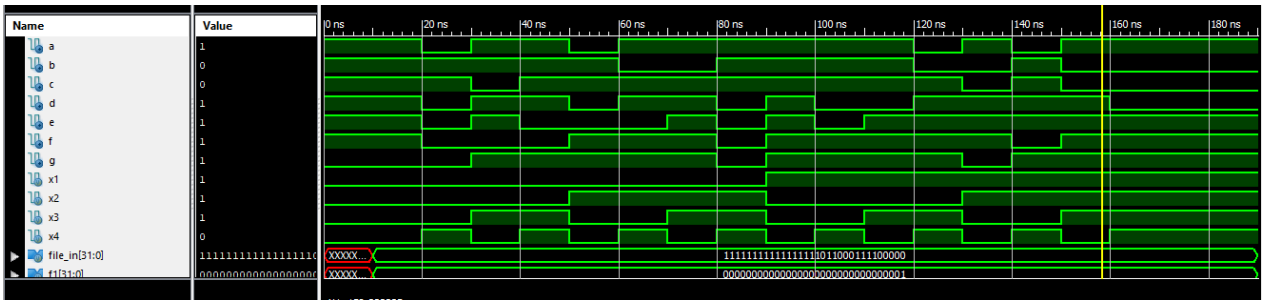
Задача 1: При първият модул , времето за закъснение е твърде малък и се получава глич , при увеличаване на времето >#1, гличът се изчиства.

При вторият модул, трябва да се опише състояние когато a=0 , тъй като симулаторът, чете стойността b като запомнена предишна , което се характеризира за латч тригерите и автоматично генерира латч.

Задача 2: https://www.edaplayground.com/x/vCHm

Задача 3:




```
/*
*           Y. Gorbounov 2021
*/
`timescale 1ps / 1fs

module bin2hex_gatelevel_fixed (
            input x1, x2, x3, x4,
            output a, b, c, d, e, f, g
            );

            wire a1, a2, a3, a4, a5, a6;
            wire b1, b2, b3, b4, b5;
            wire c1, c2, c3, c4, c5;
            wire d1, d2, d3, d4, d5;
            wire e1, e2, e3, e4;
            wire f1, f2, f3, f4, f5;
            wire g1, g2, g3, g4, g5;

            wire x1_, x2_, x3_, x4_; //direct values
            wire x1n, x2n, x3n, x4n; //invert values of inputs

            wire vcc=1'b1;
            wire gnd=1'b0;

            //corrective circuit
            xor (x1n, vcc, x1); //for signal x1
```

```
xor (x1_, gnd, x1);

xor (x2n, vcc, x2); //for signal x2
xor (x2_, gnd, x2);

xor (x3n, vcc, x3); //for signal x3
xor (x3_, gnd, x3);

xor (x4n, vcc, x4); //for signal x4
xor (x4_, gnd, x4);

//Instantiation of logic follows.

and(a1, x2_, x3_);
and(a2, x1n, x3_);
and(a3, x1_, x4n);
and(a4, x2n, x4n);
and(a5, x1n, x2_, x4_);
and(a6, x1_, x2n, x3n);
or(a, a1, a2, a3, a4, a5, a6);  // f1 <=> a

and(b1, x2n, x4n);
and(b2, x2n, x3n);
and(b3, x1n, x3_, x4_);
and(b4, x1_, x3n, x4_);
and(b5, x1n, x3n, x4n);
or(b, b1, b2, b3, b4, b5);  // f2 <=> b

and(c1, x3n, x4_);
and(c2, x1n, x4_);
and(c3, x1_, x2n);
and(c4, x1n, x2_);
and(c5, x2n, x3n);
```

```
    or(c, c1, c2, c3, c4, c5);  // f3 <=> c

    and(d1, x1n, x3_, x4n);
    and(d2, x2n, x3_, x4);
    and(d3, x2_, x3n, x4_);
    and(d4, x1_, x2_, x4n);
    and(d5, x2n, x3n, x4n);
    or(d, d1, d2, d3, d4, d5);  // f4 <=> d

    and(e1, x3_, x4n);
    and(e2, x1_, x3_);
    and(e3, x1_, x2_);
    and(e4, x2n, x4n);
    or(e, e1, e2, e3, e4);  // f5 <=> e

    and(f1, x1_, x3_);
    and(f2, x1_, x2n);
    and(f3, x2_, x4n);
    and(f4, x3n, x4n);
    and(f5, x1n, x2_, x3n);
    or(f, f1, f2, f3, f4, f5);  // f6 <=> f

    and(g1, x1_, x3_);
    and(g2, x2n, x3_);
    and(g3, x1_, x2n);
    and(g4, x2_, x3n, x4_);
    and(g5, x1n, x2_, x4n);
    or(g, g1, g2, g3, g4, g5);  // f7 <=> g

endmodule
```