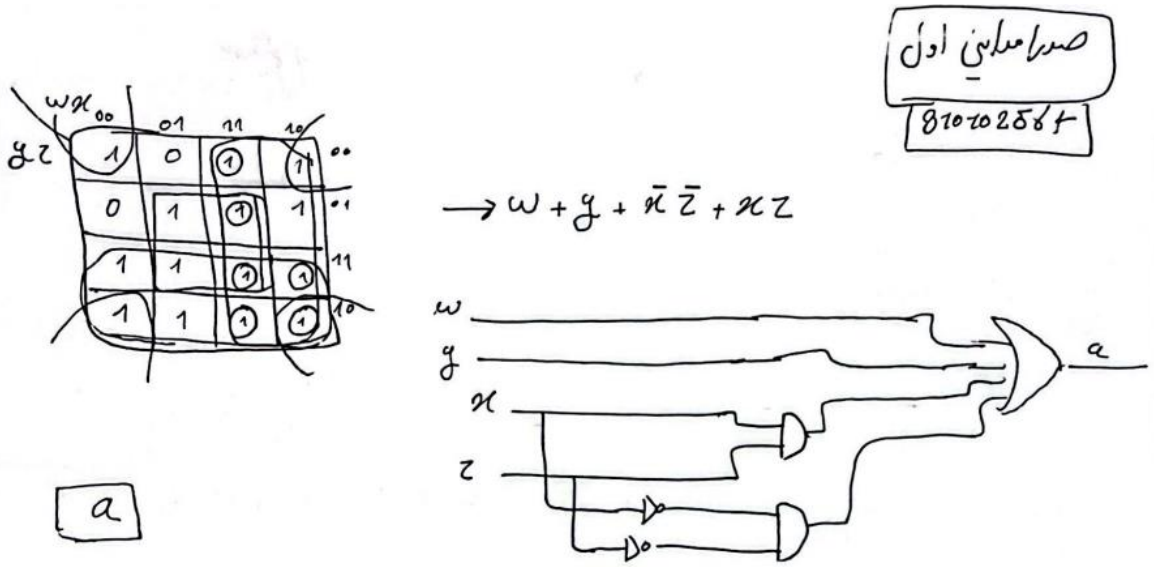
Introduction To Verilog
CA1

Sadra Madayeni Avval
810102564

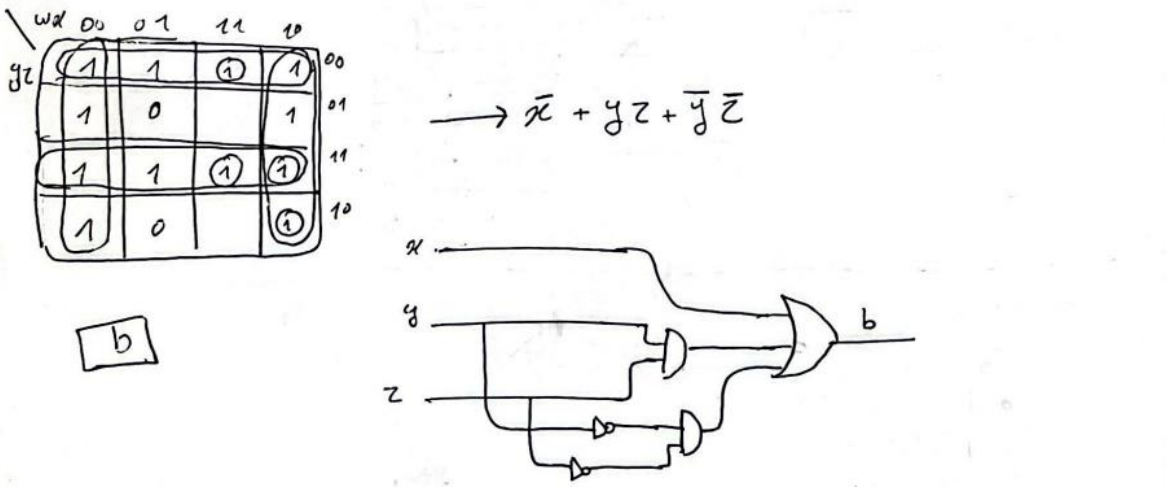
First Of All We Should Draw The Karnaugh Maps For Our Seven Segment Decoder

There Were Seven Segments For Our Karnaugh Map And We Needed To Draw 7 Different Karnaugh Maps

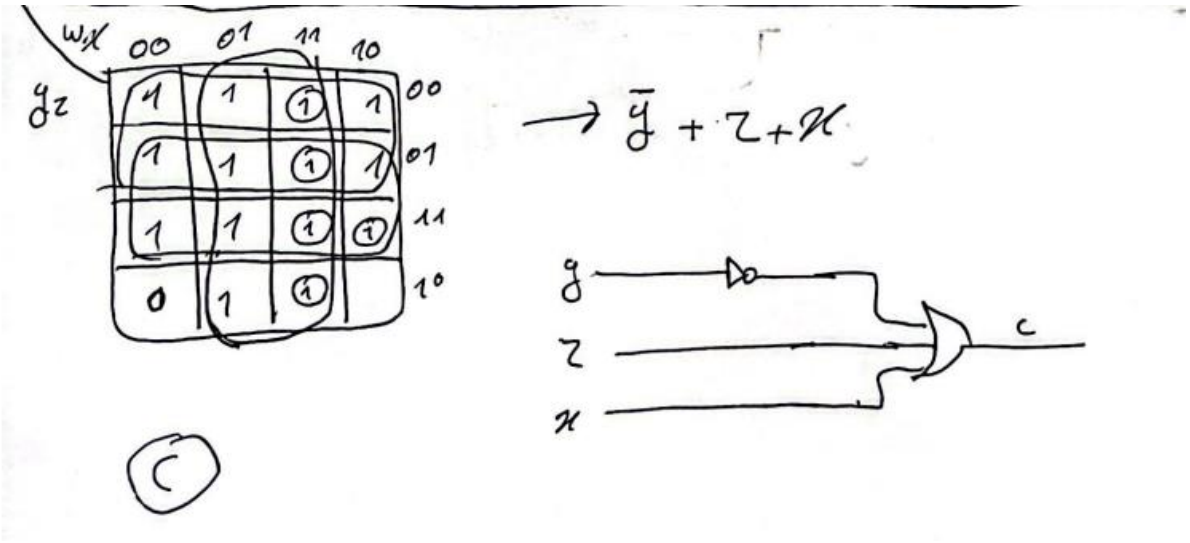
Here is the Karnaugh Map For a



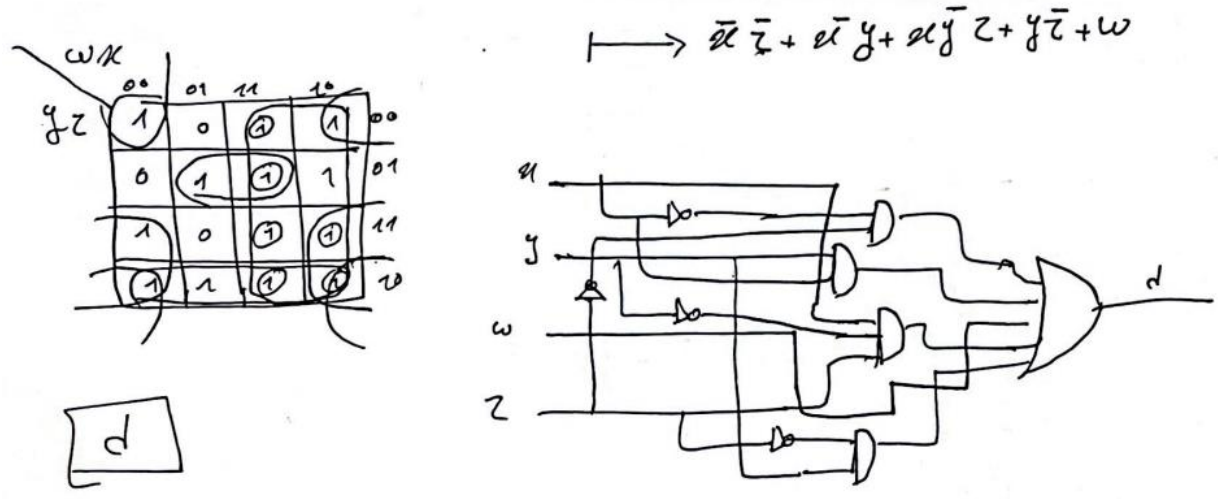
And also For B



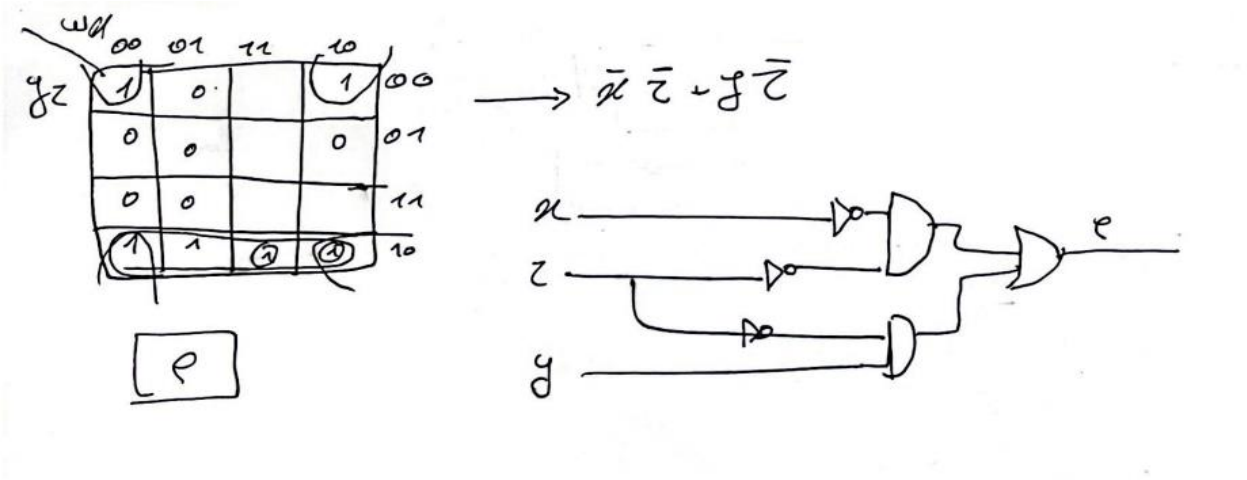
C Karnaugh Map



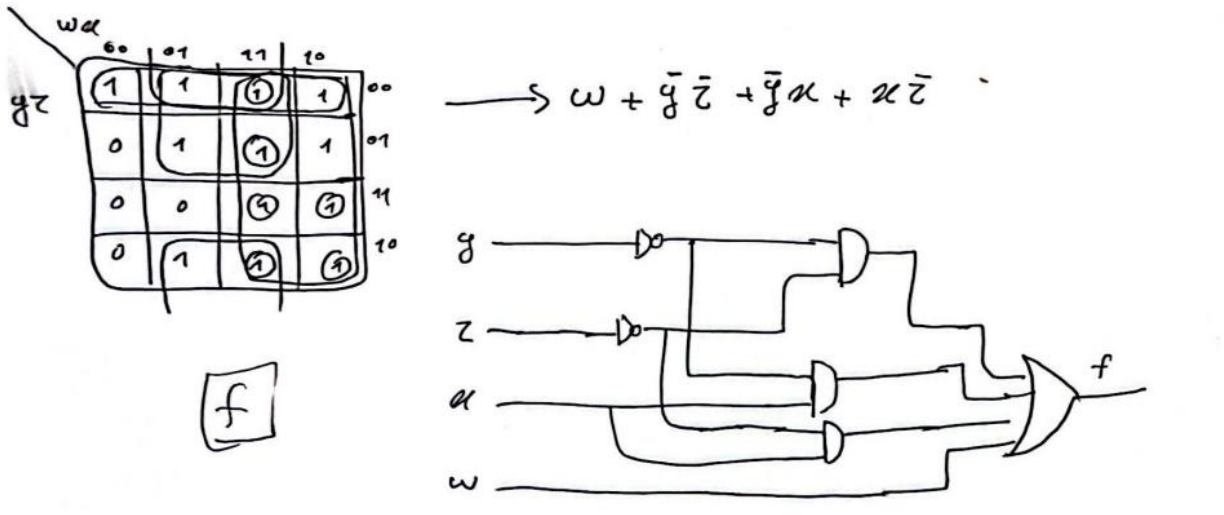
D Karnaugh Map



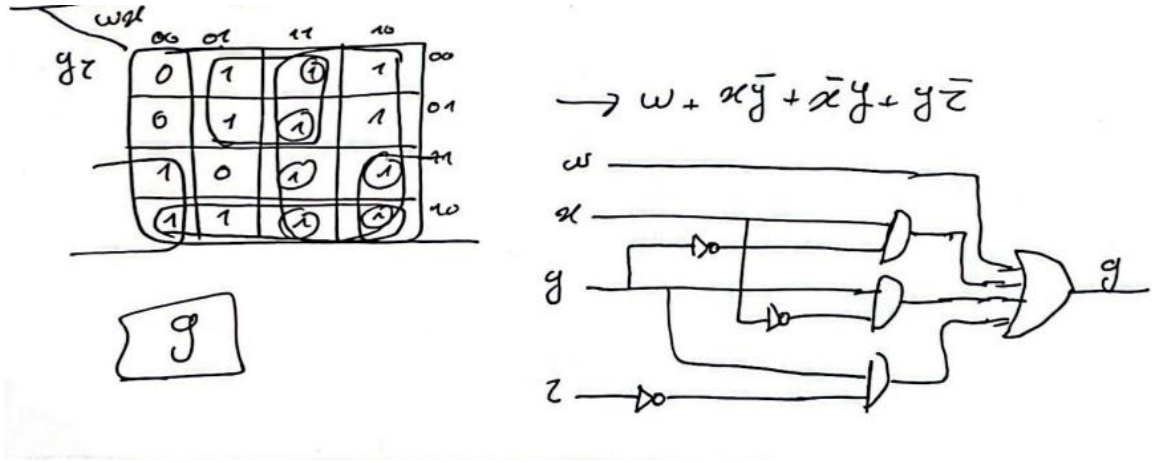
Karnaugh Map For E



F Karnaugh Map



Karnaugh Map For G



Part B

Now We Have To Implement These Karnaugh Maps Using Our Simulation Tool And We Have To Implement It Using Verilog

```
1   `timescale 1ns/1ns
2   module sevenSegmentDecoder(input w, x, y, z, output [6:0] out);
3
4       wire notx, noty, notz, notw;
5       wire xbar_zbar, xbar_w, xz, yz, not_y_z, wbarx, yzbar, yxbar, xybar, ybarzbar, xzbar, xbary, xybarz;
6
7       not #(1) g16(notx, x);
8       not #(1) g17(noty, y);
9       not #(1) g18(notz, z);
10      not #(1) g19(notw, w);
11      and #(2) g20(xybarz, x, noty, z);
12      and #(2) g1(xbar_zbar, notx, notz);
13      and #(2) g2(xbar_w, notx, w);
14      and #(2) g3(xz, x, z);
15      and #(2) g6(yz, y, z);
16      and #(2) g7(not_y_z, noty, notz);
17      and #(2) g8(yzbar, y, notz);
18      and #(2) g9(wbarx, notw, x);
19      and #(2) g10(yxbar, notx, y);
20      and #(2) g11(xybar, x, noty);
21      and #(2) g12(ybarzbar, noty, notz);
22      and #(2) g13(xzbar, x, notz);
23      and #(2) g14(xbary, notx, y);
24      or #(3) a(out[0], w, y, xz, xbar_zbar);
25      or #(3) b(out[1], notx, yz, not_y_z);
26      or #(3) c(out[2], noty, z, x);
27      or #(3) d(out[3], w, yxbar, xbar_zbar, yzbar, xybar, xybarz);
28      or #(3) e(out[4], xbar_zbar, yzbar);
29      or #(3) f(out[5], w, xybar, xzbar, ybarzbar);
30      or #(3) g(out[6], w, yzbar, xybar, xbary);
31
32  endmodule
33
34
```

We Used 1ns Timescale for Better Experience and We Have 4 Inputs w, x, y ,z and a 7 bit output that Represents A Number 0 to 9

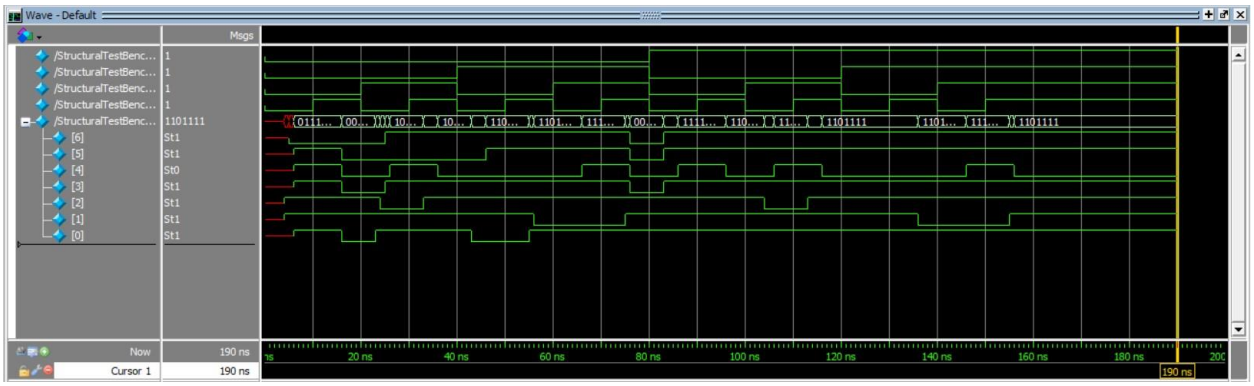
Part C

Now We Need To See That If Our Verilog Code Is Working Correctly Or Not

And For That We Need To Make A Test Bench For Our Verilog Code That We Can Implement It Like This

```
`timescale 1ns/1ns
module StructuralTestBench();
    reg w, x, y, z;
    wire [6:0]test_out;

    sevenSegmentDecoder test(w, x, y, z,test_out);

    initial begin

        w = 0; x = 0; y = 0; z = 0; #10;
        w = 0; x = 0; y = 0; z = 1; #10;
        w = 0; x = 0; y = 1; z = 0; #10;
        w = 0; x = 0; y = 1; z = 1; #10;
        w = 0; x = 1; y = 0; z = 0; #10;
        w = 0; x = 1; y = 0; z = 1; #10;
        w = 0; x = 1; y = 1; z = 0; #10;
        w = 0; x = 1; y = 1; z = 1; #10;
        w = 1; x = 0; y = 0; z = 0; #10;
        w = 1; x = 0; y = 0; z = 1; #10;
        w = 1; x = 0; y = 1; z = 0; #10;
        w = 1; x = 0; y = 1; z = 1; #10;
        w = 1; x = 1; y = 0; z = 0; #10;
        w = 1; x = 1; y = 0; z = 1; #10;
        w = 1; x = 1; y = 1; z = 0; #10;
        w = 1; x = 1; y = 1; z = 1; #10;
        #30;

    end
endmodule
```

We Give All Possible Inputs That Varies From 0 – 15 Because We Have 4 Outputs And That's Why We Have 16 Possible Changes With 10 ns Delay For Each Change And a 30 ns Delay To Make it More Clear And Our Output is A Seven Bit Number That Represents A Number Between 0 to 9, Note that for inputs 10 to 15 we might get a meaningless output and its due to Don't care parameters.



0111111 to 0000110 --→ 0 to 1

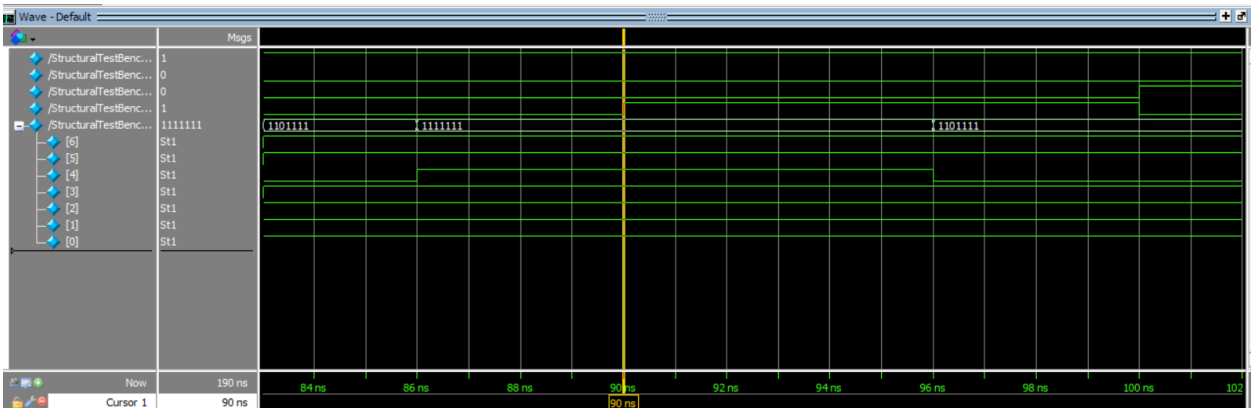
so the change is input is:

From 0000 to 0001 where only z turns 1.
When transferring 0 to 1 At 10ns Our Input Starts To Change And At 16 ns We Can See The Changes
a , c , d , e start to Transfer From 1 To 0,

If We Consider The Longest Path We Get 1 + 2 + 3 = 6ns delay for these outputs to change.

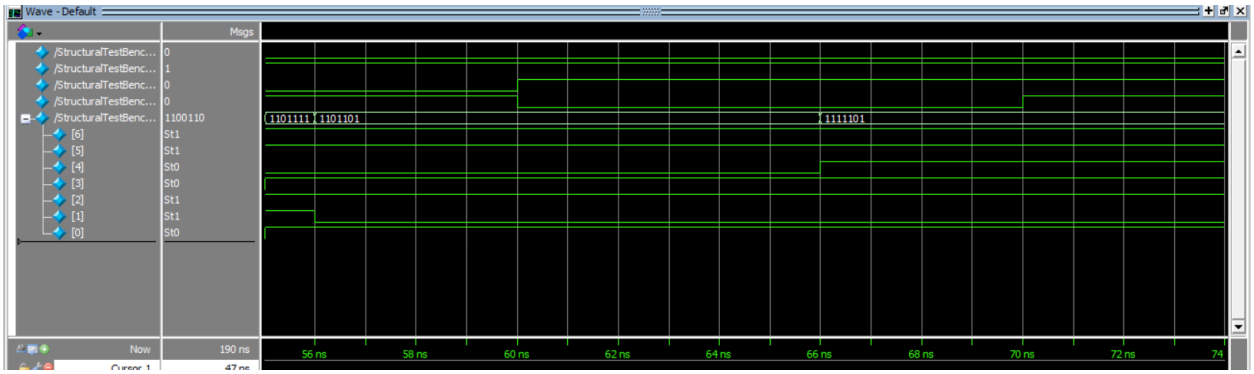
when 8 changing to 9 that means after changing z from to 1 in 90ns after 6ns our output changes from 1 to 0 so the input change is 1000 to 1001.
and for changing z all 3 gates , And , Or , Not gates comes to the game and after the 6ns delay e changes as well and our number moves from 8 to 9 → 1111111 to 1101111

By checking the changes in each output separately we see for all outputs other than e we have a gate the keeps the output 1, for a its w, for b its x-bar, for c its y-bar, and for d,f and g its w which keeps the output 1, but for e gate yz-bar turns 0 after 6ns due to a not , an or and an and gate which take 6ns in total, after that the output becomes 1101111.

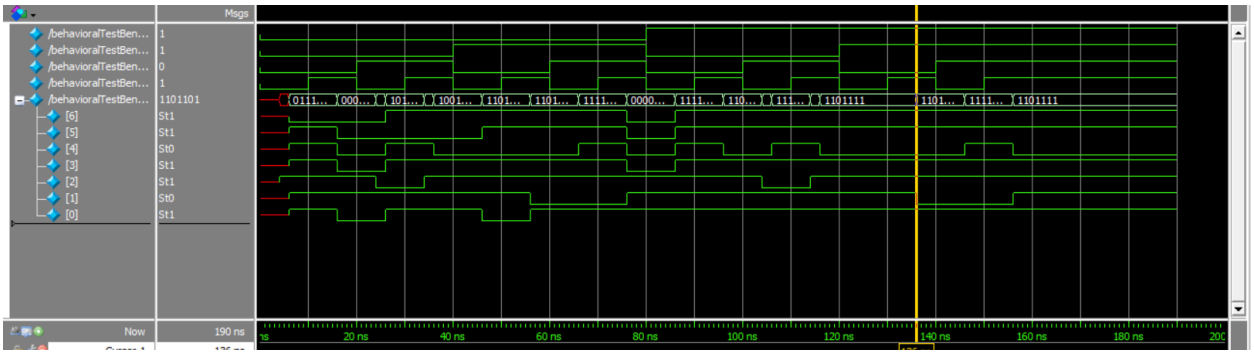


Looking at change in when turning 5 to 6, meaning input changing from 0101 to 0110, we have to changes which are z turning 0 and y turning 1,

looking closely we again only have change in one of the outputs which is e, the change is actually from 1101101 to 1111101. Here's why we don't see change in output: for a its first xz where before turning to 0 we get a 1 from y which keeps a 1 , for b all of them are 0 and don't change since the gate yz , ybar-zbar and x-bar are always 0, even for ybar-zbar since they have the same not gate delay it stays 0, for c x keeps it 1, for d first its y z-bar and then it becomes x y-bar z but since they have the same delay of turning 1 and 0 we always have a 1 to keep the output 1,for e first y z-bar keeps it 0 but after changing it takes 6ns for output to change to 1 and x-bar z-bar doesn't change, for f first x y-bar and then x z-bar keeps it 1 and since they have the same changing time we always get 1 as output for f, and for g first it's x y-bar that keeps it 1 but then its y z-bar and again since they both have a not and an and gate before going to or gate, they both have a delay of 4ns which means we always have a 1 in or gate that keeps the output 1.



Part D



In Behavioral Some Some Delays May Not Be That Much Accurate Because In Structural All Delays Are Selected But in Behavioral We Consider The Longest Path And Some Delays May Not Be Visible Because The Longest Path May Not Play