|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | signed? | width | base | int val |
| 8’b00001000 | No | 8 | bin | 8 |
| 16’hABCD | No | 16 | hex | 43981 |
| 4’sb1110 | Yes | 4 | bin | -2 |
| 4’d12 | No | 4 | dec | 12 |
| 4’b1100 | No | 4 | bin | 12 |
| 8’shFF | Yes | 8 | hex | -1 |
| 8’sb00011100 | Yes | 8 | bin | 28 |
| 12’h2A8 | No | 12 | hex | 680 |
| 6’sb111101 | Yes | 6 | bin | -3 |
| 12’o3456 | No | 12 | oct | 1838 |

**Problem 4:**

**alarm.v:**

`timescale 1ns/10ps

//top module

module alarm(PANIC, EN, EXIT, WINDOW, DOOR, GARAGE, ALARM);

    input   PANIC,EN,EXIT,WINDOW,DOOR,GARAGE;

    output  ALARM;

    wire gate1,gate2,gate3;

    nand(gate1,WINDOW,DOOR,GARAGE);

    not(gate2,EXIT);

    and(gate3,EN,gate2,gate1);

    or(ALARM,gate3,PANIC);

endmodule

**alarm\_test.v:**

`timescale 1ns/10ps

module alarm\_test;

    reg PANIC,EN,EXIT,WINDOW,DOOR,GARAGE;

    wire ALARM;

    alarm doubile(

        PANIC,EN,EXIT,WINDOW,DOOR,GARAGE,ALARM

    );

    initial

        begin

            // first case - should be alarm = 1

            assign PANIC = 1;

            assign EN = 1;

            assign EXIT = 1;

            assign WINDOW = 1;

            assign DOOR = 1;

            assign GARAGE = 0;

            #20

            // second case - should be alarm = 0

            assign PANIC = 0;

            #20

            // third case - should be alarm = 1

            assign EXIT = 0;

            #20

            // fourth case - should be alarm = 0

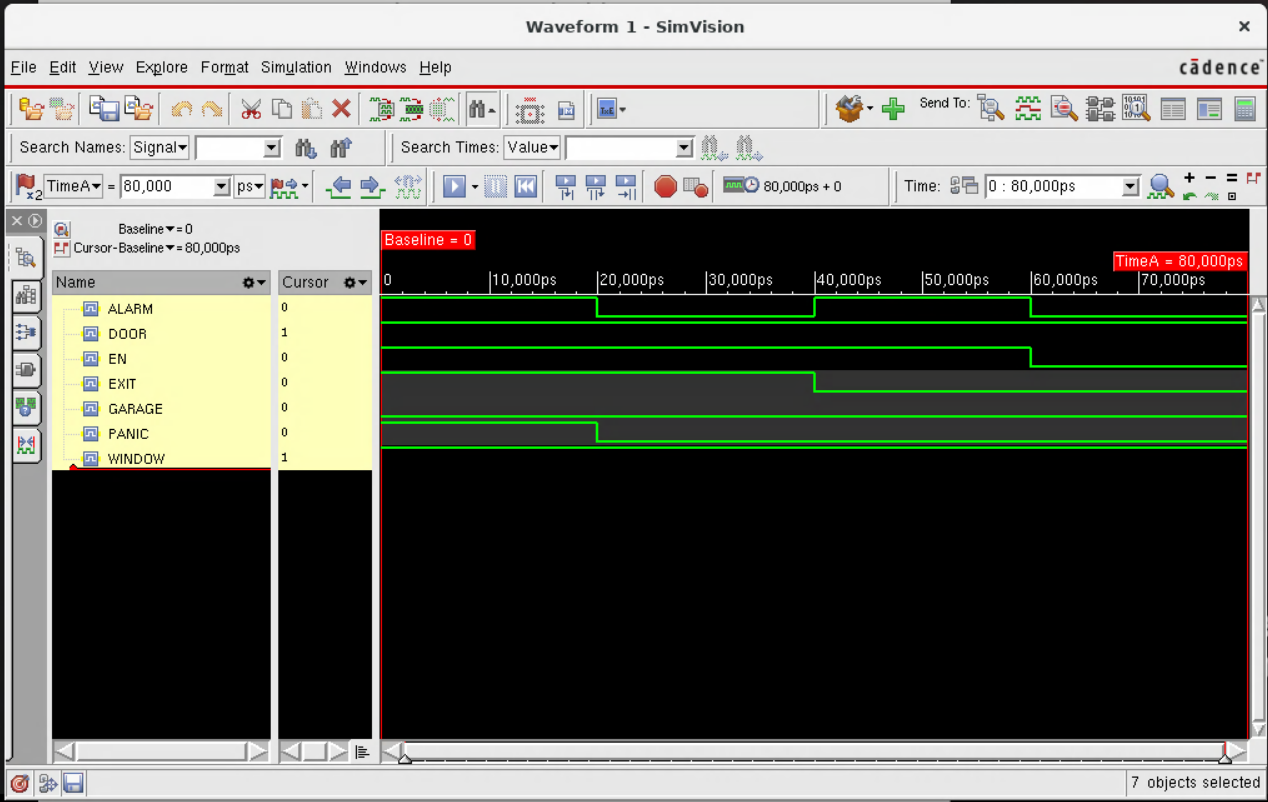
            assign EN = 0;

            #20

            $finish;

        end

endmodule



Simulation output