Always case

always_if2 Previous Next always_case2

Case statements in Verilog are nearly equivalent to a sequence of if-elseif-else that compares one expression to a list of others. Its syntax and functionality differs from the `switch` statement in C.

```
always @(*) begin     // This is a combinational circuit
    case (in)
      1'b1: begin
              out = 1'b1;  // begin-end if >1 statement
            end
      1'b0: out = 1'b0;
      default: out = 1'bx;
    endcase
end
```

- The case statement begins with `case` and each "case item" ends with a colon. There is no "switch".
- Each case item can execute *exactly one* statement. This makes the "break" used in C unnecessary. But this means that if you need more than one statement, you must use `begin ... end`.
- Duplicate (and partially overlapping) case items are permitted. The first one that matches is used. C does not allow duplicate case items.

A bit of practice

Case statements are more convenient than if statements if there are a large number of cases. So, in this exercise, create a 6-to-1 multiplexer. When `sel` is between 0 and 5, choose the corresponding data input. Otherwise, output 0. The data inputs and outputs are all 4 bits wide.

Be careful of inferring latches (See.always_if2)

Module Declaration

```
// synthesis verilog_input_version verilog_2001
module top_module (
    input [2:0] sel,
    input [3:0] data0,
    input [3:0] data1,
    input [3:0] data2,
    input [3:0] data3,
    input [3:0] data4,
    input [3:0] data5,
    output reg [3:0] out   );
```