ECEN 651 Lab Exercise 1 Report

Logic Level Modeling in Verilog

Name: Pranav Anantharam

UIN: 734003886

Date: 09/13/2023

2. Simulate a Ripple Carry Counter.

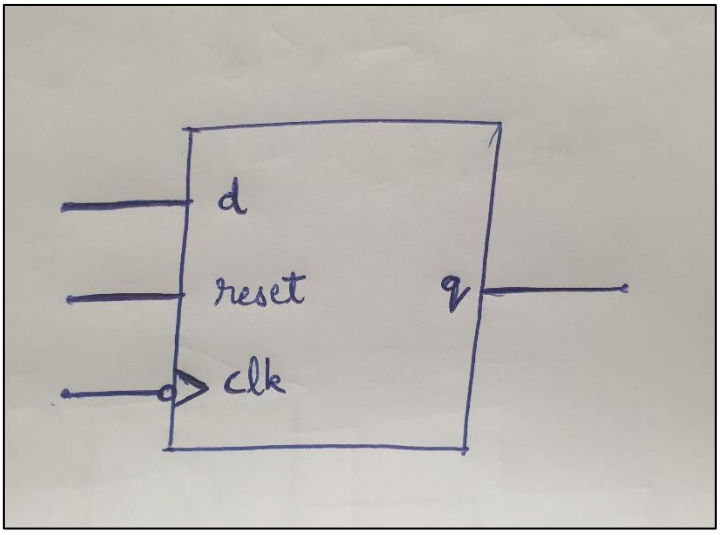
(a) Examine the code below. While attempting to understand what it is doing, draw a block diagram of the top-level module. Please use standard digital logic symbols.

Source code filename: rcc.v

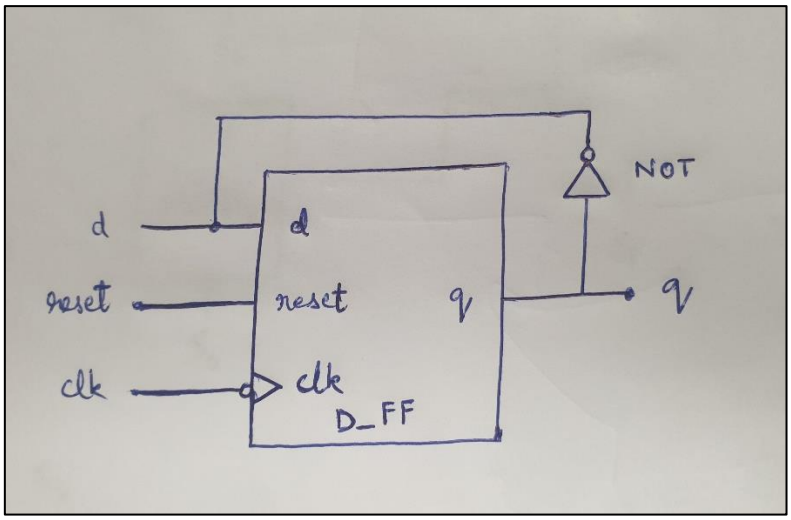
```
--------------------------------------------------------------------------------
****** Ripple Carry counter Top Block *******
--------------------------------------------------------------------------------
module RCC (q,clk,reset);
output [3:0] q;
input clk, reset;
// Four T flip flop instances are created
// The output q of each T flip flop is taken as one bit of the ripple carry counter
// The output of each T flip flop is given as input to the next T flip flop
// In this way, the T flip flops are cascaded
T_FF tff0(q[0], clk, reset);
T_FF tff1(q[1], q[0], reset);
T_FF tff2(q[2], q[1], reset);
T_FF tff3(q[3], q[2], reset);
endmodule

-----------------------------------------------
****** Flip-flop T_FF *******
-----------------------------------------------
module T_FF(q, clk, reset);
output q;
input clk, reset;
wire d;
```

```
// D flip flop instance is created
// Inputs are d, clk and reset
// Output is stored in q
D_FF dff0(q, d, clk, reset);
// Output q is complemented and used as feedback input
not n1(d, q); // not is a verilog-provided primitive
endmodule

------------------------------------------------
****** Flip_flop D_FF *******
------------------------------------------------
module D_FF(q, d, clk, reset);
output q;
input d, clk, reset;
reg q;
// Flip flop is triggered when reset is high or at negative edge of clock signal
always @(posedge reset or negedge clk)
// If reset is high, then flip flop is reset to value 0
if (reset)
q = 1'b0;
else
q = d;
endmodule
------------------------------------------------
```

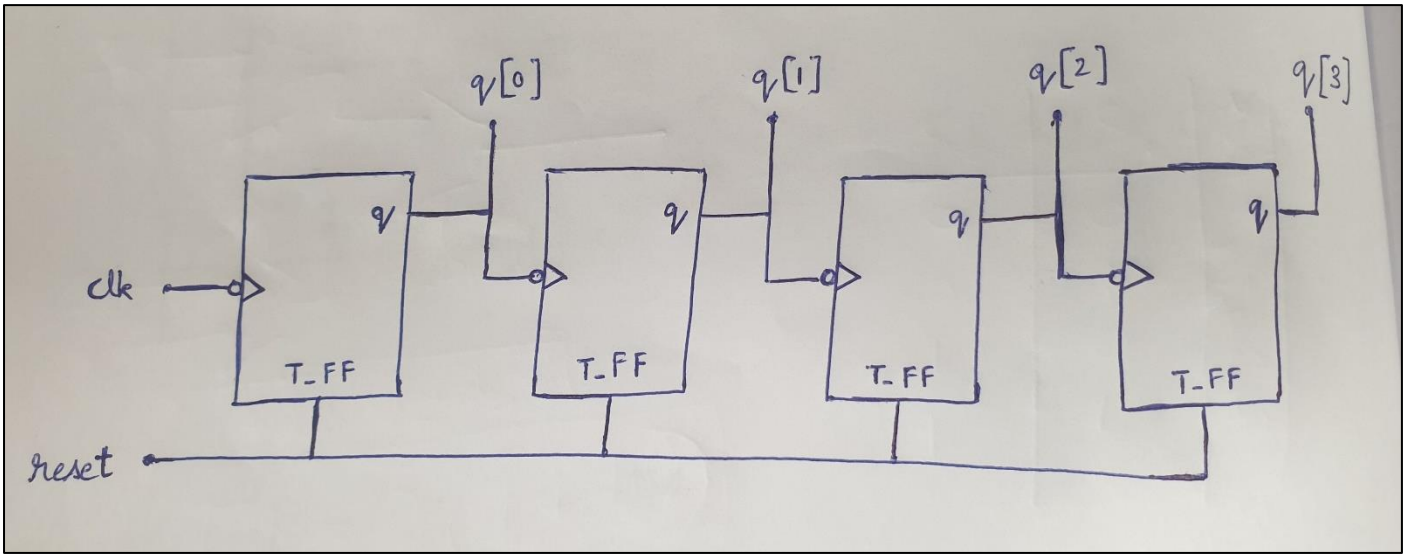
Block Diagram for D Flip-flop:



Block Diagram for T Flip-flop:



Block Diagram for Ripple Carry Counter:

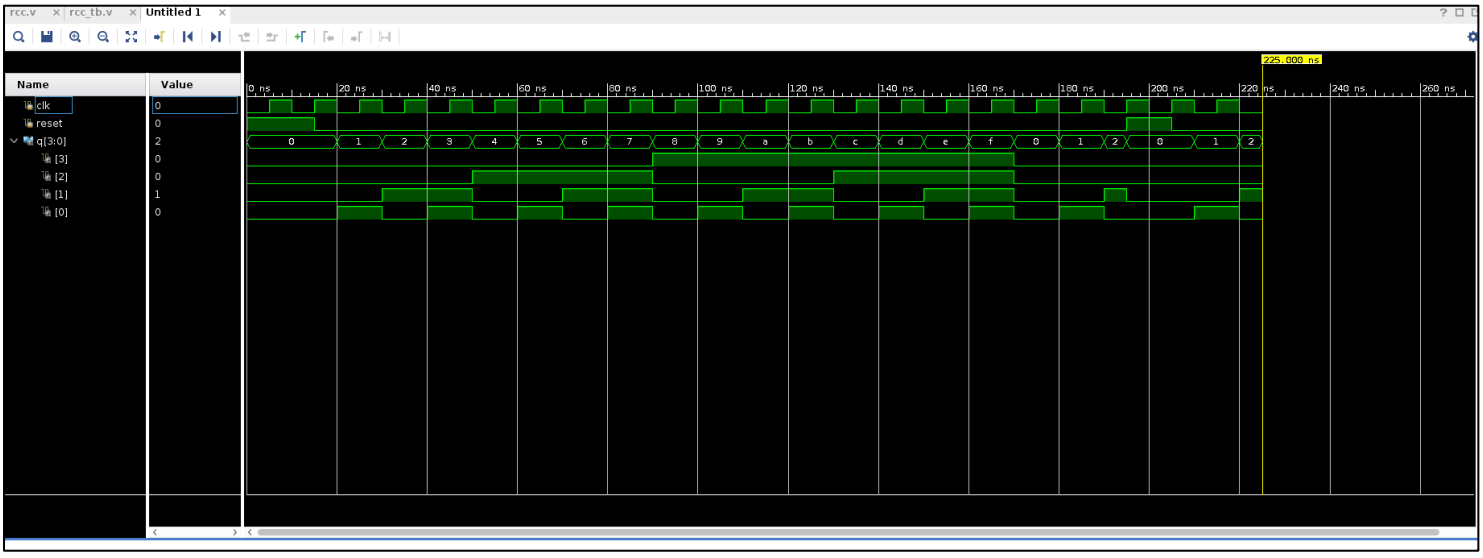


Source code filename: rcc_tb.v

```
-------------------------------------------------

****** Stimulus Block *******

-------------------------------------------------

module stimulus;

reg clk;

reg reset;

wire [3:0] q;

// instantiate the design block

RCC r1(q, clk, reset);

// control the clock signal that drives the design block. Cycle time = 10

initial
```

```
clk = 1'b0;
always
#5 clk=~clk; // toggle clk every 5 time units
// control the reset signal that drives the design block
// reset is asserted from 0 to 20 and from 200 to 220
initial
begin
reset = 1'b1;
#15 reset = 1'b0;
#180 reset = 1'b1;
#10 reset = 1'b0;
#20 $finish; // terminate the simulation
end
// Monitor the outputs
initial
$monitor($time, " Clk %b, Output q = %d",clk,q);
endmodule
-------------------------------------------------
```

Simulation Output Waveform:



Output Console Logs:

```
source
/home/grads/p/pranav_anantharam/lab1/lab1.sim/sim_1/behav/xsim/xsim.dir/stimulus_behav/web
talk/xsim_webtalk.tcl -notrace

INFO: [Common 17-206] Exiting Webtalk at Fri Sep  8 14:53:21 2023...

INFO: [USF-XSim-69] 'elaborate' step finished in '3' seconds
```

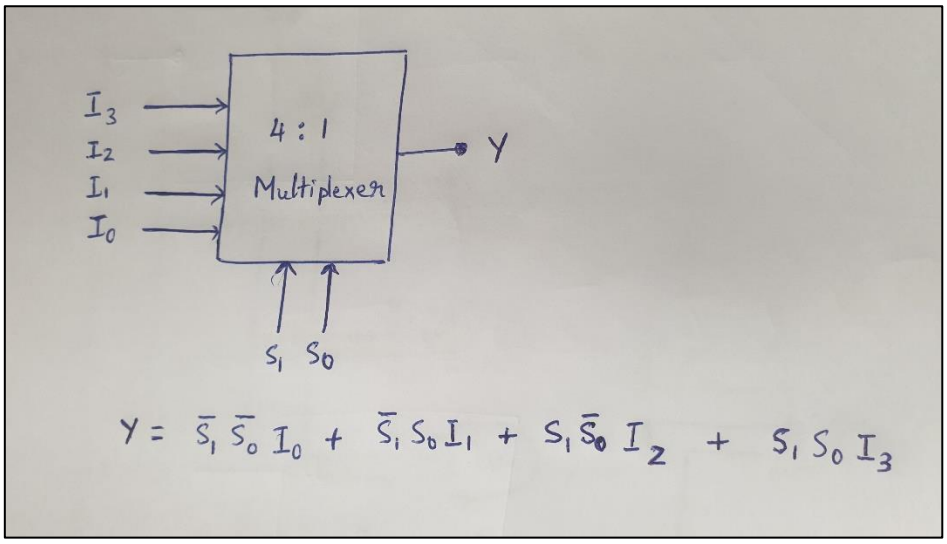
```
INFO: [USF-XSim-4] XSim::Simulate design

INFO: [USF-XSim-61] Executing 'SIMULATE' step in
'/home/grads/p/pranav_anantharam/lab1/lab1.sim/sim_1/behav/xsim'

INFO: [USF-XSim-98] *** Running xsim

   with args "stimulus_behav -key {Behavioral:sim_1:Functional:stimulus} -tclbatch
{stimulus.tcl} -log {simulate.log}"

INFO: [USF-XSim-8] Loading simulator feature

Vivado Simulator 2017.4

Time resolution is 1 ps

source stimulus.tcl

# set curr_wave [current_wave_config]

# if { [string length $curr_wave] == 0 } {

#   if { [llength [get_objects]] > 0} {

#     add_wave /

#     set_property needs_save false [current_wave_config]

#   } else {

#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start
without a wave window. If you want to open a wave window go to 'File->New Waveform
Configuration' or type 'create_wave_config' in the TCL console."

#   }

# }

# run 1000ns

                   0 Clk 0, Output q =  0

                   5 Clk 1, Output q =  0

                  10 Clk 0, Output q =  0

                  15 Clk 1, Output q =  0

                  20 Clk 0, Output q =  1

                  25 Clk 1, Output q =  1

                  30 Clk 0, Output q =  2

                  35 Clk 1, Output q =  2

                  40 Clk 0, Output q =  3

                  45 Clk 1, Output q =  3

                  50 Clk 0, Output q =  4

                  55 Clk 1, Output q =  4

                  60 Clk 0, Output q =  5

                  65 Clk 1, Output q =  5

                  70 Clk 0, Output q =  6

                  75 Clk 1, Output q =  6
```

```
 80 Clk 0, Output q =  7
 85 Clk 1, Output q =  7
 90 Clk 0, Output q =  8
 95 Clk 1, Output q =  8
100 Clk 0, Output q =  9
105 Clk 1, Output q =  9
110 Clk 0, Output q = 10
115 Clk 1, Output q = 10
120 Clk 0, Output q = 11
125 Clk 1, Output q = 11
130 Clk 0, Output q = 12
135 Clk 1, Output q = 12
140 Clk 0, Output q = 13
145 Clk 1, Output q = 13
150 Clk 0, Output q = 14
155 Clk 1, Output q = 14
160 Clk 0, Output q = 15
165 Clk 1, Output q = 15
170 Clk 0, Output q =  0
175 Clk 1, Output q =  0
180 Clk 0, Output q =  1
185 Clk 1, Output q =  1
190 Clk 0, Output q =  2
195 Clk 1, Output q =  0
200 Clk 0, Output q =  0
205 Clk 1, Output q =  0
210 Clk 0, Output q =  1
215 Clk 1, Output q =  1
220 Clk 0, Output q =  2
$finish called at time : 225 ns : File
"/home/grads/p/pranav_anantharam/lab1/lab1.srcs/sim_1/new/rcc_tb.v" Line 41
INFO: [USF-XSim-96] XSim completed. Design snapshot 'stimulus_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:05 ; elapsed = 00:00:06 . Memory (MB): peak =
6423.035 ; gain = 68.699 ; free physical = 177 ; free virtual = 17016
```

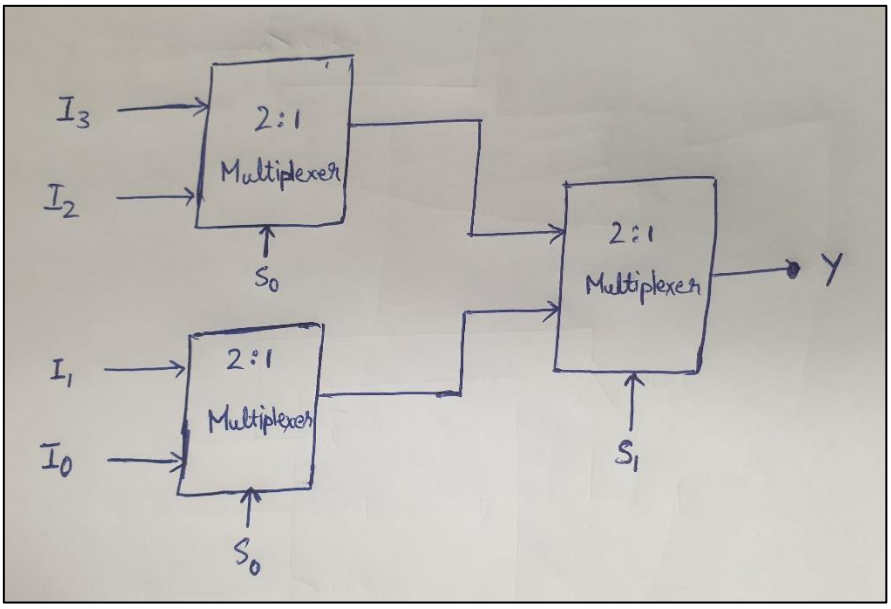
3. Design a 4-to-1 Multiplexer

(a) Draw the logic necessary to build a 4:1 Mux with select lines S0 and S1.

4:1 Multiplexer



4:1 Multiplexer Hierarchical Design (using 2:1 Multiplexers) - implemented in Verilog below



(b) Create the Verilog to describe the 4:1 Mux.

Source code filename: mux_2to1.v

```
module mux_2to1( input I0, input I1, input sel, output out);
// When sel is high, I1 is selected as output
// When sel is low, I0 is selected as output
assign out = sel ? I1 : I0;
endmodule
```

Source code filename: mux_4to1.v

```
module mux_4to1( input [3:0] I, input [1:0] sel, output out );
wire w1, w2;
// To implement a 4:1 Multiplexer, we make use of three 2:1 Multiplexers in hierarchy
// Select line sel[0] is used to select the outputs of m1(I0, I1) and m2 (I2, I3)
// Outputs of m1 and m2 are provided as inputs to m3 with sel[1] as the select line
mux_2to1 m1( I[0],I[1], sel[0], w1);
mux_2to1 m2( I[2],I[3], sel[0], w2);
mux_2to1 m3( w1,w2, sel[1], out);
endmodule
```

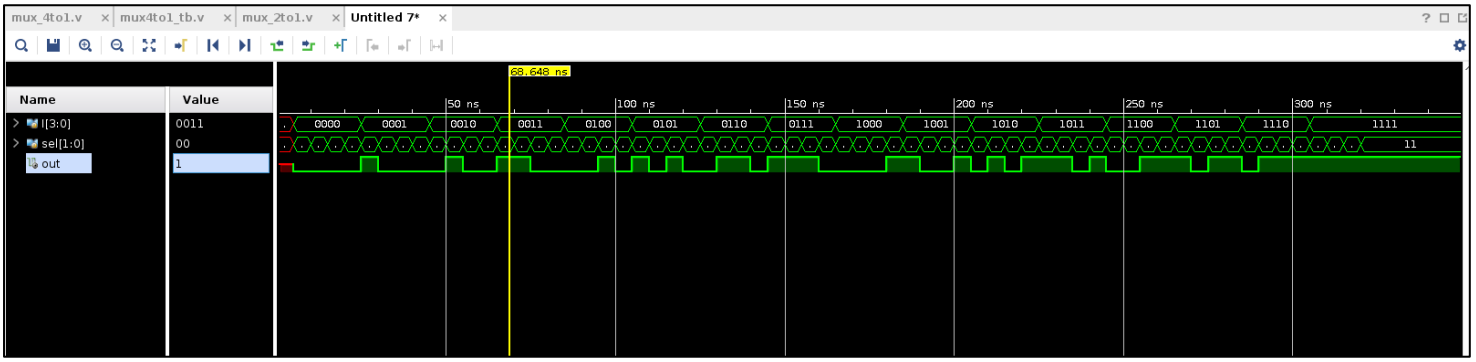
(c) Create the testbench to completely test all combinations of input.

Source code filename: mux4to1_tb.v

```
module mux4to1_tb;
reg [3:0] I;
reg [1:0] sel;
wire out;
mux_4to1 mux(.I(I), .sel(sel), .out(out));
// The data input 'I' is a 4 bit value, it ranges from 0 to 15 (16 values)
// The select line 'sel' is a 2 bit value, it ranges from 0 to 3 (4 values)
// And so, we run nested for loops to iterate through all possible inputs (64 in total)
initial begin
    for( integer i =0; i<16; i=i+1) begin
        for( integer j =0; j<4; j= j+1 ) begin
            // Providing a delay of 5 time units
            #5
            // Updating data input and select line values
            I = i;
            sel = j;
            end
        end
end
endmodule
```

(d) Simulate the testbench in ISE.

Simulation Output Waveform:



4. Answer the following review questions:

4. (a) What does the $monitor statement in the provided test bench do? What about $finish?

Answer:
The $monitor statement in the test bench is used to print the values of the arguments whenever there is a change in values of the arguments.

The $finish statement in the test bench is used to create a Verilog system task that tells the simulator to terminate the current simulation. It finishes a simulation and exits the simulation process.

4. (b) Similarly, what do the # symbols signify in the provided test bench? What about $time?

Answer:
The '#' symbol is used to provide a delay in the testbench code.

Example:

```
begin
#15
c = a + b
end
```

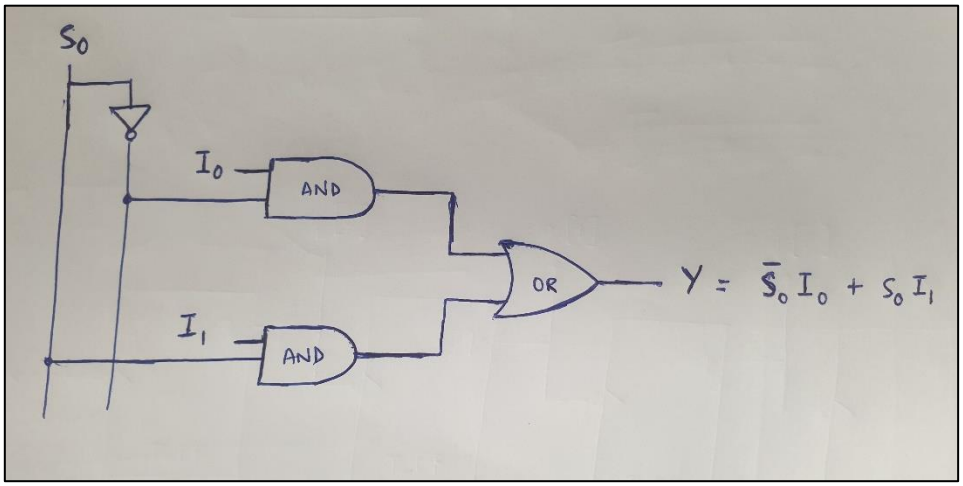
In this example, we add 15 units of time delay before executing the statement.

The $time statement in the test bench returns the current simulation run time. The current simulation run time will be reported in simulation time units.

4. (c) The 4:1 multiplexer above can be built directly using gates, or it can be constructed using three 2:1 multiplexers, of which are constructed from gates. Provide some intuition as to how the delay through the 4:1 multiplexer would differ for both of the aforementioned cases.

Answer:

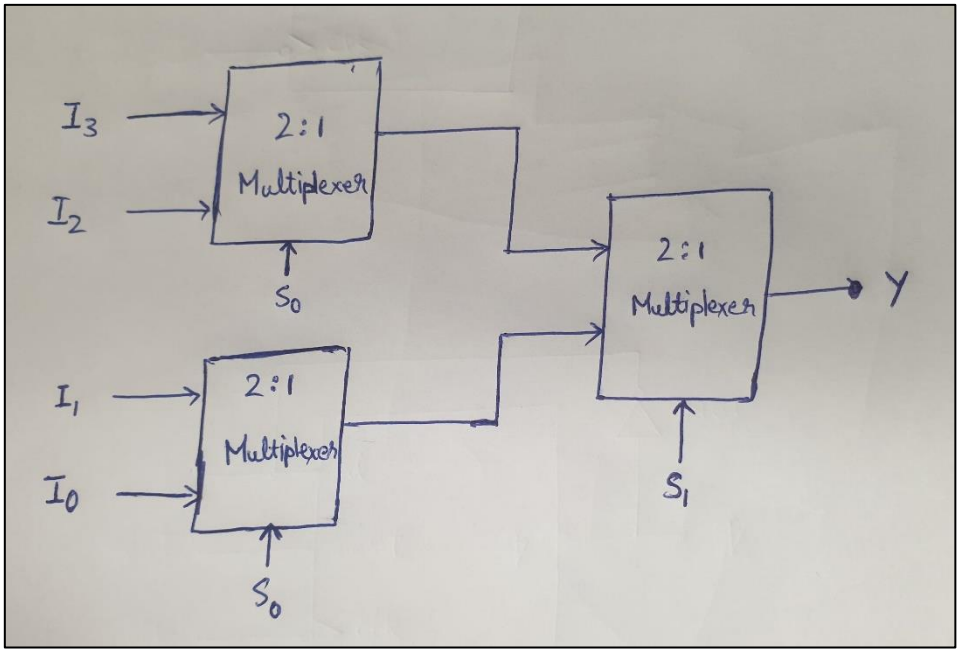
2:1 Multiplexer using logic gates



On implementing a single 2:1 Multiplexer using logic gates, we can calculate the delay of the longest path to be the sum of the delays of the NOT, AND and OR gates. Let us consider the delays to be Tnot, Tand and Tor, then the total delay for a 2:1 multiplexer using gates will be:

T(2:1 using gates) = Tnot + Tand + Tor

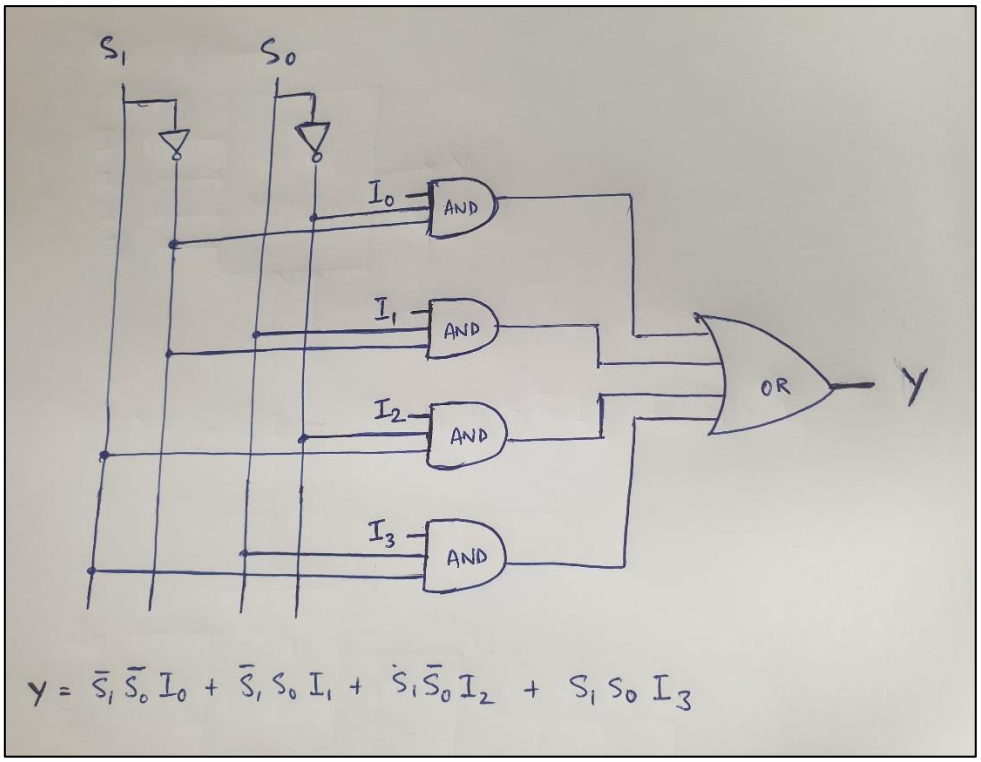
4:1 Multiplexer using 2:1 Multiplexers



And so, on constructing a 4:1 multiplexer using three 2:1 multiplexers, the overall delay will be the delay due to 1st stage 2:1 multiplexer and 2nd stage 2:1 multiplexer. Total delay for 4:1 multiplexer using three 2:1 multiplexers will be:

T(4:1 using hierarchy) = 2*Tnot + 2*Tand + 2*Tor → Result 1

4:1 Multiplexer using logic gates



On constructing a 4:1 multiplexer directly using logic gates, we can calculate the total delay for the longest path as:

T(4:1 using gates) = Tnot + Tand + Tor → Result 2

Comparing results 1 and 2, we can conclude that constructing a 4:1 multiplexer directly using logic gates would result in a lower delay than using three 2:1 multiplexers in hierarchy.