

Day-5

If, Case, For Loop and for generate

If-else statement-It is mainly used for priority Logic

Cautions with If-

An incomplete If statement results in an **Inferred Latch**. Suppose we do not give else statement in if construct, then the synthesizer will synthesize a latch which will infer to the past value of output in absence of an else block.

We should not have incomplete if statements unless it is intended.

Incomplete If's should be avoided especially in combinational circuits.

Case statement-

Caveats-

1. Incomplete case will also lead to **Inferred Latches**. To avoid this, we should add a default case in the code.

2. Partial Assignments In case- This will create an **Inferred Latch**. So, we should assign all the outputs in all the segments of case.

Comparison in If-else and case

If-else has a clear priority. Only one segment will execute either if, if -else or else whereas in case if more than one condition matches then it will execute both the condition in the order they appear. So, we should not have overlapping case statements.

For Loop-

- Used inside always block
- Evaluating expressions
- Not for instantiating hardware

Generate For –

- Always used outside the always block.
- Used for instantiating a hardware multiple time

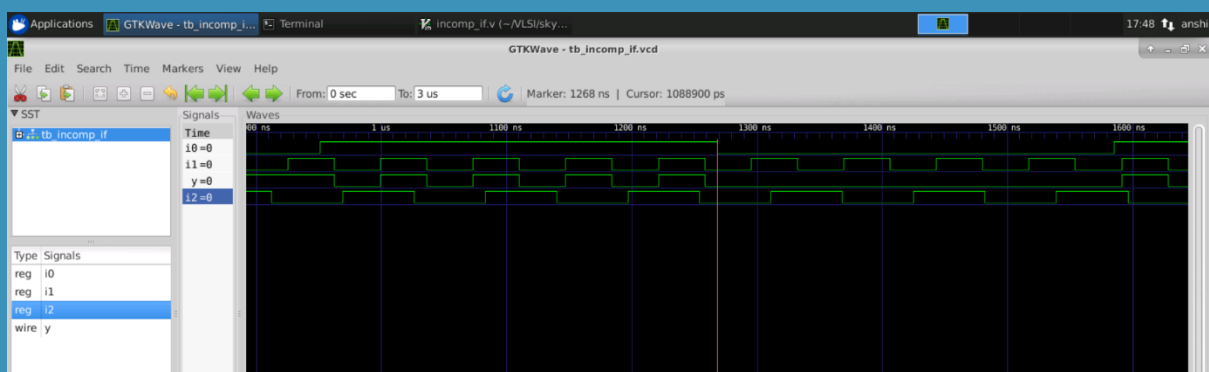
Lab cases

1. Incomplete If construct:

The code is without else.

```
incomp case.v 1,0-1 Top
module incmp_if (input i0 , input i1 , input i2 , output reg y);
always @ (*)
begin
    if(i0)
        y <= i1;
end
endmodule
```

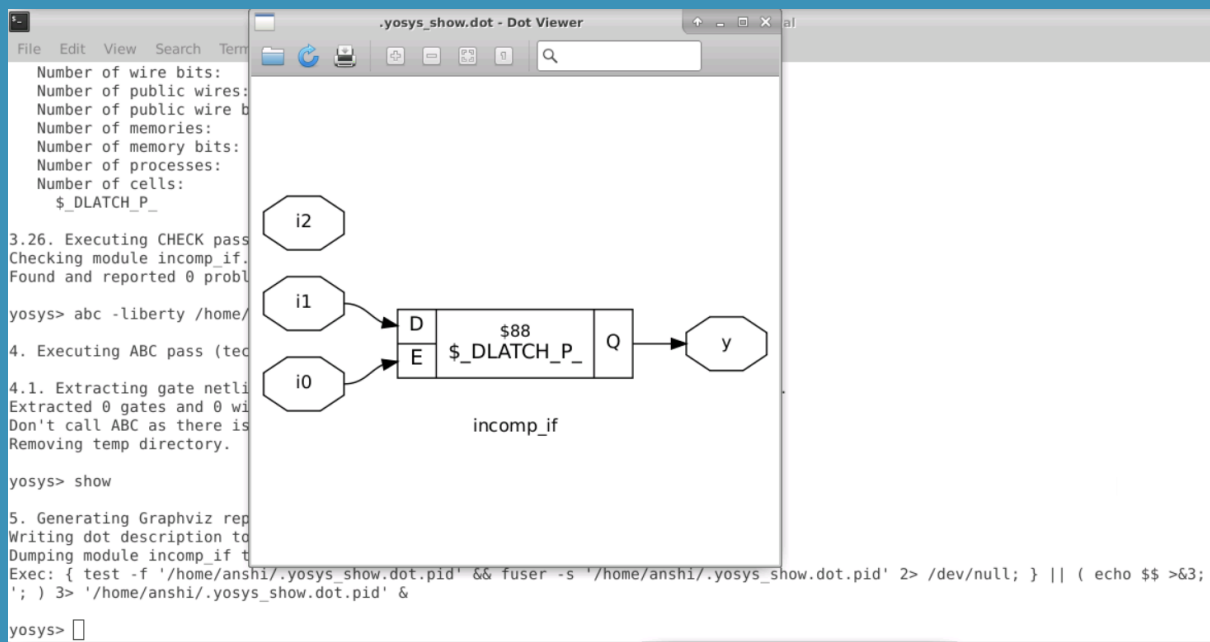
Y follows i1 if i0 is high but latching onto its previous value if i0 is low, thereby resulting in an Inferred Latch



Clearly, a D-Latch is inferred in synthesis

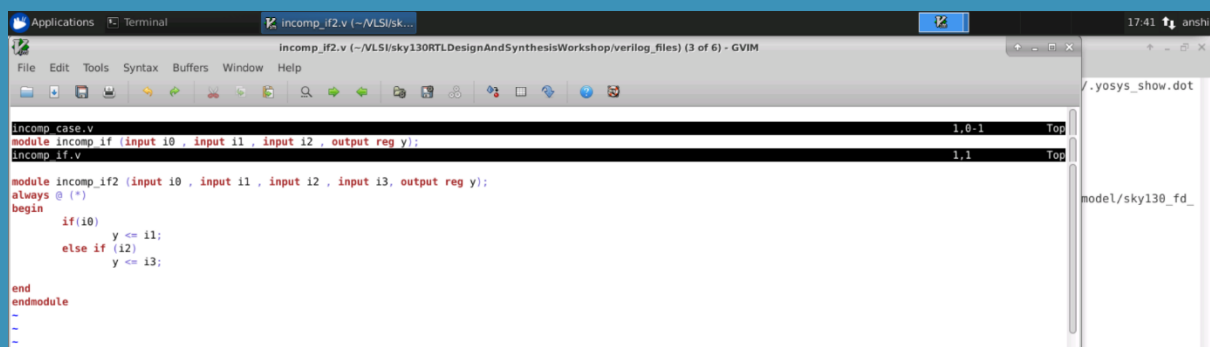
```
3.23.5. Finished fast OPT passes.
3.24. Executing HIERARCHY pass (managing design hierarchy).
3.24.1. Analyzing design hierarchy..
Top module: \incmp_if
3.24.2. Analyzing design hierarchy..
Top module: \incmp_if
Removed 0 unused modules.
3.25. Printing statistics.
=== incmp_if ===
Number of wires:          4
Number of wire bits:      4
Number of public wires:   4
Number of public wire bits: 4
Number of memories:       0
Number of memory bits:    0
Number of processes:      0
Number of cells:          1
$DLATCH_P_                1
3.26. Executing CHECK pass (checking for obvious problems).
Checking module incmp_if...
Found and reported 0 problems.
```

I2 is not used and y is latched to i1 without creating a Mux.

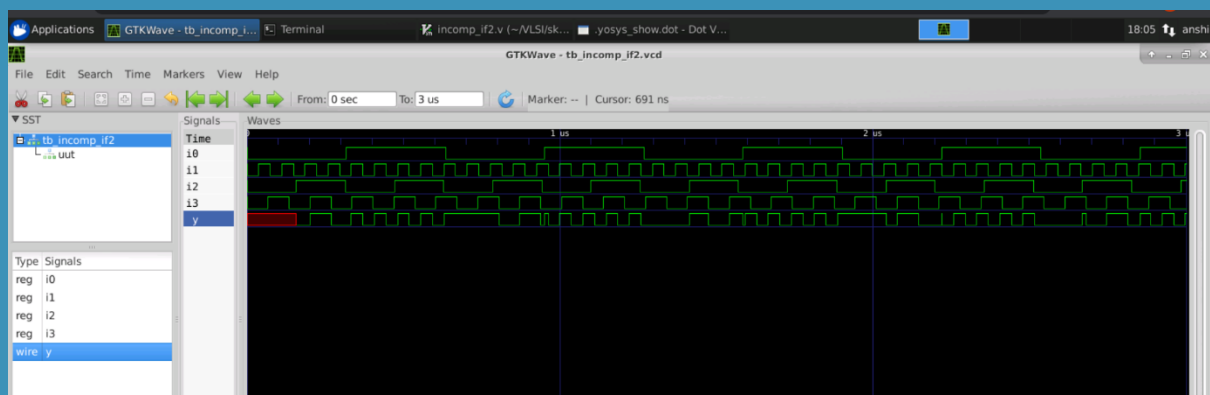


2. Second Case of Incomplete -If:

Verilog Code- If is without else block



When `i0` is high `y` follows `i1` otherwise it checks for `i2` and when `i2` and `i3` both are high `y` latches to its previous value.



Clearly, it is inferring a latch

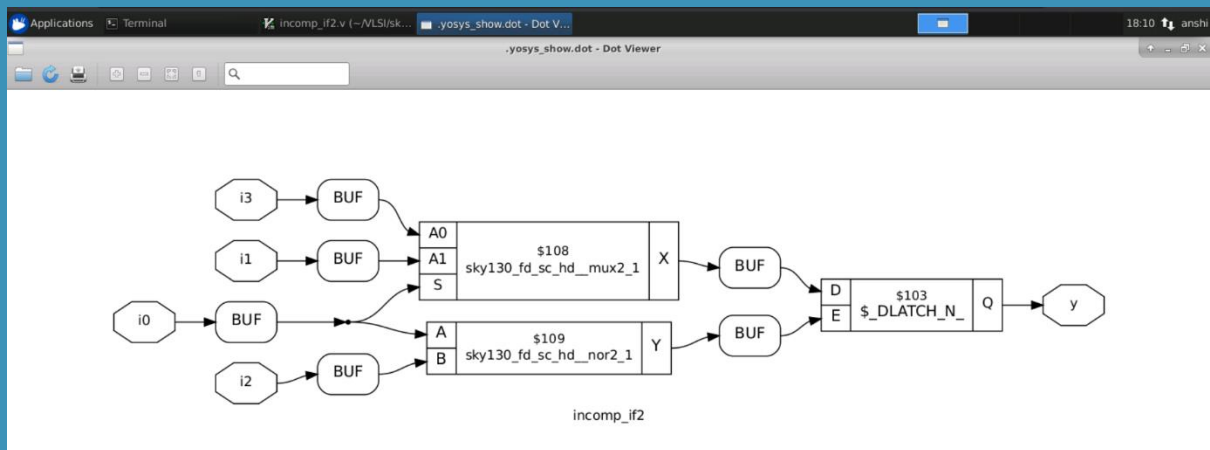
```

Applications  Terminal  incomp_if2.v (~VLSI/sk...  .yosys_show.dot - Dot V...  18:09 anshi
Terminal
File Edit View Search Terminal Help

3.24. Executing HIERARCHY pass (managing design hierarchy).
3.24.1. Analyzing design hierarchy..
Top module: \incomp_if2
3.24.2. Analyzing design hierarchy..
Top module: \incomp_if2
Removed 0 unused modules.
3.25. Printing statistics.
=== incomp_if2 ===
Number of wires:          7
Number of wire bits:      7
Number of public wires:   5
Number of public wire bits: 5
Number of memories:       0
Number of memory bits:    0
Number of processes:      0
Number of cells:          3
$ _DLATCH_N_              1
$ _MUX_                   1
$ _NOR_                   1
3.26. Executing CHECK pass (checking for obvious problems).
Checking module incomp_if2..
Found and reported 0 problems.

```

So, a nor is synthesized for the logic of the enable of D-Latch. If both i0 and i2 are low then the latch goes high. A 2:1 mux is present with i0 as select which if high then y is i1 else i3.



Hence, resulting in an **Inferred Latch** in case of incomplete if statement.

3.Case Statements-Incomplete Case

If sel is 00 , y is i0

If sel is 01 , y is i1 else y will latch.

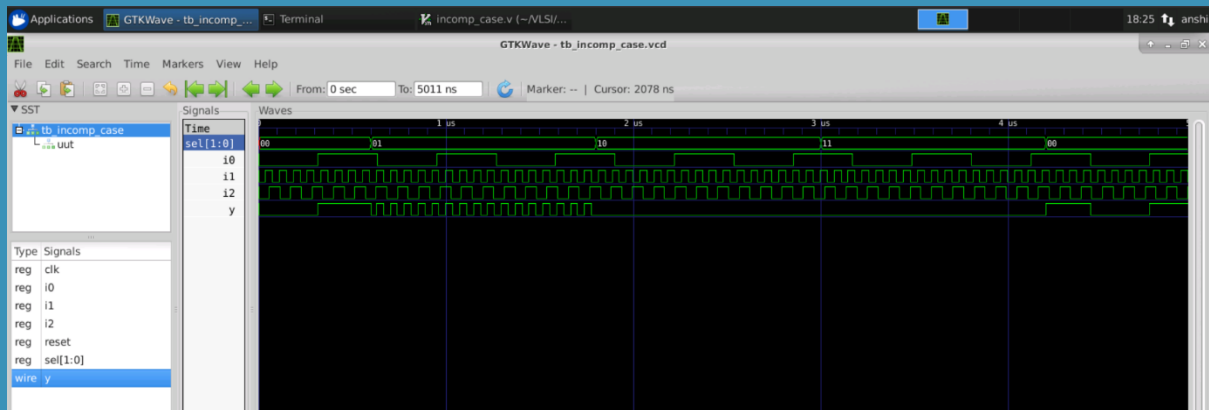
```

Applications  Terminal  incomp_case.v (~VLSI/...  18:19 anshi
incomp_case.v (~VLSI/sky130RTLDesignAndSynthesisWorkshop/verilog_files) (2 of 4) - GVIM
File Edit Tools Syntax Buffers Window Help

incomp_case.v 1,0-1 Top
module incomp_case (input i0 , input i1 , input i2 , input [1:0] sel, output reg y);
always @ (*)
begin
    case(sel)
        2'b00 : y = i0;
        2'b01 : y = i1;
    endcase
end
endmodule

```

When sel is 00 y follows i0, when 11 if follows i1 and when it is 10 or 11 then y is latching to its previous value.



Gates Synthesized

```
3.24.1. Analyzing design hierarchy..
Top module: \incomp_case

3.24.2. Analyzing design hierarchy..
Top module: \incomp_case
Removed 0 unused modules.

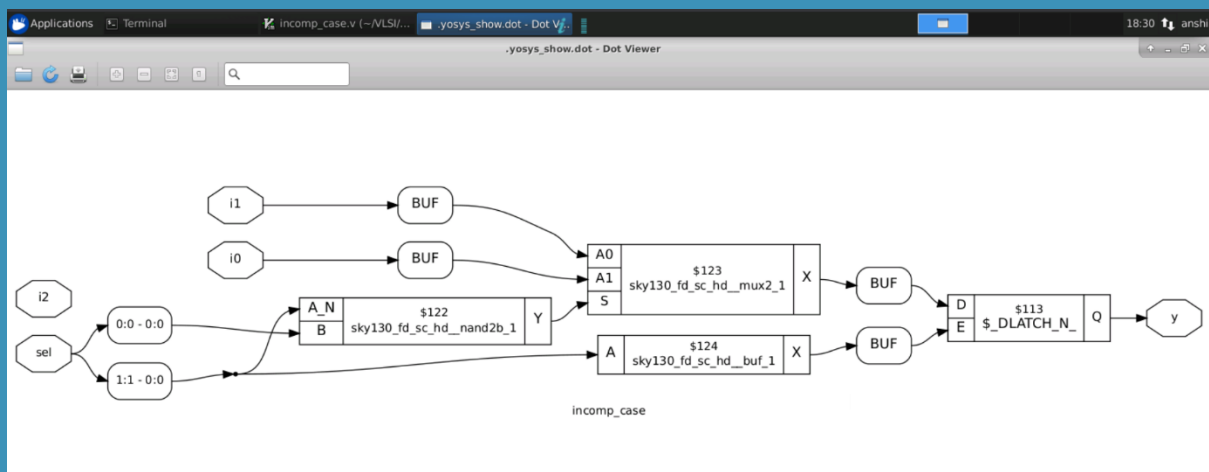
3.25. Printing statistics.

=== incomp_case ===
Number of wires:          9
Number of wire bits:      10
Number of public wires:   5
Number of public wire bits: 6
Number of memories:       0
Number of memory bits:    0
Number of processes:      0
Number of cells:          5
$ _ANDNOT_                1
$ _DLATCH_N_              1
$ _MUX_                   1
$ _NOR_                   1
$ _ORNOT_                 1

3.26. Executing CHECK pass (checking for obvious problems).
Checking module incomp_case...
Found and reported 0 problems.

yosys>
```

So, after synthesis a D-Latch is inferred in the design because of the absence of else block.



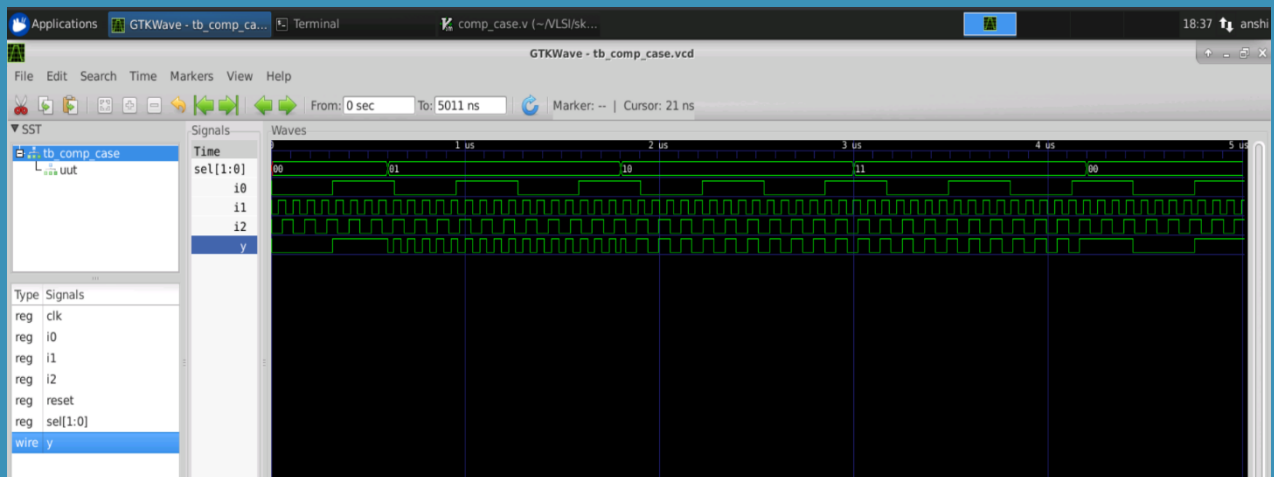
4.Complete Case with No Inferred Latch

```

module comp_case (input i0 , input i1 , input i2 , input [1:0] sel, output reg y);
always @ (*)
begin
    case(sel)
        2'b00 : y = i0;
        2'b01 : y = i1;
        default : y = i2;
    endcase
end
endmodule

```

When sel is 00 y is i0 , when 01 y is i1 else y is i2.Hence, no latching



Clearly, no latching this time all gate synthesized are combinational logic

```

3.24.1. Analyzing design hierarchy..
Top module: \comp_case

3.24.2. Analyzing design hierarchy..
Top module: \comp_case
Removed 0 unused modules.

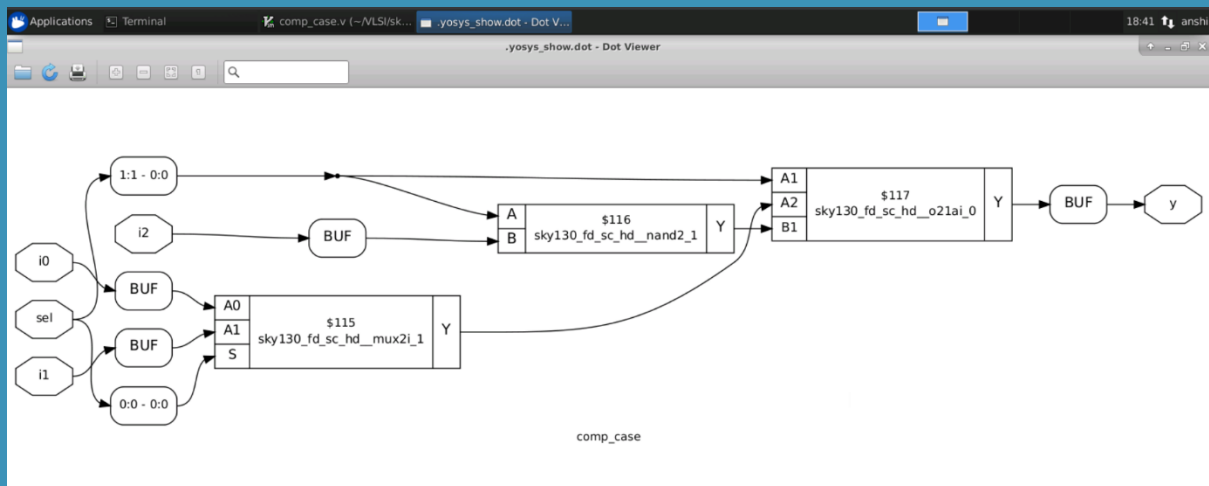
3.25. Printing statistics.

=== comp_case ===
Number of wires:          11
Number of wire bits:      12
Number of public wires:   5
Number of public wire bits: 6
Number of memories:       0
Number of memory bits:    0
Number of processes:      0
Number of cells:          7
$ _ANDNOT_                2
$ _AND_                    1
$ _MUX_                    1
$ _ORNOT_                  1
$ _OR_                     2

3.26. Executing CHECK pass (checking for obvious problems).
Checking module comp_case...
Found and reported 0 problems.

```

Design:



Hence, no latch is inferred.

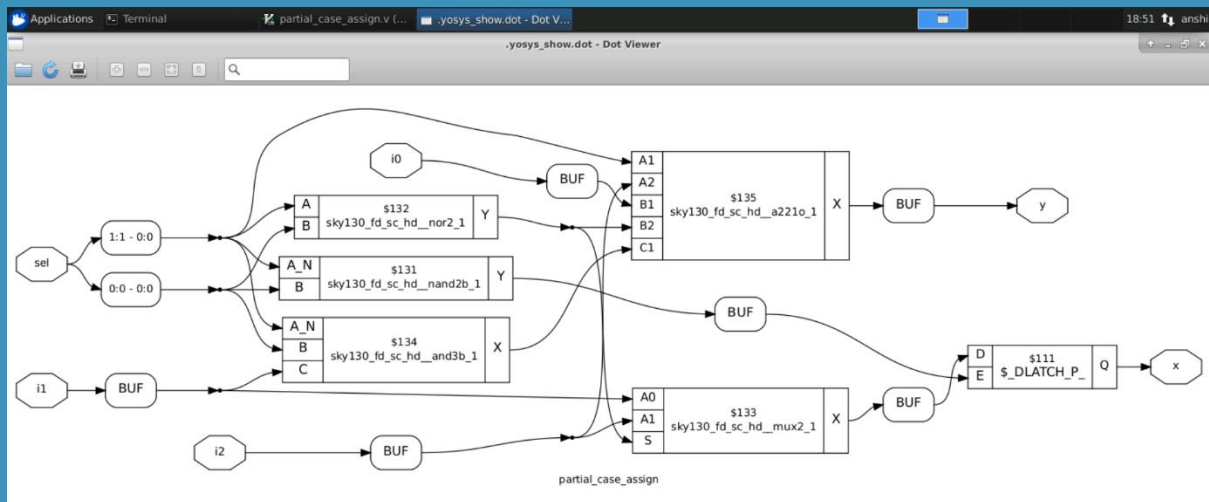
5. Partial-Assignments

```

end
comp case.v 12,9 90%
incomp case.v 10,9 Top
module partial_case_assign (input i0 , input i1 , input i2 , input [1:0] sel, output reg y , output reg x);
always @ (*)
begin
    case(sel)
        2'b00 : begin
            y = i0;
            x = i2;
        end
        2'b01 : y = i1;
        default : begin
            x = i1;
            y = i2;
        end
    endcase
end
endmodule

```

When For x, when sel is 00, then x is i2 otherwise it is i1 and latched to its previous value for nand of sel0 and not(sel1).



X in inferred with a Latch because Partial assignments

6.Overlapping case

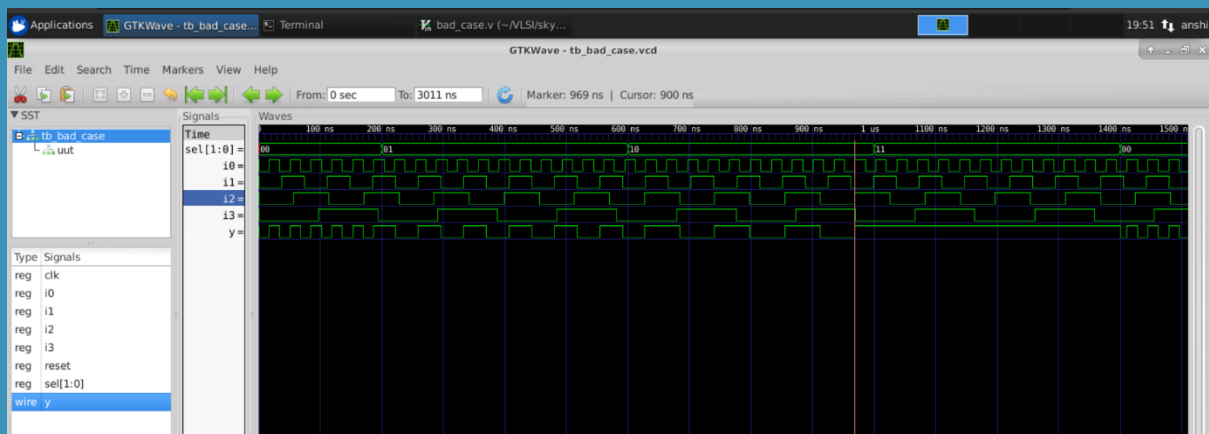
```

end
comp case.v
12,9 90%
incomp case.v
x = 12;
10,9 Top
partial_case_assign.v
7,10-31 40%
module bad_case (input i0 , input i1, input i2, input i3 , input [1:0] sel, output reg y);
always @(*)
begin
    case(sel)
        2'b00: y = i0;
        2'b01: y = i1;
        2'b10: y = i2;
        2'b11: y = i3;
        //2'b11: y = i3;
    endcase
end
endmodule

```

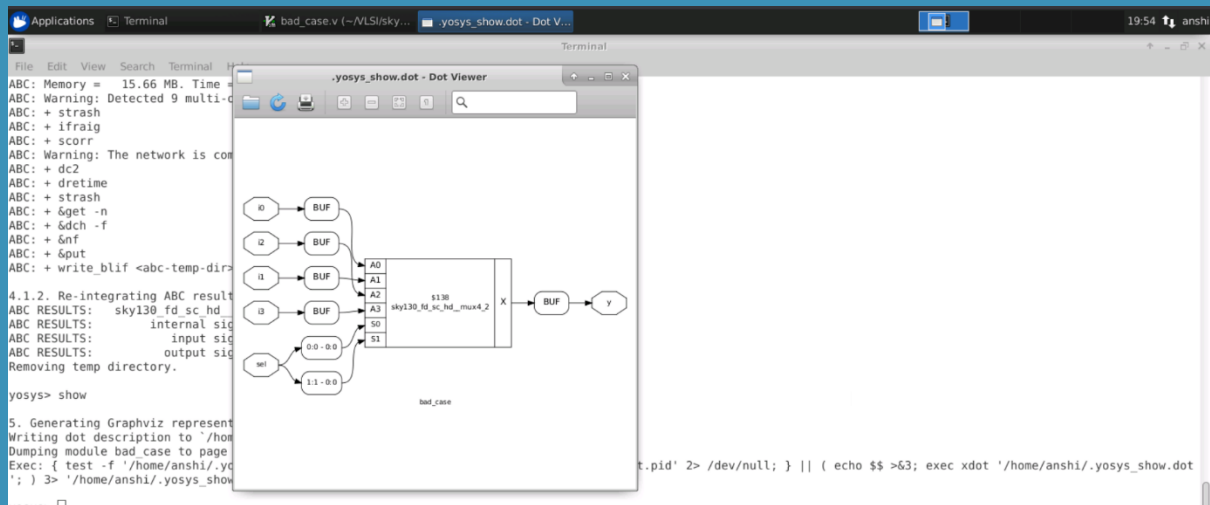
Waveform:

When sel is 11, it is latching to an output of 1 as it gets confused with the value.



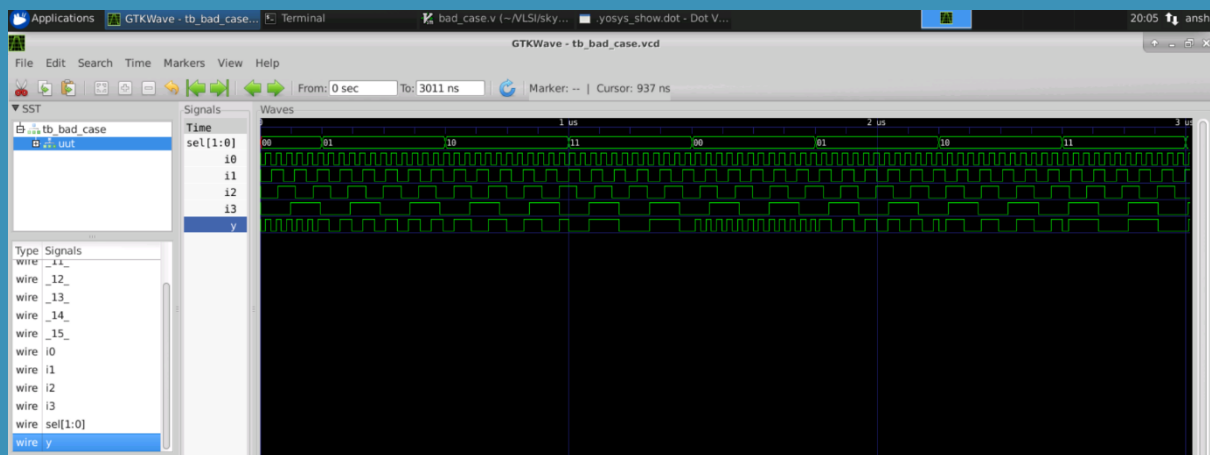
Design:

A mux_4 is created



Gtkwave of Netlist

Here y is referring to i3 when sel is 11



So, there is mismatch between synthesis and simulation wave.

Lab of For and For Generate

1. A 4:1 Mux using For

Code-Here for loop is used to create a 4:1 mux . It evaluates

```

Applications  ripple_counter.v (~VLSI...  Terminal  verilog_files - File Mana...  mux_generate.v (~VLSI...  20:54 anshi
mux_generate.v (~VLSI/sky130RTLDesignAndSynthesisWorkshop/verilog_files) - GVIM
File Edit Tools Syntax Buffers Window Help
mux_generate.v (~VLSI/sky130RTLDesignAndSynthesisWorkshop/verilog_files) - GVIM
model/sky130_fd_
model/sky130_fd_

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
~

```

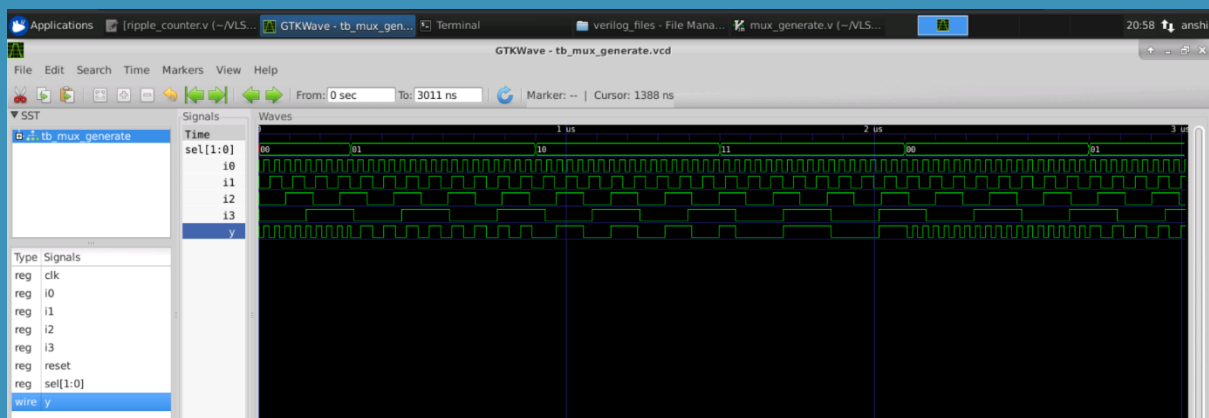
Gtkwave

So, when sel is 00 y is i0

sel is 10 y is i1

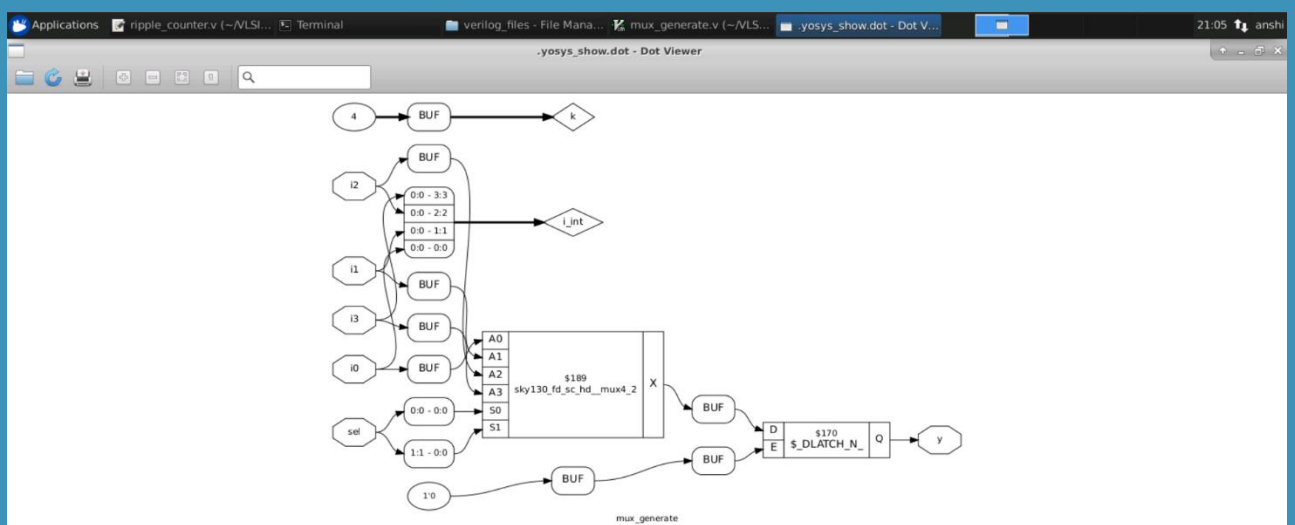
sel is 10 y is i2

sel is 11 y is i3

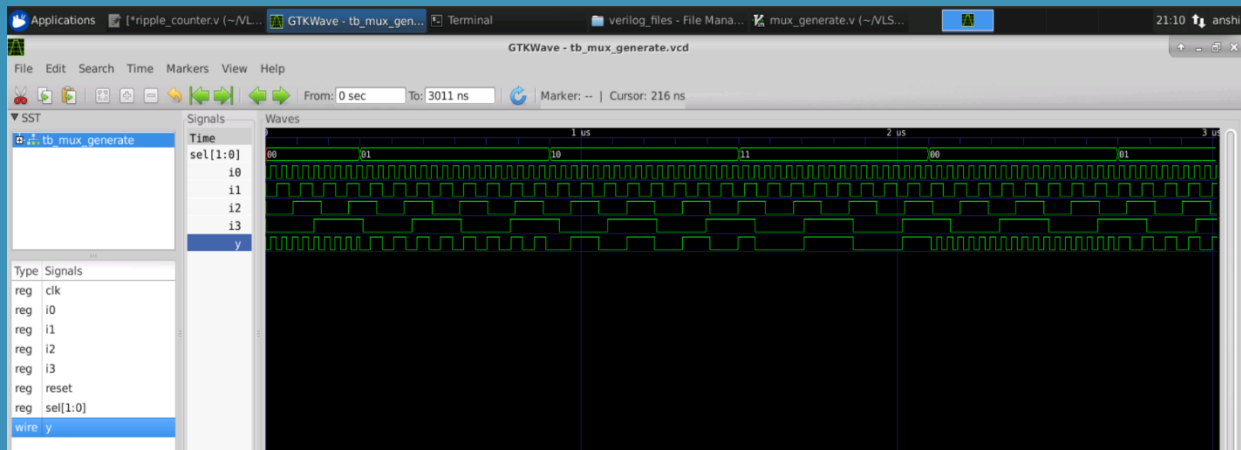


Design:

Here a 4:1 Mux is synthesized with 2 select lines and an extra D Latch because of the else block in the code.



The waveform of the netlist matches with the waveform of RTL code.



Hence, for loop generates a 4:1 mux or a mux with higher inputs and reduces the effort of writing many lines of code.

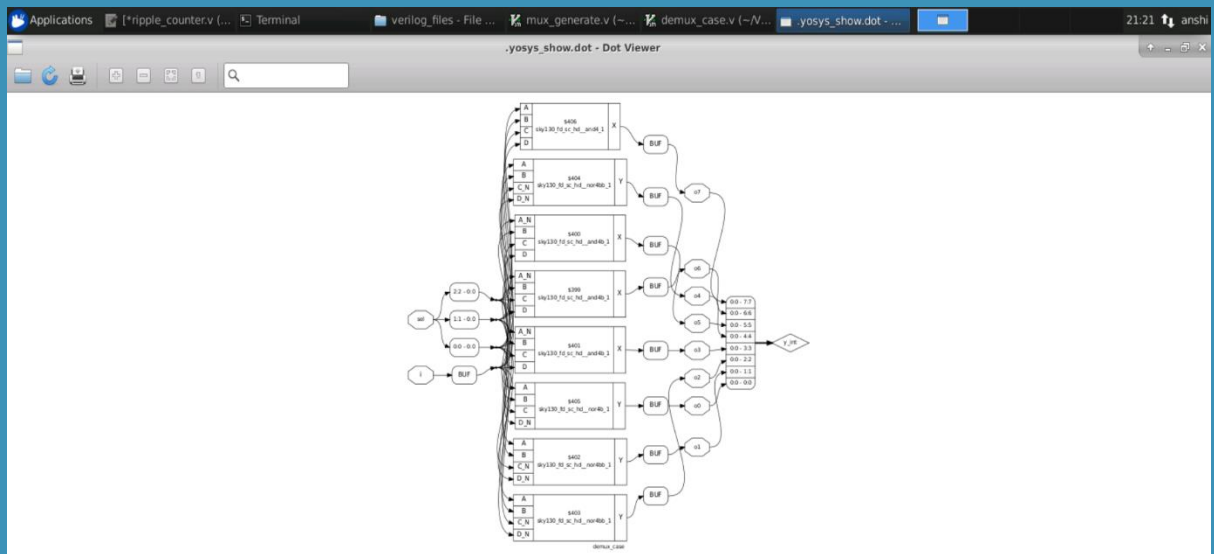
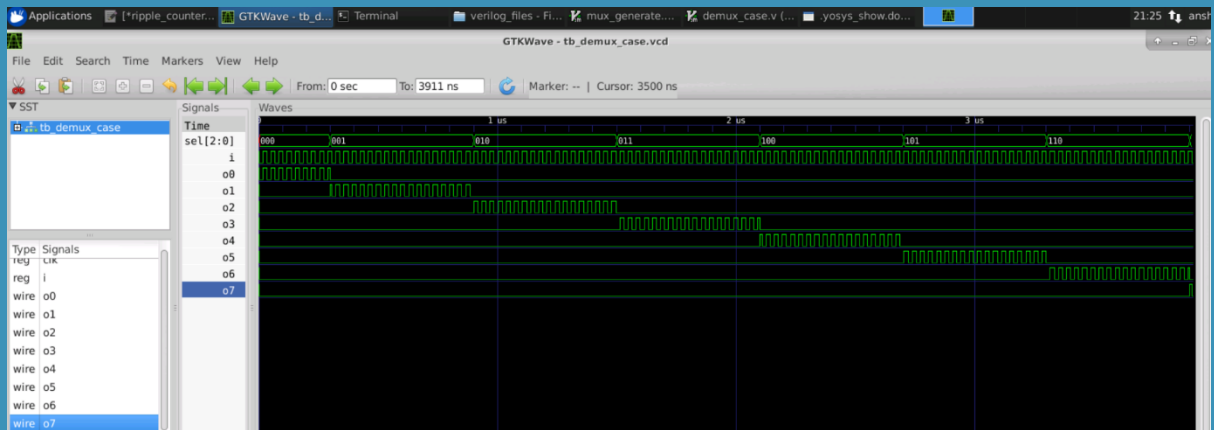
No simulation and synthesis mismatch is found.

2. Demux Using For Construct

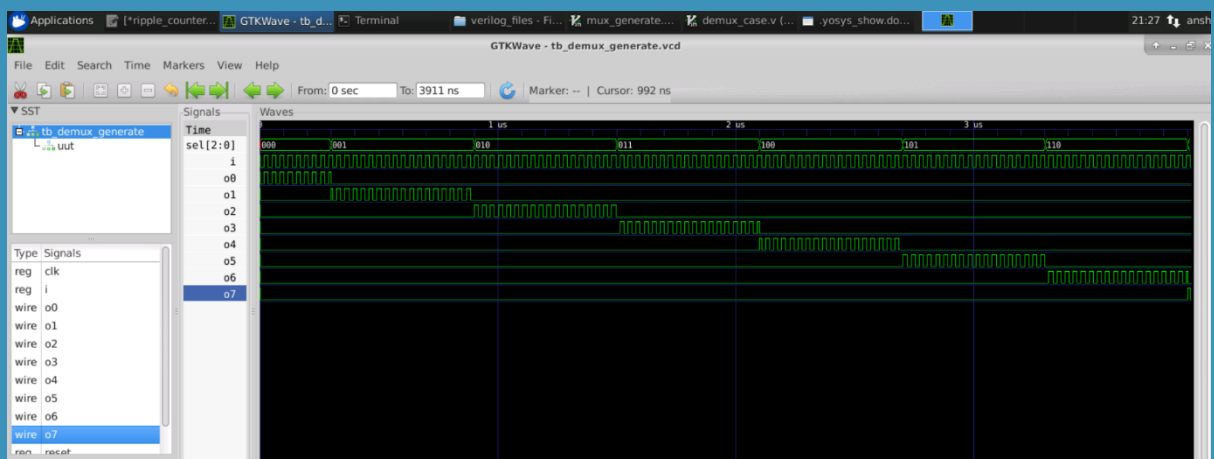
```
module demux_case (output o0 , output o1 , output o2 , output o3 , output o4 , output o5 , output o6 , output o7 , input [2:0] sel , input i);
reg [7:0] y_int;
assign {o7,o6,o5,o4,o3,o2,o1,o0} = y_int;
integer k;
always @ (*)
begin
y_int = 8'b0;
case(sel)
3'b000 : y_int[0] = i;
3'b001 : y_int[1] = i;
3'b010 : y_int[2] = i;
3'b011 : y_int[3] = i;
3'b100 : y_int[4] = i;
3'b101 : y_int[5] = i;
3'b110 : y_int[6] = i;
endcase
end
endmodule

module demux_generate (output o0 , output o1 , output o2 , output o3 , output o4 , output o5 , output o6 , output o7 , input [2:0] sel , input i);
reg [7:0] y_int;
assign {o7,o6,o5,o4,o3,o2,o1,o0} = y_int;
integer k;
always @ (*)
begin
y_int = 8'b0;
for(k = 0; k < 8; k++) begin
if(k == sel)
y_int[k] = i;
end
end
endmodule
```

Demux_case



Waveform of demux with for loop



3. Ripple Carry Adder using For Generate

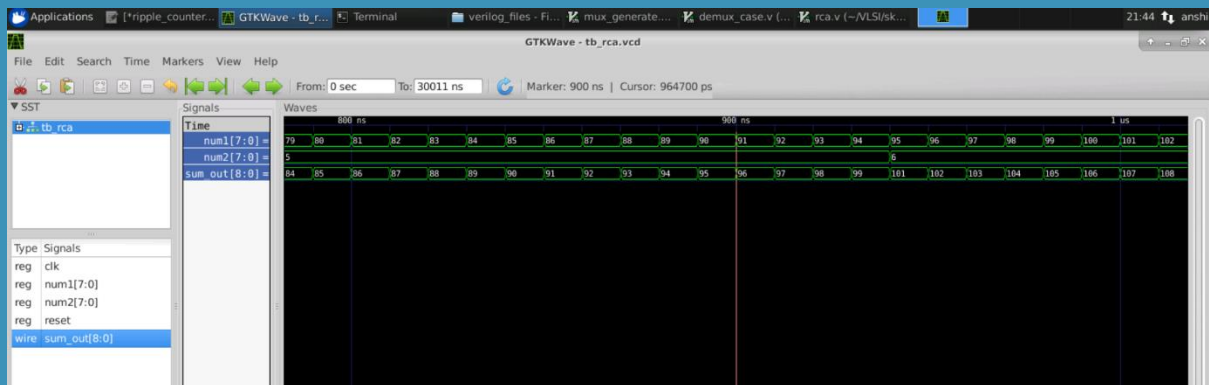
```
Applications  [ripple_counter...] Terminal  verilog_files - File ... mux_generate... demux_case.v (~V... fa.v (~VLSI/sky130... 21:35 ansh
fa.v (~VLSI/sky130RTLDesignAndSynthesisWorkshop/verilog_files) (1 of 2) - GVIM2
File Edit View Search  File Edit Tools Syntax Buffers Window Help
4.1.2. Re-integrat
ABC RESULTS: sky
ABC RESULTS: sky
ABC RESULTS: sky
ABC RESULTS: sky
ABC RESULTS: sky
ABC RESULTS: sky
ABC RESULTS: sky
Removing temp dire
yosys> write_veril
5. Executing Veril
Dumping module `d
yosys> show
6. Generating Graph
Writing dot descri
Dumping module dem
Exec: ( test -f '/f
'; ) 3> '/home/ans
yosys> exit
End of script. Log
Yosys 0.9+4081 (g

module fa (input a, input b, input c, output co, output sum);
    assign (co,sum) = a + b + c;
endmodule

fa.v
module rca (input [7:0] num1, input [7:0] num2, output [8:0] sum);
    wire [7:0] int_sum;
    wire [7:0] int_co;

    genvar i;
    generate
        for (i = 1; i < 8; i=i+1) begin
            fa_u_fa_1 (.a(num1[i]),.b(num2[i]),.c(int_co[i-1]),.co(int_co[i]),.sum(int_sum[i]));
        end
    endgenerate
    fa_u_fa_0 (.a(num1[0]),.b(num2[0]),.c(1'b0),.co(int_co[0]),.sum(int_sum[0]));
    assign sum[7:0] = int_sum;
    assign sum[8] = int_co[7];
endmodule
```

Gtkwave Waveform



Design of Ripple Carry Adder:

8 instances of Full-Adder have been created in Ripple Carry Adder using for generate.

