Vectorr ○

← vector3 ✓ (/wiki/vector3)

vector4 ○ (/wiki/vector4) →

Given an 8-bit input vector [7:0], reverse its bit ordering.

See also: Reversing a longer vector (/wiki/Vector100r).

Module Declaration

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

Hint...

- `assign out[7:0] = in[0:7];` does not work because Verilog does not allow vector bit ordering to be flipped.
- The concatenation operator may save a bit of coding, allowing for 1 assign statement instead of 8.

Write your solution here

[Load a previous submission] ▼ Load

```
module top_module(
    input [7:0] in,
    output [7:0] out
);
    assign out = {in[0],in[1],in[2],in[3],in[4],in[5],in[6],in[7]};
    //assign out[7:0] = in[0:7];

endmodule

```

Submit Submit (new window)

Upload a source file... ⌄

Solution

Show solution

```
module top_module (
    input [7:0] in,
    output [7:0] out
```

```
);

    assign {out[0],out[1],out[2],out[3],out[4],out[5],out[6],out[

    /*
    // I know you're dying to know how to use a loop to do this:

    // Create a combinational always block. This creates combinat
    // as sequential code. for-loops describe circuit *behaviour*
    // inside procedural blocks (e.g., always block).
    // The circuit created (wires and gates) does NOT do any iter
    // AS IF the iteration occurred. In reality, a logic synthesi
    // figure out what circuit to produce. (In contrast, a Verilo
    // during simulation.)
    always @(*) begin
        for (int i=0; i<8; i++) // int is a SystemVerilog type. U
            out[i] = in[8-i-1];
    end


    // It is also possible to do this with a generate-for loop. G
    // but are quite different in concept, and not easy to unders
    // of "things" (Unlike procedural loops, it doesn't describe
    // module instantiations, net/variable declarations, and proc
    // a procedure). Generate loops (and genvars) are evaluated e
    // blocks as a form of preprocessing to generate more code, w
    // In the example below, the generate-for loop first creates
    // synthesized.
    // Note that because of its intended usage (generating code a
    // on how you use them. Examples: 1. Quartus requires a gener
    // attached (in this example, named "my_block_name"). 2. Insi
    generate
        genvar i;
        for (i=0; i<8; i = i+1) begin: my_block_name
            assign out[i] = in[8-i-1];
        end
    endgenerate
    */

endmodule

```

vectorr — Compile and simulate

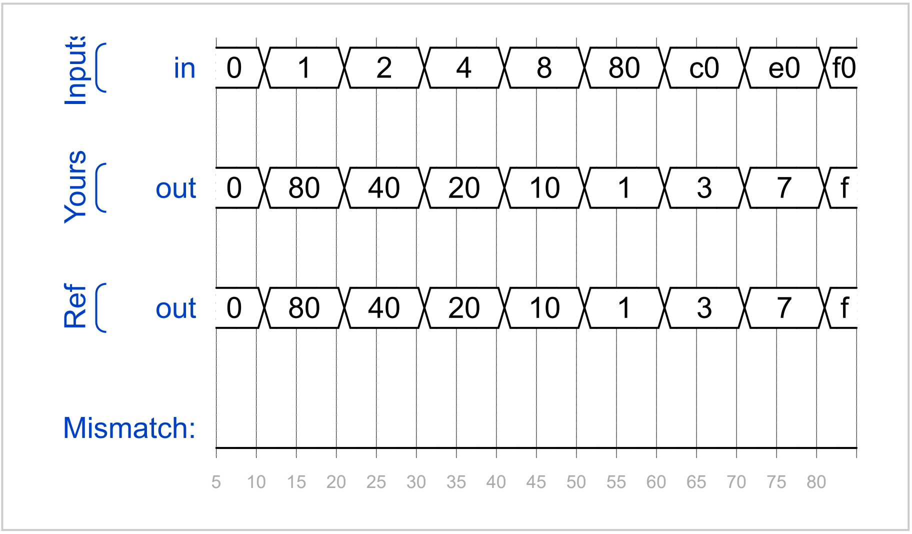
Running Quartus synthesis. Show Quartus messages...
Running ModelSim simulation. Show Modelsim messages...

Status: Success!

You have solved 17 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).



🡐 vector3 ✓ (/wiki/vector3)

vector4 ○ (/wiki/vector4) 🡒

Problem Set Contents