;
wire   [7:0]   rcon, rcon2,rcon3,rcon4,rcon5, rcon6,
rcon7,rcon8,rcon9,rcon10;
```

```
always @(posedge clk)
begin

w0 =  key[127:096];
w1 =  key[095:064];
w2 =  key[063:032];
w3 =  key[031:000];

w4 =  key[127:096]^subword^{8'h01,24'b0};
w5 =
key[095:064]^key[127:096]^subword^{8'h01,24'b0};
w6 =
key[063:032]^key[095:064]^key[127:096]^subword^{8'h0
1,24'b0};
w7 =
key[127:096]^key[095:064]^key[063:032]^key[031:000]^
subword^{8'h01,24'b0};

w8  =  w4^subword2^{rcon2,24'b0};
w9  =  w5^w4^subword2^{rcon2,24'b0};
w10 =  w6^w5^w4^subword2^{rcon2,24'b0};
w11 =  w7^w6^w5^w4^subword2^{rcon2,24'b0};
```

ELABORATED DESIGN



ENCRYPTED DATA WAVEFORM



SYNTHESIS DESIGN USING GENUS

AES ENCRYPTION DESIGN	PARAMETER
MAXIMUM FREQUENCY	1.39GHZ
CRITICAL PATH	Start-point : r4_a2_data_out_reg[110]/CK End-point : r5_a1_s_data_out_reg[100]/D
DELAY	715ps
MINIMUM AREA UTILIZATION	327836.520

POWER COMPONENT USING GENUS

```
Instance: /aes_main
Power Unit: W
PDB Frames: /stim#0/frame#0
```

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	3.74607e-06	8.05550e-02	1.28693e-02	9.34280e-02	16.85%
latch	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
logic	2.89024e-05	2.09520e-01	2.51353e-01	4.60902e-01	83.15%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	3.26485e-05	2.90075e-01	2.64222e-01	5.54330e-01	100.00%
Percentage	0.01%	52.33%	47.67%	100.00%	100.00%

NCSIM POST SYNTHESIS SIMULATION

DELAY-300PS

- It has been observed that there is lot of delay in the out change with respect to change in input.
- We can do pipelining to reduce the delay.

THANK YOU!