گزارش کار آزمایش 5 مدار منطقی

سروین نامی 9931103
آبتین زندی 9931071

سوال 1:
در این بخش باید سه مدار ابتدای آزمایش را با استفاده از زبان وریلاگ رسم کنیم و سپس نمودار های زمانی هر کدام از کدها را تحلیل می کنیم :

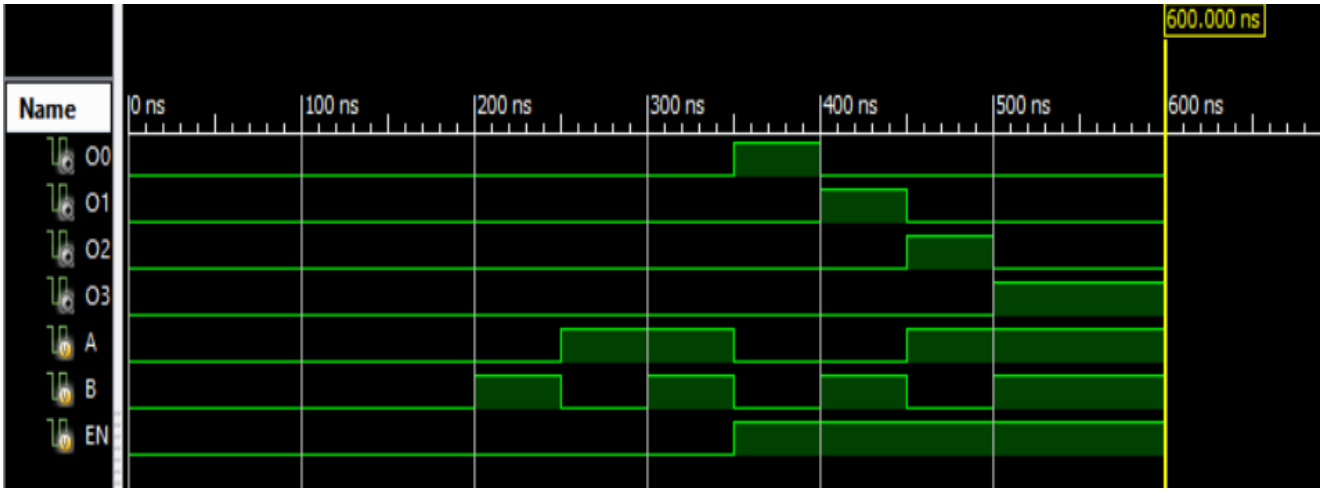
Decoder 2X4 Verilog code :

```
module Decoder_2X4(
    input A,
    input B,
    input EN,
    output O0,
    output O1,
    output O2,
    output O3
    );

assign O0 = (~A) & (~B) & EN;
assign O1 = (~A) & B & EN;
assign O2 = A & (~B) & EN;
assign O3 = A & B & EN;


endmodule
```

Decoder 2X4 Verilog Test Bench :

```
    initial begin
        // Initialize Inputs
        A = 0;
        B = 0;
        EN = 0;

        // Wait 100 ns for global reset to finish
        #100;
        #50 A=0; B=0; EN=0;
        #50 A=0; B=1; EN=0;
        #50 A=1; B=0; EN=0;
        #50 A=1; B=1; EN=0;

        #50 A=0; B=0; EN=1;
        #50 A=0; B=1; EN=1;
        #50 A=1; B=0; EN=1;
        #50 A=1; B=1; EN=1;

        #100;

        $finish;
        // Add stimulus here

    end

endmodule
```

Decoder 2X4 Verilog Timeline :



Priority_Encoder_4X2 Verilog code:

```
module Priority_Encoder_4X2(
    input D0,
    input D1,
    input D2,
    input D3,
    output O0,
    output O1
    );

assign O0 = ((~D2) & D1) + D3;
assign O1 = D2 + D3;

endmodule
```

Priority_Encoder_4X2 Verilog Test Bench :

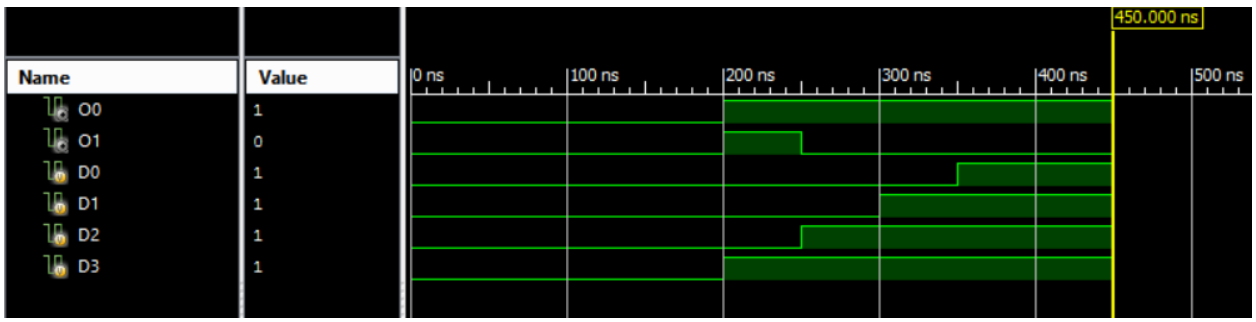
TABLE IV
LOGIC TRUTH TABLE OF 4X2 PRIORITY ENCODER

Input				Output	
I3	I2	I1	I0	Y1	Y0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

در جدول ارزش گزارش های آمده در این گزارش ما می توانیم مشاهده کنیم که در چند خانه X را داریم یعنی در تست بنچ ها می توانیم آنها را یک یا صفر بگذاریم :

```
initial begin
    // Initialize Inputs
    D0 = 0;
    D1 = 0;
    D2 = 0;
    D3 = 0;

    // Wait 100 ns for global reset to finish
    #100;
    #50 D0=0; D1=0; D2=0; D3=0;
    #50 D0=0; D1=0; D2=0; D3=1;
    #50 D0=0; D1=0; D2=1; D3=1;
    #50 D0=0; D1=1; D2=1; D3=1;
    #50 D0=1; D1=1; D2=1; D3=1;
    #100;
    $finish;
    // Add stimulus here

end
```

Priority_Encoder_4X2 Verilog Timeline:



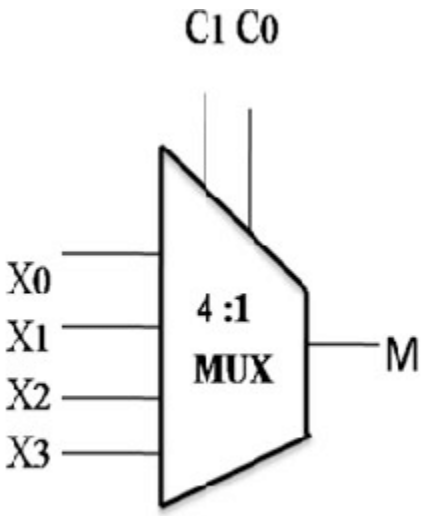
Multiplexer 4X1 Verilog code:

```
module MULTIPLEXER_4X1(
    input W0,
    input W1,
    input W2,
    input W3,
    input S1,
    input S2,
    output MuxOutput
    );

assign MuxOutput = S2 ? (S1 ? W3 : W2) : (S1 ? W1 : W0);

endmodule
```

Multiplexer 4X1 Verilog Test Bench:

```
    initial begin
        // Initialize Inputs
        W0 = 0;
        W1 = 0;
        W2 = 0;
        W3 = 0;
        S1 = 0;
        S2 = 0;

        // Wait 100 ns for global reset to finish
        #100;
        #50 W0=1; W1=0; W2=0; W3=0; S1=0; S2=0;
        #50 W0=1; W1=0; W2=1; W3=1; S1=1; S2=0;
        #50 W0=0; W1=0; W2=1; W3=0; S1=0; S2=1;
        #50 W0=0; W1=0; W2=0; W3=1; S1=1; S2=1;
        #100;
        $finish;
        // Add stimulus here

    end

endmodule
```

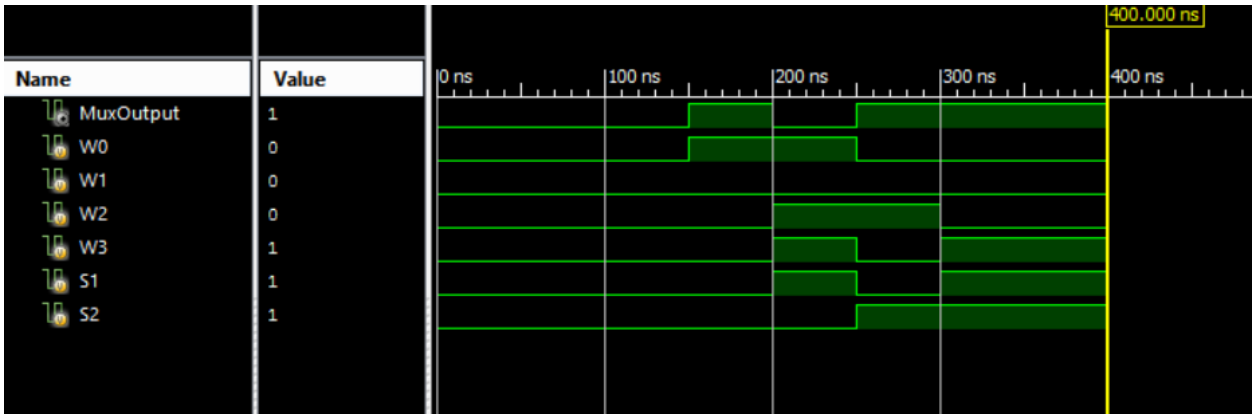
Multiplexer 4X1 Verilog Truth Table:



C1	C0	M
0	0	X0
0	1	X1
1	0	X2
1	1	X3

(b)

Multiplexer 4X1 Verilog Timeline:



تبدیل Decoder 2X4 به Decoder 4X16 :

با استفاده از چهار لیترال می توانیم تابع مذکور را پیاده سازی کنیم :

Decoder 2X4 to Decoder 4X16 Verilog code :

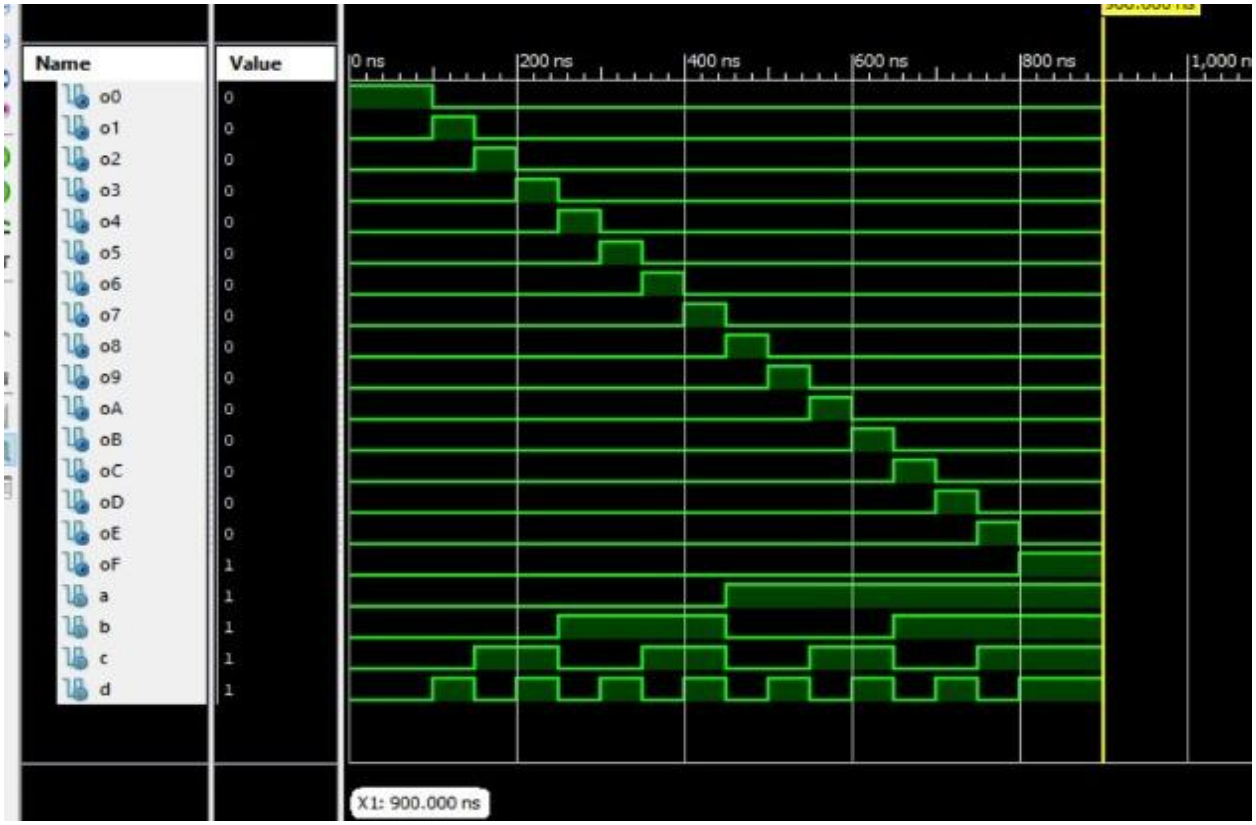
```
36      output oA,
37      output oB,
38      output oC,
39      output oD,
40      output oE,
41      output oF
42      );
43  assign o0 = (~a)&(~b)&(~c)&(~d);
44  assign o1 = (~a)&(~b)&(~c)& d ;
45  assign o2 = (~a)&(~b)& c &(~d);
46  assign o3 = (~a)&(~b)& c & d ;
47  assign o4 = (~a)& b &(~c)&(~d);
48  assign o5 = (~a)& b &(~c)& d ;
49  assign o6 = (~a)& b & c &(~d);
50  assign o7 = (~a)& b & c & d ;
51  assign o8 =  a &(~b)&(~c)&(~d);
52  assign o9 =  a &(~b)&(~c)& d ;
53  assign oA =  a &(~b)& c &(~d);
54  assign oB =  a &(~b)& c & d ;
55  assign oC =  a & b &(~c)&(~d);
56  assign oD =  a & b &(~c)& d ;
57  assign oE =  a & b & c &(~d);
58  assign oF =  a & b & c & d ;
59  endmodule
60
```

تست بنچ ها باید دقیقا متناظر با ورودی هر کدام از خروجی های ما باشند :

Decoder 2X4 to Decoder 4X16 Verilog Test Bench:

```
78        b = 0;
79        c = 0;
80        d = 0;
81        #50 a=0;b=0;c=0;d=0;
82        #50 a=0;b=0;c=0;d=1;
83        #50 a=0;b=0;c=1;d=0;
84        #50 a=0;b=0;c=1;d=1;
85        #50 a=0;b=1;c=0;d=0;
86        #50 a=0;b=1;c=0;d=1;
87        #50 a=0;b=1;c=1;d=0;
88        #50 a=0;b=1;c=1;d=1;
89        #50 a=1;b=0;c=0;d=0;
90        #50 a=1;b=0;c=0;d=1;
91        #50 a=1;b=0;c=1;d=0;
92        #50 a=1;b=0;c=1;d=1;
93        #50 a=1;b=1;c=0;d=0;
94        #50 a=1;b=1;c=0;d=1;
95        #50 a=1;b=1;c=1;d=0;
96        #50 a=1;b=1;c=1;d=1;
97
98        // Wait 100 ns for global reset to finish
99        #100;
00          $finish;
01        // Add stimulus here
```

Decoder 2X4 to Decoder 4X16 Verilog Timeline:



تابع متناظر ساخته شده مبتنی بر Decoder 4X16 :

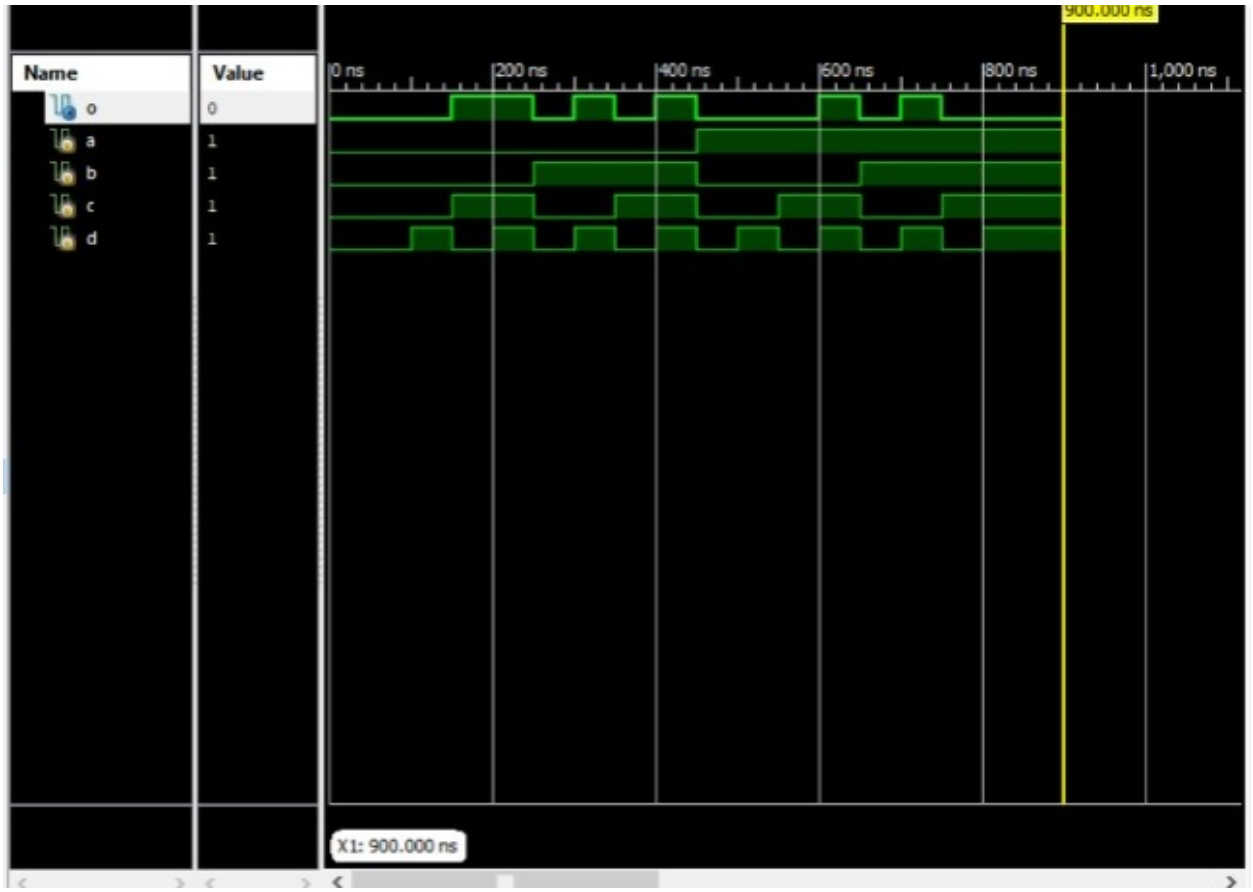
Function Verilog code :

```
//
// Create Date:    18:29:09 11/08/2021
// Design Name:
// Module Name:    F
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module F(
    input a,
    input b,
    input c,
    input d,
    output o
    );

   assign o = ((~a)&(~b)& c &(~d))|((~a)&(~b)& c & d)|((~a)& b &(~c)& d)|((~a)& b & c & d)|(a &(~b)& c & d)|(a & b &(~c)& d);

endmodule
```

Function Verilog Test Bench:

```
        // Initialize Inputs
        a = 0;
        b = 0;
        c = 0;
        d = 0;
        #50 a=0;b=0;c=0;d=0;
        #50 a=0;b=0;c=0;d=1;
        #50 a=0;b=0;c=1;d=0;
        #50 a=0;b=0;c=1;d=1;
        #50 a=0;b=1;c=0;d=0;
        #50 a=0;b=1;c=0;d=1;
        #50 a=0;b=1;c=1;d=0;
        #50 a=0;b=1;c=1;d=1;
        #50 a=1;b=0;c=0;d=0;
        #50 a=1;b=0;c=0;d=1;
        #50 a=1;b=0;c=1;d=0;
        #50 a=1;b=0;c=1;d=1;
        #50 a=1;b=1;c=0;d=0;
        #50 a=1;b=1;c=0;d=1;
        #50 a=1;b=1;c=1;d=0;
        #50 a=1;b=1;c=1;d=1;

        // Wait 100 ns for global reset to finish
        #100;
          $finish;
```

Function Verilog Timeline:



$F(A,B,C,D) = \sum m(2,3,5,7,11,13) = 0[2]+0[3]+0[5]+0[7]+0[11]+0[13]$