

# Always casez ○

← [always\\_case2](#) ✓ Previous

Next [always\\_nolatches](#) ○ →

Build a priority encoder for 8-bit inputs. Given an 8-bit vector, the output should report the first (least significant) bit in the vector that is 1. Report zero if the input vector has no bits that are high. For example, the input 8'b10010000 should output 3'd4, because bit[4] is first bit that is high.

From the previous exercise ([always\\_case2](#) ✓), there would be 256 cases in the case statement. We can reduce this (down to 9 cases) if the case items in the case statement supported don't-care bits. This is what case**z** is for. It treats bits that have the value z as don't-care in the comparison.

For example, this would implement the 4-input priority encoder from the previous exercise:

```
always @(*) begin
  casez (in[3:0])
    4'bzzz1: out = 0;    // in[3:1] can be anything
    4'bzz1z: out = 1;
    4'bz1zz: out = 2;
    4'b1zzz: out = 3;
    default: out = 0;
  endcase
end
```

A case statement behaves as though each item is checked sequentially (in reality, a big combinational logic function). Notice how there are certain inputs (e.g., 4'b1111) that will match more than one case item. The first match is chosen (so 4'b1111 matches the first item, out = 0, but not any of the later ones).

- There is also a similar casex that treats both x and z as don't-care. I don't see much purpose to using it over casez.
- The digit ? is a synonym for z. so 2'bz0 is the same as 2'b?0

It may be less error-prone to explicitly specify the priority behaviour rather than rely on the ordering of the case items. For example, the following will still behave the same way if some of the case items were reordered, because any bit pattern can only match at most one case item:

```
casez (in[3:0])
  4'bzzz1: ...
  4'bzz10: ...
  4'bz100: ...
  4'b1000: ...
  default: ...
endcase
```

## Module Declaration

```
// synthesis verilog_input_version verilog_2001
module top_module (
  input [7:0] in,
  output reg [2:0] pos );
```