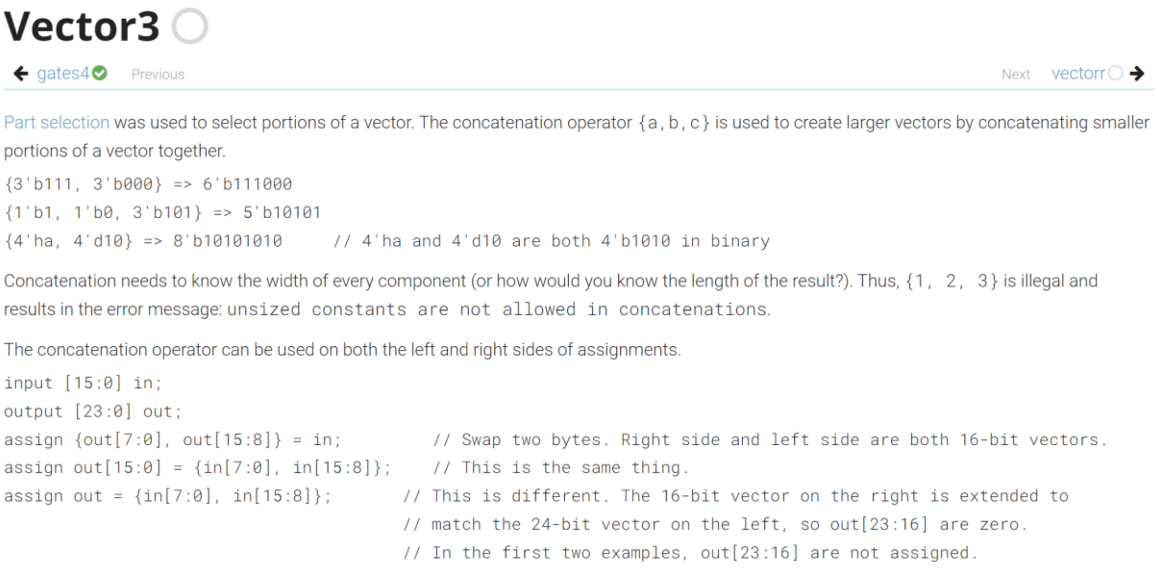


Vector3 ○

← gates4 ✓ Previous Next vectorr ○ →

Part selection was used to select portions of a vector. The concatenation operator {a,b,c} is used to create larger vectors by concatenating smaller portions of a vector together.

```
{3'b111, 3'b000} => 6'b111000
{1'b1, 1'b0, 3'b101} => 5'b10101
{4'ha, 4'd10} => 8'b10101010     // 4'ha and 4'd10 are both 4'b1010 in binary
```

Concatenation needs to know the width of every component (or how would you know the length of the result?). Thus, {1, 2, 3} is illegal and results in the error message: `unsized constants are not allowed in concatenations`.

The concatenation operator can be used on both the left and right sides of assignments.

```
input [15:0] in;
output [23:0] out;
assign {out[7:0], out[15:8]} = in;         // Swap two bytes. Right side and left side are both 16-bit vectors.
assign out[15:0] = {in[7:0], in[15:8]};    // This is the same thing.
assign out = {in[7:0], in[15:8]};       // This is different. The 16-bit vector on the right is extended to
                                        // match the 24-bit vector on the left, so out[23:16] are zero.
                                        // In the first two examples, out[23:16] are not assigned.
```

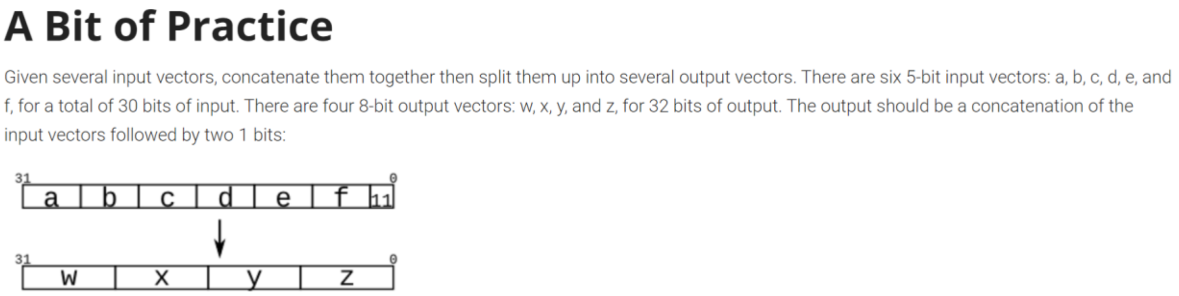
설명

1. 중괄호로 bit를 묶으면 더 큰 vector를 만들 수 있다. -> Concatenation operator

2. 다른 진수 타입의 비트를 묶어도 됨. 결과는 bit의 합과 이진수로 표현된 값이다.

주의점

- 모든 요소들의 너비를 알아야 함.



A Bit of Practice

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:

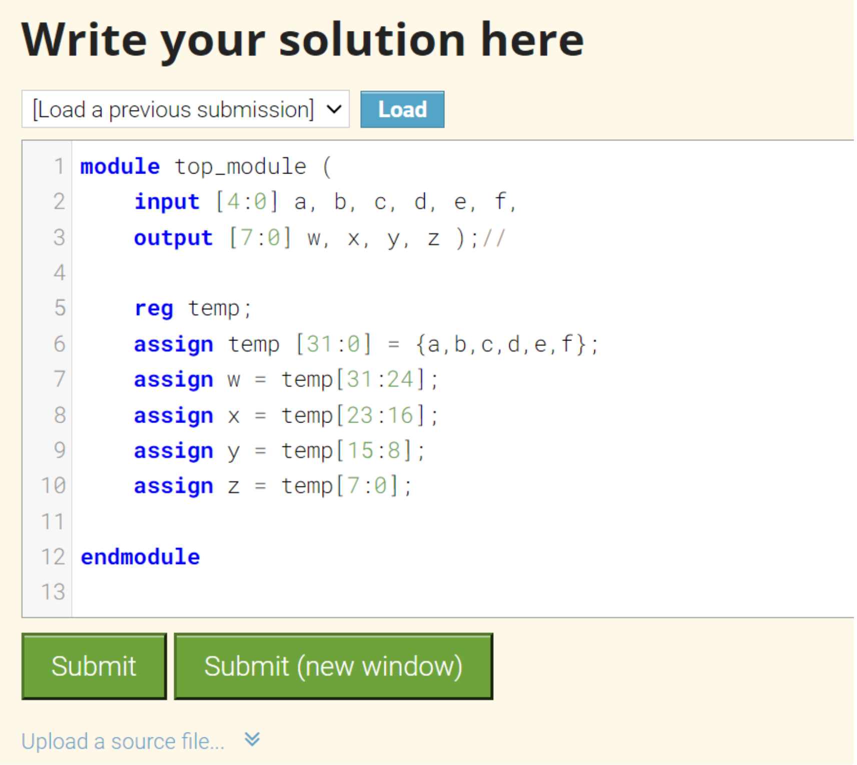
A Bit of Practice

- 몇 개의 입력 벡터가 주어지는데, 이들을 묶은 후, 몇 개의 출력 벡터로 분리해라. 여기선 5-bit를 가진 6개의 요소를 8-bit를 가진 4개의 요소로 변경하라고 했음.

풀이

요소 a,b,c,d,e,f를 한번에 묶어 32bit로 만든 후에 w,x,y,z로 분리해보자.

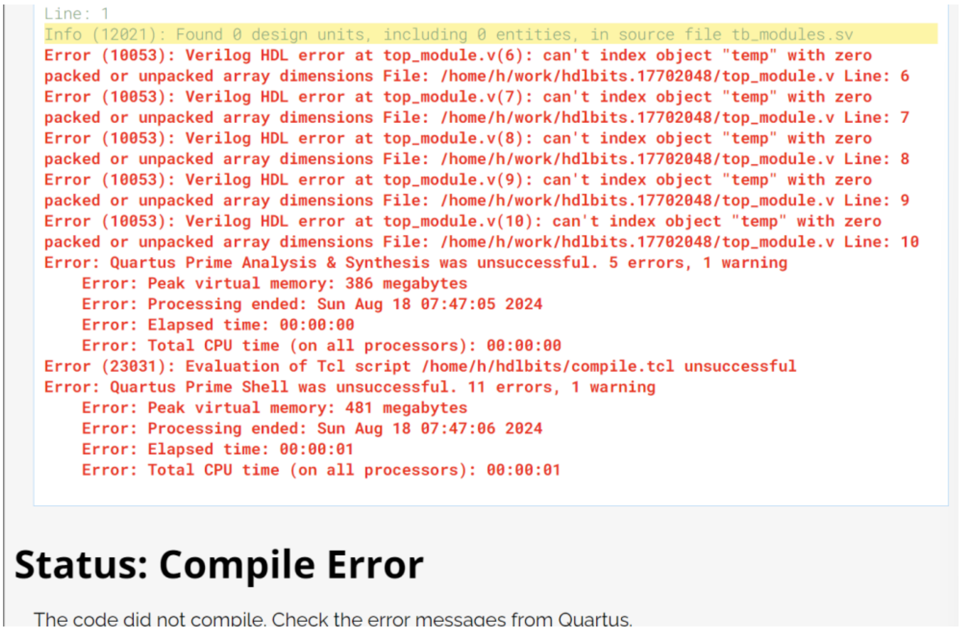
처음 작성한 코드



Write your solution here

```
 1  module top_module (
 2      input [4:0] a, b, c, d, e, f,
 3      output [7:0] w, x, y, z );//
 4
 5      reg temp;
 6      assign temp [31:0] = {a,b,c,d,e,f};
 7      assign w = temp[31:24];
 8      assign x = temp[23:16];
 9      assign y = temp[15:8];
10      assign z = temp[7:0];
11
12  endmodule
13
```

결과



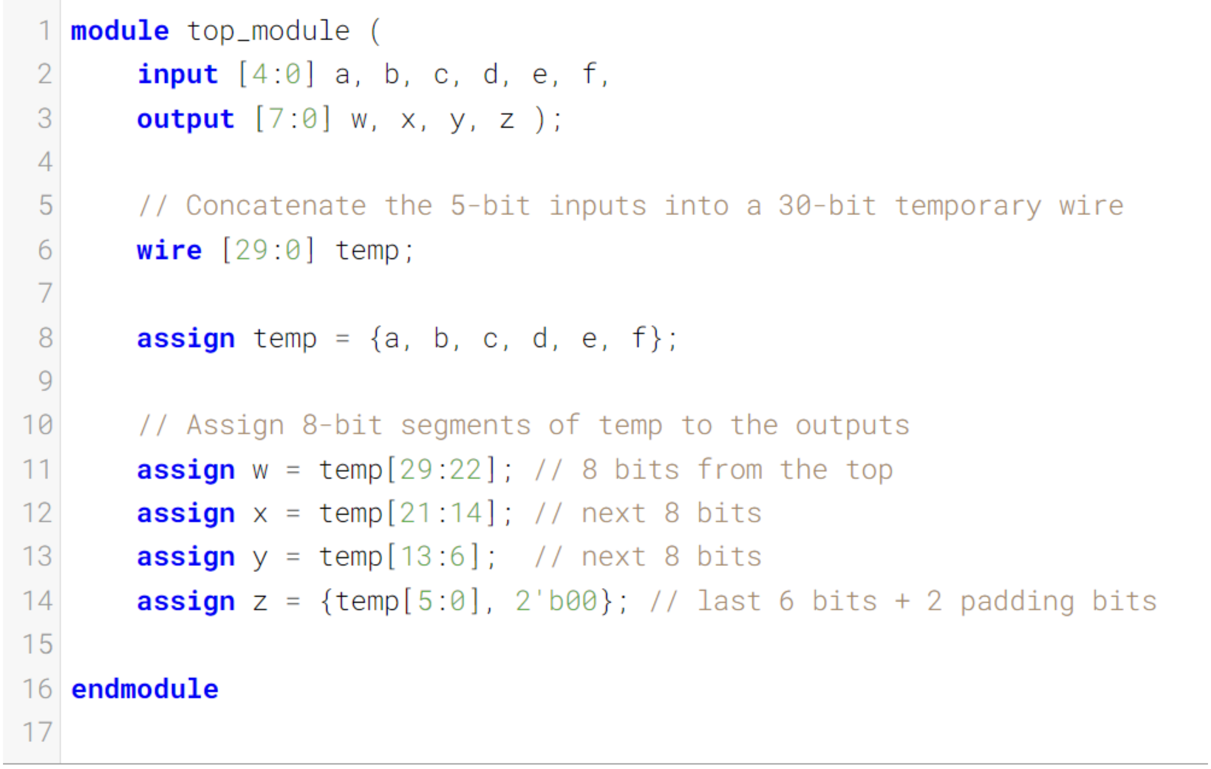
```
Line: 1
Info (12021): Found 0 design units, including 0 entities, in source file tb_modules.sv
Error (10053): Verilog HDL error at top_module.v(6): can't index object "temp" with zero
packed or unpacked array dimensions File: /home/h/work/hdlbits.17702048/top_module.v Line: 6
Error (10053): Verilog HDL error at top_module.v(7): can't index object "temp" with zero
packed or unpacked array dimensions File: /home/h/work/hdlbits.17702048/top_module.v Line: 7
Error (10053): Verilog HDL error at top_module.v(8): can't index object "temp" with zero
packed or unpacked array dimensions File: /home/h/work/hdlbits.17702048/top_module.v Line: 8
Error (10053): Verilog HDL error at top_module.v(9): can't index object "temp" with zero
packed or unpacked array dimensions File: /home/h/work/hdlbits.17702048/top_module.v Line: 9
Error (10053): Verilog HDL error at top_module.v(10): can't index object "temp" with zero
packed or unpacked array dimensions File: /home/h/work/hdlbits.17702048/top_module.v Line: 10
Error: Quartus Prime Analysis & Synthesis was unsuccessful. 5 errors, 1 warning
    Error: Peak virtual memory: 386 megabytes
    Error: Processing ended: Sun Aug 18 07:47:05 2024
    Error: Elapsed time: 00:00:00
    Error: Total CPU time (on all processors): 00:00:00
Error (23031): Evaluation of Tcl script /home/h/hdlbits/compile.tcl unsuccessful
Error: Quartus Prime Shell was unsuccessful. 11 errors, 1 warning
    Error: Peak virtual memory: 481 megabytes
    Error: Processing ended: Sun Aug 18 07:47:06 2024
    Error: Elapsed time: 00:00:01
    Error: Total CPU time (on all processors): 00:00:01
```

Status: Compile Error

The code did not compile. Check the error messages from Quartus.

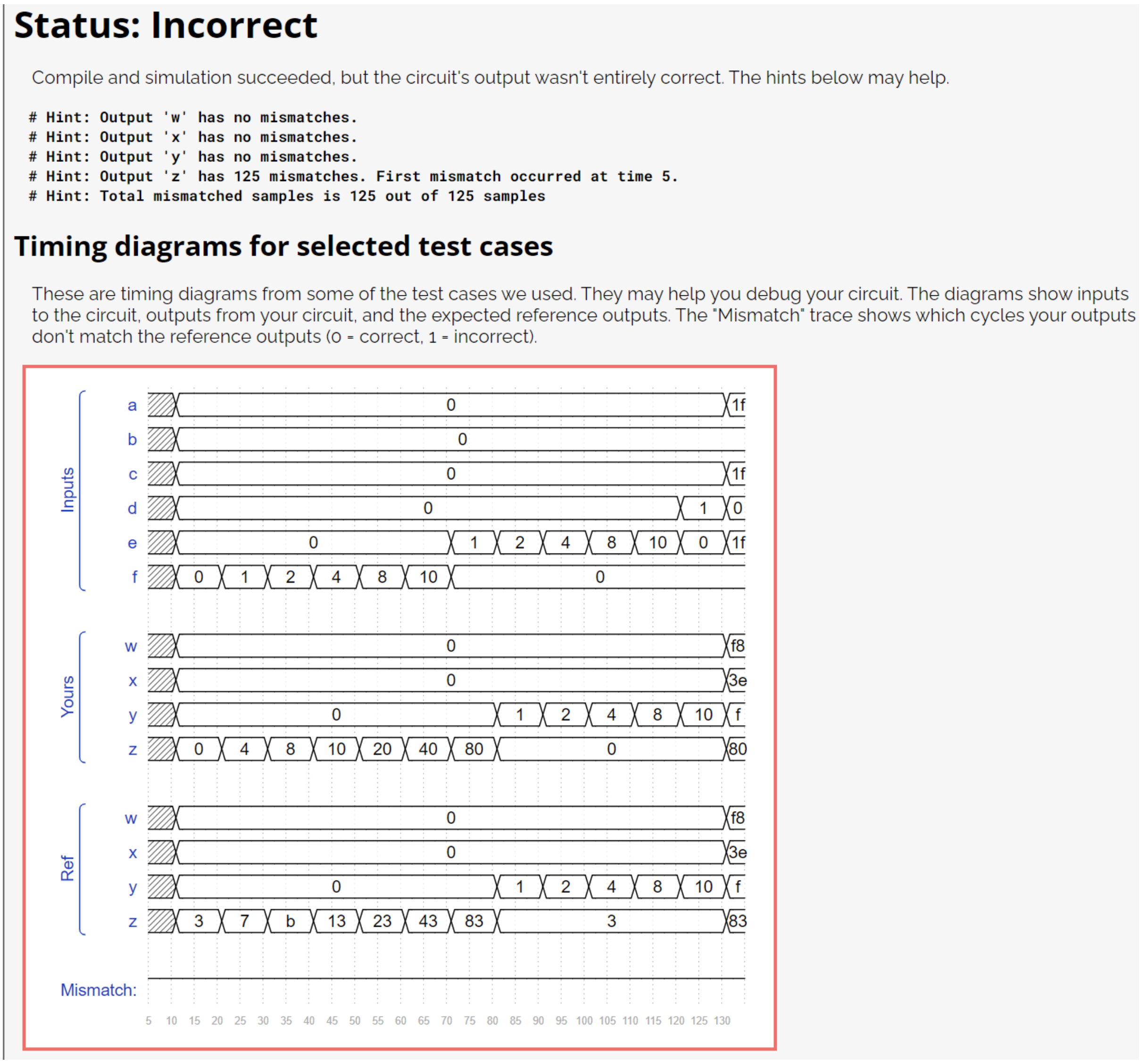
틀린 이유: assign인데 reg로 자료형을 지정했음. Wire로 선언해야 한다. 그리고 문제를 잘못 이해했음. 출력에 padding을 추가해야한다.

수정한 코드



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

    // Concatenate the 5-bit inputs into a 30-bit temporary wire
    wire [29:0] temp;

    assign temp = {a, b, c, d, e, f};

    // Assign 8-bit segments of temp to the outputs
    assign w = temp[29:22]; // 8 bits from the top
    assign x = temp[21:14]; // next 8 bits
    assign y = temp[13:6];  // next 8 bits
    assign z = {temp[5:0], 2'b00}; // last 6 bits + 2 padding bits

endmodule
```

결과



Status: Incorrect

Compile and simulation succeeded, but the circuit's output wasn't entirely correct. The hints below may help.

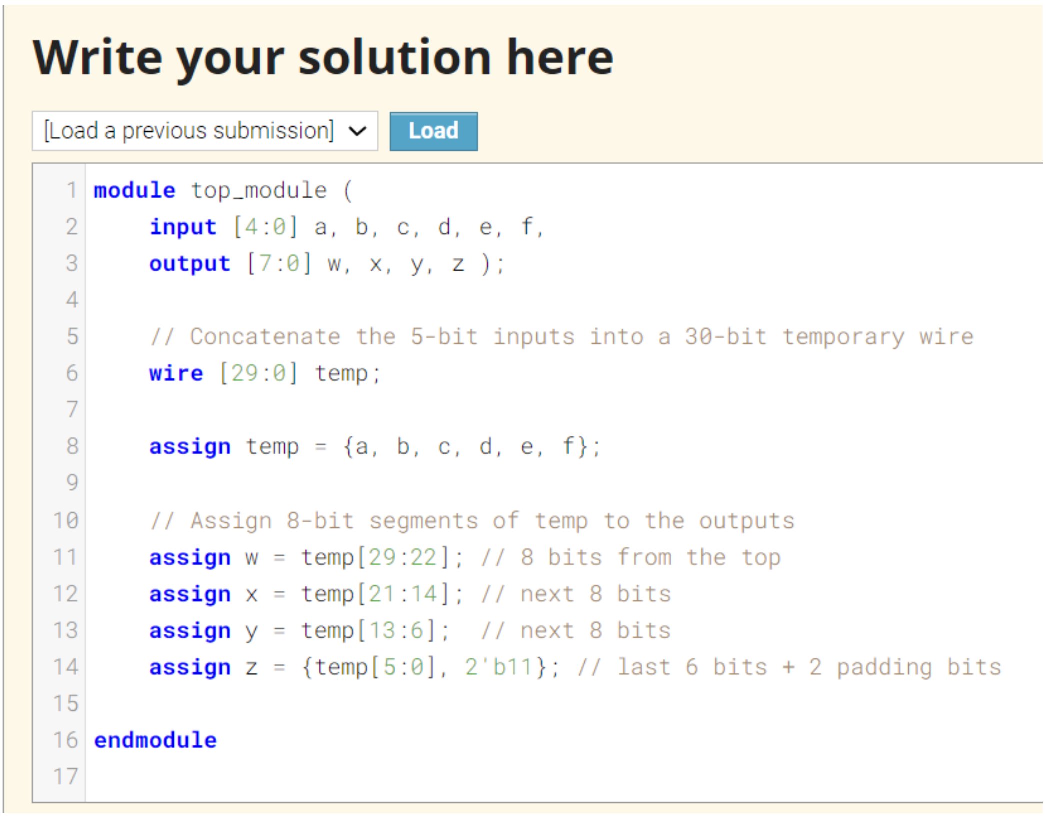
```
# Hint: Output 'w' has no mismatches.
# Hint: Output 'x' has no mismatches.
# Hint: Output 'y' has no mismatches.
# Hint: Output 'z' has 125 mismatches. First mismatch occurred at time 5.
# Hint: Total mismatched samples is 125 out of 125 samples
```

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).

반례가 발생했다. 출력 z에서 대단한 mismatch가 나왔으므로, z를 위주로 코드를 수정하자.

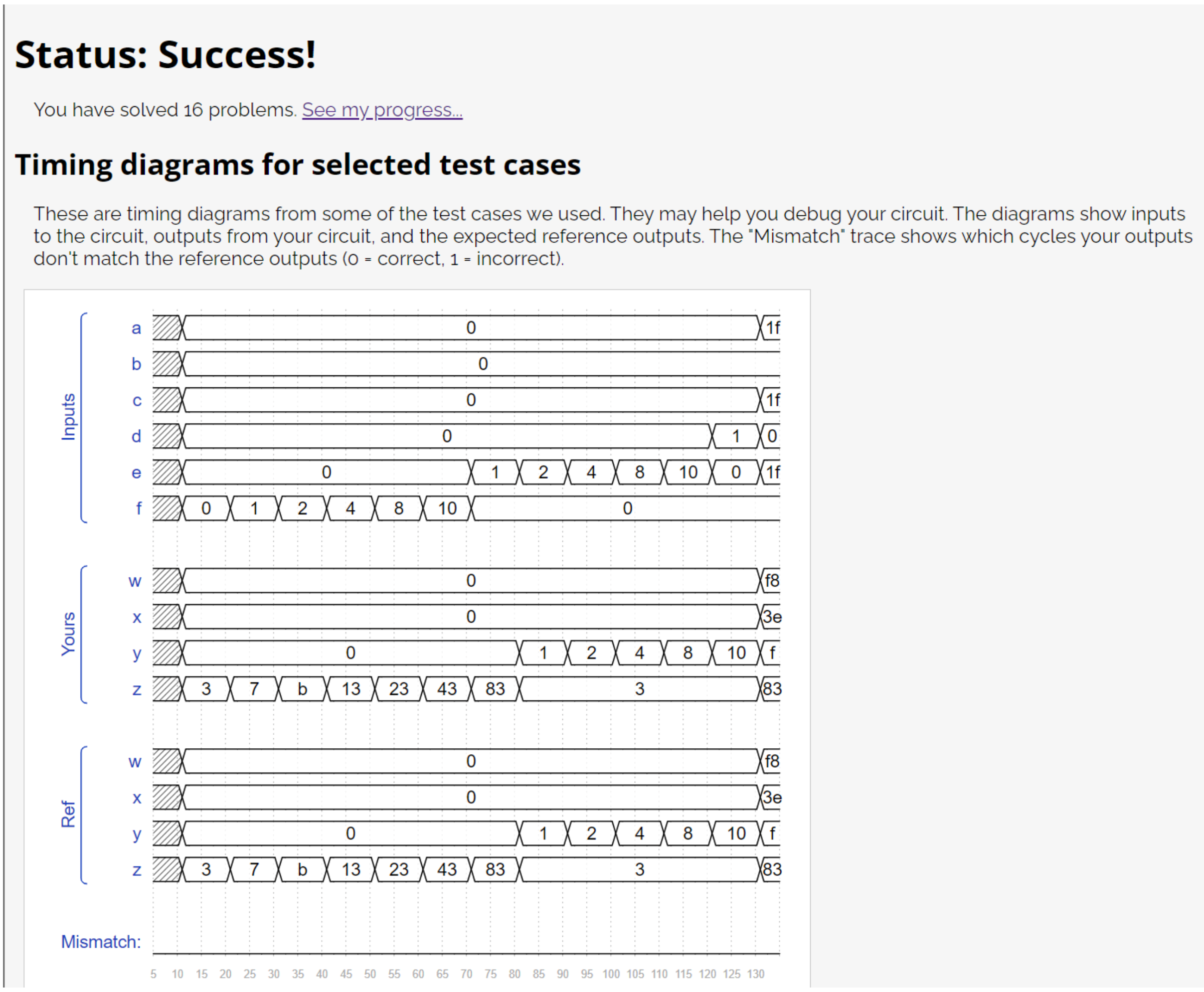
답과 내 코드는 z에서 3만큼 차이가 나게 출력이 된 걸 알 수 있다. 즉, z에 값을 할당하는 과정에서 padding을 zero가 아닌 binary 11 = 3으로 채워야 한다.



Write your solution here

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

    // Concatenate the 5-bit inputs into a 30-bit temporary wire
    wire [29:0] temp;

    assign temp = {a, b, c, d, e, f};

    // Assign 8-bit segments of temp to the outputs
    assign w = temp[29:22]; // 8 bits from the top
    assign x = temp[21:14]; // next 8 bits
    assign y = temp[13:6];  // next 8 bits
    assign z = {temp[5:0], 2'b11}; // last 6 bits + 2 padding bits

endmodule

```

결과



Status: Success!

You have solved 16 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).