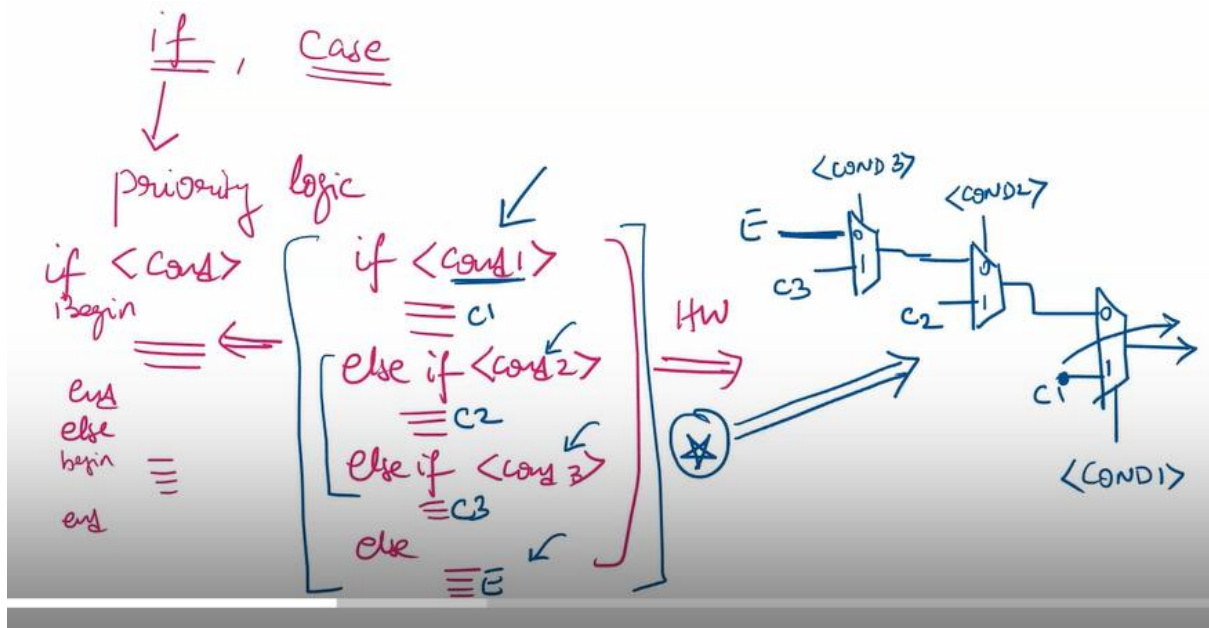
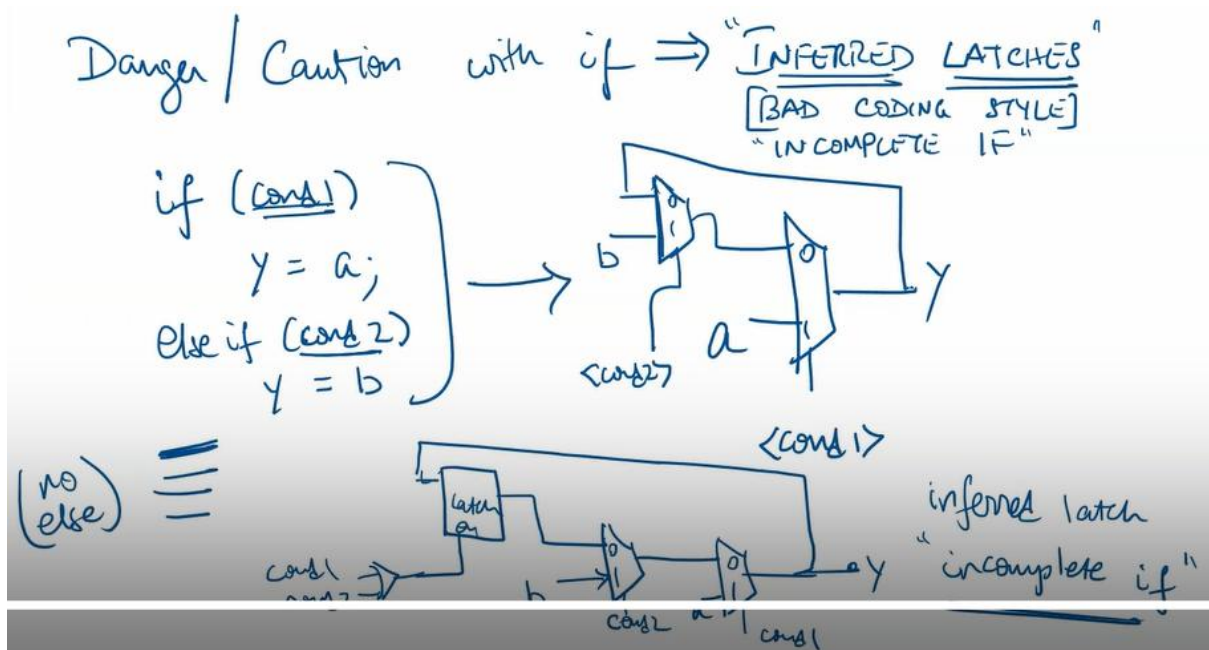


If and Case



Danger with if statement if the 2 conditions are not met then it will act as latch

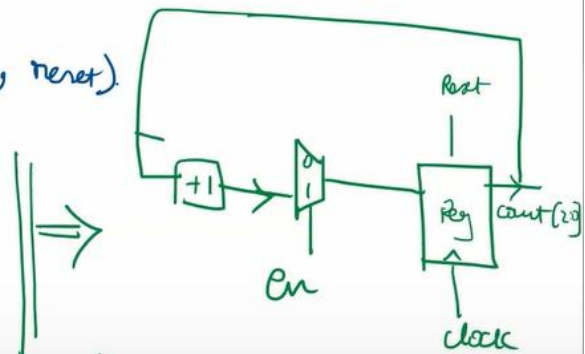


Counter Example

Counter

$\text{reg}[2:0] \text{ count}$
 always @ (posedge clk, posedge reset)
 begin
 if (reset)
 count <= 3'b000;
 else if (len)
 count <= count + 1;
 end

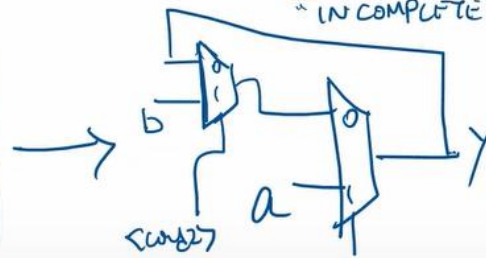
✓
complete
if



if no en; count should latch on to the previous value.

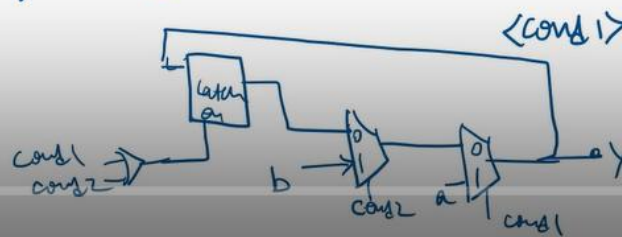
Danger / Caution with if \Rightarrow "INFERRED LATCHES"
 [BAD CODING STYLE]
 "INCOMPLETE IF"

if (cond1)
 y = a;
 else if (cond2)
 y = b



COMBINATIONAL
CIRCUIT

(no else)



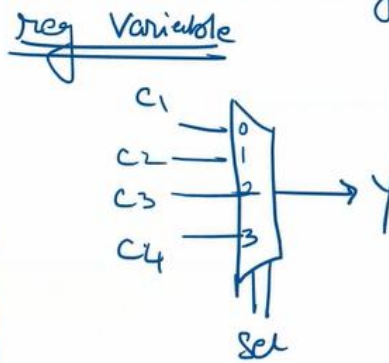
inferred latch
"incomplete if"

Case Statement [if, case are used inside always block]

```

reg y
always @ (*).
begin
    case (sel)
        2'b00 : begin
                    c1
                end
        2'b01 : begin
                    c2
                end
        2'b10 :
        2'b11 :
    endcase
end

```



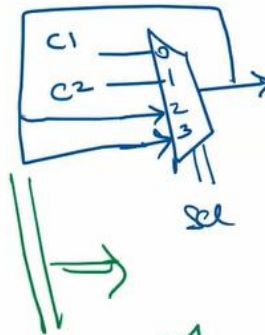
Caveats with case

① Incomplete Case ⇒ Inferred latches → CODE CASE WITH DEFAULT]

```

reg [1:0] sel
always @ (*).
begin
    case (sel)
        2'b00 :
        2'b01 :
    endcase
end

```



```

begin
    case (sel)
        2'b00 :
        2'b01 :
        default :
    endcase
end

```

With default ⇒ we avoid inferred latches. (X)

Case - Cauter (2)

partial assignments in case.

```
reg [1:0] sel;
```

```
reg x, y;
```

```
always @(*)
```

```
begin
```

```
case (sel)
```

```
2'b00:
```

```
begin
  x = a;
  y = b;
end
```

```
2'b01:
```

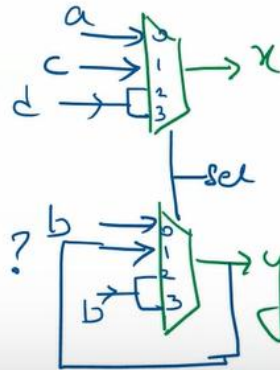
```
begin
  x = c;
  y = ?? ☹️
end
```

```
2'b10:
```

```
begin
  x = d;
  y = b;
end
```

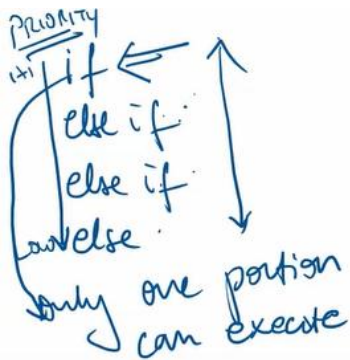
```
end
```

```
end case
```



⊗

Note:- Assign all the outputs in all the segments of case



Case(sel)

```
2'b00: =
```

```
2'b01: =
```

```
2'b10: ✓
```

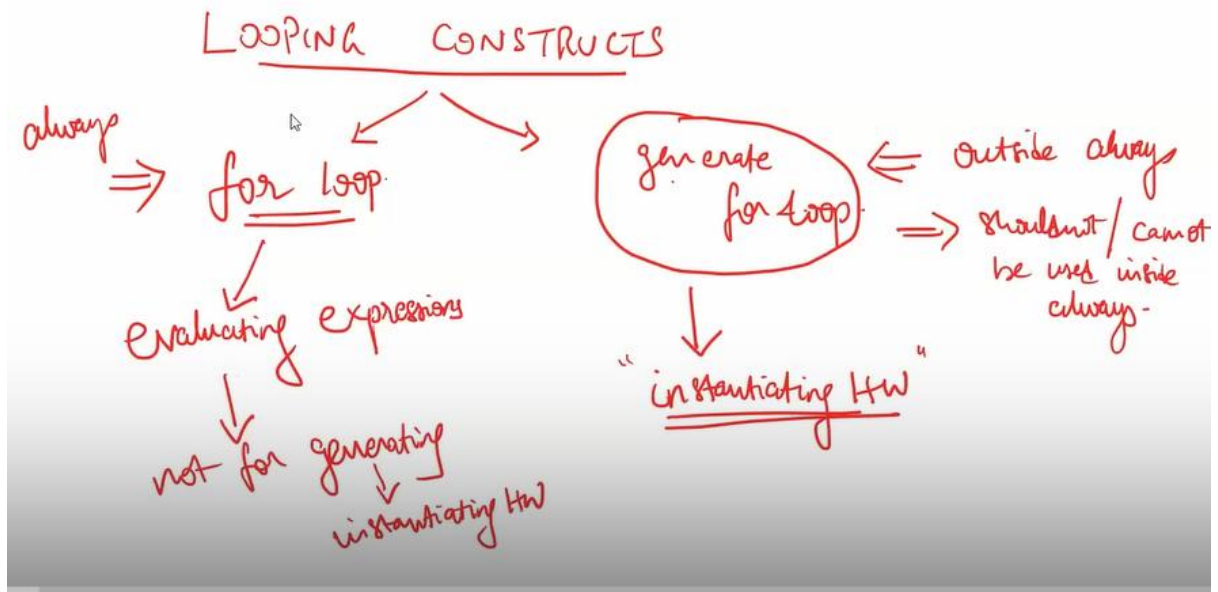
```
2'b1?: ✓
```

unpredictable outputs ⊗

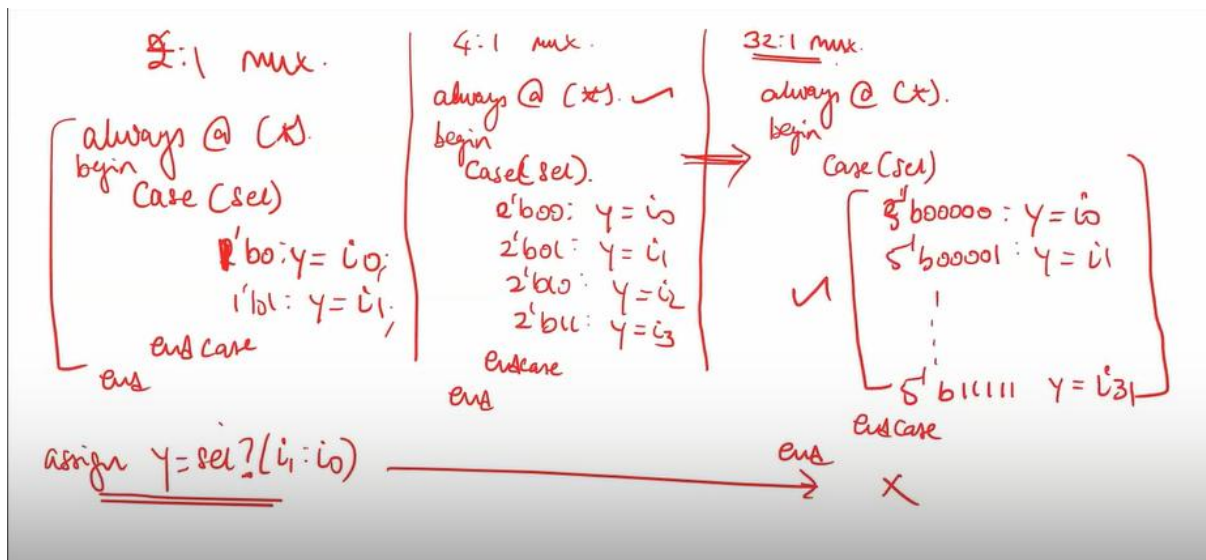
endcase

⊗ not have overlapping case ⊗

For loop




Example 2:1 mux



More mux

integer i
always @ (*)
begin
for (i=0; i < ²⁵⁶32; i=i+1) begin
if (i == sel)
y = ip[i];
end
end
256 → 1 mux

assumption:- $inp[31:0]$
bus

$inp[255:0]$ bus
32x1
mux


Demux

DEMUX 8 x 8 Demux

assumption $op_bus[7:0] \Rightarrow op$
input $\Rightarrow ip$
sel $[2:0]$

integer i;
always @ (*)
begin
 $op_bus[7:0] = 8'b0;$ ✓
for (i=0; i < 8; i=i+1) begin
if (i == sel)
op_bus[i] = input;
end

$op_bus[7] = 0$ ✓
 $[6] = 0$ ✓
 $[5] = 0$ ✓
input
 $[0] = 0$ ✓

Very wide mux / demux

for statement
is very handy

For generate loop