Lint Rules
p697-p870

8 謝明翰 林正硯 劉祐瑋

Simulation Rules

W526 Use *case* statements rather than *if/else*, where feasible, if performance is important.

p725 Warning

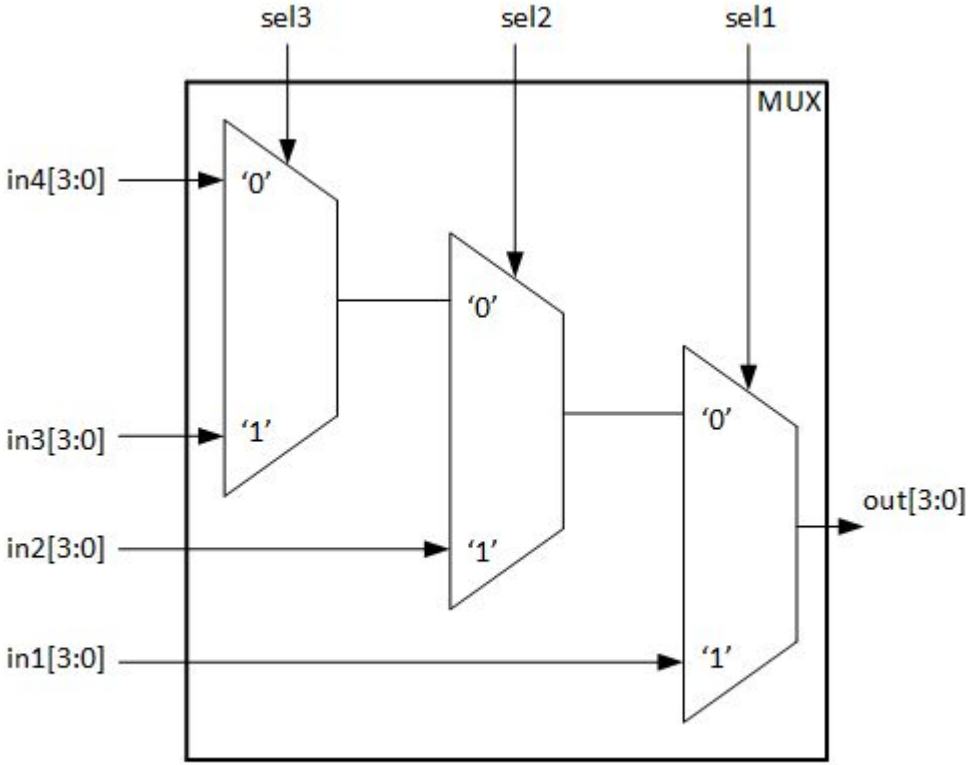
The W526 rule flags those *if-else* constructs should be changed to *case* constructs to improve simulation performance.

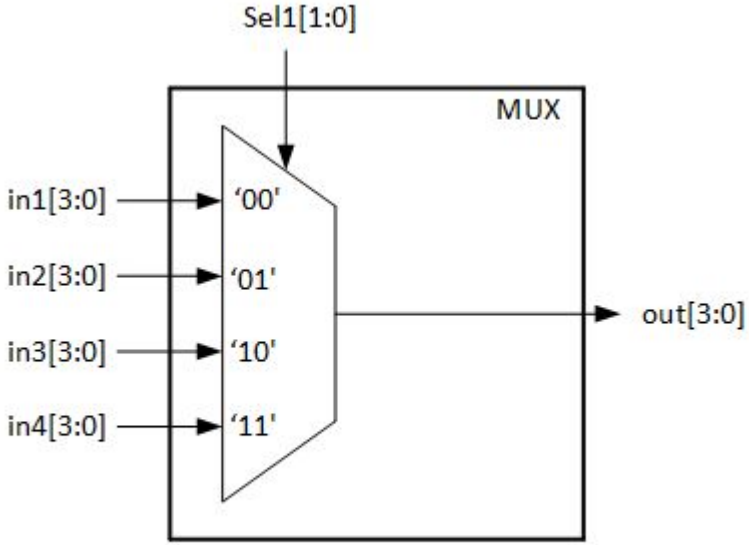
A chian of *if-else* on the same test signal is inferred as a priority encoder.

The same logic build using *case* is inferred as a multiplexer, which is likely to simulate faster.

```
if(sel == 4'd0)
        ...
else if(sel == 4'd1)
        ...
else if(sel == 4'd2)
        ...
else if(sel == 4'd3)
        ...
else
        ...
```

```
case(sel)
        4'd0: ...
        4'd1: ...
        4'd2: ...
        4'd3: ...
        default: ...
endcase
```


sel3
sel2
sel1
MUX
in4[3:0]
'0'
'1'
in3[3:0]
'0'
'1'
in2[3:0]
'0'
'1'
in1[3:0]
out[3:0]


Sel1[1:0]
MUX
in1[3:0]
in2[3:0]
in3[3:0]
in4[3:0]
'00'
'01'
'10'
'11'
out[3:0]

Event Rules

W245 Probably intended "*or*", not "|" or "||" in sensitivity list

p731 Warning

The W245 rules flags **bit-wise or** operator (|) or **logical or** operator (||) used in the sensitivity list of an *always* construct.
As the sensitivity list should be sensitive to the events in the control signal, it's recommended to use the "*or*" operator

always @(a || b)

→Evaluate the value a||b and trigger the event.

This is generally not the design intent.

always @(a or b)

→Trigger when either signal a or signal b changes which is generally the design intent.

W326 Event variable appearing in a posedge/negedge expression

p736 Fatal Note: The W326 rulse is switched off by default.

The W326 rule flags event variables used with edges.
Event don't have edges and therefore, should not appear in edge-based expressions.

```
module test (clk, in, cntr...);
        input clk, in, cntr;
        event test;
        always @(clk or test)
        ...
        always @(posedge test)
        ...
endmodule
```

Event variable should not appear in posedge/negedge expressions.

Events are generally used for synchronization and communication purposes rather than clocking control.

Loop Rules

p741-761

for (init-statement ; cond-expression ; loop-expression)

Lint_Elab Rules

p762-844

LHS matches RHS in bit width, ensure no data extension or truncation.
Especially in data copy operations.

```
C = A + B;
```

```
reg = DataIn;
DataOut = reg;
```

```
out[6:0] = 100 << 4;    // RHS = 11

out[4:0] = 100 >> 2;    // RHS = 15

out[12:0] = 12'b0;

out[1:0] = in1[1:0] + in2[1:0] + in3[1:0];
// Max value = 3 + 3 + 3 = 9, width = 4

out[2:0] = in1[1:0] + (3/3);
// Max value = 3 + 1 = 4, width = 3

out[3:0] = in1[3:0] - in2[1:0];
// -3(width 3) ~ 15(width 4), width = 4

out[3:0] = in1[3:0] - 4'b1010;
// -10(width 5) ~ 5(width 3), width = 5
```

Lint_Multiassign_blocking_sig, Lint_Multiassign_Nonblocking_sig

p872, 874 Warning

This rule is equivalent to W415, W415a. Signal may be multiply assigned in blocking manner.

```
always @(a or b)
begin
        d = b;
        d = a;
end
```

```
always @(a)
begin
        c <= c;
if(a)
        c <= c + 1;
else
        c <= c - 1;
end
```