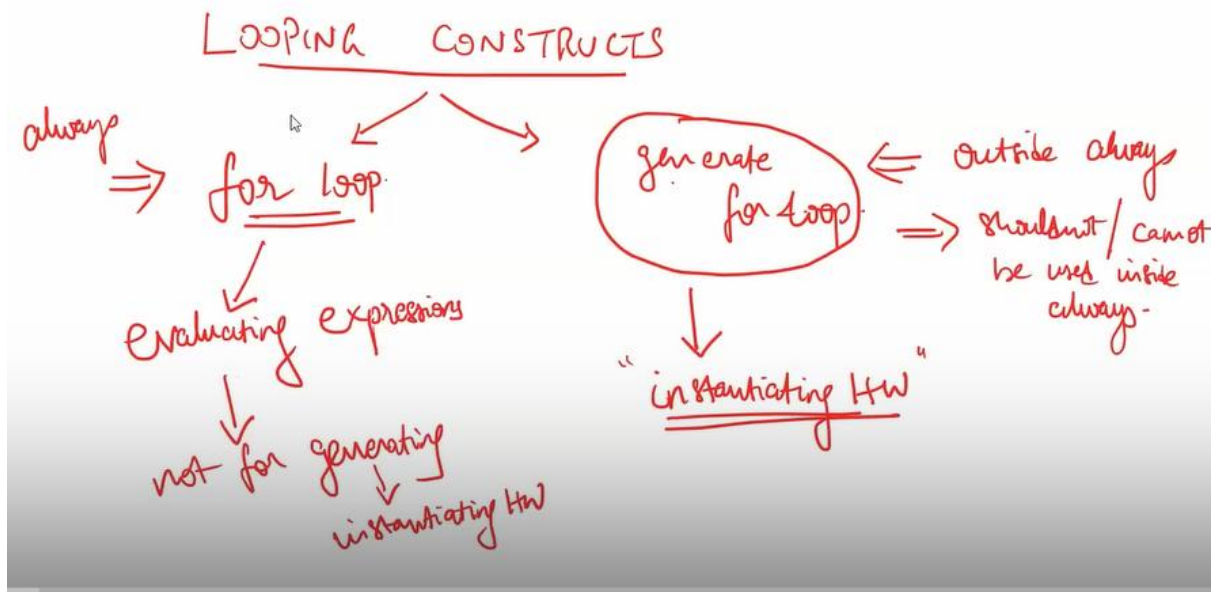
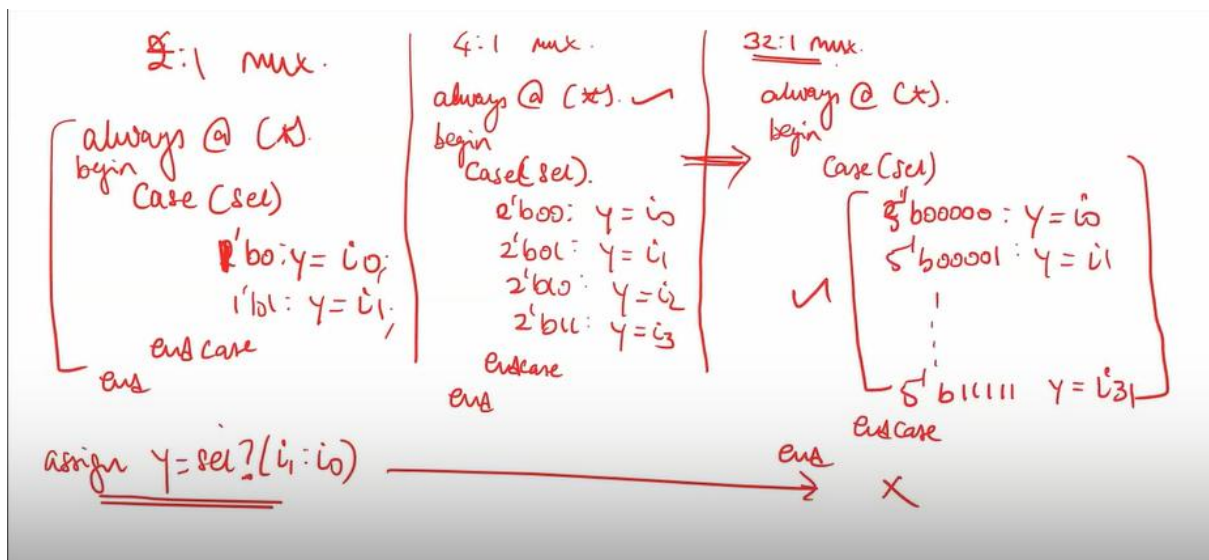


For loop




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
end

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

Very wide mux / demux

for statement
is very handy

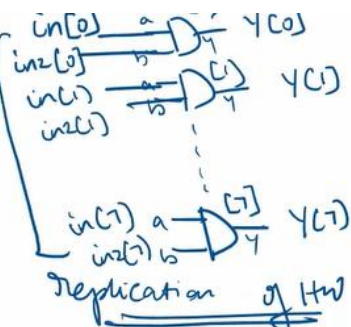
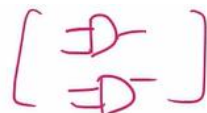
For generate loop

for generate:-

```

and u-ans1 (.ac), .bc), .y(1));
and u-ans2 (.ac), .bc), .y(1));

```



for-generate \Rightarrow replicating the HW. [outside always block]

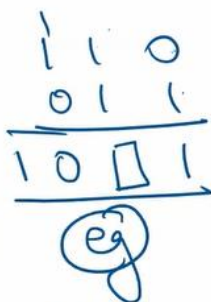
```

genvar i;
generate
  for (i=0; i<8; i=i+1) begin
    and u-ans (.a(in1[i]), .b(in2[i]), .y(y[i]));
  end
end

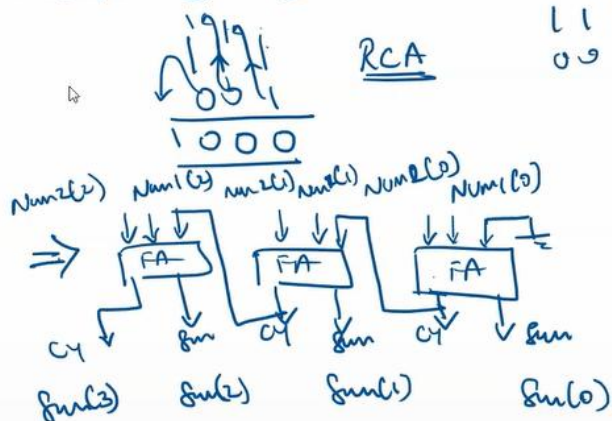
```

Generate outside always

Ripple carry ADDER (RCA)



Clearly a single block may be instantiated multiple times \rightarrow



for generate

Difference between the for and generate

for
inside always
multiple evaluations

for generate
outside always
HW replication

X

```
root@vsd:/home/mjcet/Desktop/raheem/week1/day1/vsdfLOW/sky130RTLDesignAndSynthe
sisWorkshop/verilog_files# iverilog mux_generate.v tb_mux_generate.v
root@vsd:/home/mjcet/Desktop/raheem/week1/day1/vsdfLOW/sky130RTLDesignAndSynthe
sisWorkshop/verilog_files# ./a.out
VCD info: dumpfile tb_mux_generate.vcd opened for output.
root@vsd:/home/mjcet/Desktop/raheem/week1/day1/vsdfLOW/sky130RTLDesignAndSynthe
sisWorkshop/verilog_files# gtkwave tb_mux_generate.vcd
Gtk-Message: 12:28:50.780: Failed to load module "canberra-gtk-module"
```

GTKWave Analyzer v3.3.103 (w)1999-2019 BSI

```

module mux_generate (input i0 , input i1, input i2 , input i3 , input [1:0] sel
, output reg y);
wire [3:0] i_int;
assign i_int = {i3,i2,i1,i0};
integer k;
always @ (*)
begin
for(k = 0; k < 4; k=k+1) begin
    if(k == sel)
        y = i_int[k];
end
end
endmodule

```

File Edit Search Time Markers View Help

From: 0 sec To: 3011 ns Marker:

SST

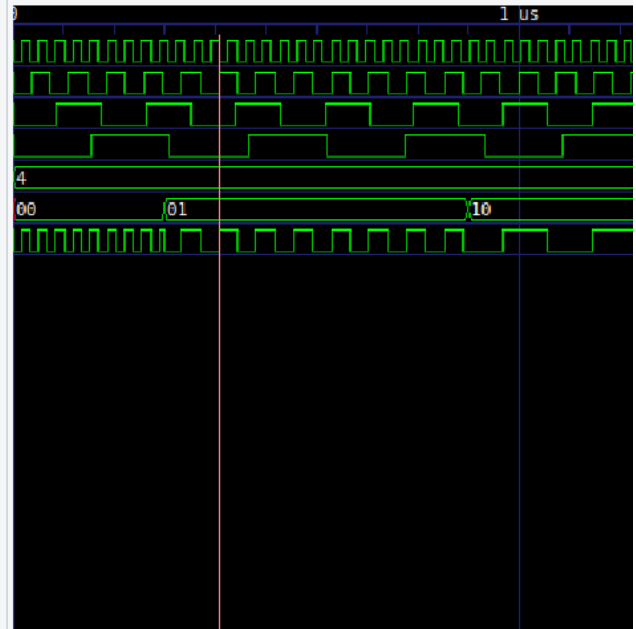
tb mux_generate
uut

Type	Signals
reg	clk
reg	i0
reg	i1
reg	i2
reg	i3
reg	reset
reg	sel[1:0]
wire	y

Signals

Time
i0=
i1=
i2=
i3=
k=
4
sel[1:0]=
y=

Waves



Filter:

```
=== mux_generate ===
```

Number of wires:	16
Number of wire bits:	51
Number of public wires:	8
Number of public wire bits:	43
Number of memories:	0
Number of memory bits:	0
Number of processes:	0
Number of cells:	9
\$_ANDNOT_	2
\$_AND_	1
\$_DLATCH_P_	1
\$_MUX_	3
\$_NAND_	1
\$_OR_	1

3.27. Executing CHECK pass (checking for obvious problems).
checking module mux_generate..
found and reported 0 problems.

```
ABC: + &put
```

```
ABC: + write_blif <abc-temp-dir>/output.blif
```

4.1.2. Re-integrating ABC results.

```
ABC RESULTS:      _const1_ cells:      1
ABC RESULTS:  sky130_fd_sc_hd__mux4_2 cells:      1
ABC RESULTS:      internal signals:      6
ABC RESULTS:      input signals:      6
ABC RESULTS:      output signals:      2
Removing temp directory.
```

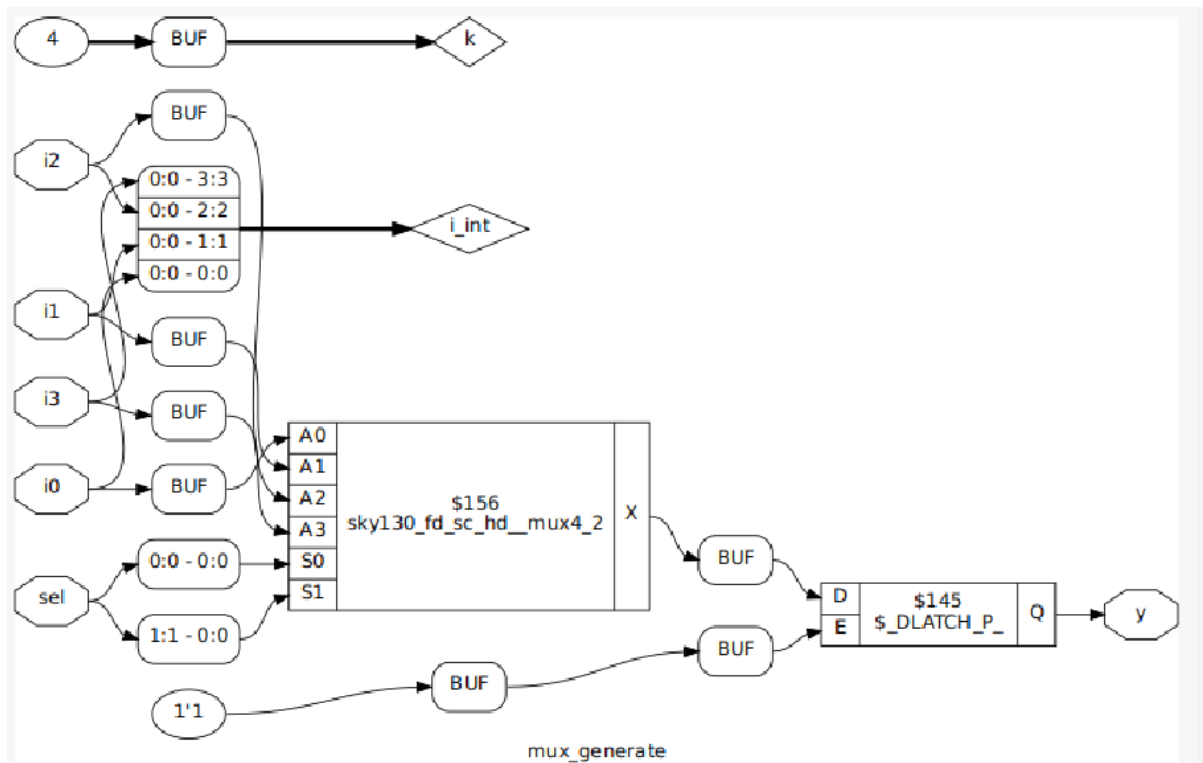
4.1.2. Re-integrating ABC results.

```
ABC RESULTS:      _const1_ cells:      1
ABC RESULTS:  sky130_fd_sc_hd__mux4_2 cells:      1
ABC RESULTS:      internal signals:      6
ABC RESULTS:      input signals:      6
ABC RESULTS:      output signals:      2
Removing temp directory.
```

```
yosys> write_verilog -noattr mux_generate.net.v
```

5. Executing Verilog backend.
Dumping module '\mux_generate'.

```
yosys> show
```

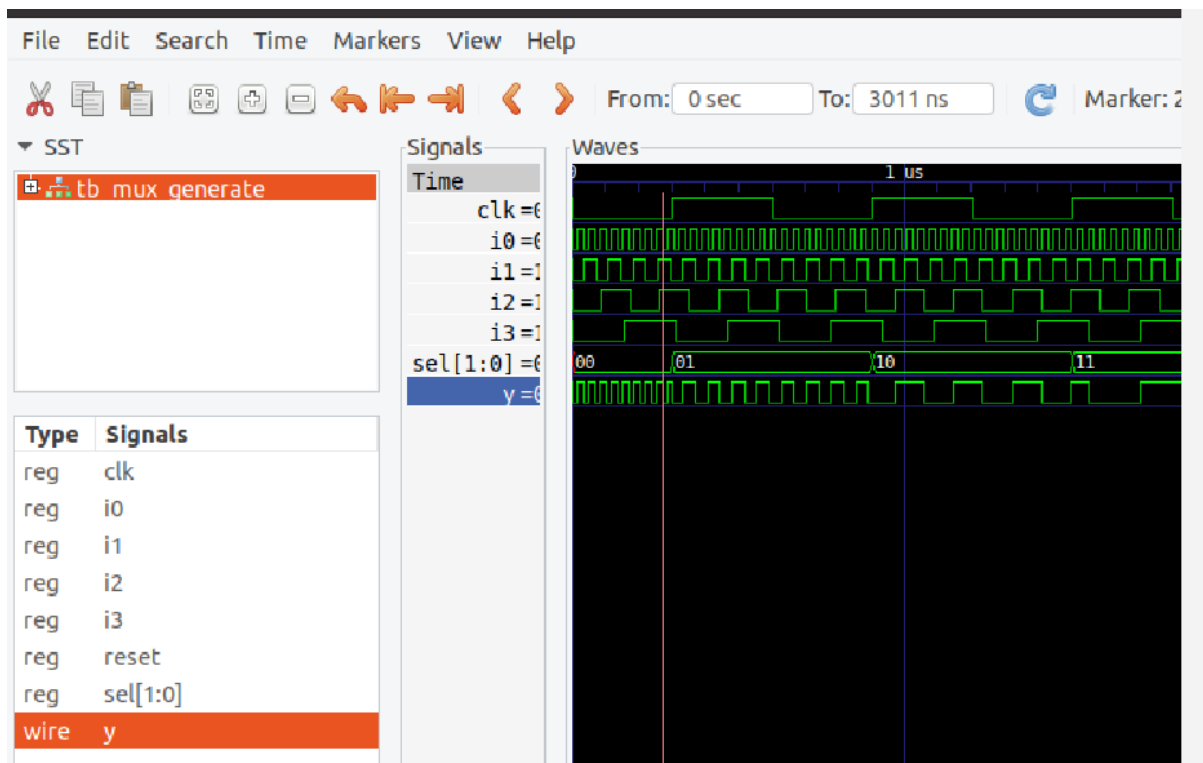


Mux_generator 256

```

module mux_generate (input i0 , input i1, input i2 , input i3 , input [1:0] sel
, output reg y);
wire [3:0] i_int;
assign i_int = {i3,i2,i1,i0};
integer k;
always @ (*)
begin
for(k = 0; k < 256; k=k+1) begin
if(k == sel)
y = i_int[k];
end
end
endmodule

```

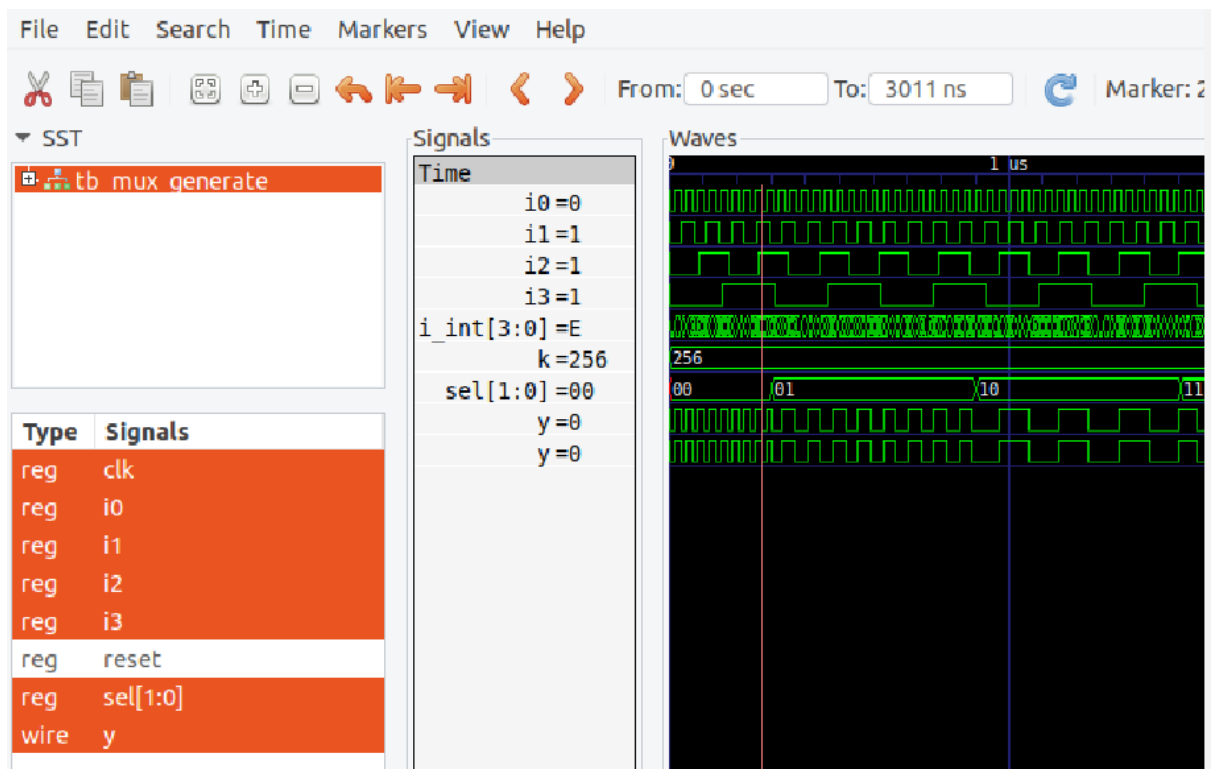


mux_generator 256

```

File Edit Tools Syntax Buffers Window Help
[Icons]
module mux_generate (input i0 , input i1, input i2 , input i3 , input [1:0] sel
    , output reg y);
wire [7:0] i_int;
assign i_int = {i3,i2,i1,i0};
integer k;
always @ (*)
begin
for(k = 0; k < 256; k=k+1) begin
    if(k == sel)
        y = i_int[k];
end
end
endmodule

```

```
=== mux_generate ===
```

```

Number of wires:           16
Number of wire bits:       51
Number of public wires:    8
Number of public wire bits: 43
Number of memories:        0
Number of memory bits:     0
Number of processes:       0
Number of cells:           9
  $_ANDNOT_                 2
  $_AND_                    1
  $_DLATCH_P_               1
  $_MUX_                    3
  $_NAND_                   1
  $_OR_                     1

```

```

3.27. Executing CHECK pass (checking for obvious problems).
checking module mux_generate..
found and reported 0 problems.

```

5.1.2. Re-integrating ABC results.

```
ABC RESULTS:      _const1_ cells:      1
ABC RESULTS:  sky130_fd_sc_hd__mux4_2 cells:      1
ABC RESULTS:      internal signals:      6
ABC RESULTS:      input signals:      6
ABC RESULTS:      output signals:      2
Removing temp directory.
```

