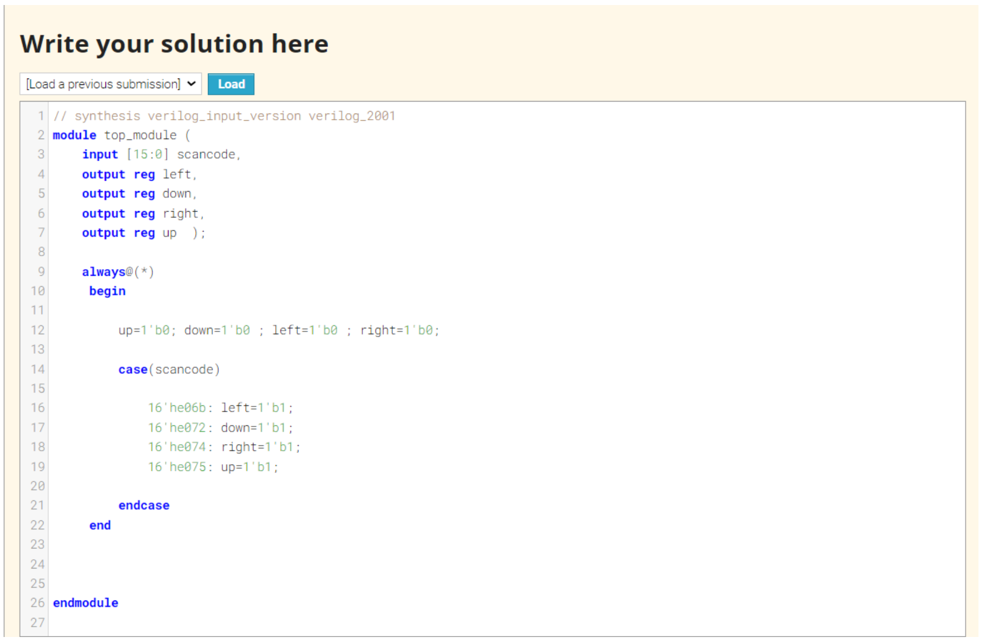
DAY-6
#100DAYSOFRTL

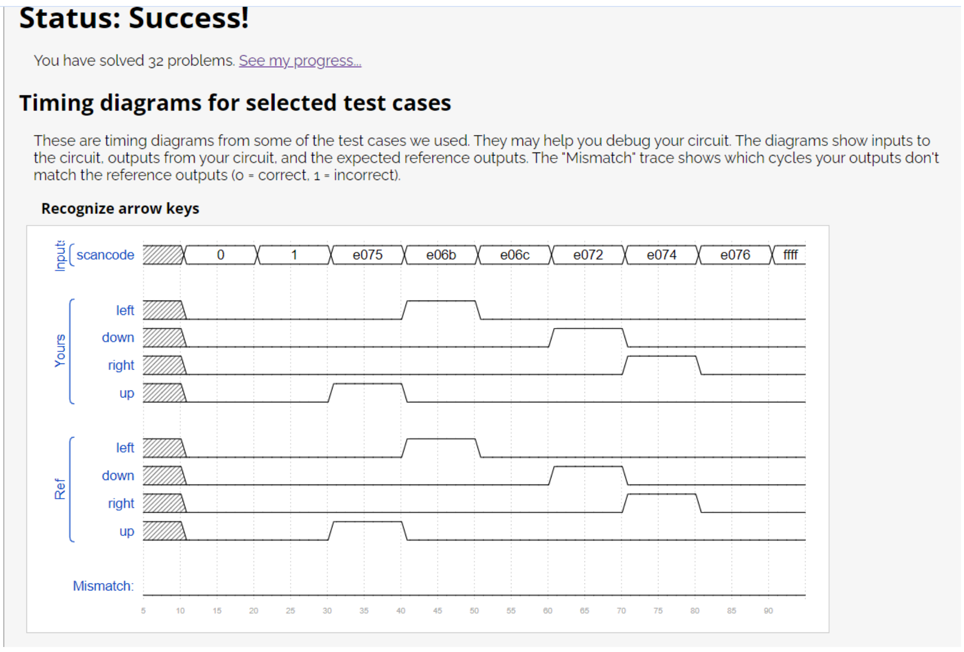
Problem statement :-

1. Your circuit has one 16-bit input, and four outputs. Build this circuit that recognizes these four scancodes and asserts the correct output.



Write your solution here

```
// synthesis verilog_input_version verilog_2001
module top_module (
    input [15:0] scancode,
    output reg left,
    output reg down,
    output reg right,
    output reg up  );

    always@(*)
     begin

         up=1'b0; down=1'b0 ; left=1'b0 ; right=1'b0;

         case(scancode)

             16'he06b: left=1'b1;
             16'he072: down=1'b1;
             16'he074: right=1'b1;
             16'he075: up=1'b1;

         endcase
     end


endmodule
```



Status: Success!

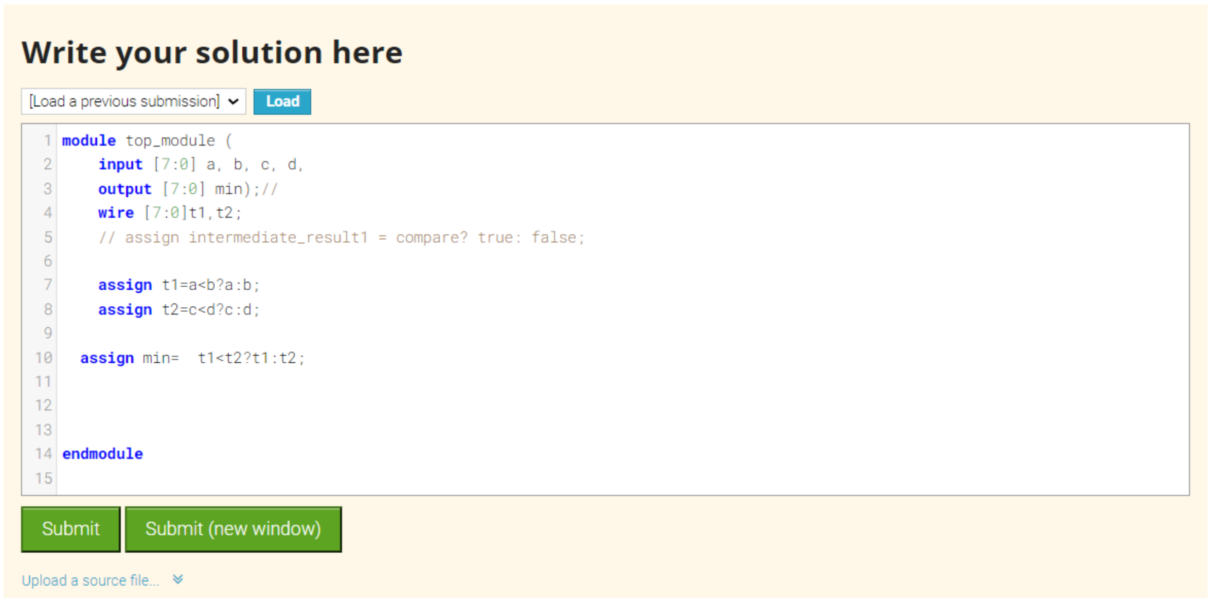
You have solved 32 problems. See my progress...

Timing diagrams for selected test cases

These are timing diagrams from some of the test cases we used. They may help you debug your circuit. The diagrams show inputs to the circuit, outputs from your circuit, and the expected reference outputs. The "Mismatch" trace shows which cycles your outputs don't match the reference outputs (0 = correct, 1 = incorrect).

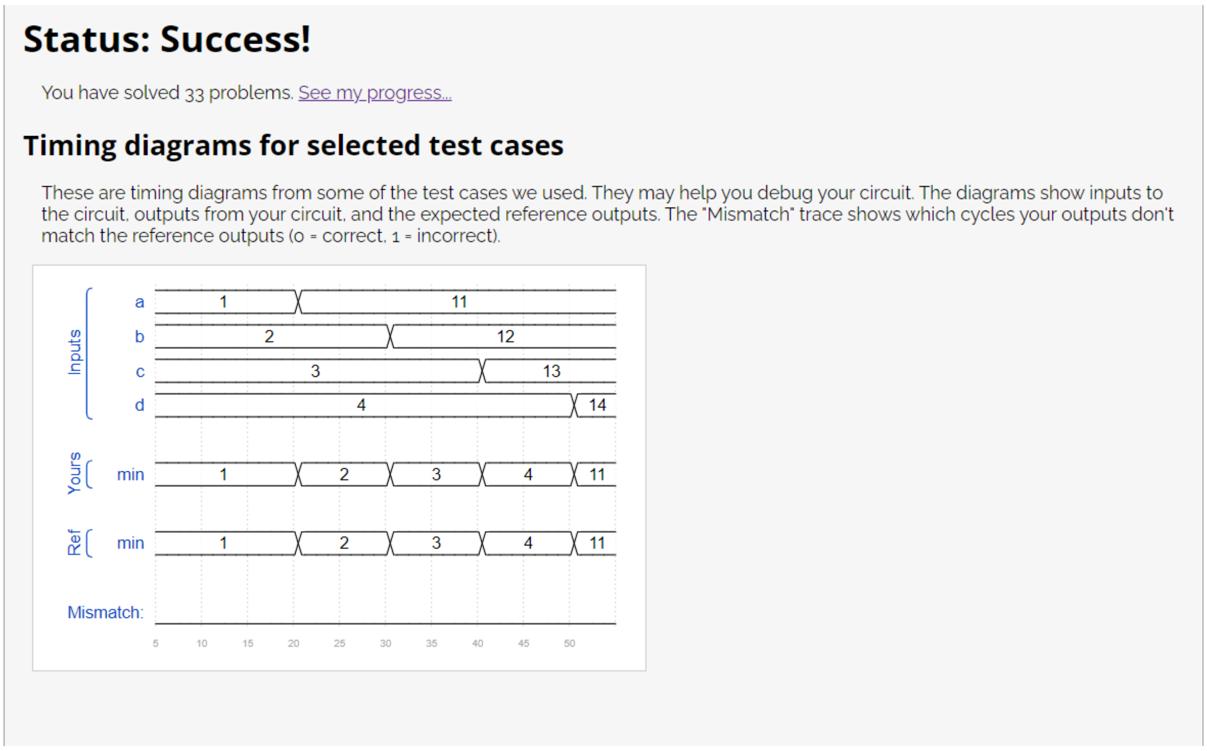
Recognize arrow keys

2. Given four unsigned numbers, find the minimum. Unsigned numbers can be compared with standard comparison operators (a < b). Use the conditional operator to make two-way *min* circuits, then compose a few of them to create a 4-way *min* circuit. You'll probably want some wire vectors for the intermediate results.



Write your solution here

```verilog
module top_module (
    input [7:0] a, b, c, d,
    output [7:0] min);//
    wire [7:0]t1,t2;
    // assign intermediate_result1 = compare? true: false;

    assign t1=a<b?a:b;
    assign t2=c<d?c:d;

  assign min=  t1<t2?t1:t2;



endmodule

```

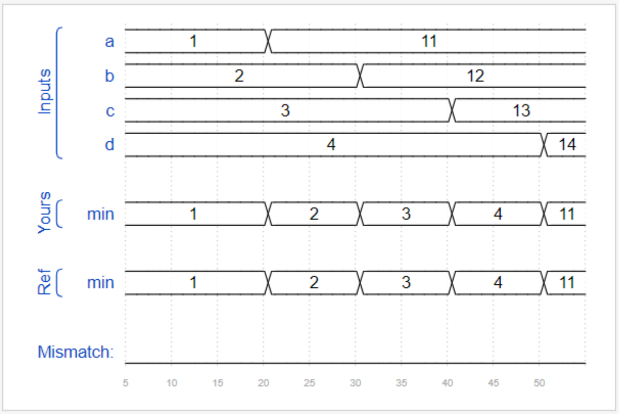


Status: Success!

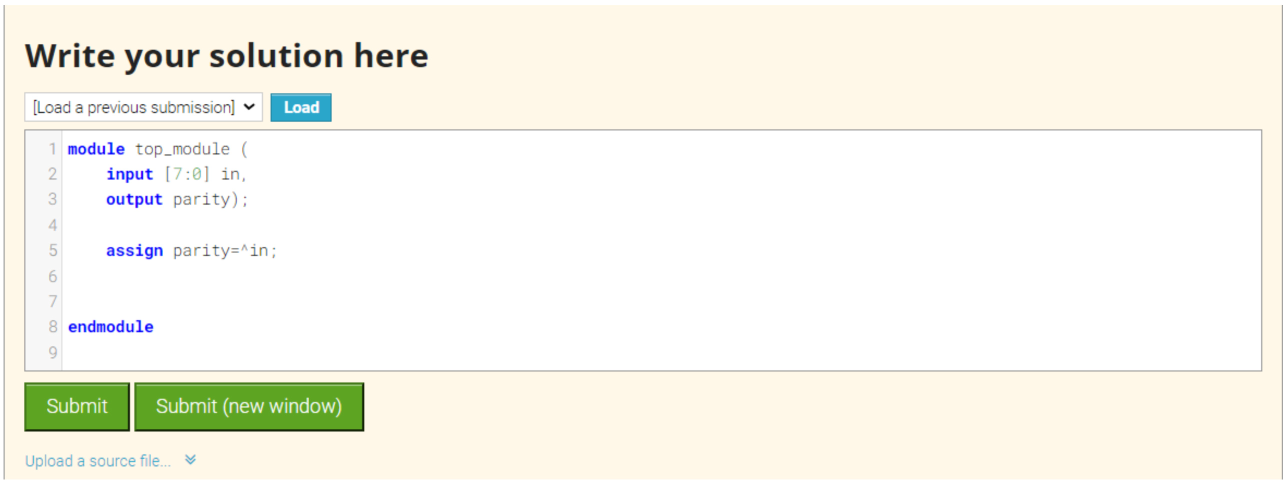
You have solved 33 problems. See my progress...

Timing diagrams for selected test cases

These are timing diagrams from some of the test cases we used. They may help you debug your circuit. The diagrams show inputs to the circuit, outputs from your circuit, and the expected reference outputs. The "Mismatch" trace shows which cycles your outputs don't match the reference outputs (0 = correct, 1 = incorrect).

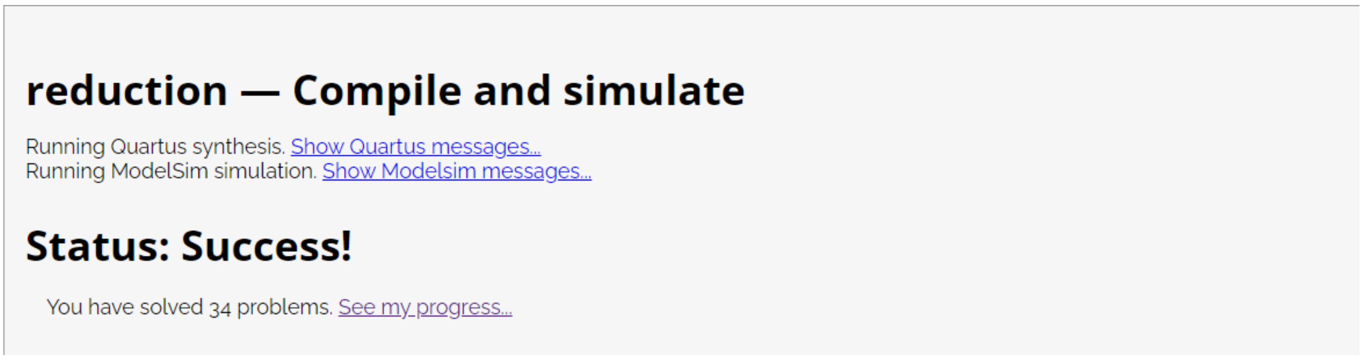


3. Parity checking is often used as a simple method of detecting errors when transmitting data through an imperfect channel. Create a circuit that will compute a parity bit for a 8-bit byte (which will add a 9th bit to the byte). We will use "even" parity, where the parity bit is just the XOR of all 8 data bits.



Write your solution here

```
module top_module (
    input [7:0] in,
    output parity);

    assign parity=^in;


endmodule

```



reduction — Compile and simulate

Running Quartus synthesis. Show Quartus messages...
Running ModelSim simulation. Show Modelsim messages...

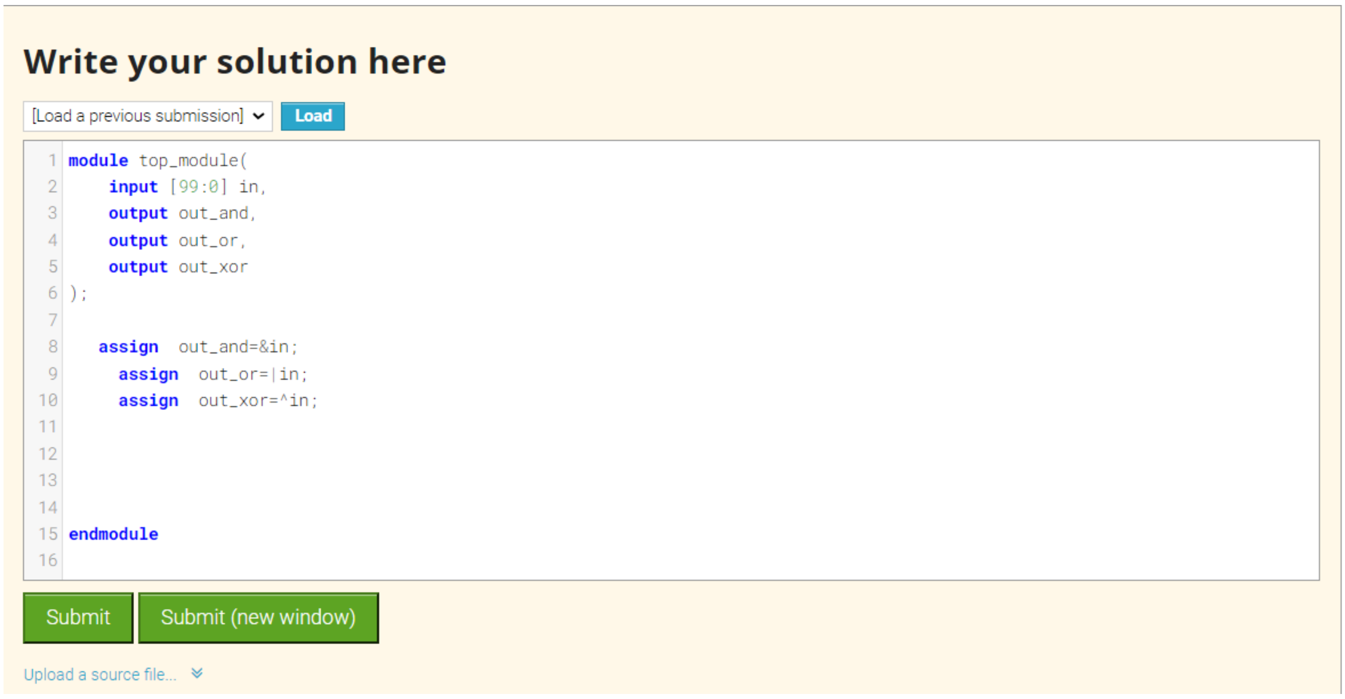
Status: Success!

You have solved 34 problems. See my progress...

4. Build a combinational circuit with 100 inputs, `in[99:0]`.

There are 3 outputs:

- out_and: output of a 100-input AND gate.
- out_or: output of a 100-input OR gate.
- out_xor: output of a 100-input XOR gate.



```
module top_module(
    input [99:0] in,
    output out_and,
    output out_or,
    output out_xor
);

   assign  out_and=&in;
     assign  out_or=|in;
     assign  out_xor=^in;




endmodule

```

gates100 — Compile and simulate

Running Quartus synthesis. Show Quartus messages...
Running ModelSim simulation. Show Modelsim messages...

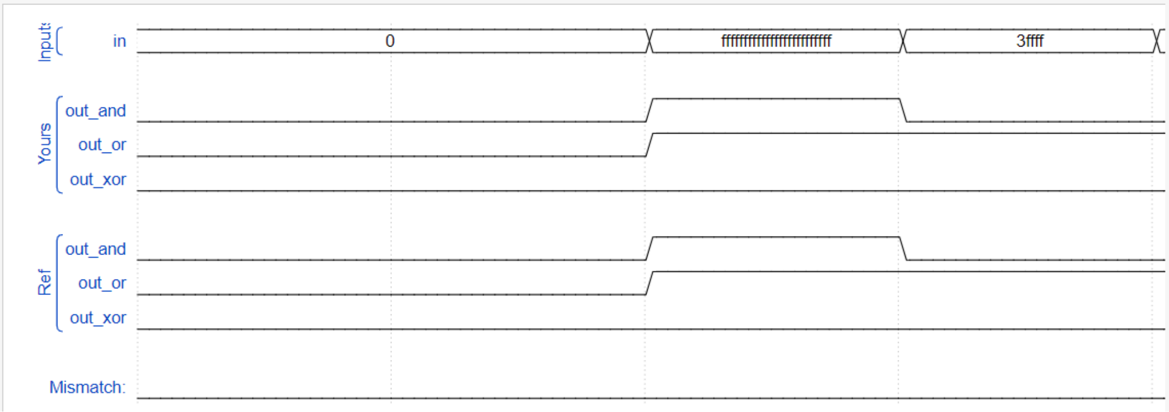
Status: Success!

You have solved 35 problems. See my progress...

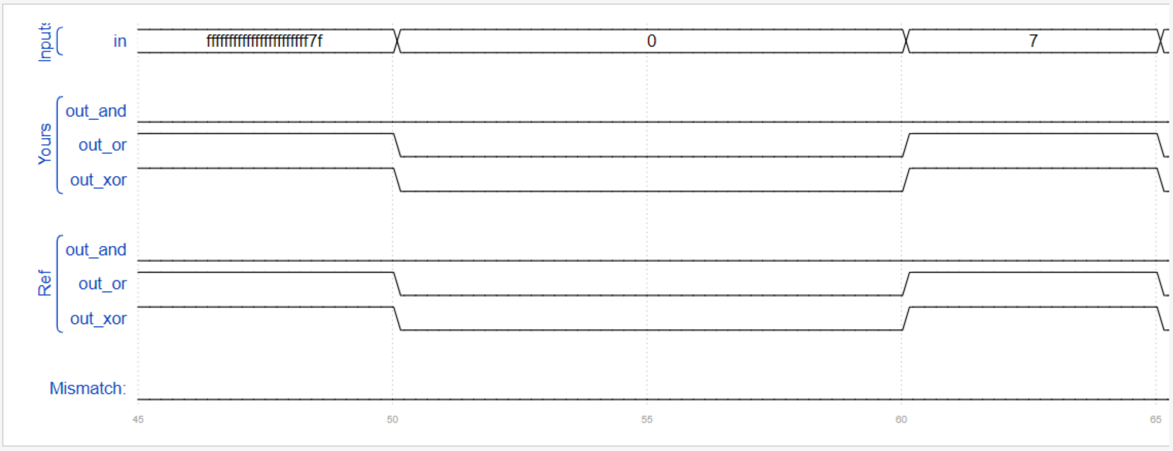
Timing diagrams for selected test cases

These are timing diagrams from some of the test cases we used. They may help you debug your circuit. The diagrams show inputs to the circuit, outputs from your circuit, and the expected reference outputs. The "Mismatch" trace shows which cycles your outputs don't match the reference outputs (0 = correct, 1 = incorrect).

Test AND gate



Test OR and XOR gates


Inputs
in
fffffffffffffffffffffff7f
Yours
out_and
out_or
out_xor
Ref
out_and
out_or
out_xor
Mismatch: