Digital Assignment Report

EEE F313: Analog & Digital VLSI Design Problem Set 85

Birla Institute of Technology & Science, Pilani - Pilani Campus First Semester - Academic Year 2024-25

Vaidik A Sharma 2021B5A82509P Akshit Saxena 2021B5A30886P Deeksha Agarwal 2022A3PS0500P

September 25, 2024

**Abstract**

The problem statements, design approaches, and optimization processes used to accomplish the objectives outlined in the problem statement are all detailed in this report. Out of all the problems that were offered, this group decided to use Problem Set 85 for the project. There are two components to the problem: (1) Design of a three-input XOR/XNOR function based on CPL (implemented in MICROWIND software) (2) Using Verilog HDL, a machine is designed to detect three consecutive heads. The report displays the input and output waveforms as a result of the design process. We would like to express our gratitude to Prof. Anu Gupta for giving our team this exceptional chance. The team members express their gratitude to individuals who provided indirect assistance, especially with the MICROWIND Software lessons.

# CPL Implementation of Three Input XOR/XNOR Gate

**Problem Statement** Design of a 3-input XOR/NXOR logic gate using CPL (Complementary pass-transistor logic) logic style at 1GHz with load capacitance of 500 fF.

**Theory & Calculations** The primary design goal is to make the gate operational at 1GHz frequency. The parameters concerned with this are the *τpHL* which is the time taken for the high to low transition of the output upon input change, and the *τpLH* which is the time taken for the low to high transition of the output upon input change.

The definitions of *τpHL* & *τpLH* are given as follows:

*τ* = *Cload*  2 |*VT*0*,n*| + *ln*  4(*VOH* − |*VT*0*,n*|) − 1 (1)

*pHL*

*kn*(*VOH* − |*VT*0*,n*|)

*VOH* − |*VT*0*,n*|

*VOH* + *VOL*

*τ* = *Cload*  2 |*VT*0*,p*| + *ln*  4(*VOH* − |*VT*0*,p*|) − 1 (2)

*pLH*

*kp*(*VOH* − |*VT*0*,p*|)

*VOH* − |*VT*0*,p*|

*VOH* + *VOL*

The values of *kn* and *kp* are given by:

*kn* = *µn*

*C Wn ox Ln*

(3)

*kp* = *µpC*

*Wp ox Lp*

(4)

The value of *Cload* is the Load capacitance(provided in the Problem Statement as 500pF) in parallel with parasitic capacitances(*CGD* & *CDB*) of the output MOSFETs. Since the Load Capacitance is considerably higher than the typical values of *CGD* & *CDB*, hence value of *Cload* can be approximated to 500pF without incresing error percentage.

For a balanced output, design parameters *kn* & *kp* were assumed to be equal. Hence, the relation *Wn* = *µp*

is obtained.

**Optimisation / Innovation** The 180nm technology node used in this project had a *µn*

*µ*

*p*

*Wp µn*

ratio of 1.9 .

The MICROWIND Software introduces a constraint of minimum length of any drawing to be of 200nm, thus

making the gate size to be 200nm in all MOSFETs.

The *W* ratios were optimised such as to minimise Drain Currents and thus power. However pure optimisation of drain currents leads to reduced levels of *VOH* and increased levels of *VOL*. Hence, a combined optimisation of the two parameters was undertaken.

*L*

**Schematic** The Schematic of the Three Input XOR/XNOR Gate implementation in CPL technology is shown in Figure [1](#_bookmark0).

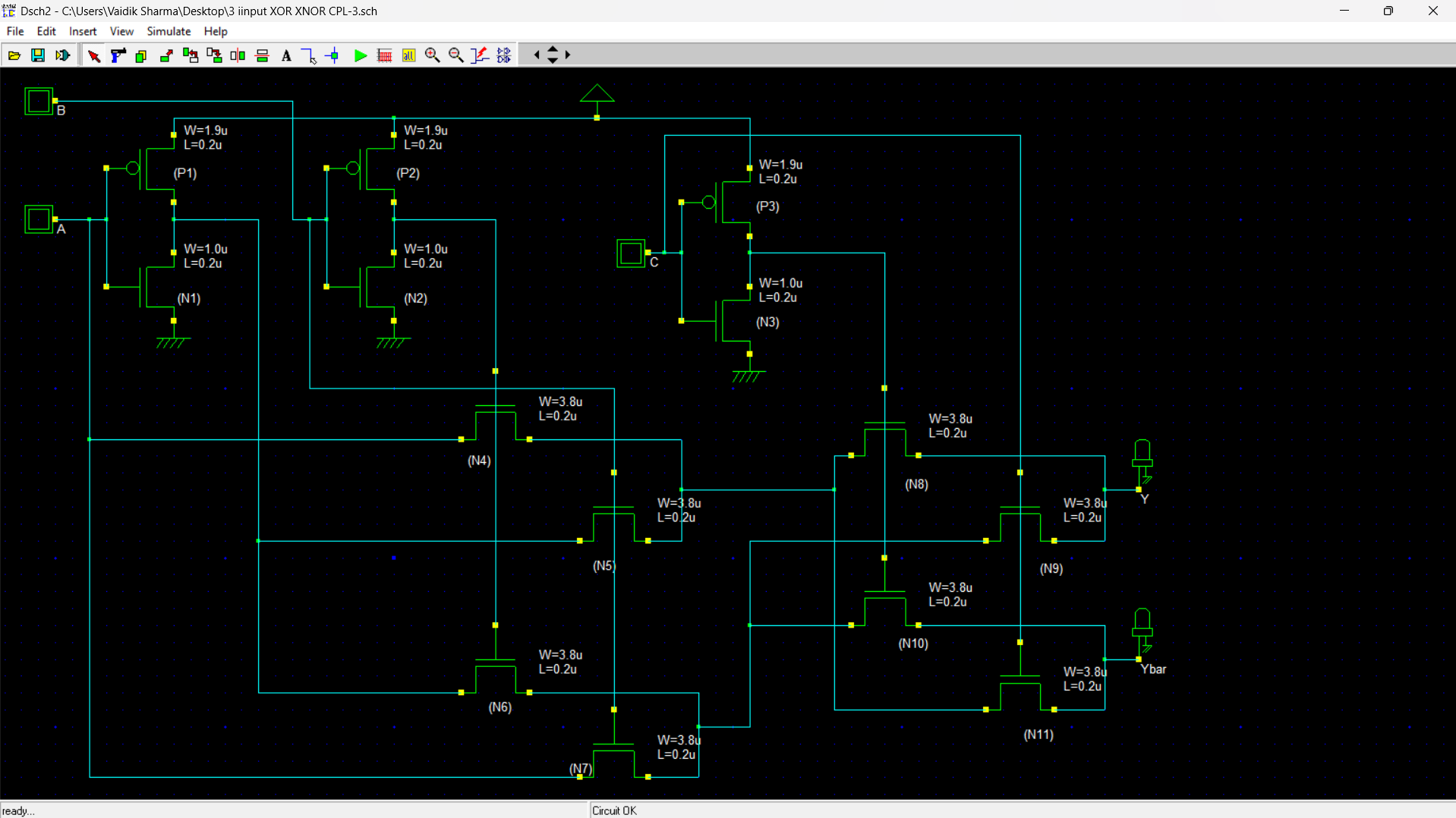


Figure 1: Schematic of CPL Implementation of Three Input XOR/XNOR.

*W* **of MOSFETs** The Table [1](#_bookmark1) gives the values of *W*

*L*

*L*

ratios of all the PMOSs and NMOSs used in the

design and implementation of CPL three input XOR/XNOR gate.

Table 1: *W*

*L*

Ratios of all MOSFETs used in Schematic.

|  |  |  |  |
| --- | --- | --- | --- |
| **Inverter Label** | *Wn Ln* | **CPL Label** | *Wp*  *Lp* |
| N1 | 5 | N4 | 19 |
| N2 | 5 | N5 | 19 |
| N3 | 5 | N6 | 19 |
| P1 | 9.5 | N7 | 19 |
| P2 | 9.5 | N8 | 19 |
| P3 | 9.5 | N9 | 19 |
|  |  | N10 | 19 |
|  |  | N11 | 19 |

**Layout of XOR/XNOR Gate** The Silicon layout of the three input CPL implementation XOR/XNOR Gate is shown in Figure [2](#_bookmark2).

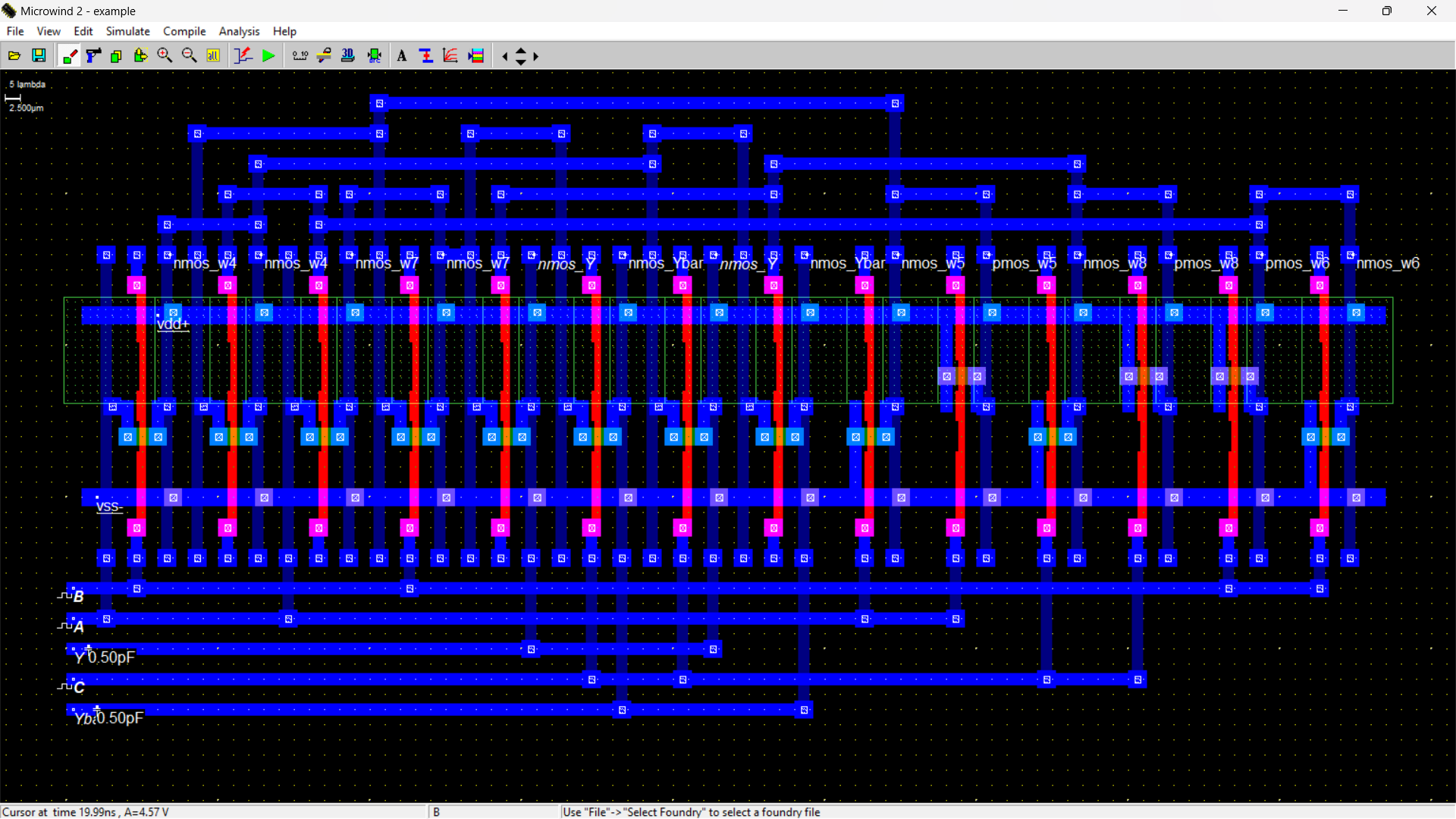


Figure 2: Layout of CPL Implementation of Three Input XOR/XNOR.

**Gate Function Output Waveforms** The XOR/XNOR Gate implemented in CPL Implementation has the function F *=* A⊕B⊕C/ . The Signal A has a time period of 1ns, while the B signal has a time period of 4ns and C signal has a time period of 2ns which corresponds to 1GHz frequency of input signal. Both the input signals have *τrise* = *τfall* = 0*.*050*ps*. The output of the XOR/XNOR Gate can be verified in Figure [3](#_bookmark3)(a), 3(b).

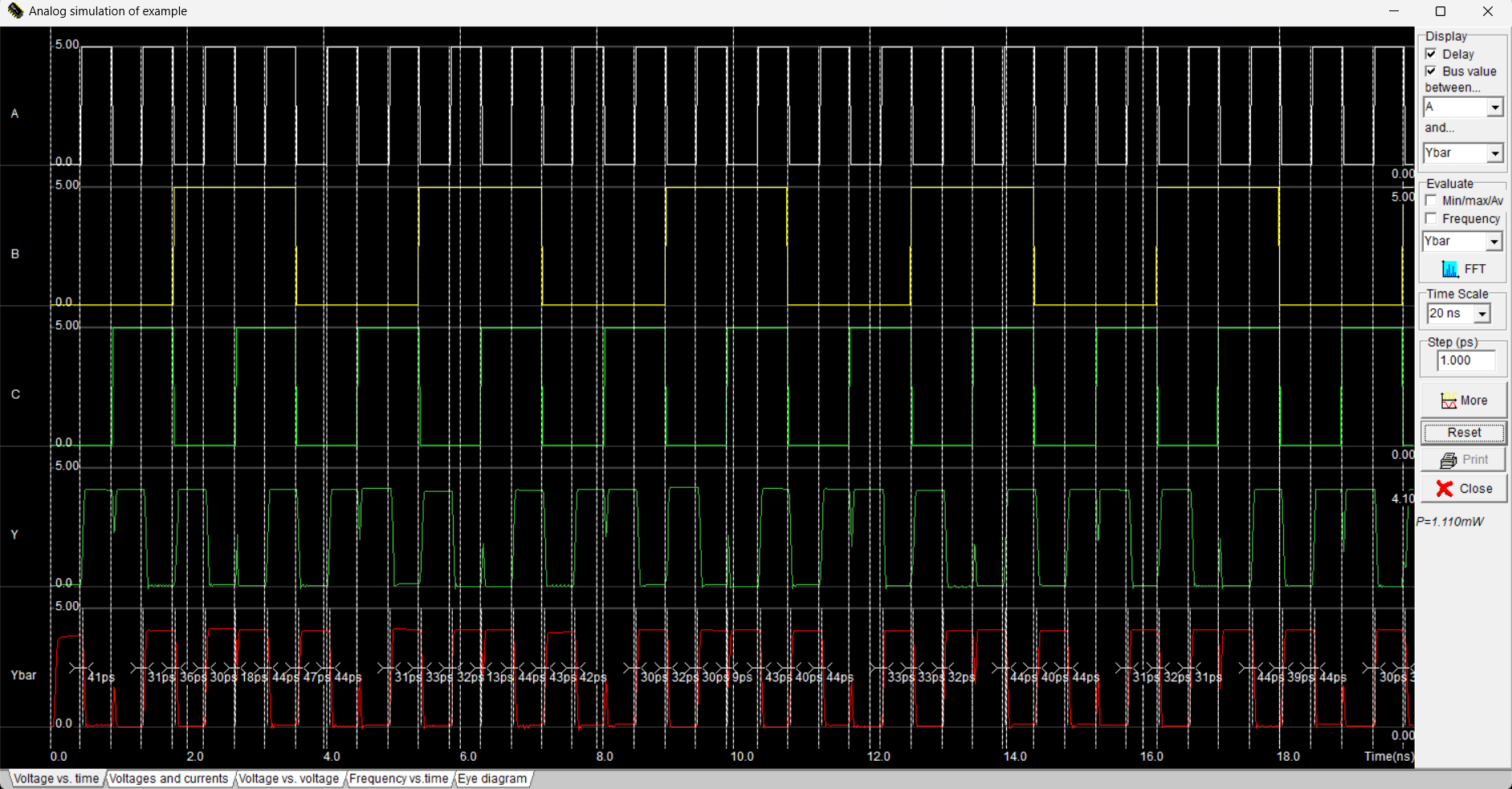


Figure 3(a): Time signal verification of Function Implementation of XOR/XNOR Gate without Load Capacitance.

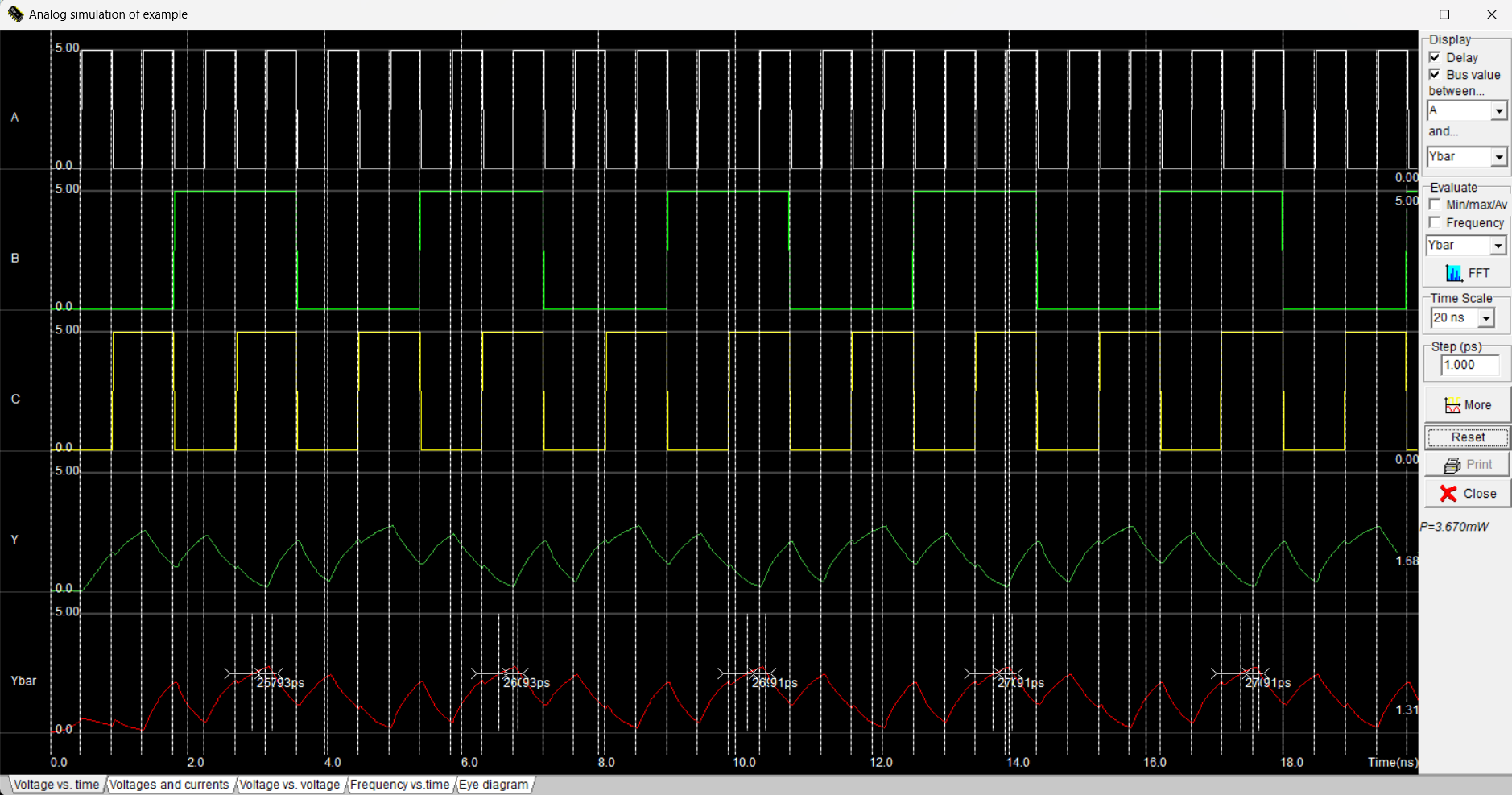
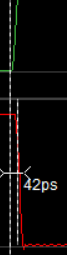


Figure 3(b): Time signal verification of Function Implementation of XOR/XNOR Gate with Load Capacitance as 500fF or 0.5pF.

The*τpHL* = 42*ps* & *τpLH* = 31*ps* for the designed gate can be seen in Figures [4a](#_bookmark4) & [4b](#_bookmark4) respectively.



1. *τpHL* =42*ps* (b) *τpLH* = 31*ps*

**Conclusion** The CPL implementation of the three-input XOR/XNOR Gate designed satisfies the require- ments of the problem statement of providing functionality at 1GHz rate of input by providing *τpLH* = 31*ps* & *τpHL* = 42*ps*. The power consumed by the gate is 3.670mW. This power is mainly due to short circuit power +due to *τrise* = *τfall* =0.01nsof the input signals.

# Heads in Toss Detection Machine

**Problem Statement** Design a Machine which can detect 3 consecutive heads in a sequence of random trials of tossing fair coin. Use one hot assignment for defining the states.

**INTRODUCTION:**

In this task, we aim to design a state machine (finite state automaton) that detects 3 consecutive heads (HHH) in a sequence of coin tosses. The coin is fair, meaning that the probability of heads (H) or tails (T) in any individual toss is 0.5. The machine should output a signal indicating when three heads occur in succession.

The design will use **one-hot assignment** for defining the states, where each state will be represented by a unique bit configuration. The machine will have a finite number of states, and transitions between these states will occur based on the result of each coin toss. The final state will indicate that 3 consecutive heads have been observed.

**STATE MACHINE DESIGN:**

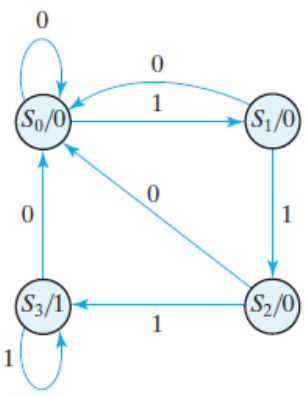
We define the following states for our machine:

* **State S0:** The initial state where no heads have been detected.
* **State S1:** A state indicating that one head (H) has been detected.
* **State S2:** A state indicating that two consecutive heads (HH) have been detected.
* **State S3:** A state indicating that three consecutive heads (HHH) have been detected. This is the accepting state, where the machine detects the target sequence.
* **Reset States:** Transitions will occur back to previous states upon detecting tails (T) or breaking the sequence of consecutive heads.

**STATE TRANSITIONS:**

The transitions between states depend on the outcome of each coin toss. The machine should transition between the states as follows:

* **S0 (Initial State)**:
  + If the toss is T, stay in S0.
  + If the toss is H, transition to S1.
* **S1 (One Head Detected)**:
  + If the toss is T, return to S0.
  + If the toss is H, transition to S2.
* **S2 (Two Heads Detected)**:
  + If the toss is T, return to S0.
  + If the toss is H, transition to S3.
* **S3 (Three Consecutive Heads Detected)**:
  + Once this state is reached, the machine has detected the desired sequence of three consecutive heads. It will remain in this state or reset based on the design preference.



Verilog Icarus 12.0 Code-

module ThreeConsecutiveHeads (

input wire clk,

input wire reset,

input wire coin, // Input coin toss (H = '1', T = '0')

output reg detected // Output (1 when 3 consecutive heads detected)

);

// State encoding using one-hot encoding

typedef enum reg [3:0] {

S0 = 4'b0001, // No heads

S1 = 4'b0010, // One head

S2 = 4'b0100, // Two consecutive heads

S3 = 4'b1000 // Three consecutive heads

} state\_type;

state\_type state, next\_state;

// State register process

always @(posedge clk or posedge reset) begin

if (reset) begin

state <= S0; // Start in S0 state

end else begin

state <= next\_state;

end

end

// Next state logic and output logic

always @(\*) begin

// Default assignments

next\_state = S0;

detected = 1'b0;

case (state)

S0: begin

if (coin == 1'b1) // H

next\_state = S1;

else // T

next\_state = S0;

end

S1: begin

if (coin == 1'b1) // H

next\_state = S2;

else // T

next\_state = S0;

end

S2: begin

if (coin == 1'b1) // H

next\_state = S3;

else // T

next\_state = S0;

end

S3: begin

detected = 1'b1; // Three consecutive heads detected

if (coin == 1'b1) // H

next\_state = S3;

else // T

next\_state = S0;

end

default: begin

next\_state = S0;

end

endcase

end

endmodule

//TestBench

module tb\_ThreeConsecutiveHeads;

// Inputs

reg clk;

reg reset;

reg coin;

// Output

wire detected;

// Instantiate the Unit Under Test (UUT)

ThreeConsecutiveHeads uut (

.clk(clk),

.reset(reset),

.coin(coin),

.detected(detected)

);

// Clock generation

always #5 clk = ~clk; // Clock with a period of 10 time units

// Stimulus block

initial begin

// Initialize Inputs

clk = 0;

reset = 1;

coin = 0;

// Apply reset

#10 reset = 0;

// Apply test vectors

#10 coin = 1; // First head (H)

#10 coin = 1; // Second head (H)

#10 coin = 0; // Tail (T)

#10 coin = 1; // First head (H)

#10 coin = 1; // Second head (H)

#10 coin = 1; // Third head (H) - Should detect 3 consecutive heads

#10 coin = 0; // Tail (T)

#10 coin = 1; // First head (H)

#10 coin = 1; // Second head (H)

#10 coin = 1; // Third head (H) - Should detect 3 consecutive heads again

#10 $finish; // End the simulation

end

// Monitor the output

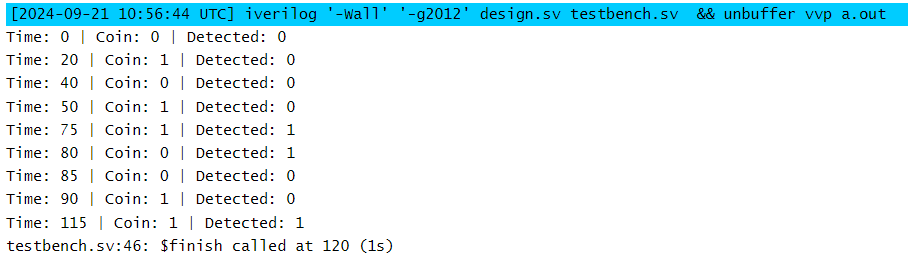
initial begin

$monitor("Time: %0t | Coin: %b | Detected: %b", $time, coin, detected);

end

endmodule

Output-



**CONCLUSION:**

The state machine designed above efficiently detects three consecutive heads in a sequence of coin tosses. By using one-hot encoding for the states, the transitions between states are simplified, and the design can be implemented using basic digital logic components such as flip-flops and combinational circuits.

The detection process is summarized as follows:

* The machine starts in state S0.
* It transitions between states based on the outcome of the coin toss (H or T).
* Upon detecting three consecutive heads, the machine reaches state S3, where it outputs a detection signal.
* The machine can be reset after detection to look for the next occurrence of three consecutive heads.