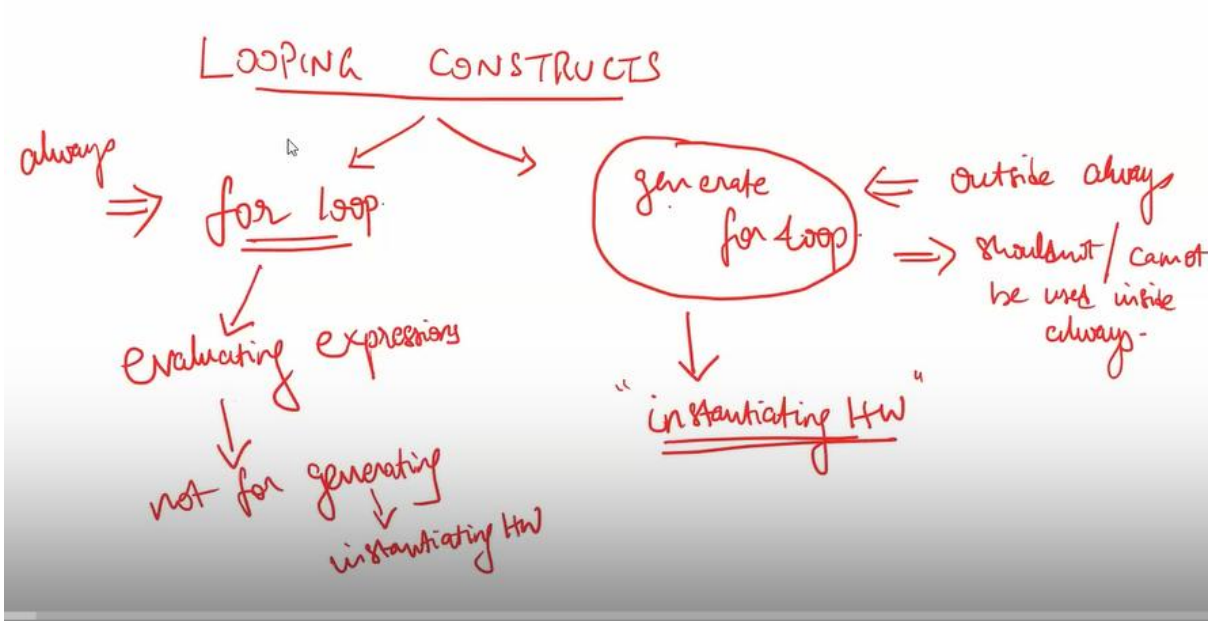
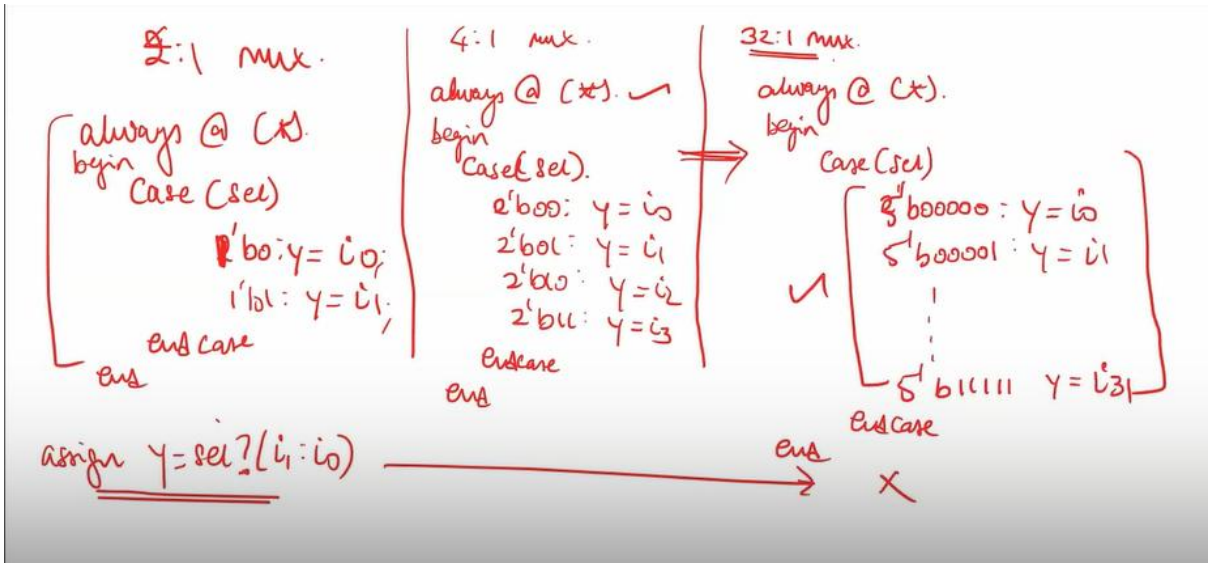
For loop



Example 2:1 mux



More mux

```
integer i
always @ (*)
begin
    for (i=0; i<32; i=i+1) begin
        if (i==sel)
            y = inp[i];
    end
end
```

256

assumption:- inp[31:0] 32 bus

inp[255:0] bus

32x1 mux

256 → 1 mux

Demux

```
DEMUX 1x8 Demux
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;
```

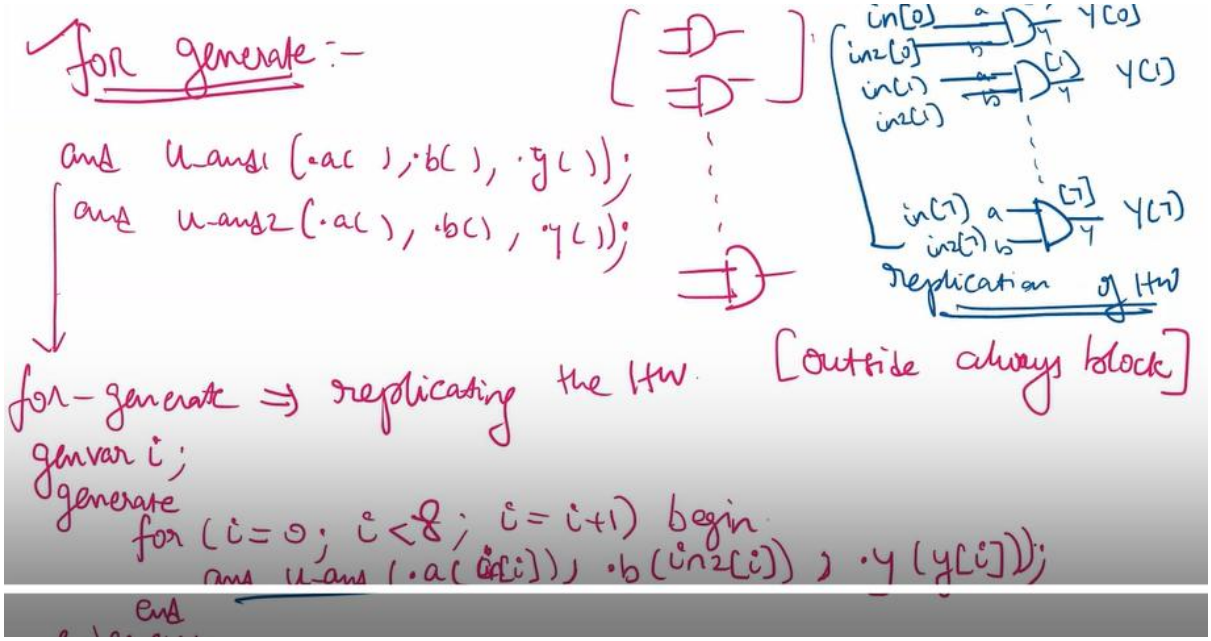
Evaluation

assumption op_bus[7:0] ⇒ op
input ⇒ ip 1 bit
sel [2:0]

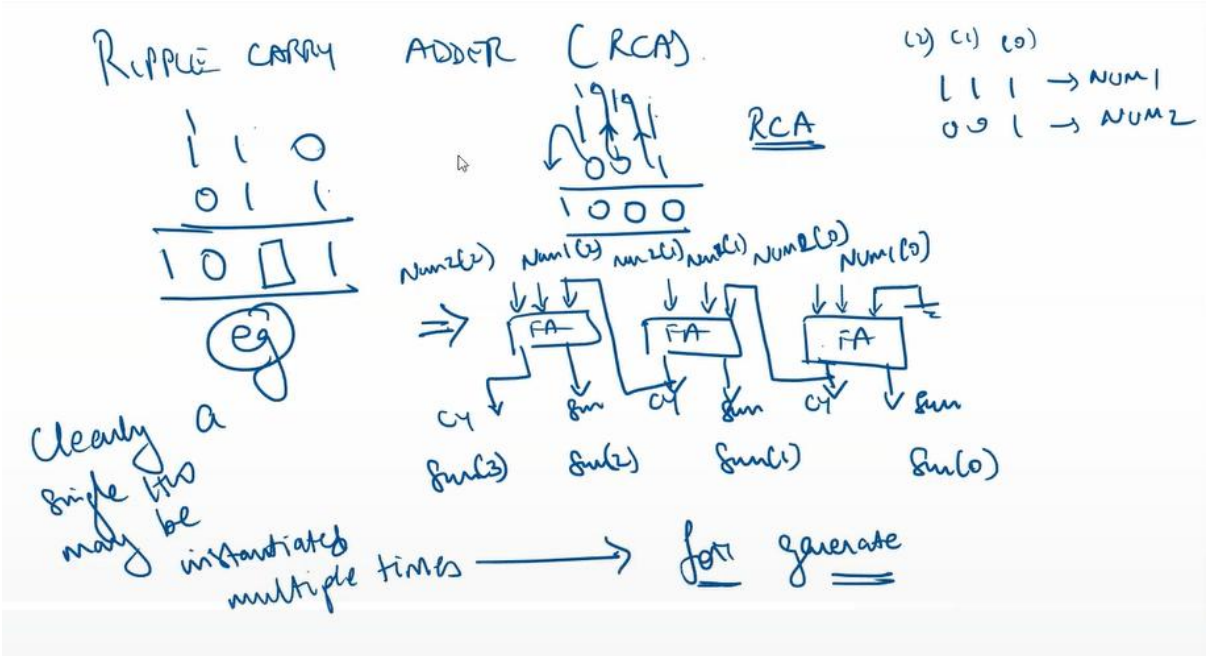
op_bus[7] = 0
[6] = 0
[5] = 0
input
[0] = 0

Very wide Mux / Demux for statement is very handy

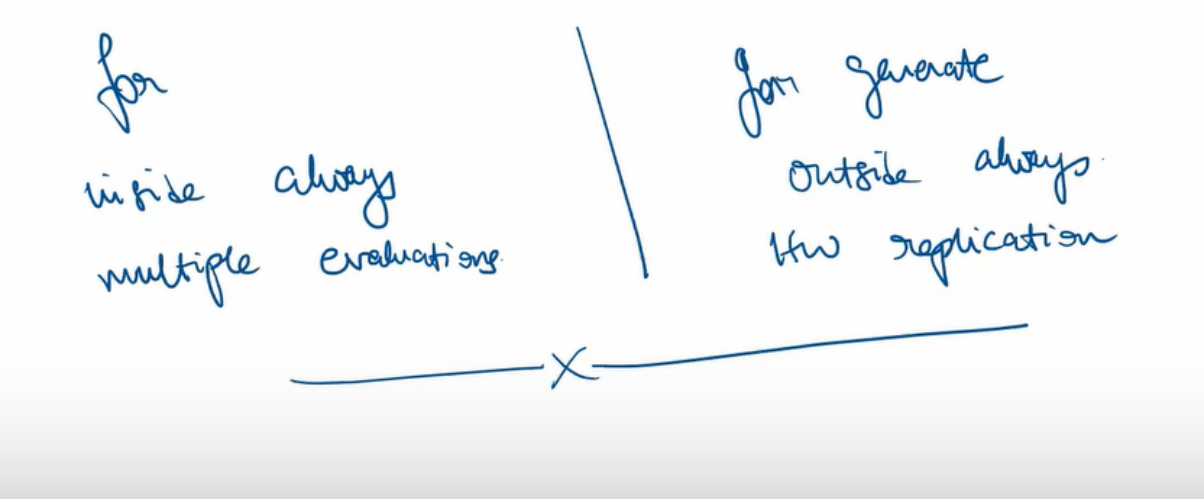
For generate loop



Gernerate oustside always

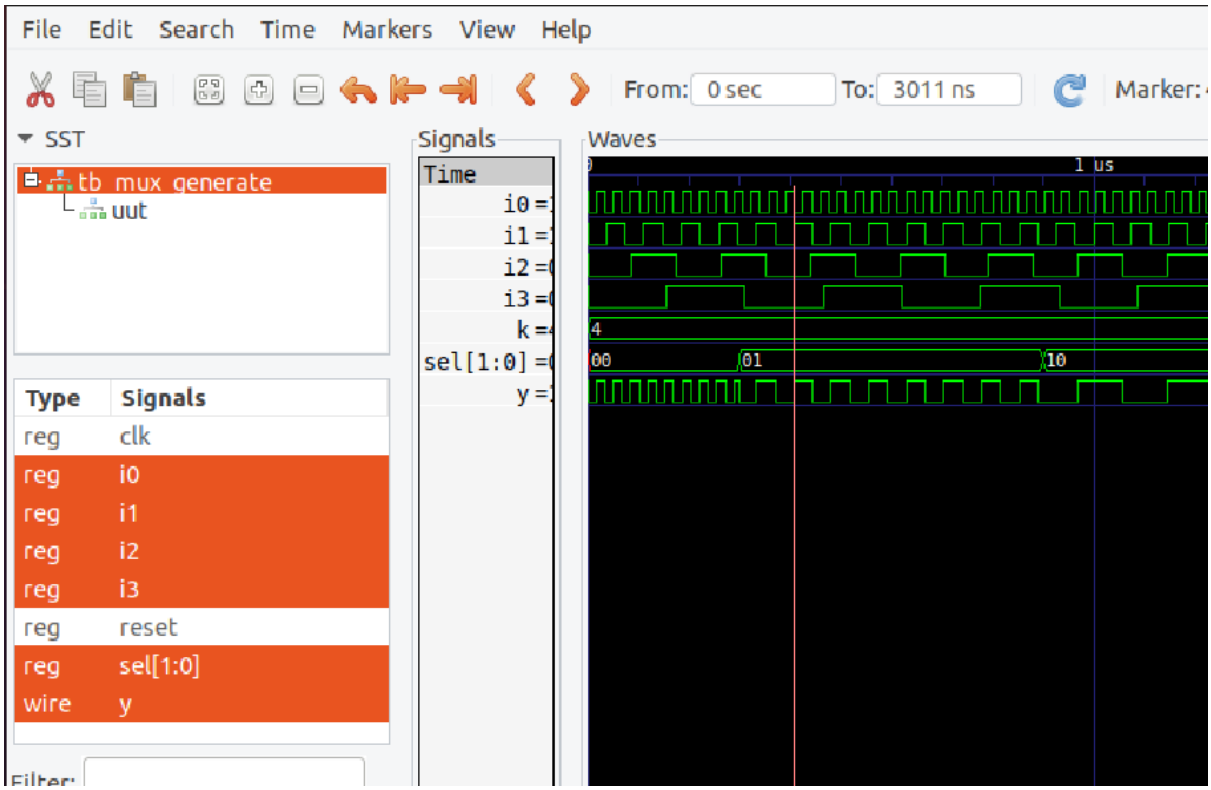


Difference between the for and generate




```
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
```

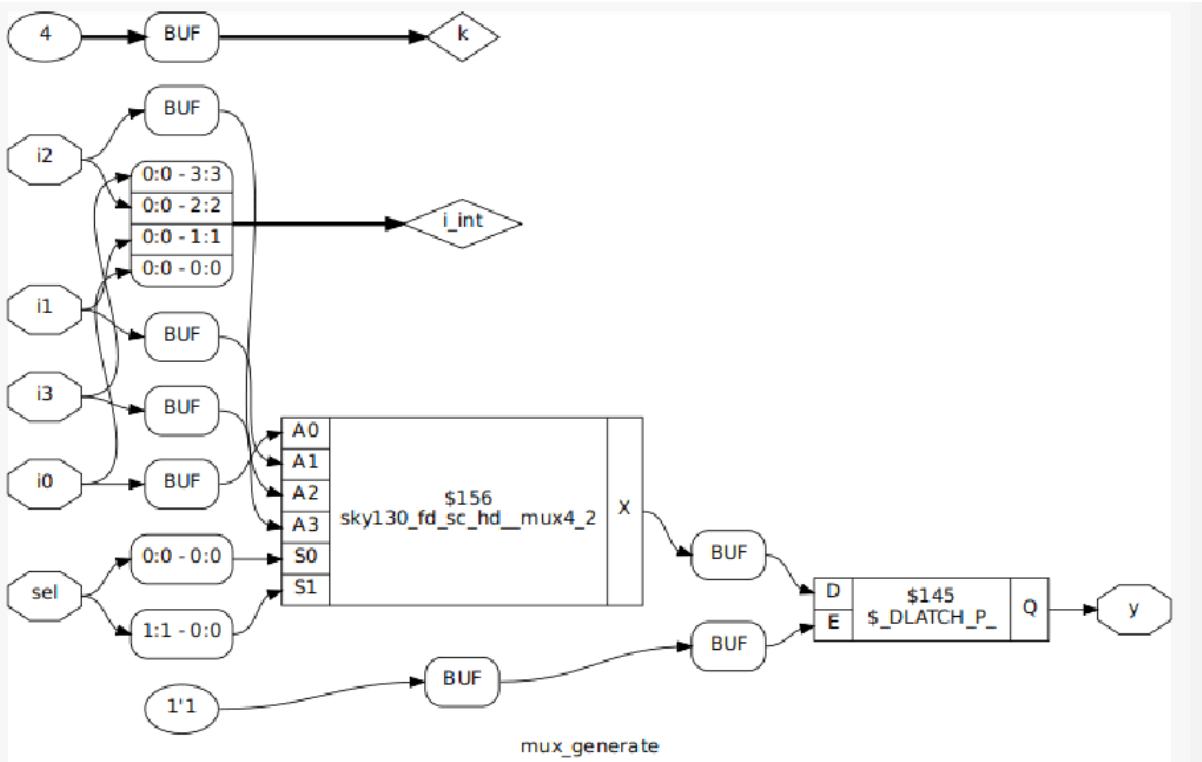
```
module mux_generate (input i0 , input i1, input i2 , input i3 , input [1:0] sel
  , output reg y);
wire [3:0] i_int;
assign i_int = {i3,i2,i1,i0};
integer k;
always @ (*)
begin
for(k = 0; k < 4; k=k+1) begin
        if(k == sel)
                y = i_int[k];
end
end
endmodule
```



```
=== 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.
```

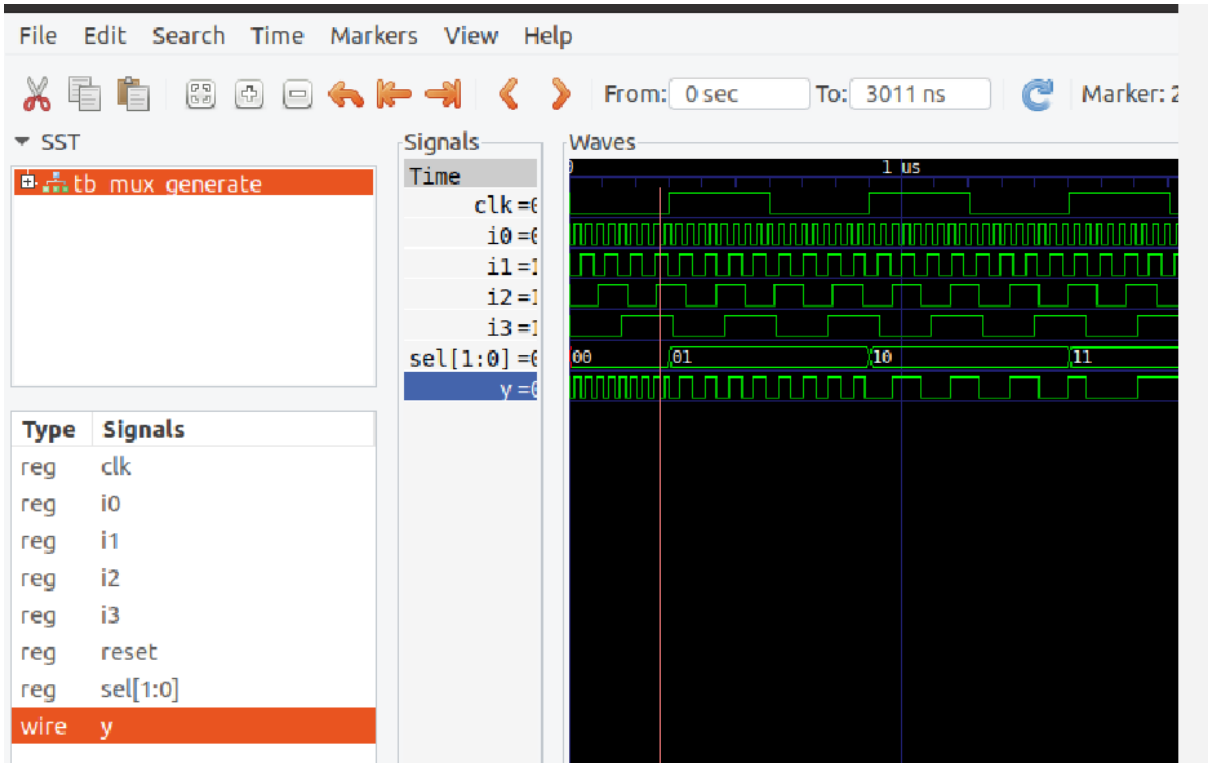
```
ABC: + &put
ABC: + write_blif <abc-temp-dir>/output.blif

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
```

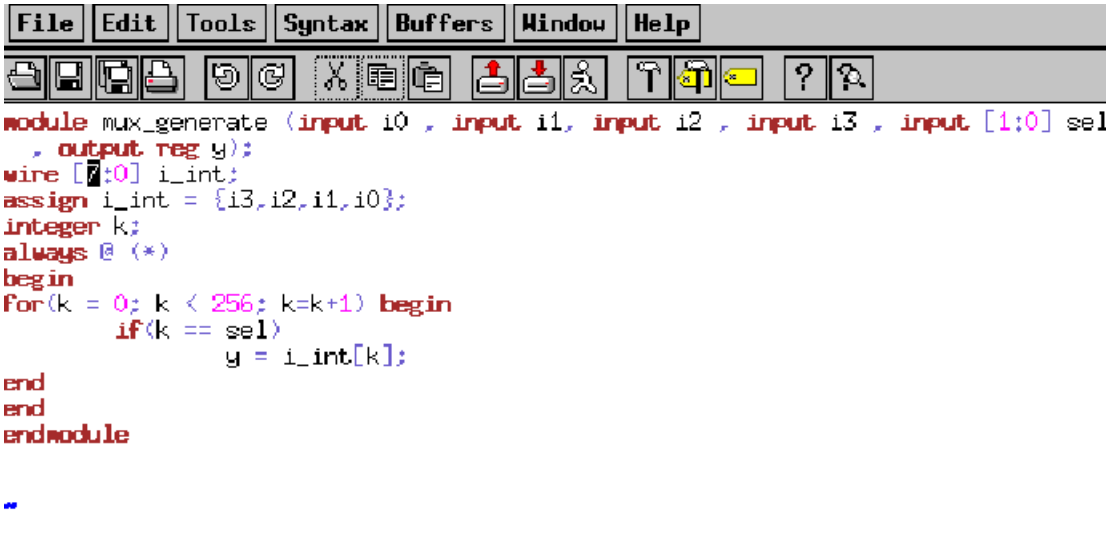


Mux_generator 256

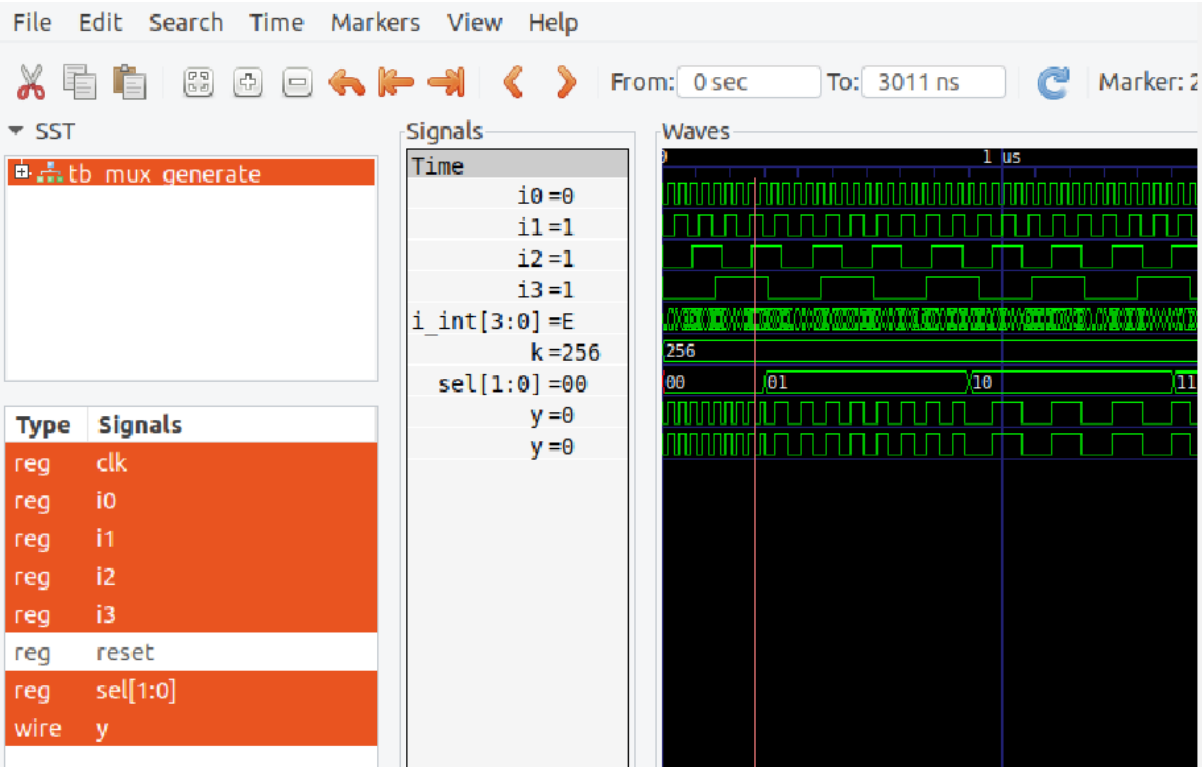
```
module mux_generate (input i0 , input i1, input i2 , input i3 , input [1:0] sel
  , output reg y);
wire [3:0] i_int;
assign i_int = {i3,i2,i1,i0};
integer k;
always @ (*)
begin
for(k = 0; k < 256; k=k+1) begin
        if(k == sel)
                y = i_int[k];
end
end
endmodule
```



mux_generator 256




```
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.
```