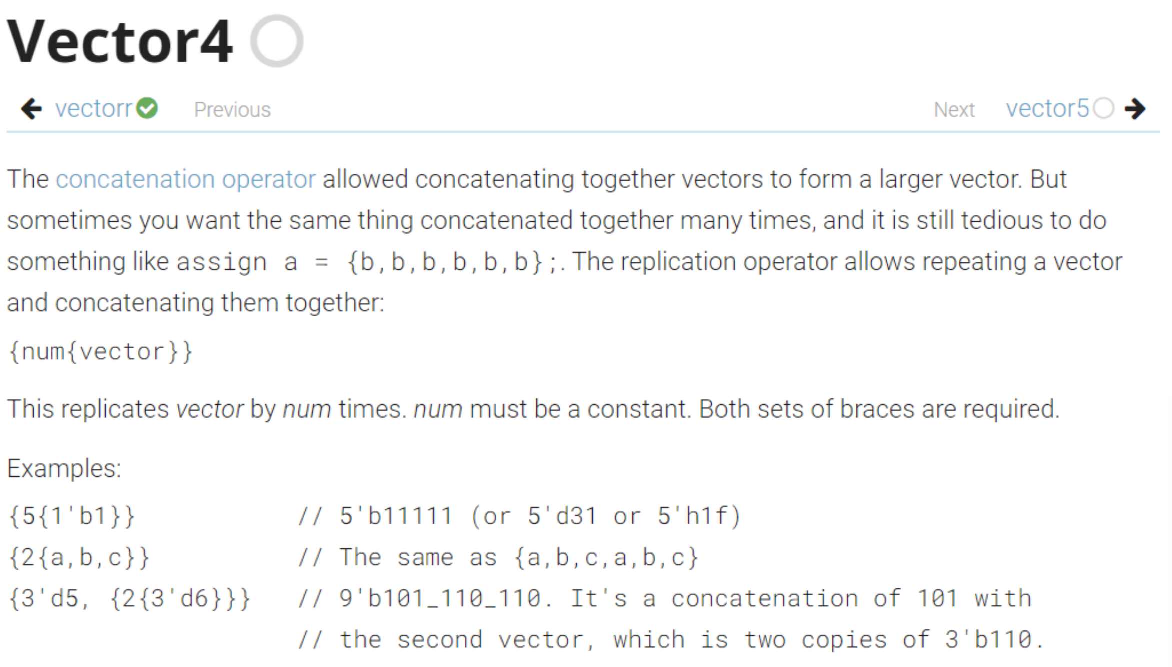


Vector4

← vectorr ✓ Previous Next vector5 ○ →

The concatenation operator allowed concatenating together vectors to form a larger vector. But sometimes you want the same thing concatenated together many times, and it is still tedious to do something like `assign a = {b,b,b,b,b,b};`. The replication operator allows repeating a vector and concatenating them together:

```
{num{vector}}
```

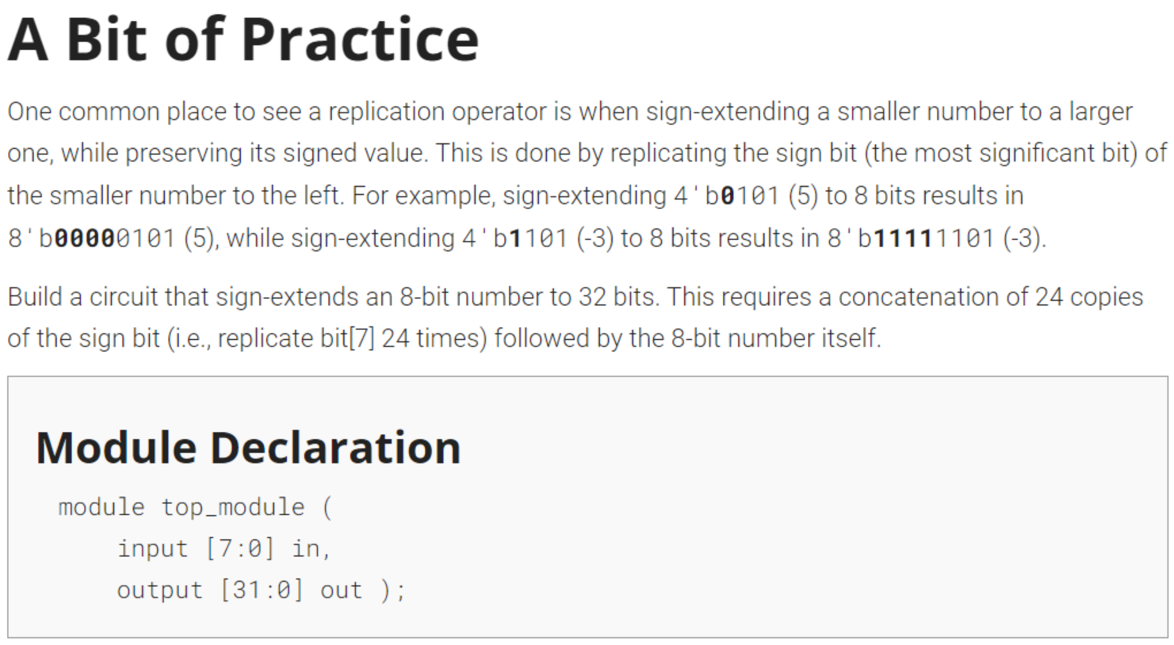
This replicates *vector* by *num* times. *num* must be a constant. Both sets of braces are required.

Examples:

```
{5{1'b1}}           // 5'b11111 (or 5'd31 or 5'h1f)
{2{a,b,c}}          // The same as {a,b,c,a,b,c}
{3'd5, {2{3'd6}}}   // 9'b101_110_110. It's a concatenation of 101 with
                    // the second vector, which is two copies of 3'b110.
```

설명

- 새로운 방법을 사용해 concatenation operator를 사용해보자



A Bit of Practice

One common place to see a replication operator is when sign-extending a smaller number to a larger one, while preserving its signed value. This is done by replicating the sign bit (the most significant bit) of the smaller number to the left. For example, sign-extending 4'b**0**101 (5) to 8 bits results in 8'b**0000**0101 (5), while sign-extending 4'b**1**101 (-3) to 8 bits results in 8'b**1111**1101 (-3).

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.

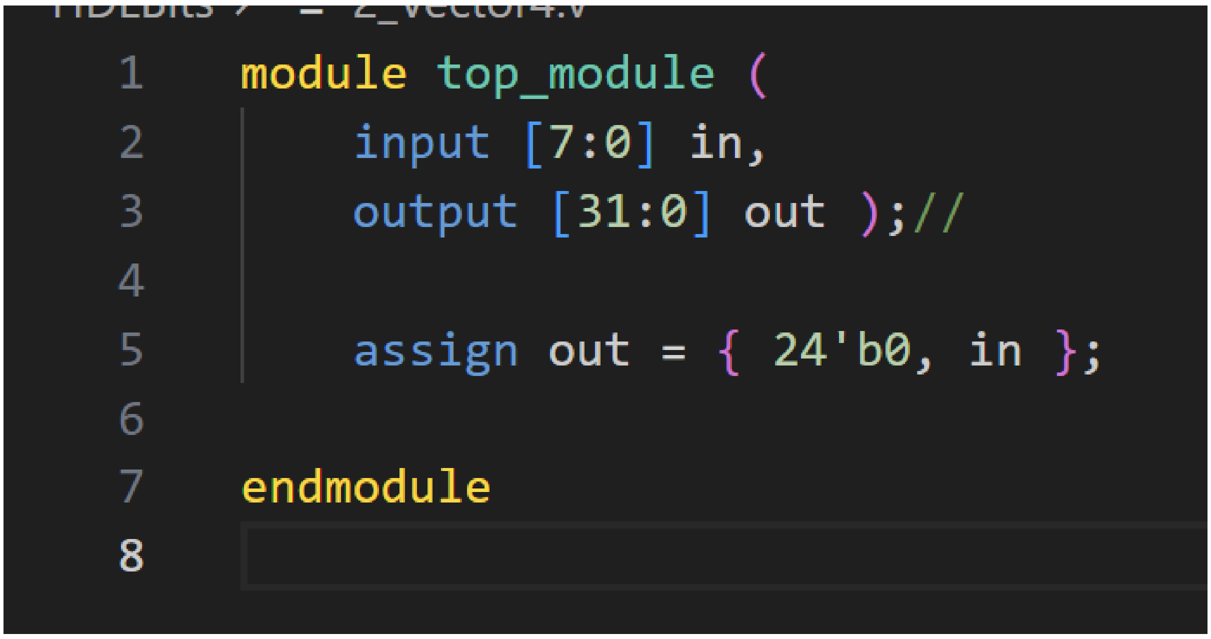
Module Declaration

```
module top_module (
    input [7:0] in,
    output [31:0] out );
```

A bit of practice

- 8-bit 입력을 32-bit 입력으로 확장. 근데 양수 음수를 구분할 수 있게 해야한다.

먼저 테스트 코드를 입력해봤다.



```
module top_module (
    input [7:0] in,
    output [31:0] out );//

    assign out = { 24'b0, in };

endmodule

```

vector4 — Compile and simulate

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

Status: Incorrect

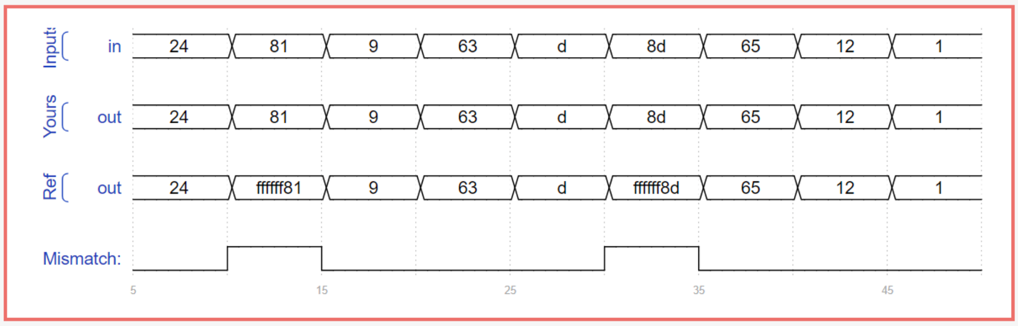
Compile and simulation succeeded, but the circuit's output wasn't entirely correct. The hints below may help.

```
# Hint: Output 'out' has 47 mismatches. First mismatch occurred at time 10.
# Hint: Total mismatched samples is 47 out of 98 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).

Near first mismatch at time 10



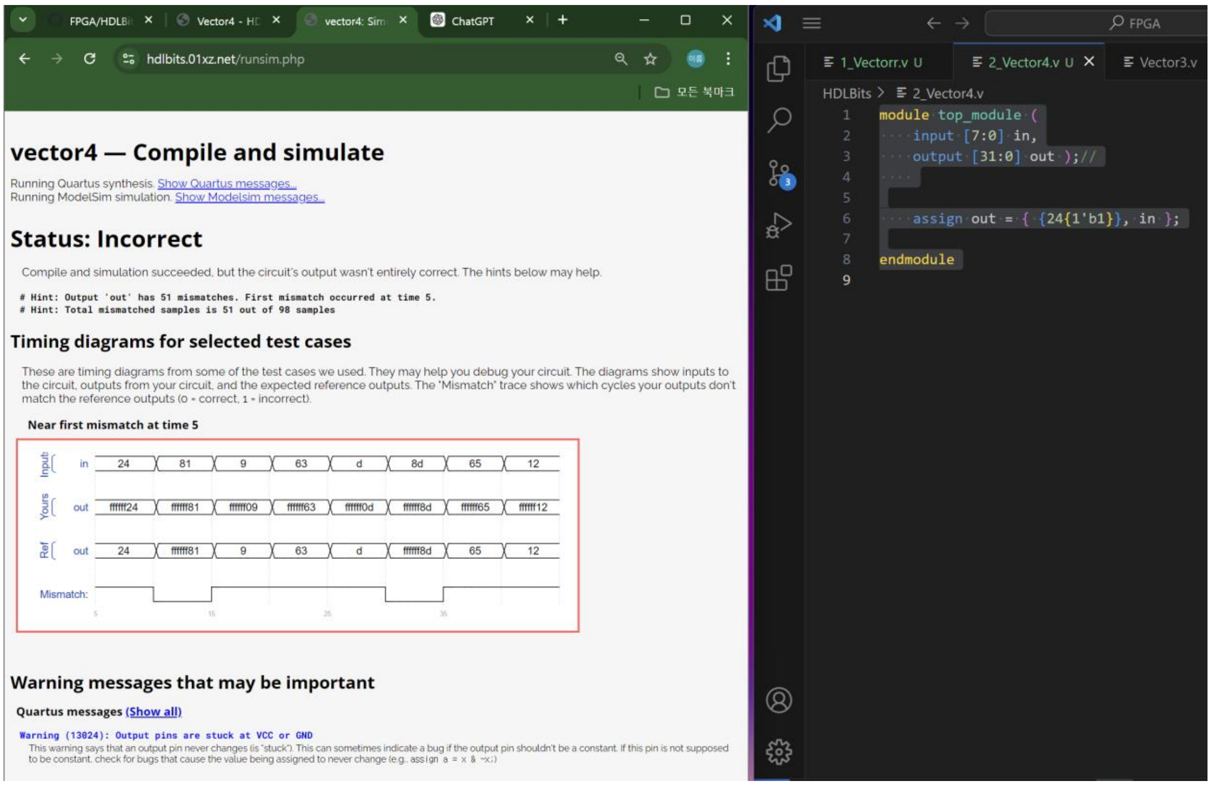
Warning messages that may be important

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;`)

이와 같이 오류가 나온 것을 확인할 수 있었다. 부호 반전만 추가하면 코드가 완성될 것이다.



vector4 — Compile and simulate

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

Status: Incorrect

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

```
# Hint: Output 'out' has 51 mismatches. First mismatch occurred at time 5.
# Hint: Total mismatched samples is 51 out of 98 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).

Near first mismatch at time 5

Inputs in 24 81 9 63 d 8d 65 12
Yours out ffffff24 ffffff81 ffffff09 ffffff63 ffffff0d ffffff8d ffffff65 ffffff12
Ref out 24 ffffff81 9 63 d ffffff8d 65 12
Mismatch:
5 15 25 35

Warning messages that may be important

Quartus messages (Show all)

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

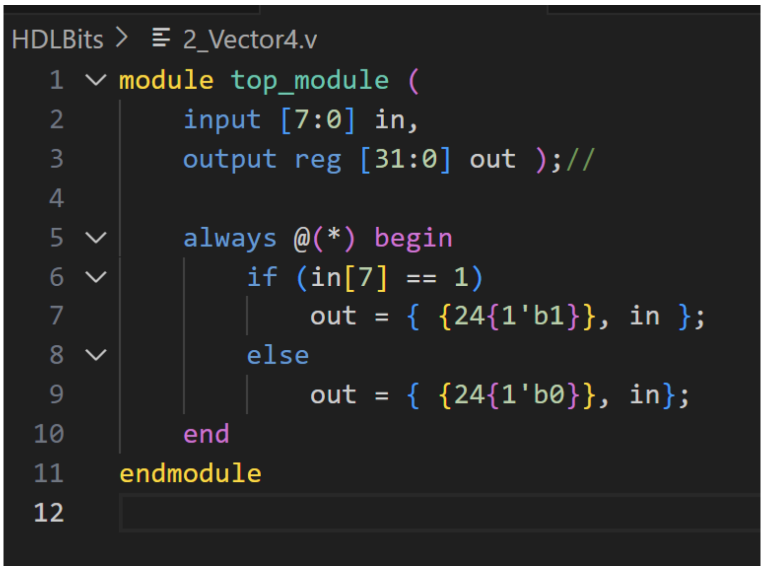
This warning says that an output pin never changes (is "stuck"). This can sometimes indicate a bug if the output pin shouldn't be a constant. If this pin is not supposed to be constant, check for bugs that cause the value being assigned to never change (e.g. `assign a = x & ~x;`)

```verilog
module top_module (
    input [7:0] in,
    output [31:0] out );//
    

    assign out = { {24{1'b1}}, in };

endmodule
```

부호를 반전하여 테스트를 해보니, 이전 값과 상반된 결과를 얻을 수 있었다.

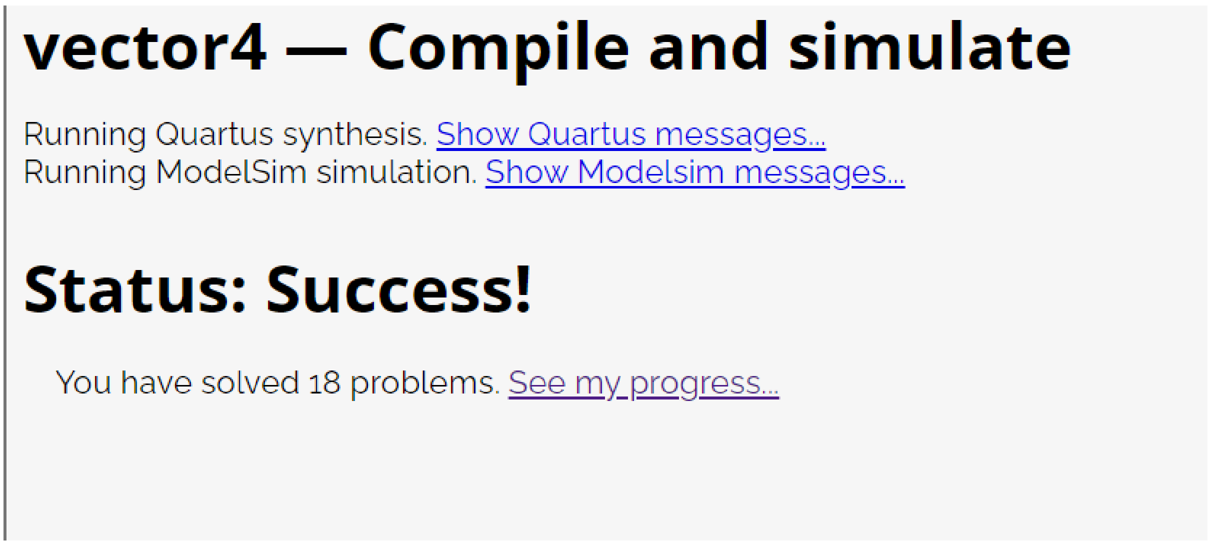
Input의 MSB가 1이면 부호 반전이 되도록 조건문을 추가한다.

최종 코드



```verilog
module top_module (
    input [7:0] in,
    output reg [31:0] out );//

    always @(*) begin
        if (in[7] == 1)
            out = { {24{1'b1}}, in };
        else
            out = { {24{1'b0}}, in};
    end
endmodule
```

결과



vector4 — Compile and simulate

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

Status: Success!

You have solved 18 problems. See my progress...

테이블이 출력되지 않았다.

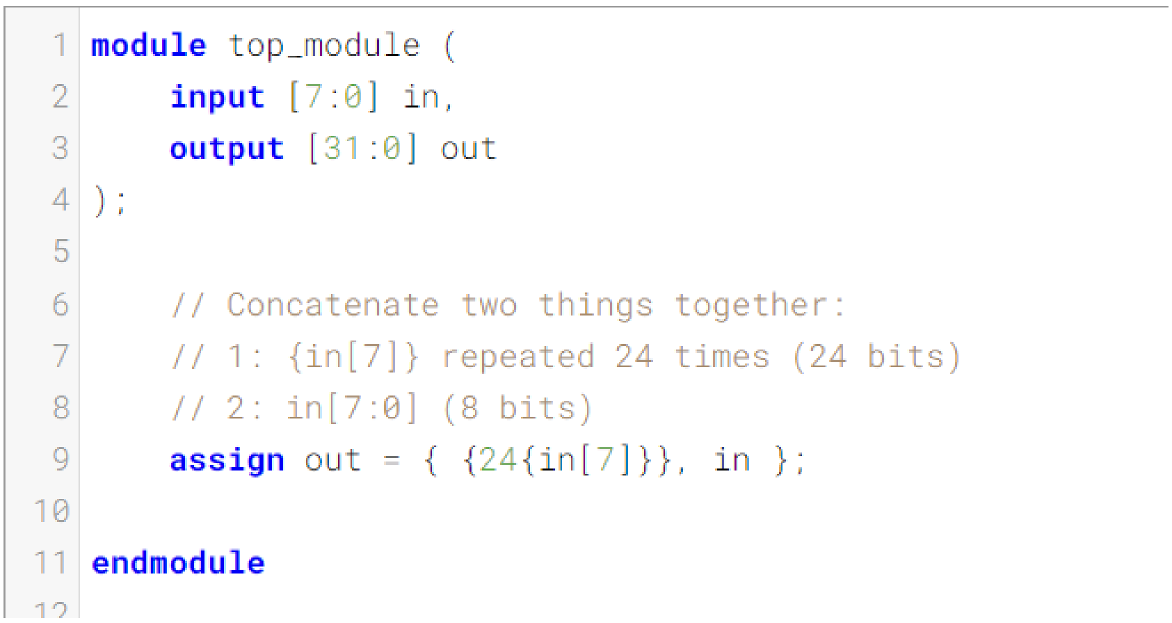
내가 틀렸던 점은,

1. always 내부 할당은 reg로 받아야 하는 것.

2. always는 내부에서 assgin을 쓰지 않는다는 것. Assign은 연속 할당문으로, wire를 연결하는 것이다. 값을 할당하는 것이 아니라..

Solution

Solution



```verilog
module top_module (
    input [7:0] in,
    output [31:0] out
);

    // Concatenate two things together:
    // 1: {in[7]} repeated 24 times (24 bits)
    // 2: in[7:0] (8 bits)
    assign out = { {24{in[7]}}, in };

endmodule
```

굳이 if로 bit를 지정하지 않아도, 입력 값의 MSB를 따라가는 특성을 이용해 코드를 짰다. 훨씬 깔끔