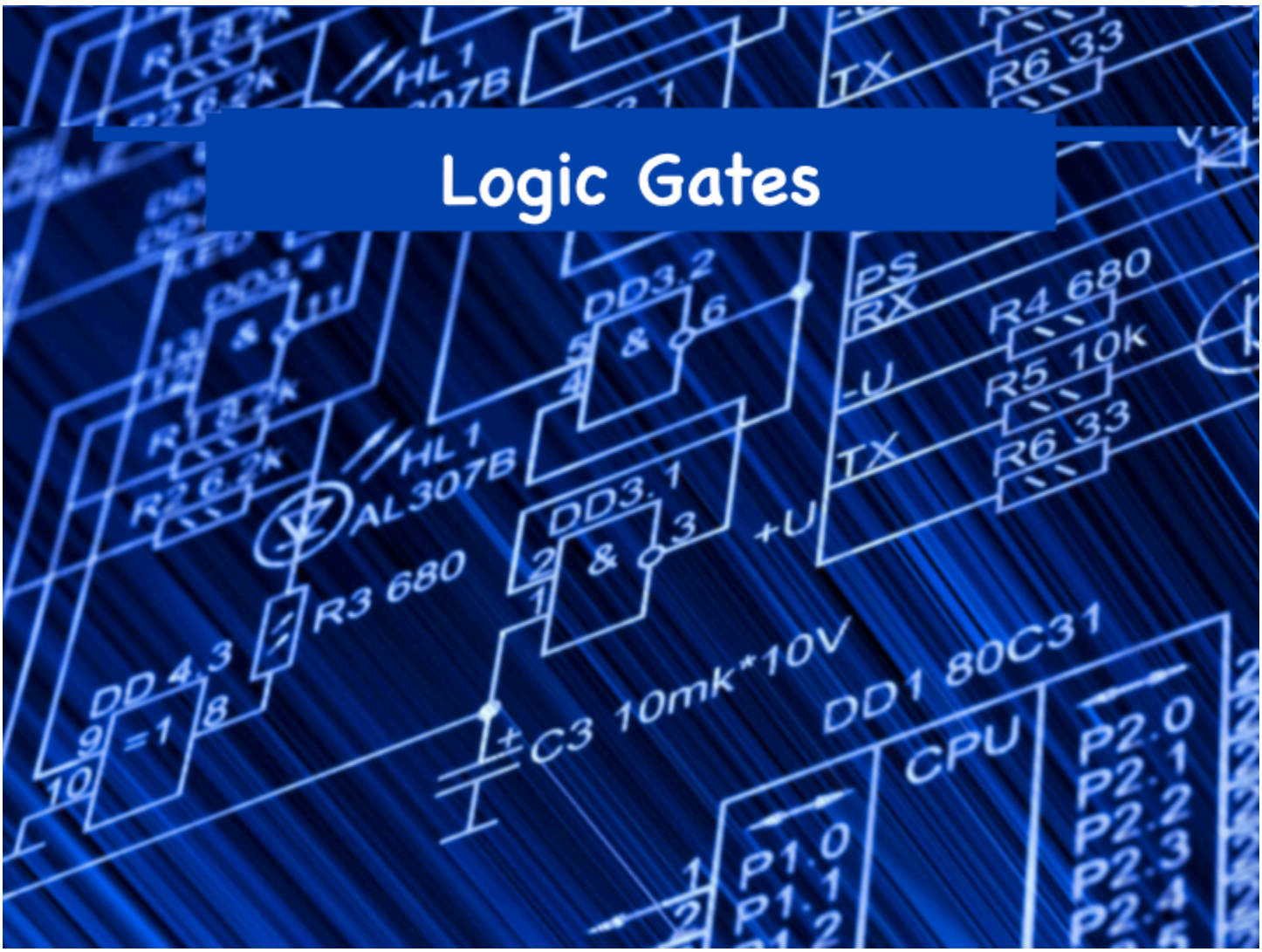
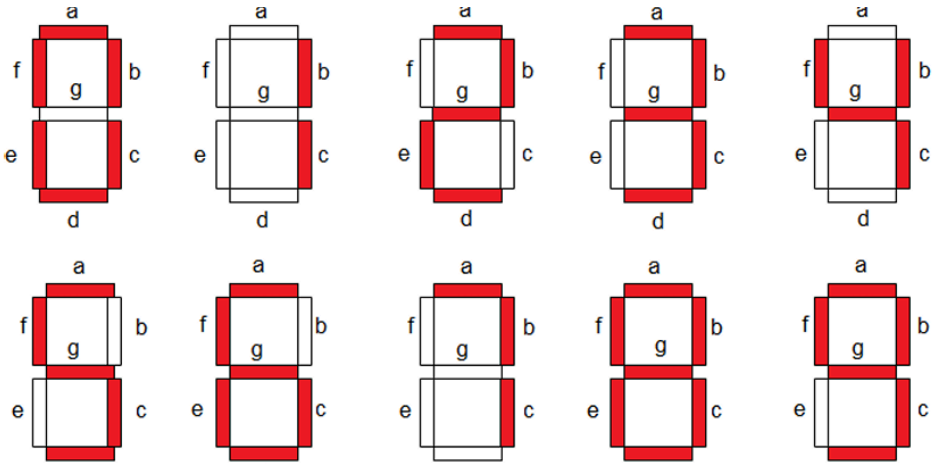
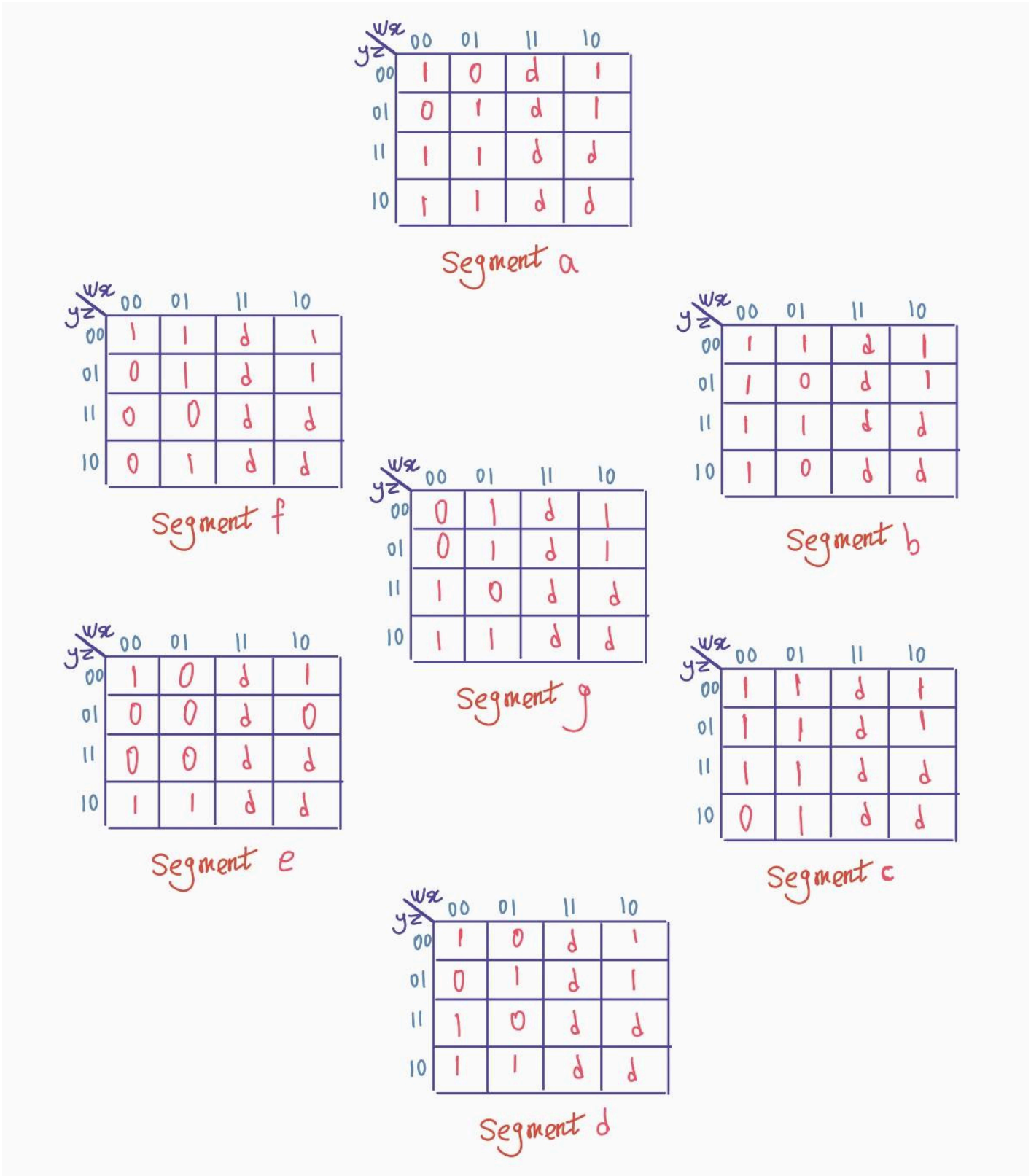
پروژه کامپیوتری اول درس مدار منطقی

محمد امین رشید 810102454

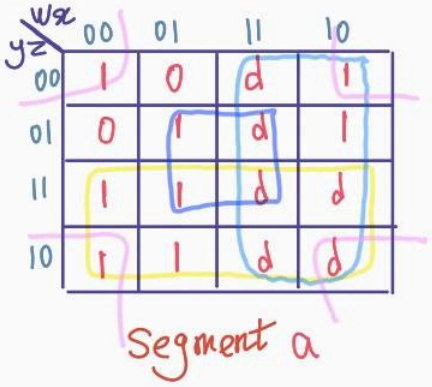


الف) در این بخش ابتدا جدول های کارنو مپ را برای هر خروجی رسم کردم(مجموعا 7 تا) و با توجه به عکس زیر ،مشخص کردم که هر یک از خروجی ها به ازای چه مقادیری روشن(1) و به ازای چه مقادیری خاموش میماند(0) و به ازای چه مقادیری don’t care است یعنی فرقی نمی کند که آن segment روشن باشد یا خاموش باشد.منظورم از مقادیر عدد ورودی است که به صورت wxyz قرار گرفته است.طبیعتا این عدد بین 0 تا 15 می باشد پس مقادیر 10 تا 15 don’t care تولید میکنند زیرا نهایت خروجی ما عدد 9 است .


a
f
b
g
e
c
d

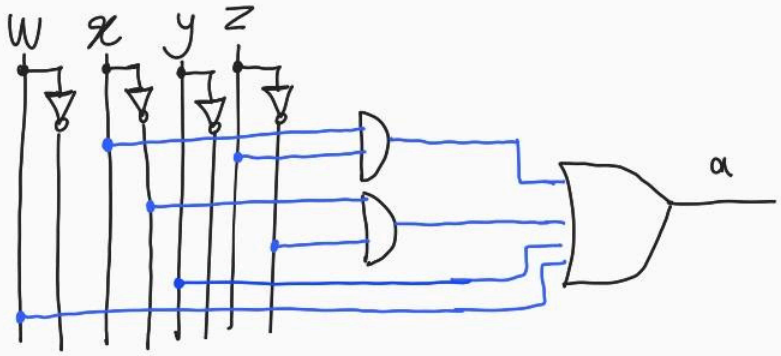


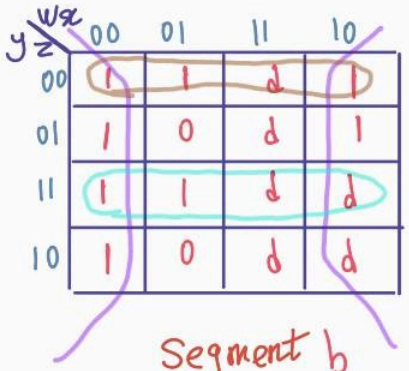
سپس با پیدا کردن min SOP هر خروجی با استفاده از کارنو مپ میتوانیم ، مدار گیت لول هر segment را به صورت مجزا رسم کنیم.برای پیدا کردن min SOP از don’t care ها هم استفاده شده تا به صورت حداکثری مدار optimize شود.



Segment a

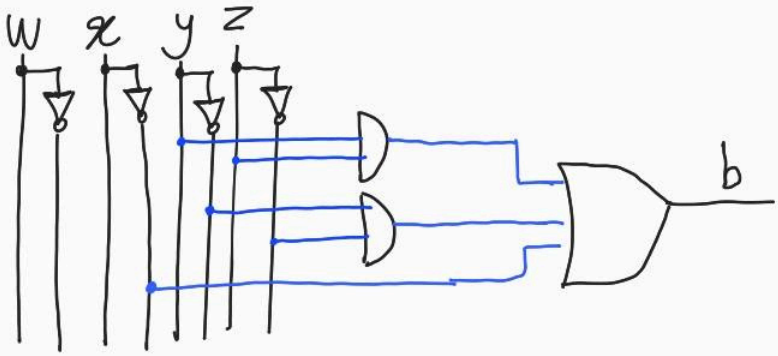
$$a = w + y + xz + \bar{x}\bar{z}$$

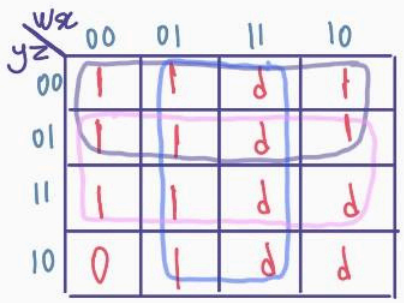




Segment b

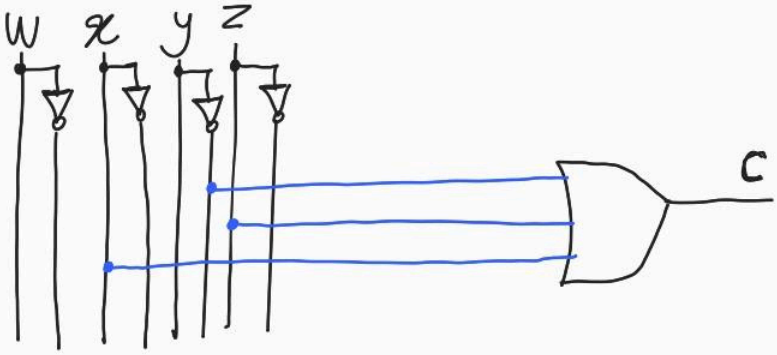
$$b = \bar{x} + yz + \bar{y}\bar{z}$$

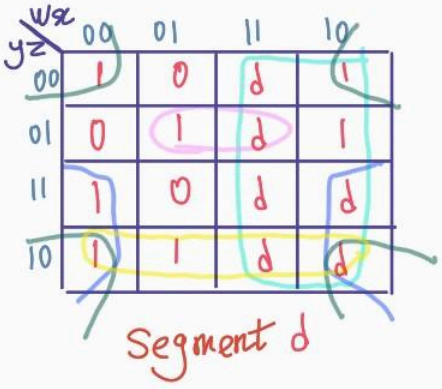




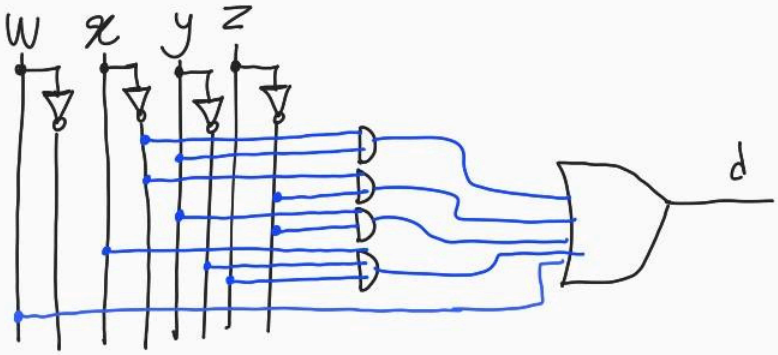
Segment c

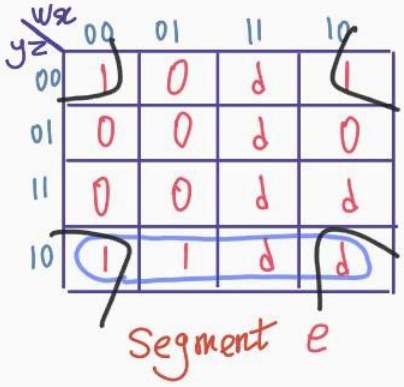
$$c = \bar{y} + z + x$$



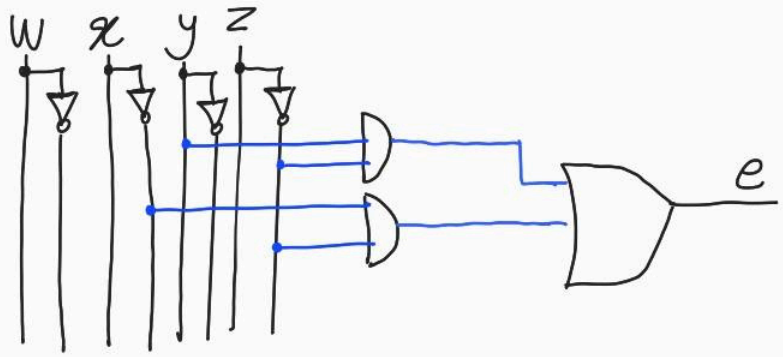


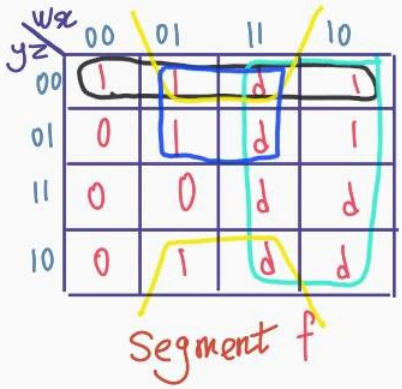
$$d = w + \bar{x}y + \bar{x}\bar{z} + y\bar{z} + x\bar{y}z$$



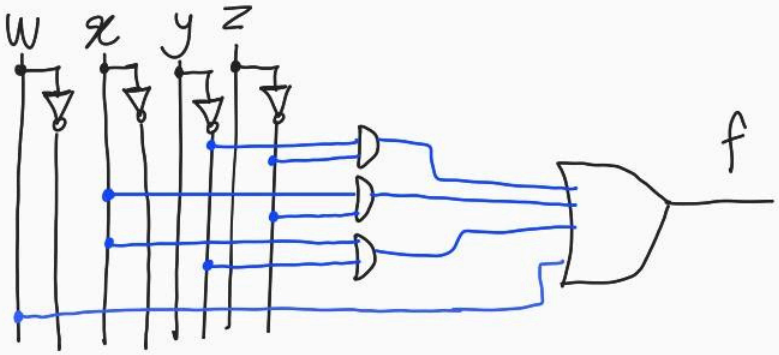


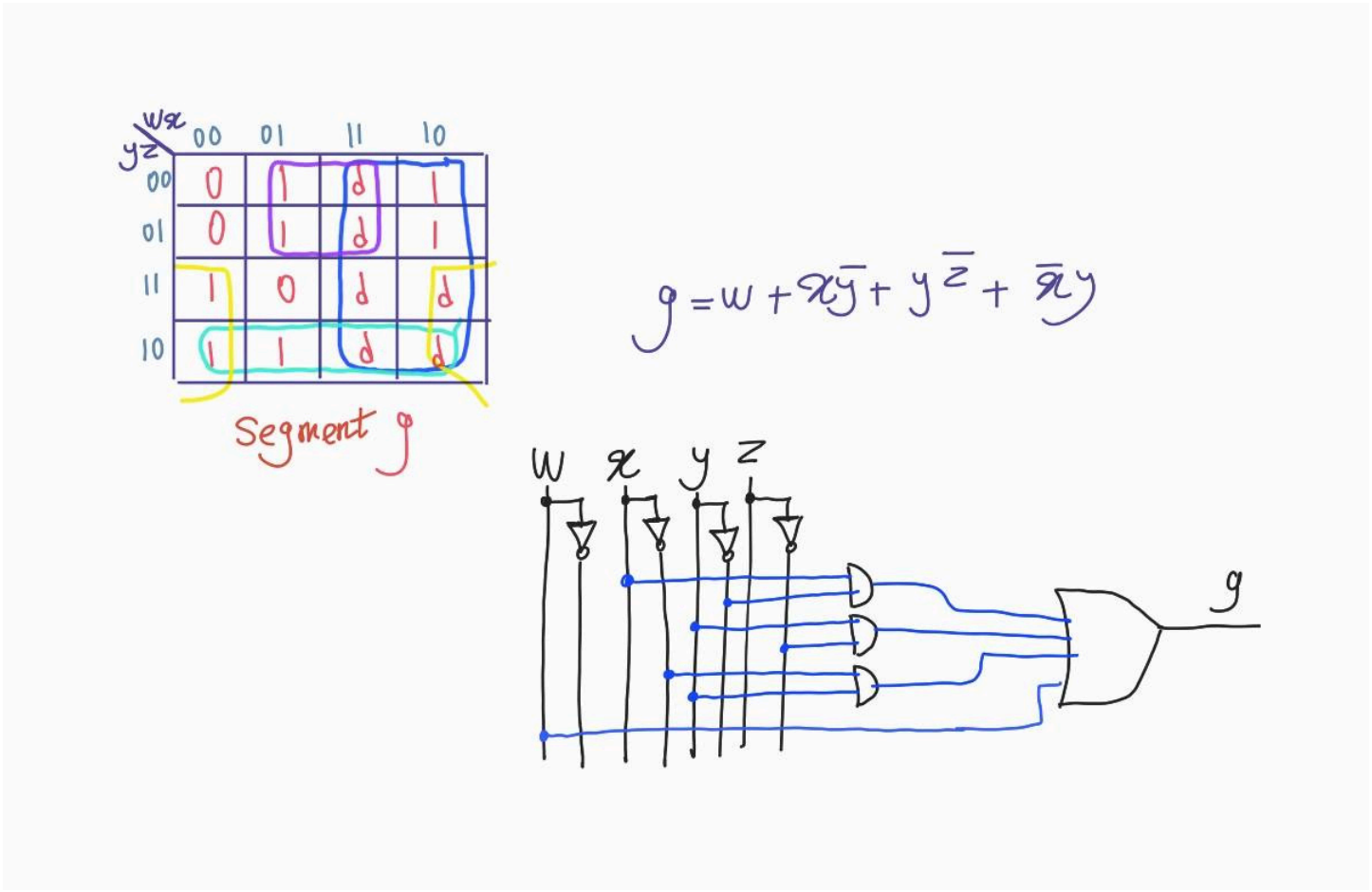
$$e = y\bar{z} + \bar{x}\bar{z}$$





$$f = w + \bar{y}\bar{z} + x\bar{z} + x\bar{y}$$



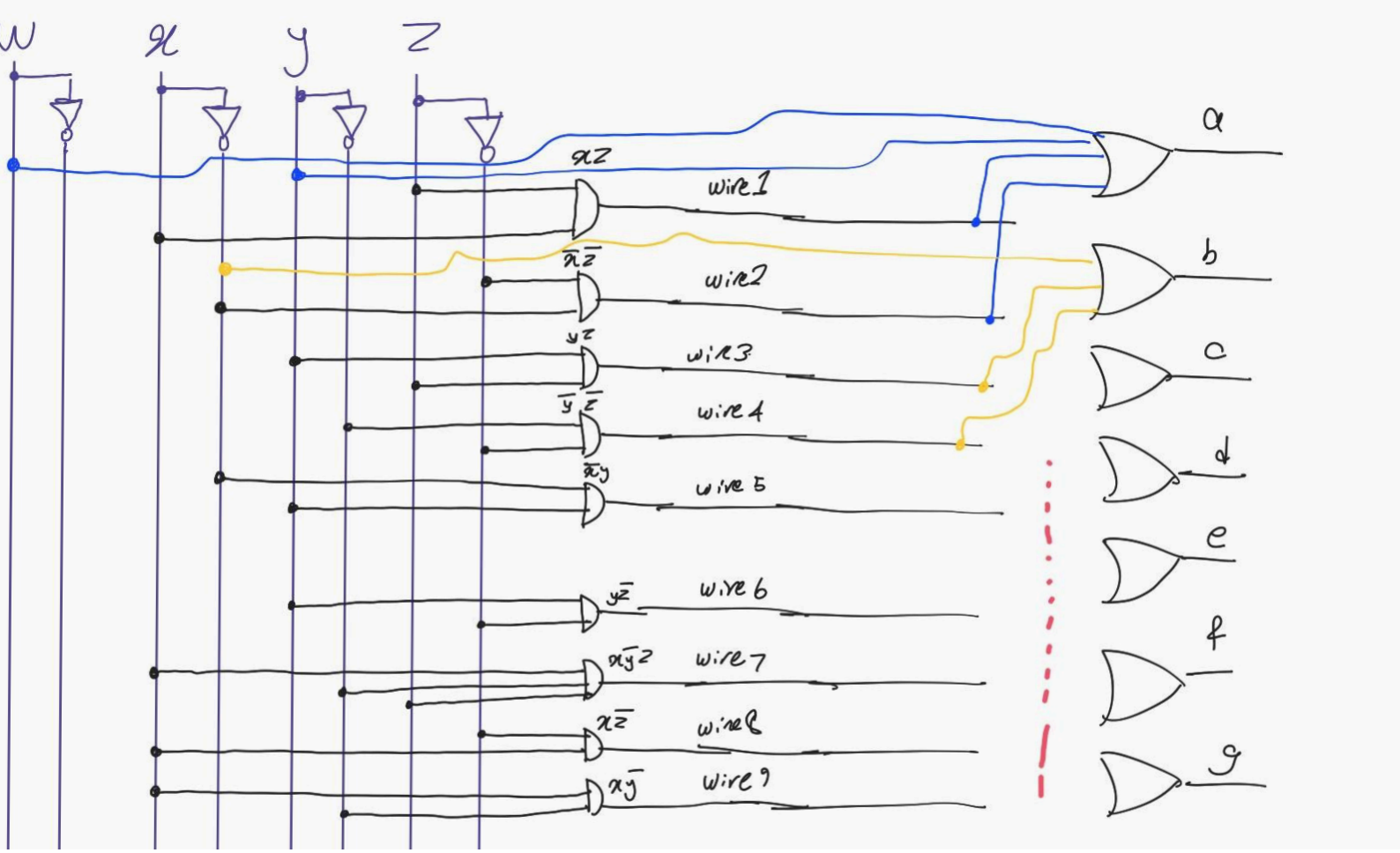


در رسم مدار ها از گیت های NOT و AND و OR استفاده شده است تا به ازای هر خروجی Min Sum of Products داشته باشیم.

ب)در این بخش با استفاده از زبان system verilog شروع به کدزنی این ماژول میکنم. ابتدا 3 تا سیم(wire) ایجاد میکنم تا خروجی not ورودی هایم باشندکه به صورت زیر میباشد:

```
`timescale 1ns/1ns
module sevenSegmentDecoder(input logic w,x,y,z , output logic [6:0] out);

    wire not_w , not_x , not_y, not_z;

    not #(2) N1 (not_w, w);
    not #(2) N2 (not_x, x);
    not #(2) N3 (not_y, y);
    not #(2) N4 (not_z, z);

endmodule
```

سپس باید با استفاده از wxyz و نقیض هر یک از این 4 تا ورودی ،ترم های لازم برای مدارمان بسازیم.برای اینکار از گیت AND استفاده میکنم .با توجه به شکل زیر که نمای کلی مدار را نشان میدهد،متوجه میشویم که به 9 تا گیت and نیاز داریم و واضح است که باید 9 تا سیم برای خروجی گیت ها تولید کنیم که اینجا از یک آرایه 9 تایی از جنس wire تشکیل میدهم.



آرایه را terms نامگذاری کردم و هر سیم را به خروجی هر یک از گیت ها متصل کردم و ورودی های هر گیت را مشخص کردم.

```
    wire [1:9] terms;

    and #(2) (terms[1],x,z);  //xz
    and #(2) (terms[2],not_z,not_x);  //~x.~z
    and #(2) (terms[3],y,z);  //yz
    and #(2) (terms[4],not_y,not_z);  //~y.~z
    and #(2) (terms[5],not_x,y);  //~x.y
    and #(2) (terms[6],y,not_z);   //y.~z
    and #(2) (terms[7],x,not_y,z);  //x.~y.z
    and #(2) (terms[8],x,not_z);  //x.~z
    and #(2) (terms[9],x,not_y); //x.~y
```

در نهایت با توجه به اینکه a,b,c,d,e,f,g با or شدن چه چیز هایی تشکیل شده اند، از 7 تا گیت or استفاده میکنیم تا خروجی آن ها مقادیر out را مشخص کند.

```
or #(3) (out[0],w,not_x,terms[1],terms[2]); //a
or #(3) (out[1],not_x,terms[3],terms[4]);  //b
or #(3) (out[2],not_y,not_z,not_x); //c
or #(3) (out[3],w,terms[5],terms[2],terms[6],terms[7]); //d
or #(3) (out[4],terms[6],terms[2]);  //e
or #(3) (out[5],w,terms[4],terms[8],terms[9]); //f
or #(3) (out[6],w,terms[9],terms[6],terms[5]); //g
```

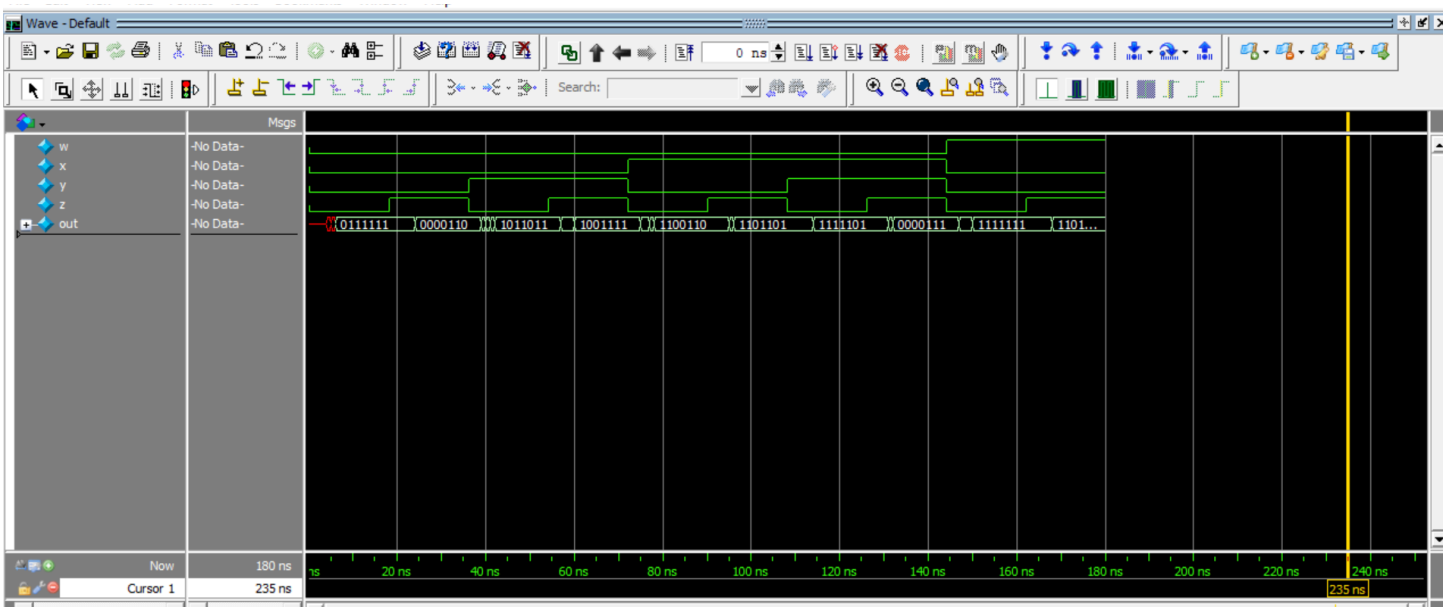
ج)با نوشتن تست بنچ مناسب همه حالات ممکن را تست میکنم.دوباره timescale را همانند ماژول قبلی روی نانوثانیه تنظیم میکنیم.سپس با ورودی ها و خروجی را با logic تعریف میکنیم.در خط ششم این کد ، این ماژول را به ماژول اصلی وصل میکینم تا از اینجا بتوانیم مقادیر آنجارا تغییر دهیم.سپس با استفاده از initial ، فرآیند اجرای کد در این بخش را به صورت ترتیبی میکنیم.در آن یک for استفاده کردم تا بتوانم هر بار با delay مناسب ، ورودی ام را تغییر دهم.حدود number از صفر تا 9 است . delay را هم برابر 18ns قرار دادم تا خروجی ها در هر مرحله تثبیت شود.

{w,x,y,z}=number

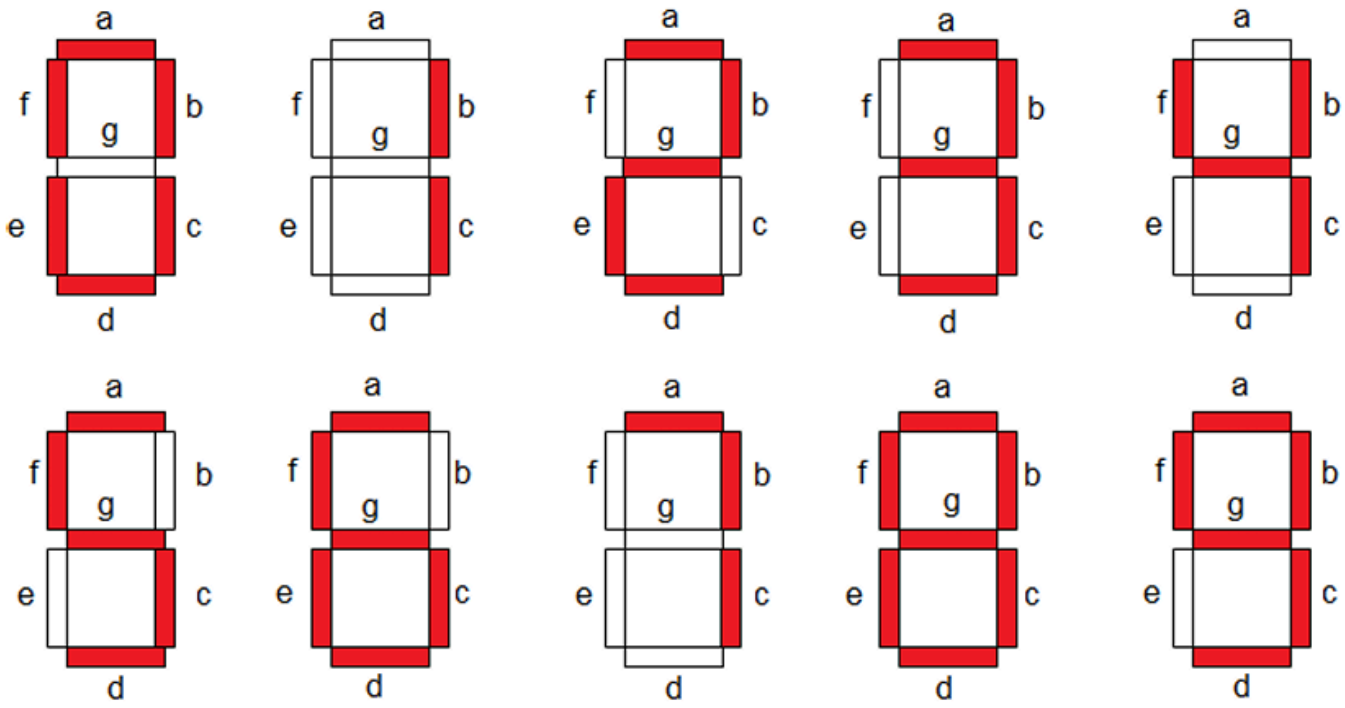
این سینتکس ، هر بیت number در حالت باینری را به ورودی هایمان وصل میکند.

```
`timescale 1ns/1ns
module sevensegmentDecoder_TB;
    logic w,x,y,z;
    logic [6:0] out;

    sevenSegmentDecoder SSD(w,x,y,z,out);

    initial begin
        for(int number=0 ; number<10 ; number+=1)begin

            {w,x,y,z}=number;
            #18;
        end
    end
endmodule
```

پس از simulate کردن و اضافه کردن wave ، نمودار زمانی قابل مشاهده می باشد.همانطور که معلوم است مقادیر 0 تا 9 به درستی خروجی داده شده است.برای مثال در ثانیه 20ns خروجی ما به شکل 0000110 می باشد و نشان میدهد وقتی عدد 1 باشد،فقط b و c یک می باشند و بقیه 0 هستند که کاملا با sevensegment مطابقت دارد.



به راحتی میتوانیم مقادیر سیگنال را با شکل زیر مقایسه کنیم. (مقادیر سیگنال از چپ به راست به صورت 1,2,3,4,5,6,7,8,9 است..

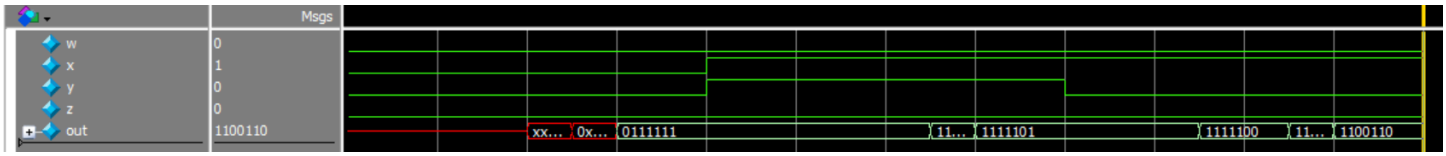


اکنون میخواهیم تست بنچ جدیدی بنویسم که 3 تا ورودی مختلف بدهم و تاخیر هارا و نحوه محاسبه آن هارا تحلیل کنم.تست بنچ جدید را به صورت زیر مینویسم:

```
1   `timescale 1ns/1ns
2   module new_sevensegmentDecoder_TB;
3       logic w=0,x=0,y=0,z=0;
4       logic [6:0] out;
5
6       sevenSegmentDecoder SSD(w,x,y,z,out);
7
8       initial begin
9
10              #8;
11              {w,x,y,z}=6;
12              #8;
13              {w,x,y,z}=4;
14              #8;
15          end
16
17  endmodule
18
```

که ابتدا رشته ورودی به صورت 0000 می باشد سپس در خط بعد از 8 نانوثانیه این رشته به صورت 0110 می شود.و بعد از 8 نانو ثانیه رشته ورودی به 0100 تغییر میکند.

خروجی timing diagram به صورت زیر می باشد:



تغییر اول :مقادیر خروجی ازxxxxxxx به 0111111 تغییر میکند و علتش هم مقدار دهی اولیه wxyz است که در خط سوم نوشته ام. که در نهایت 6 نانو ثانیه طول میکشد تا مقادیر پایدار شود به این

دلیل که critical path از یک not و از یک and و از یک or تشکیل شده است که مقدار تاخیر آن ها به ترتیب 1 و 2 و 3 است که در نهایت جمع آن ها 6 است.

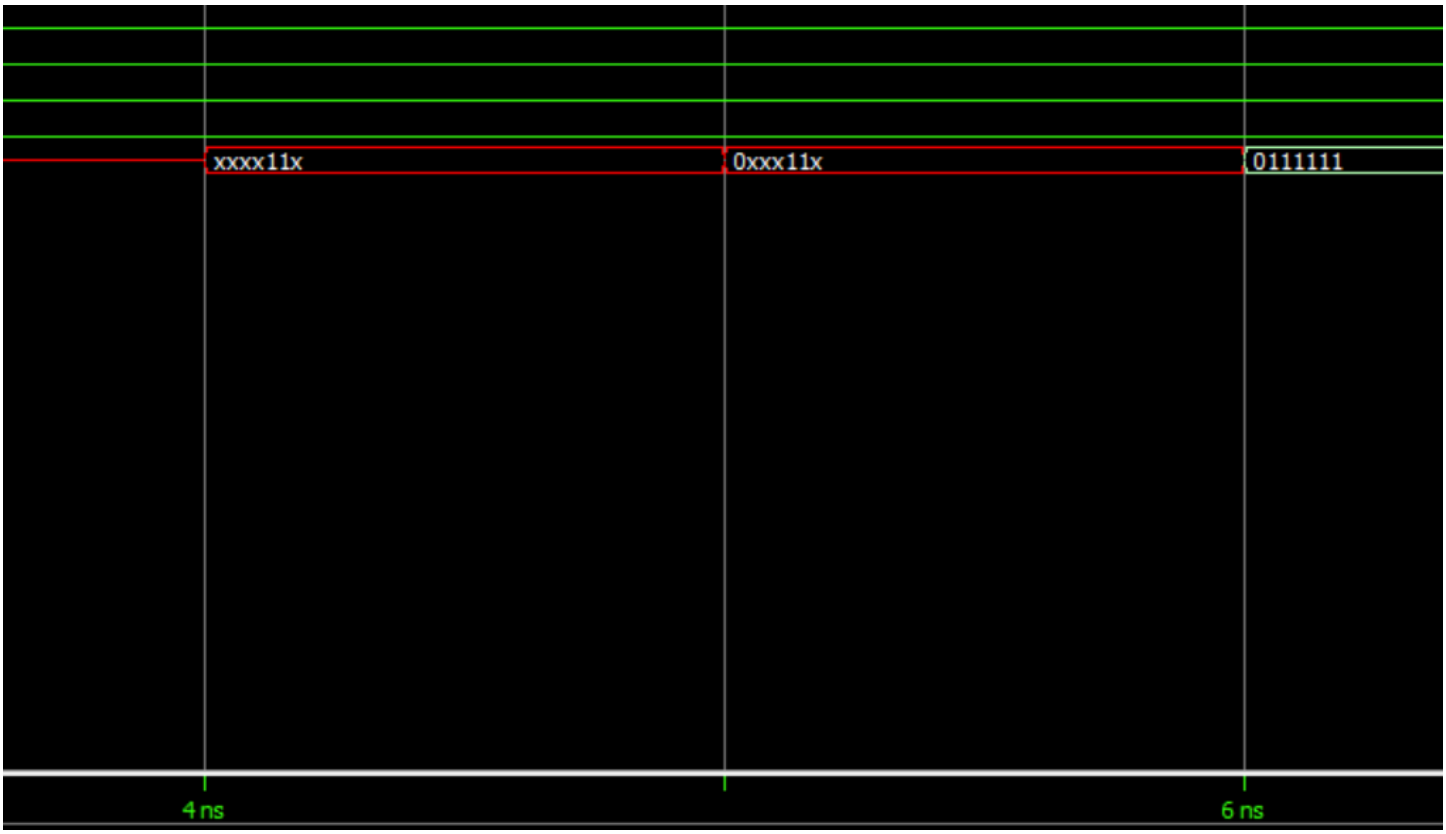
در کل میتوان گفت که مسیر های ممکن از یک متغیر تا خارج شدنش به شکل زیر است:

1. ->not-> and->or (critical path) (6ns)
2. ->and->or (5ns)
3. ->not->or (4ns)
4. ->or(3ns)

میدانیم بیت های خروجی به صورت gfedcba و به ازای صفر شدن 4 بیت ورودی باید تنها g صفر شود و بقیه 1 شوند.از ثانیه صفر مقداردهی ما شروع میشود.پس از 4 نانو ثانیه بیت های b و c یک میشوند.علتش با توجه به مدار های رسم شده در بالا مشخص است . critical path گیت های وابسته به c مسیر 3 است. و گیت های وابسته به بیت b , طولانی ترین مسیر، همان مسیر 1 است ولی با توجه به اینکه برای گیت or یک شدن یکی از ورودی ها برای یک شدن خروجی کافیست 4 نانو ثانیه طول میکشد.

سپس در لجظه 5 نانو ثانیه بیت g صفر می شود.زیرا همه ورودی های گیت or باید صفر شوند.و وردی گیت OR همان خروجی گیت AND است.برای گیت اند صفر شدن یکی از ورودی ها کفایت میکند.هر گیت اند در این مدار حداقل یک ورودی دارد که به گیت not وابسته نیست پس مدت زمان این مسیر همان مسیر 2 است و 5 نانوثانیه طول میکشد.

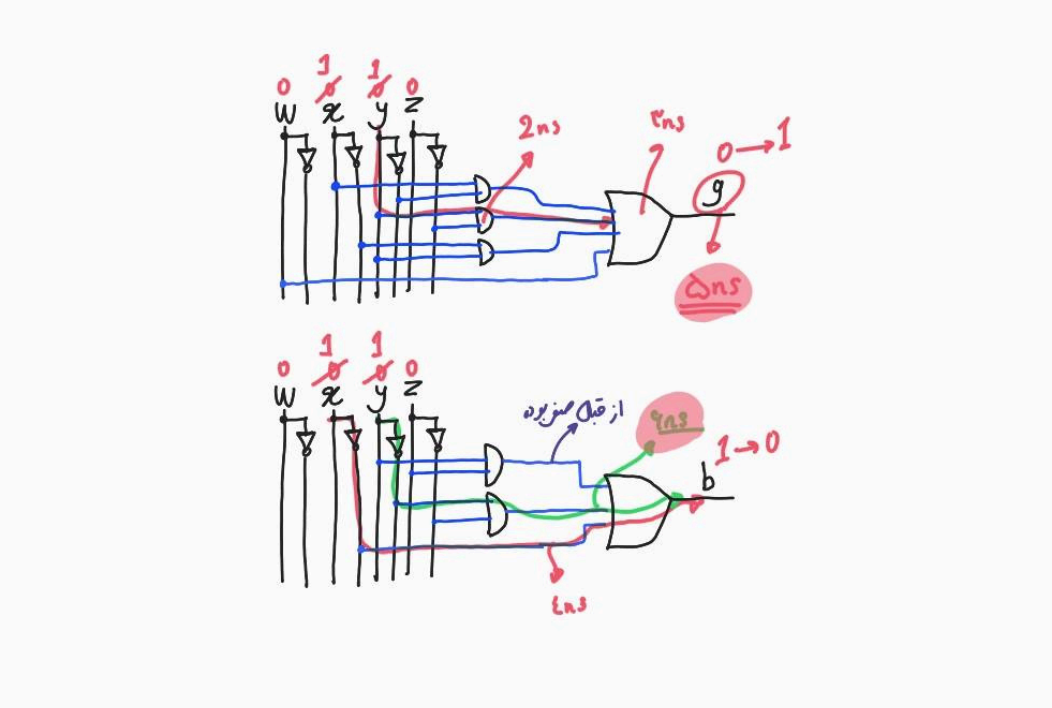
اما سایر بیت ها برای 1 شدن نیاز دارند تا طولانی ترین مسیر هم طی شود تا مقادیرشان پایدار شود که یان با توجه به مدار های رسم شده کاملا مشخص است.

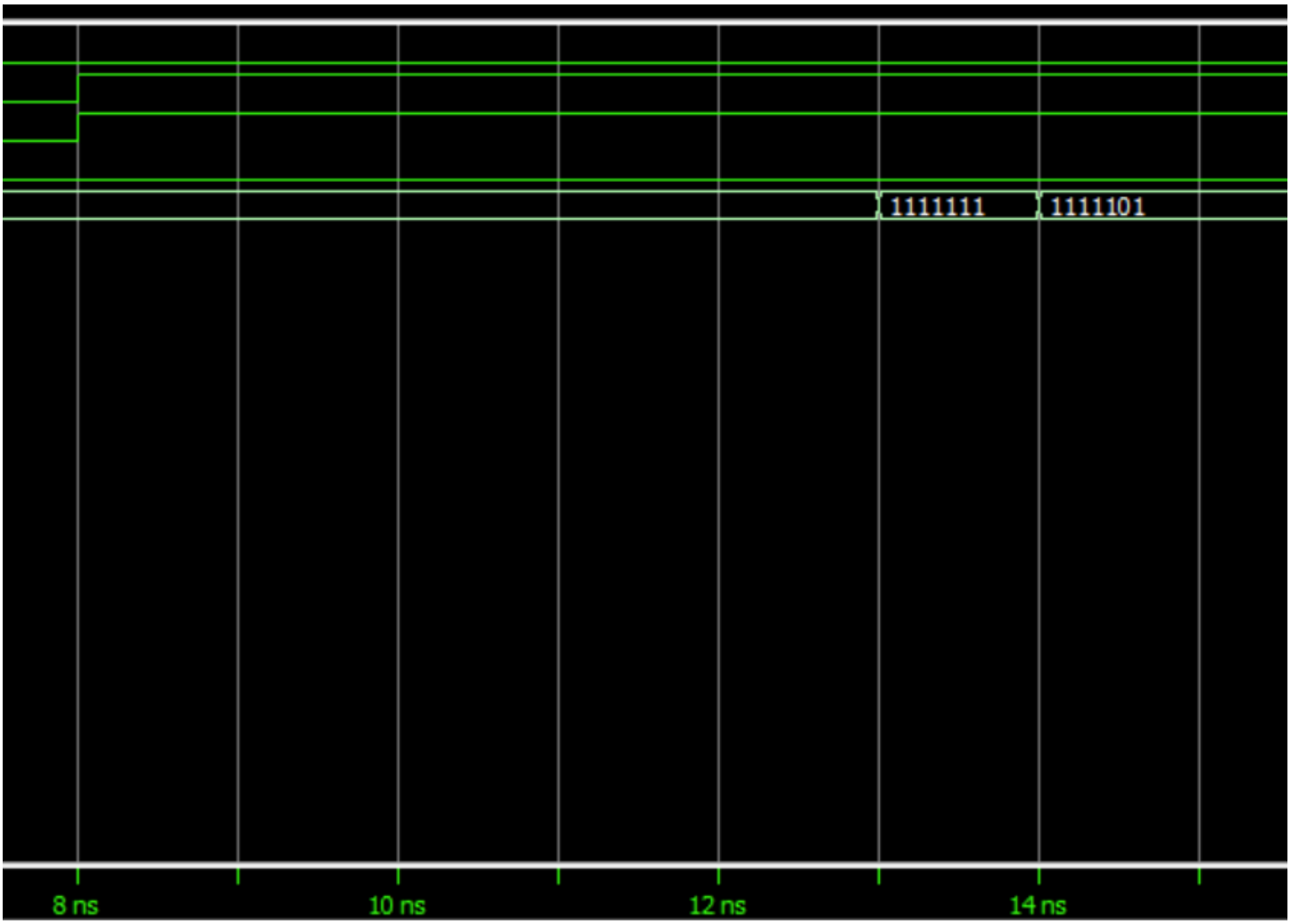


تغییر دوم:

در ثانیه 8 سیگنال های ورودی را تغییر میدهیم(x و y را 1 میکنیم).میدانیم که خروجی ما از 0111111 باید به 1111101 تغییر پیدا کند.یعنی بیت g یک شود و بیت b صفر شود.

با توجه به عکس زیر، بیت g پس از 5 نانوثانیه تغییر میکند و بیت b پس از 6 نانوثانیه تغییر میکند.

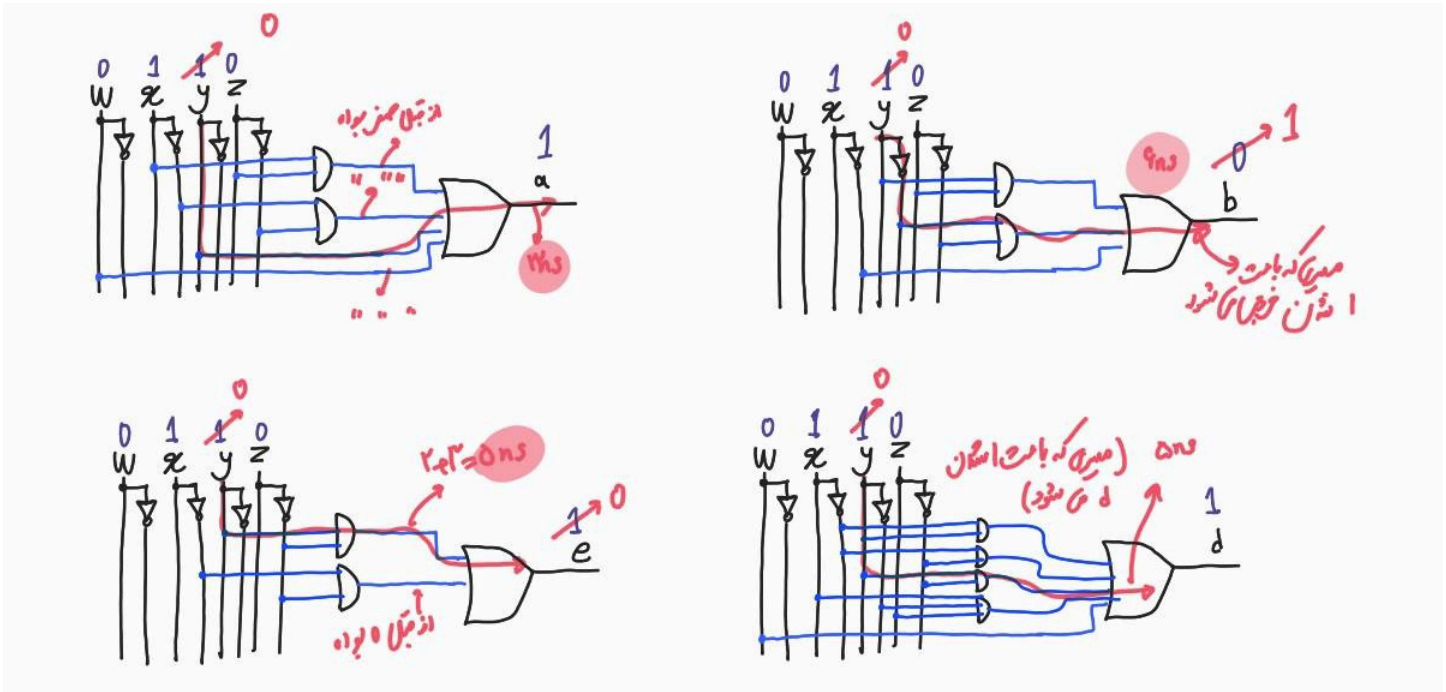




تغییر سوم:

در این قسمت تنها بیت y صفر میشود.و میدانیم که خروجی ما باید از 1111101 به 1100110 تغییر پیدا کند.یعنی بیت a صفر شود و بیتb یک شود و بیت های d و e هم صفر شوند.

تحیلل زمانی روی مدار:


W
x
y
z
a
b
e
d
1111100
1100100
1100110
16 ns
18 ns
20 ns
22 ns

د)اکنون در یک فایل دیگر رفتار منطقی گیت هارا به صورت behavioral تعریف میکنم و از | برای or و & برای and و از ~ برای گیت not استفاده میکنم و با توجه به معادله های منطقی بالا کد زیر را میزنم:

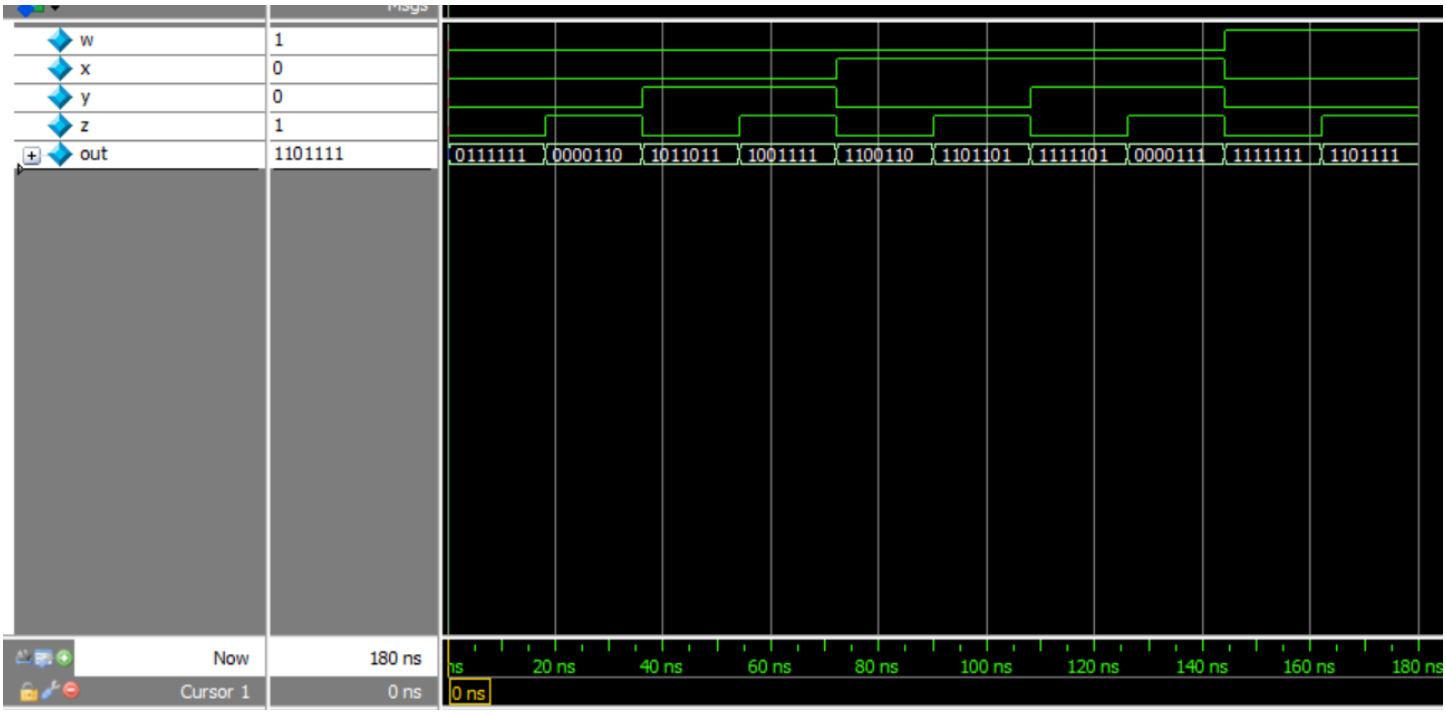
```
`timescale 1ns/1ns
module behavioral_sevenSegmentDecoder(input logic w,x,y,z , output logic [6:0] out);
    assign out[0]= w|y|(x&z)|(~x&~z); //a
    assign out[1]=~x|(y&z)|(~y&~z);  //b
    assign out[2]=~y|z|x; //c
    assign out[3]=w|(~x&y)|(~x&~z)|(y&~z)|(x&~y&z); //d
    assign out[4]=(y&~z)|(~x&~z);  //e
    assign out[5]=w|(~y&~z)|(x&~z)|(x&~y); //f
    assign out[6]=w|(x&~y)|(y&~z)|(~x&y); //g

endmodule
```

سپس تست بنچ را به صورت زیر تغییر میدهیم(در یک فایل جدید):

```
`timescale 1ns/1ns
module behavioral_sevensegmentDecoder_TB;
    logic w,x,y,z;
    logic [6:0] out;

    behavioral_sevenSegmentDecoder SSD(w,x,y,z,out);

    initial begin
       for(int number=0 ; number<10 ; number+=1)begin

            {w,x,y,z}=number;
            #18;
       end
    end
endmodule

```

کد را کامپایل کرده و simulate میکنم

در نهایت داریم:



در این بخش چون تاخیر نگذاشتیم ،به محض تغییر سیگنال های ورودی ،output ما به شکل صحیح درست خواهد شد.

The end.