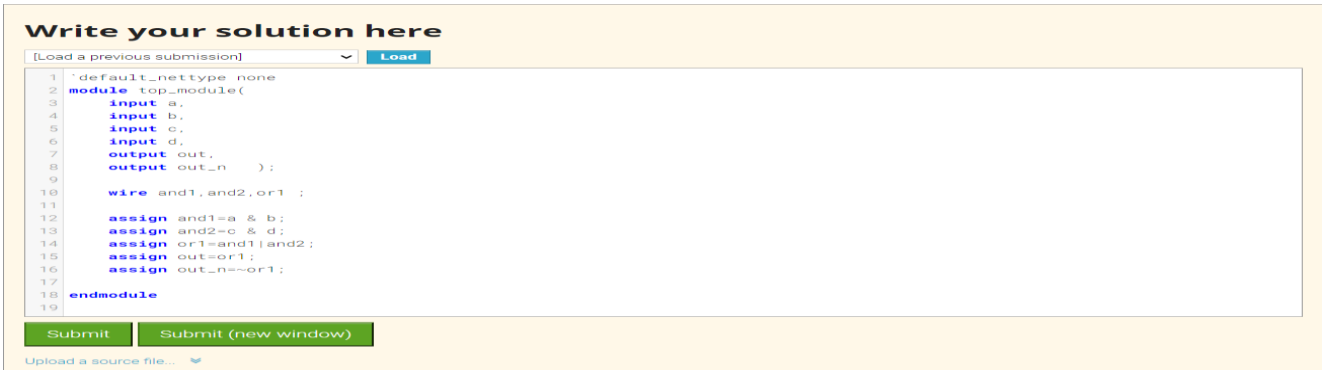
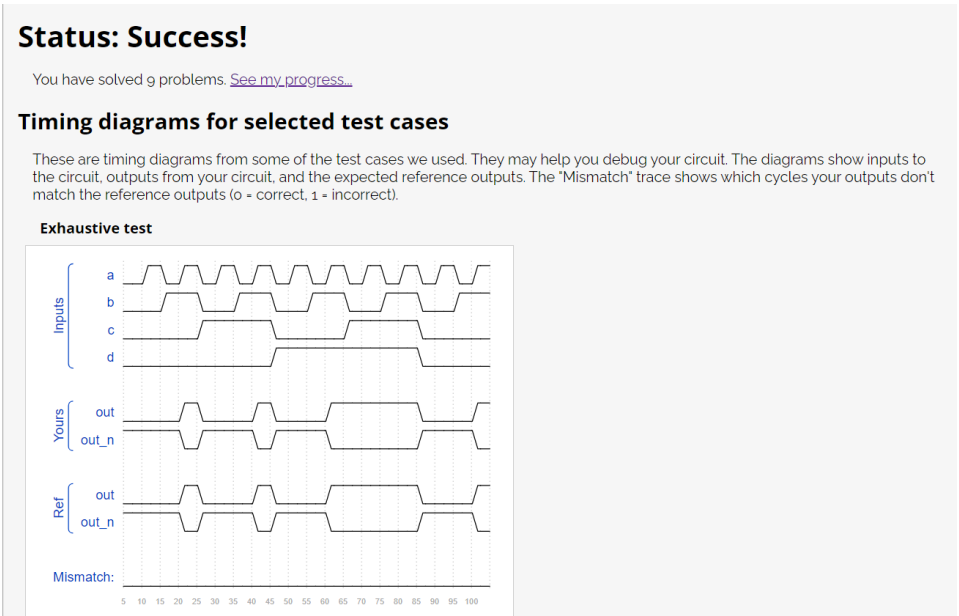
DAY-2
#100DAYSOFRTL

Problem statement :-

1. Implement the following circuit. Create two intermediate wires (named anything you want) to connect the AND and OR gates together. Note that the wire that feeds the NOT gate is really wire `out`, so you do not necessarily need to declare a third wire here. Notice how wires are driven by exactly one source (output of a gate), but can feed multiple inputs.
   If you're following the circuit structure in the diagram, you should end up with four assign statements, as there are four signals that need a value assigned.





2. The 7458 is a chip with four AND gates and two OR gates. This problem is slightly more complex than 7420.
   Create a module with the same functionality as the 7458 chip. It has 10 inputs and 2 outputs. You may choose to use an `assign` statement to drive each of the output wires, or you may choose to declare (four) wires for use as intermediate signals, where each internal wire is driven by the output of one of the AND gates. For extra practice, try it both ways.

Write your solution here

[Load a previous submission] Load

```
module top_module (
    input p1a, p1b, p1c, p1d, p1e, p1f,
    output p1y,
    input p2a, p2b, p2c, p2d,
    output p2y );

    wire i1,i2,i3,i4,i5,i6;

    assign i1=p2a & p2b;
    assign i2=p2c & p2d;
    assign i3=p1a & p1c & p1b;
    assign i4= p1f & p1e & p1d;
    assign i5=i1 | i2;
    assign i6=i3 | i4;

    assign p2y=i5;
    assign p1y=i6;


endmodule

```

Submit | Submit (new window)

Upload a source file...

7458 — Compile and simulate

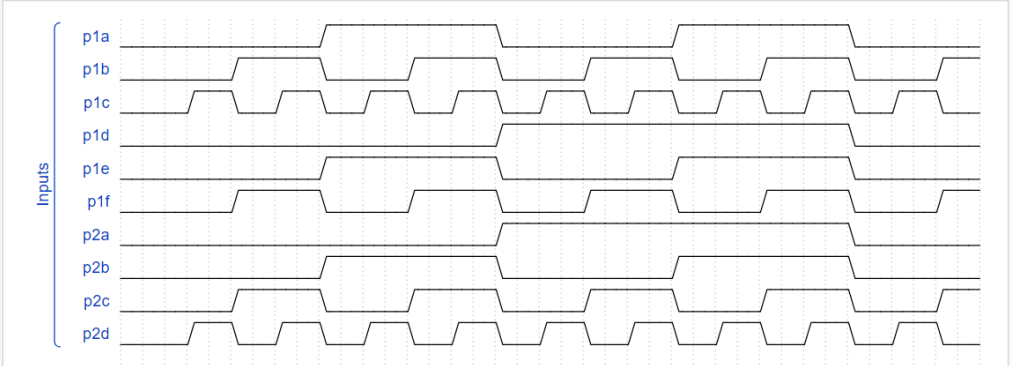
Running Quartus synthesis. Show Quartus messages...
Running ModelSim simulation. Show Modelsim messages...

Status: Success!

You have solved 10 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. Vectors are used to group related signals using one name to make it more convenient to manipulate. For example, `wire [7:0] w;` declares an 8-bit vector named `w` that is functionally equivalent to having 8 separate wires.

   Notice that the *declaration* of a vector places the dimensions *before* the name of the vector, which is unusual compared to C syntax. However, the *part select* has the dimensions *after* the vector name as you would expect.

Write your solution here

[Load a previous submission] Load

```
1  module top_module (
2      input wire [2:0] vec,
3      output wire [2:0] outv,
4      output wire o2,
5      output wire o1,
6      output wire o0  ); // Module body starts after module declaration
7
8      assign o0=vec[0];
9      assign o1=vec[1];
10     assign o2=vec[2];
11     assign outv=vec;
12
13 endmodule
14
```

Submit | Submit (new window)

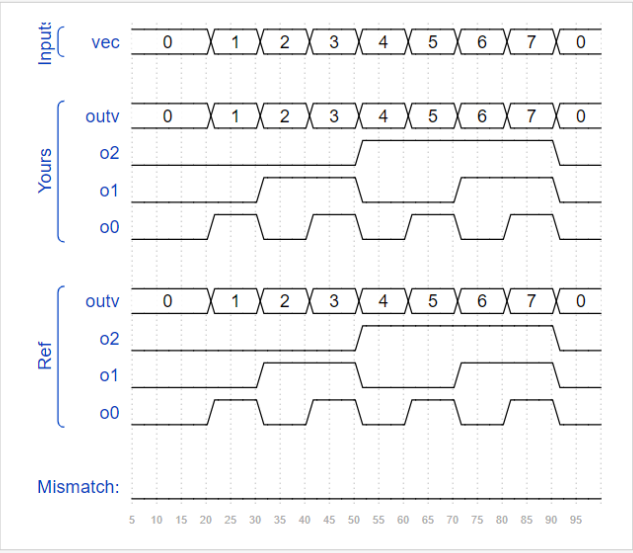
Upload a source file...

Status: Success!

You have solved 11 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).



4. Build a combinational circuit that splits an input half-word (16 bits, [15:0]) into lower [7:0] and upper [15:8] bytes.

Write your solution here

[Load a previous submission] Load

```
1  `default_nettype none     // Disable implicit nets. Reduces some types of bugs.
2  module top_module(
3      input wire [15:0] in,
4      output wire [7:0] out_hi,
5      output wire [7:0] out_lo );
6
7      assign out_hi=in[15:8];
8      assign out_lo=in[7:0];
9
10
11 endmodule
12
```

Submit | Submit (new window)

Upload a source file...

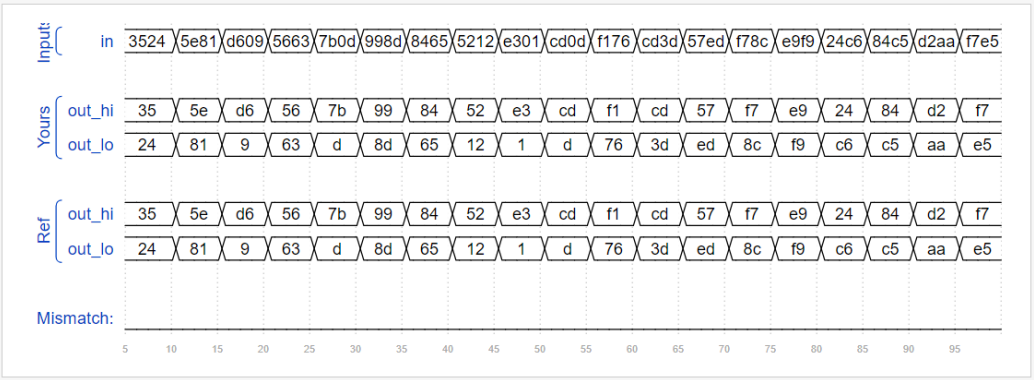
Status: Success!

You have solved 12 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).

Random inputs



5. A 32-bit vector can be viewed as containing 4 bytes (bits [31:24], [23:16], etc.). Build a circuit that will reverse the *byte* ordering of the 4-byte word.
   AaaaaaaaBbbbbbbbCcccccccDddddddd => DddddddCccccccBbbbbbbbAaaaaaaa

Write your solution here

[Load a previous submission] Load

```
1  module top_module(
2      input [31:0] in,
3      output [31:0] out );
4
5      assign out[31:24]= in[7:0];
6      assign out[23:16]= in[15:8];
7      assign out[15:8]= in[23:16];
8      assign out[7:0]= in[31:24];
9
10
11 endmodule
12
```

Submit Submit (new window)

Upload a source file...

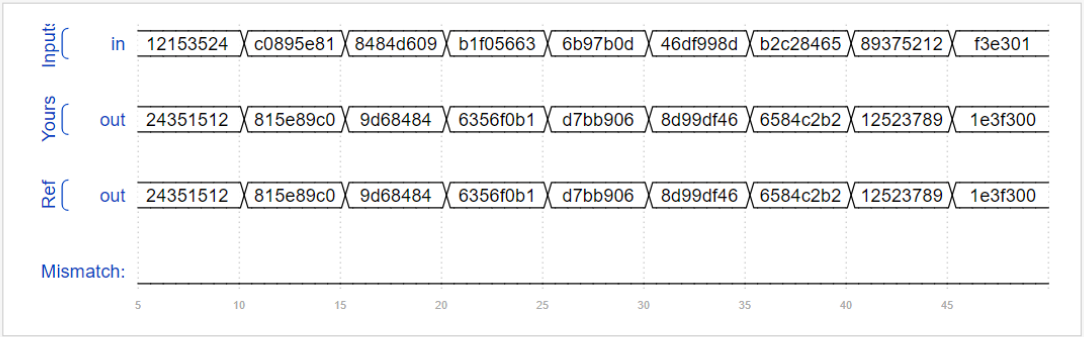
Status: Success!

You have solved 13 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).

Random inputs



6. Build a circuit that has two 3-bit inputs that computes the bitwise-OR of the two vectors, the logical-OR of the two vectors, and the inverse (NOT) of both vectors. Place the inverse of b in the upper half of `out_not` (i.e., bits [5:3]), and the inverse of a in the lower half.

Write your solution here

[Load a previous submission] Load

```
module top_module(
    input [2:0] a,
    input [2:0] b,
    output [2:0] out_or_bitwise,
    output out_or_logical,
    output [5:0] out_not
);

    assign out_or_bitwise[2:0]=a[2:0] | b[2:0];
    assign out_or_logical=a[2:0] || b[2:0];
    assign out_not[5:3]=~b[2:0];
    assign out_not[2:0]=~a[2:0];

endmodule

```

Submit Submit (new window)

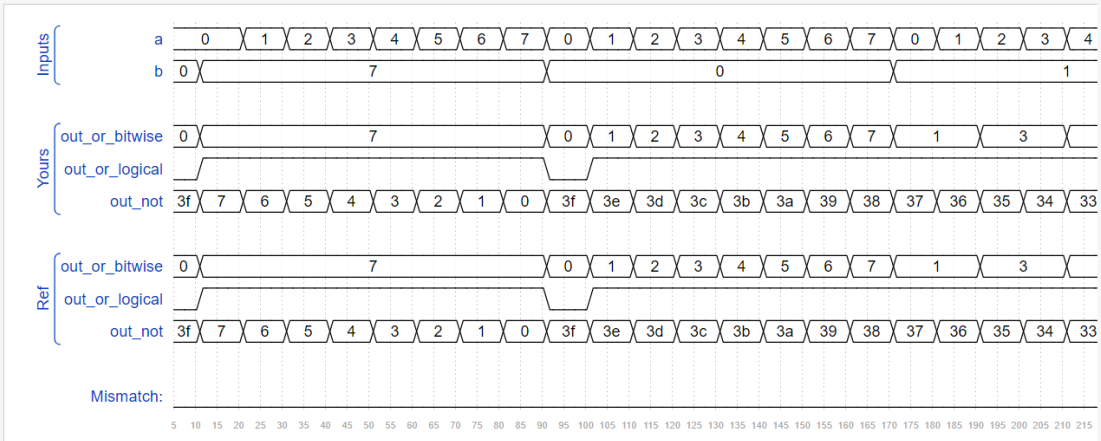
Upload a source file...

Status: Success!

You have solved 14 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).



7. Build a combinational circuit with four inputs, `in[3:0]`.

There are 3 outputs:

- out_and: output of a 4-input AND gate.
- out_or: output of a 4-input OR gate.
- out_xor: output of a 4-input XOR gate.

Write your solution here

[Load a previous submission] Load

```
module top_module(
    input [3:0] in,
    output out_and,
    output out_or,
    output out_xor
);

    assign out_and=in[0] && in[1] && in[2] && in[3];
    assign out_or= in[0] || in[1] || in[2] || in[3];
    assign out_xor= in[0] ^ in[1] ^ in[2] ^ in[3];


endmodule

```

Submit | Submit (new window)

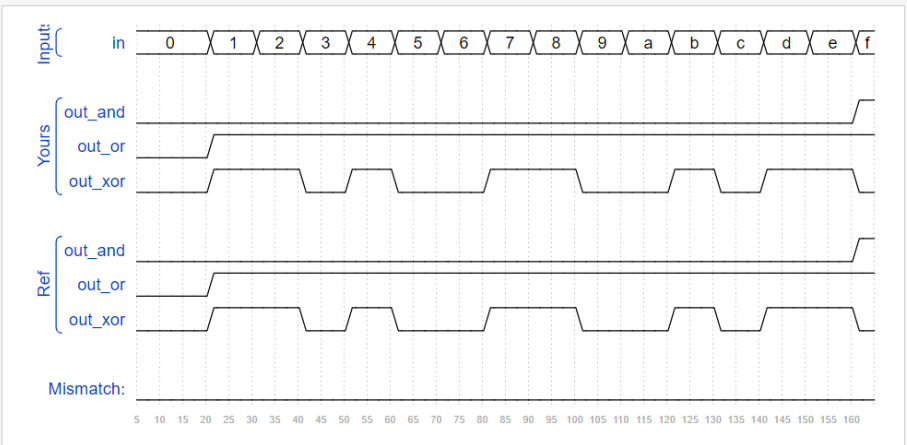
Status: Success!

You have solved 15 problems. See my progress..

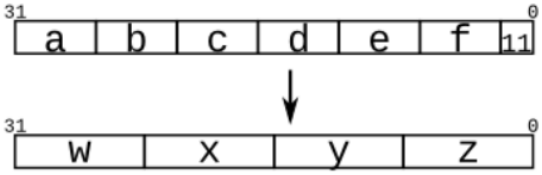
Timing diagrams for selected test cases

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).

All combinations



8. Given several input vectors, concatenate them together then split them up into several output vectors. There are six 5-bit input vectors: a, b, c, d, e, and f, for a total of 30 bits of input. There are four 8-bit output vectors: w, x, y, and z, for 32 bits of output. The output should be a concatenation of the input vectors followed by two 1 bits:



Write your solution here

[Load a previous submission] Load

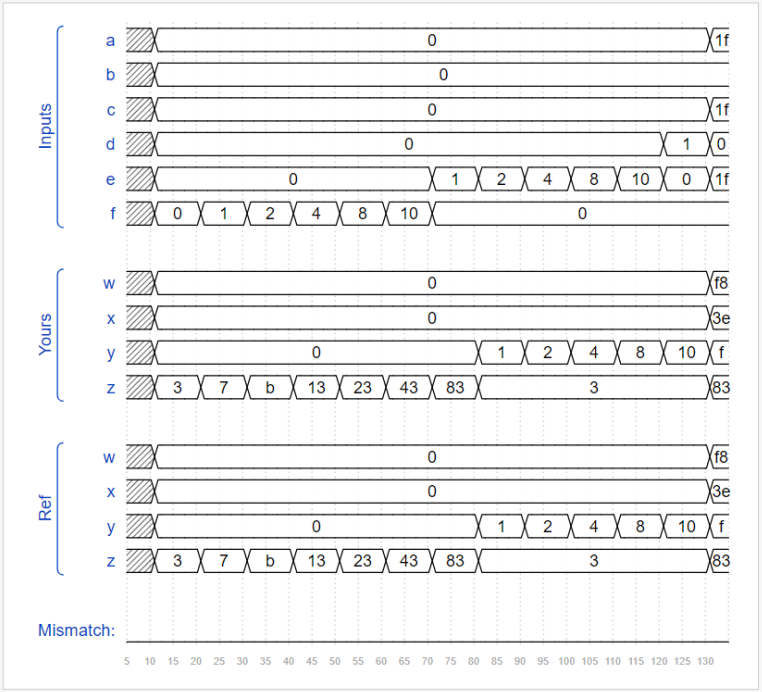
```
module top_module (
    input [4:0] a, b, c, d, e, f,
    output [7:0] w, x, y, z );

    assign {w,x,y,z}={a,b,c,d,e,f,2'b11};

endmodule
```

Submit | Submit (new window)

Upload a source file...



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

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]};

endmodule

```

Submit Submit (new window)

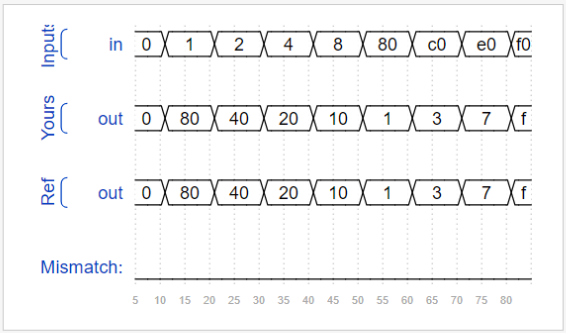
Upload a source file...

Status: Success!

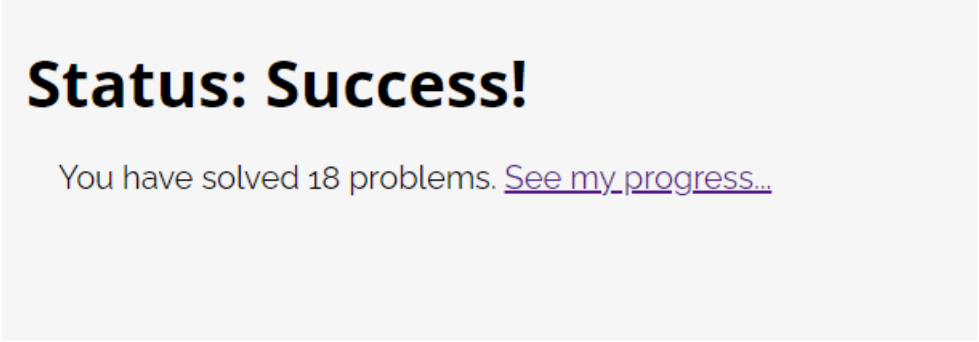
You have solved 17 problems. See my progress...

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).



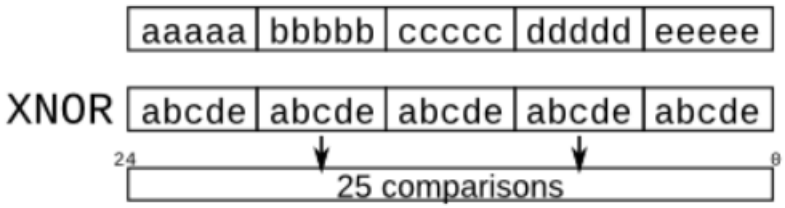
10. Build a circuit that sign-extends an 8-bit number to 32 bits. This requires a concatenation of 24 copies of the sign bit (i.e., replicate bit[7] 24 times) followed by the 8-bit number itself.



11. Given five 1-bit signals (a, b, c, d, and e), compute all 25 pairwise one-bit comparisons in the 25-bit output vector. The output should be 1 if the two bits being compared are equal.

```
out[24] = ~a ^ a;   // a == a, so out[24] is always 1.
out[23] = ~a ^ b;
out[22] = ~a ^ c;

out[ 1] = ~e ^ d;
out[ 0] = ~e ^ e;
```



Write your solution here

[Load a previous submission] Load

```
module top_module (
    input a, b, c, d, e,
    output [24:0] out );

    assign out=~{{5{a}}, {5{b}}, {5{c}}, {5{d}}, {5{e}}} ^ {5{a,b,c,d,e}};

endmodule
```

Submit | Submit (new window)

Upload a source file...

Status: Success!

You have solved 19 problems. See my progress...

Warning messages that may be important

Quartus messages (Show all)

Warning (13024): Output pins are stuck at VCC or GND

This warning says that an output pin never changes (is "stuck"). This can sometimes indicate a bug if the output pin shouldn't be a constant. If this pin is not supposed to be constant, check for bugs that cause the value being assigned to never change (e.g., `assign a = x & ~x;`)