DAY-1
#100DAYSOFRTL

Problem statement :-

(1) Build a circuit with no inputs and one output. That output should always drive 1 (or logic high).

Write your solution here

[Load a previous submission] Load

```
1  module top_module( output one );
2
3  // Insert your code here
4      assign one = [fixme];
5
6  endmodule
7
```

Submit | Submit (new window)

Upload a source file...

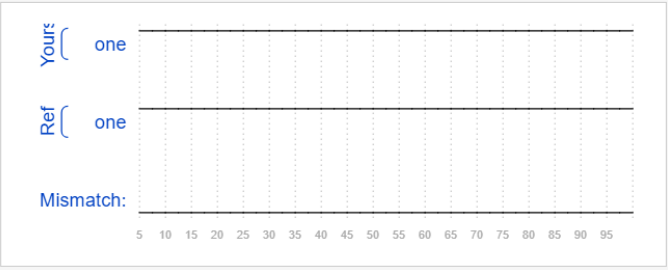
Status: Success!

You have solved 4 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).

Output should be 1



(2) Build a circuit with no inputs and one output that outputs a constant 0

Write your solution here

[Load a previous submission] Load

```
1 module top_module(output zero);
2 assign zero=0;
3 endmodule
4
```

Submit | Submit (new window)

Upload a source file...

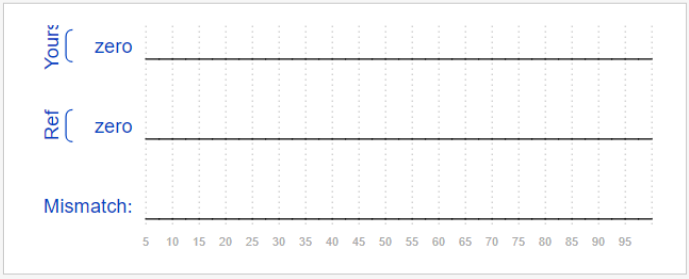
Status: Success!

You have solved 4 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).

Output should 0



(3) Create a module with one input and one output that behaves like a wire.

Write your solution here

[Load a previous submission] Load

```
1 module top_module( input in, output out );
2
3     assign out=in;
4
5 endmodule
6
```

Submit | Submit (new window)

Upload a source file...

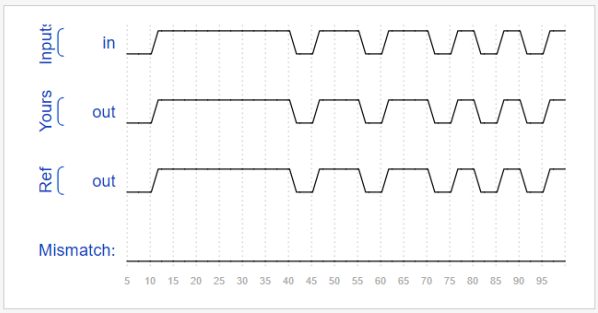
Status: Success!

You have solved 8 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).

Output should follow input



(4) Create a module with 3 inputs and 4 outputs that behaves like wires that makes these connections:

Write your solution here

[Load a previous submission] Load

```
module top_module(
    input a,b,c,
    output w,x,y,z );

    assign w=a;
    assign x=b;
    assign y =b;
    assign z=c;

endmodule

```

Submit | Submit (new window)

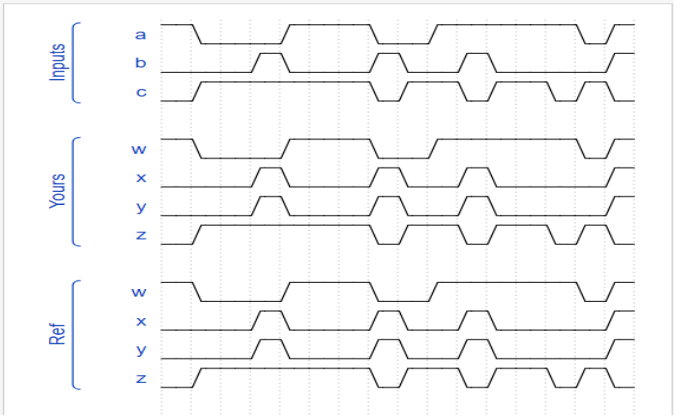
Upload a source file...

Status: Success!

You have solved 4 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).



(5) Create a module that implements a NOT gate.

Write your solution here

[Load a previous submission] Load

```
1 module top_module( input in, output out );
2
3     assign out=~in;
4
5 endmodule
6
```

Submit | Submit (new window)

Upload a source file...

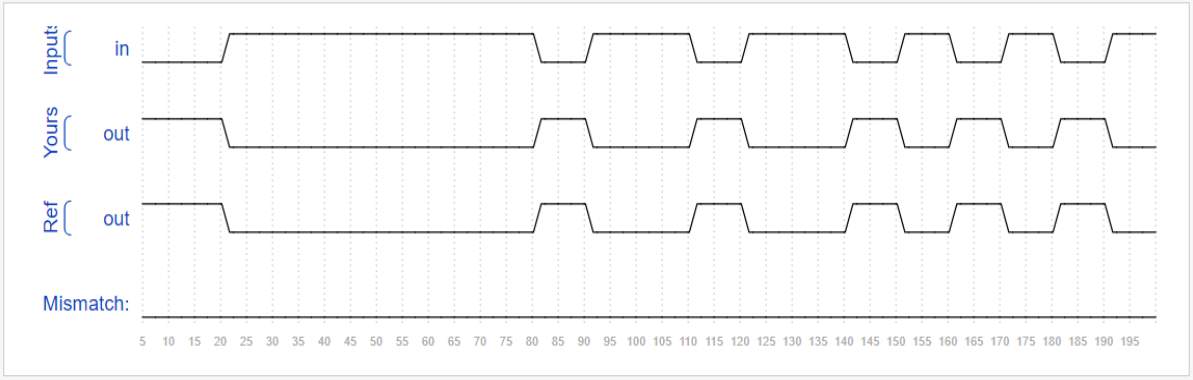
Status: Success!

You have solved 5 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).

Inversion



(6) Create a module that implements an AND gate.

Write your solution here

[Load a previous submission] Load

```
module top_module(
    input a,
    input b,
    output out );

    assign out=a&b;

endmodule

```

Submit | Submit (new window)

Upload a source file...

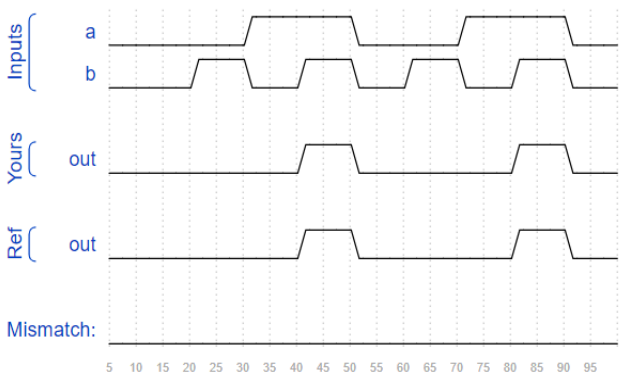
Status: Success!

You have solved 6 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).

AND gate



(7) Create a module that implements a NOR gate. A NOR gate is an OR gate with its output inverted. A NOR function needs two operators when written in Verilog.

Write your solution here

[Load a previous submission] Load

```
module top_module(
    input a,
    input b,
    output out );

    assign out=~(a|b);

endmodule

```

Submit | Submit (new window)

Upload a source file...

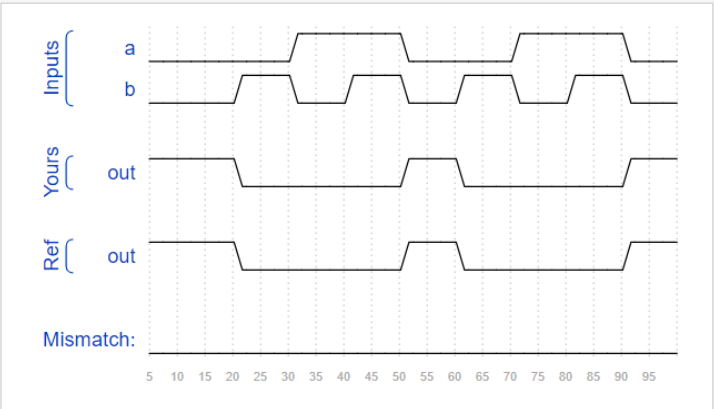
Status: Success!

You have solved 7 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).

NOR gate



(8) Create a module that implements an XNOR gate.

Write your solution here

[Load a previous submission] Load

```
module top_module(
    input a,
    input b,
    output out );

    assign out=~(a^b);

endmodule

```

Submit | Submit (new window)

Upload a source file...

Status: Success!

You have solved 8 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).

XNOR gate