Special assignment VLSI

Two Bit Counter

Yash Patel

Department of Engineering, Nirma University, Ahmedabad, Gujarat, India

E-mail: 22BEC093@nirmauni.ac.in

Received 11 September 2024; revised 09 October2024; accepted 16 October 2024

INDEX:

1. Objective and Motivation

2. Introduction

3. Optimised Boolean equation.

4. Optimised gate-level circuit diagram.

5. The transistor level schematic for CMOS/MOS implementation.

6. Stick a diagram for the above implementation level using the proper color code

7. The layout using the Microwind tool.

8. Simulate it for various combinations of inputs.

9. Measure the rise time, fall time, propagation delay and other parameters.

10. Conclusion

11. Reference

12. Verilog code

1. Objective and Motivation:

The objective of this report is to design and implement a 2-bit counter using transistor-level logic, focusing on the construction of a counter that cycles through four states (00, 01, 10, 11). This involves demonstrating the use of basic logic gates and flip-flops at the transistor level, specifically using NMOS and PMOS transistors. The report also aims to analyze the counter's behaviour during state transition. Additionally, the report seeks to explore the practical applications of such counters in digital systems, like timers and control units.

The motivation for this project arises from a desire to deepen understanding of digital design principles by moving beyond abstract logic to transistor-level implementation. It bridges the gap between theory and practical applications in IC design, essential for system-level development. This project is also a step towards gaining expertise in VLSI design, as counters are foundational in digital electronics. By engaging in this design, the report promotes skills in circuit simulation and verification, preparing for more advanced digital systems and future opportunities in hardware design.

2. Introduction:

A 2-bit up-down counter is a fundamental digital circuit used to count in both ascending (up) and descending (down) sequences, depending on the selected mode of operation. In this report, we focus on the design and implementation of a 2-bit up-down counter, with Mode 0 functioning as the up-counting mode and Mode 1 as the down-counting mode. This counter sequentially cycles through four possible states, represented in binary as 00, 01, 10, and 11, with the ability to either increment or decrement based on the mode input.

The importance of up-down counters lies in their wide range of applications in digital electronics, such as frequency dividers, digital clocks, and control systems. By switching between up and down counting, they offer flexible functionality for tasks like reversible counting, position tracking in control systems, and bidirectional event counting.

Asynchronous counters, often called ripple counters, have certain advantages over synchronous counters due to their simpler design and reduced hardware requirements. In an asynchronous counter, only the first flip-flop is driven by the clock, and each subsequent flip-flop is triggered by the previous one’s output. This creates a chain-like or "ripple" effect, where the changes propagate through the flip-flops in sequence, rather than all at once.

This ripple effect means that asynchronous counters are generally simpler to implement, needing fewer components and requiring less wiring compared to synchronous counters. Because of this, asynchronous counters are often more cost-effective and take up less space on a circuit board. The reduced component count also helps with minimizing power consumption, making asynchronous counters more energy-efficient, which is particularly beneficial in low-power or resource-constrained applications.

Another advantage is the flexibility of design. Since asynchronous counters do not require each flip-flop to switch simultaneously, they are easier to set up for different counting sequences. This allows asynchronous counters to handle a wide range of counting patterns, making them adaptable for various applications without extensive redesigning.

However, while asynchronous counters are simpler and consume less power, they may encounter issues with propagation delay, as the output takes time to propagate through each flip-flop in sequence. For applications where precision timing is essential, a synchronous counter would generally be preferable, even if it requires more complex circuitry. But for many simpler or lower-frequency applications, the asynchronous counter's advantages in terms of simplicity, efficiency, and cost-effectiveness make it the preferred choice.

3. The optimized Boolean expression:

To implement the 2-bit counter we need to refer to the finite state machine concepts. So before constructing the truth table, we are going to create a state diagram as per our requirement. We are going to implement the truth table using a D flip-flop.
We are considering an asynchronous counter where the first flip-flop is only driven by a clock and other are driven by the output of the first flip-flop.

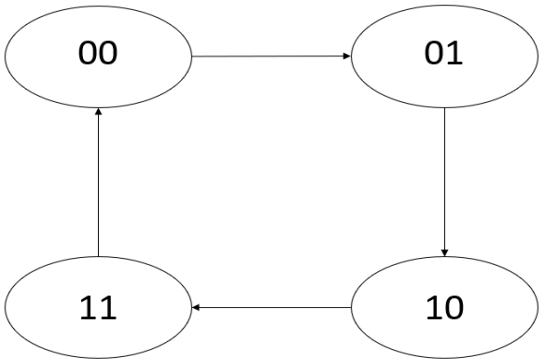

Fig1. The state diagram for up counter

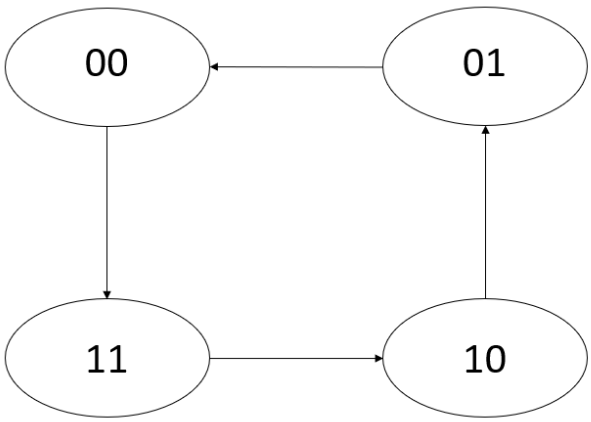

Fig2. The state diagram for down counter

Now let's look at the circuit diagram for the counter and the timing diagram. In the asynchronous counter, the clock is provided to the Q1 D flip-flop while the other corresponding flip-flops are driven by the Q1 invert output for the up counter and Q1 as the clock will give us the down counter. We are going to implement both counters using different methodologies. The up counter will be constructed using the dsch and we transfer that as a Verilog file and simulate the Verilog file in microwind while the down counter will be constructed manually.

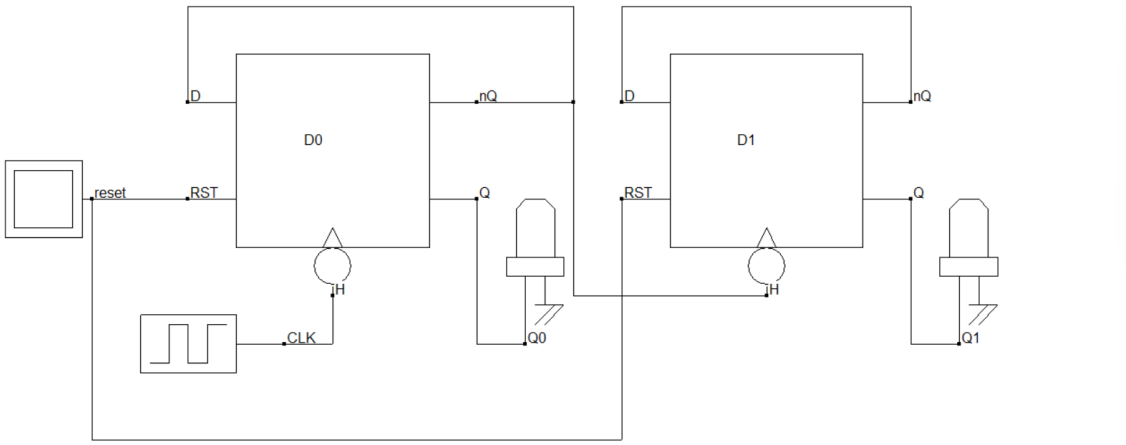

Fig3. The circuit diagram for the 2-bit up counter

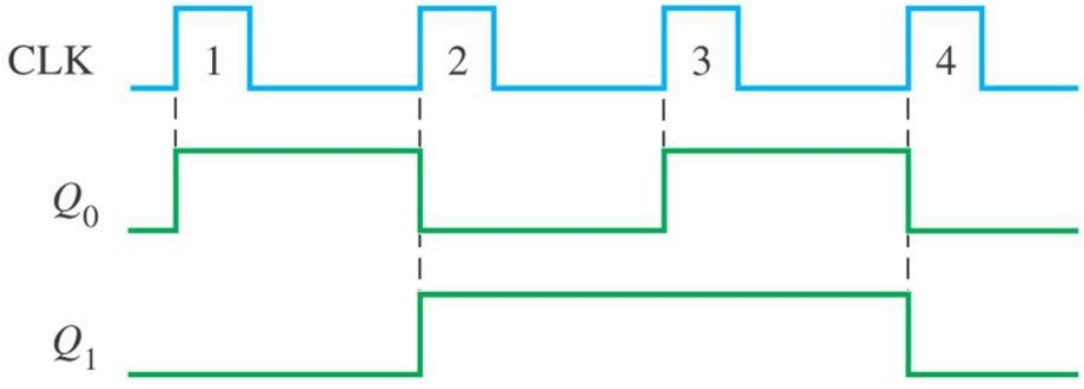

Fig4. The timing diagram for the counter

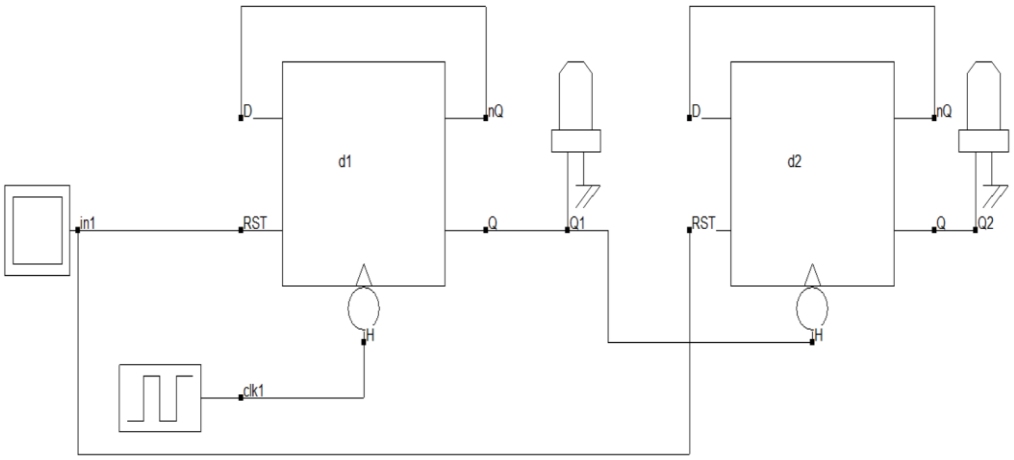

Fig5. The circuit diagram for the 2-bit down counter

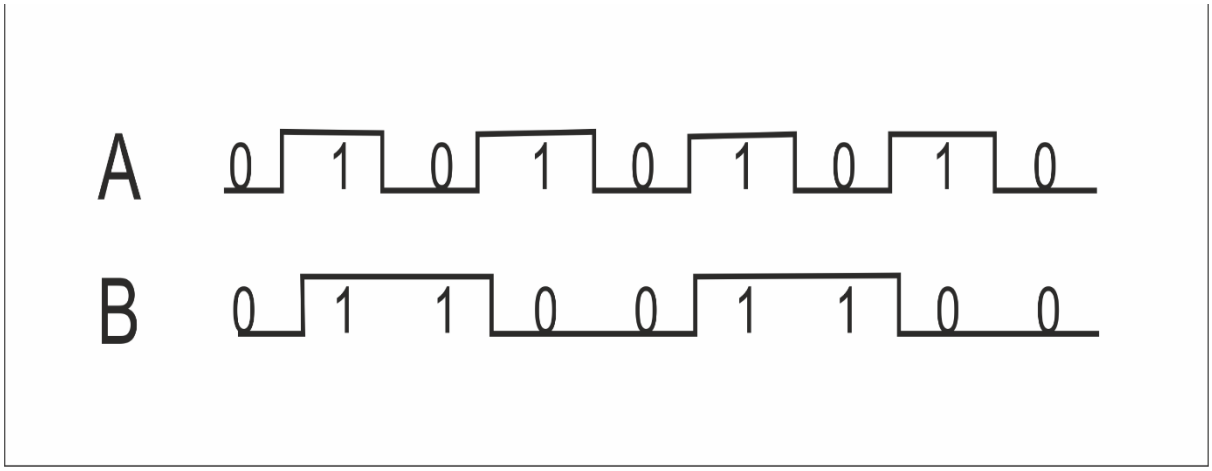

Fig6. The timing diagram

4. Optimized gate-level circuit diagram:

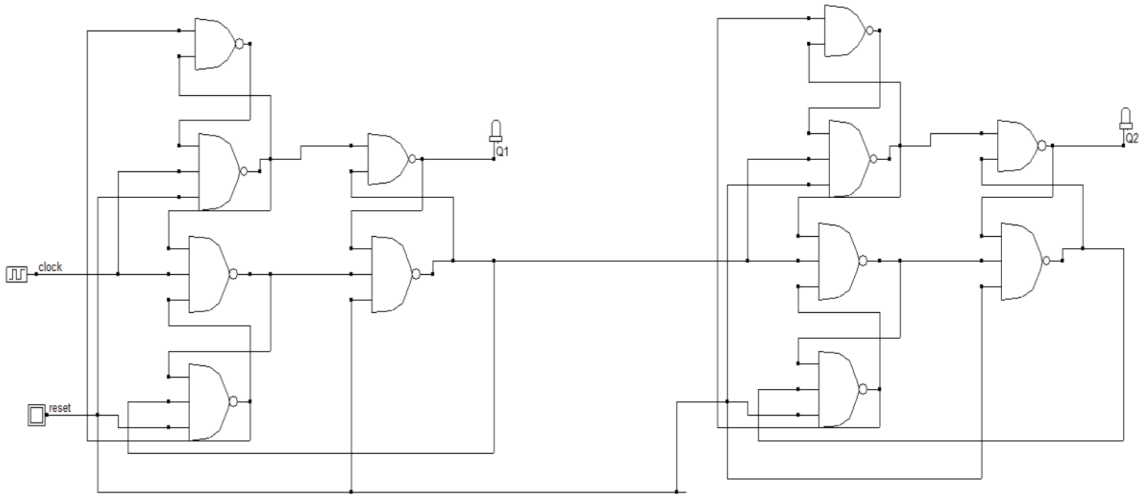

Fig7. Gate level circuit for up counter

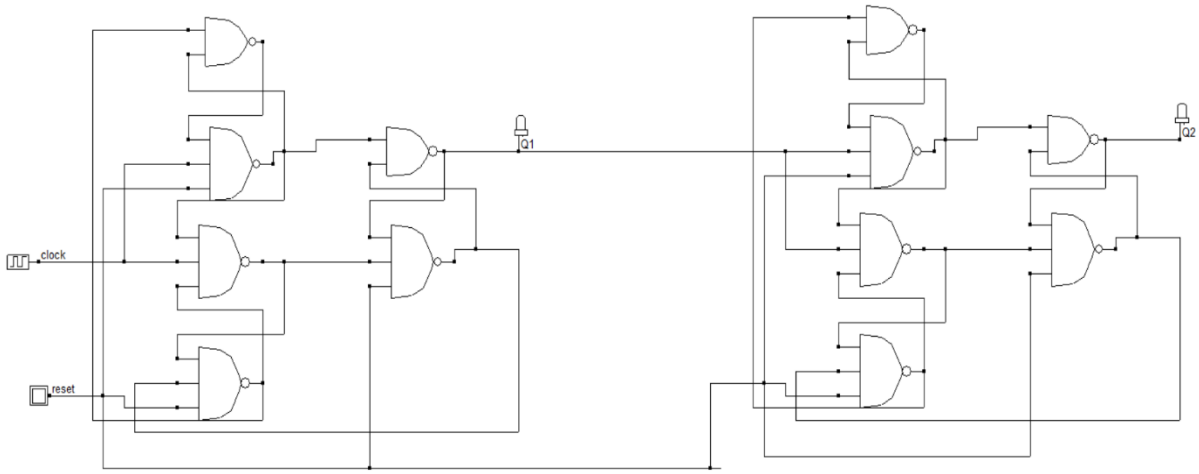

Fig8. Gate level circuit for down counter

5. Transistor level CMOS circuit for down counter:

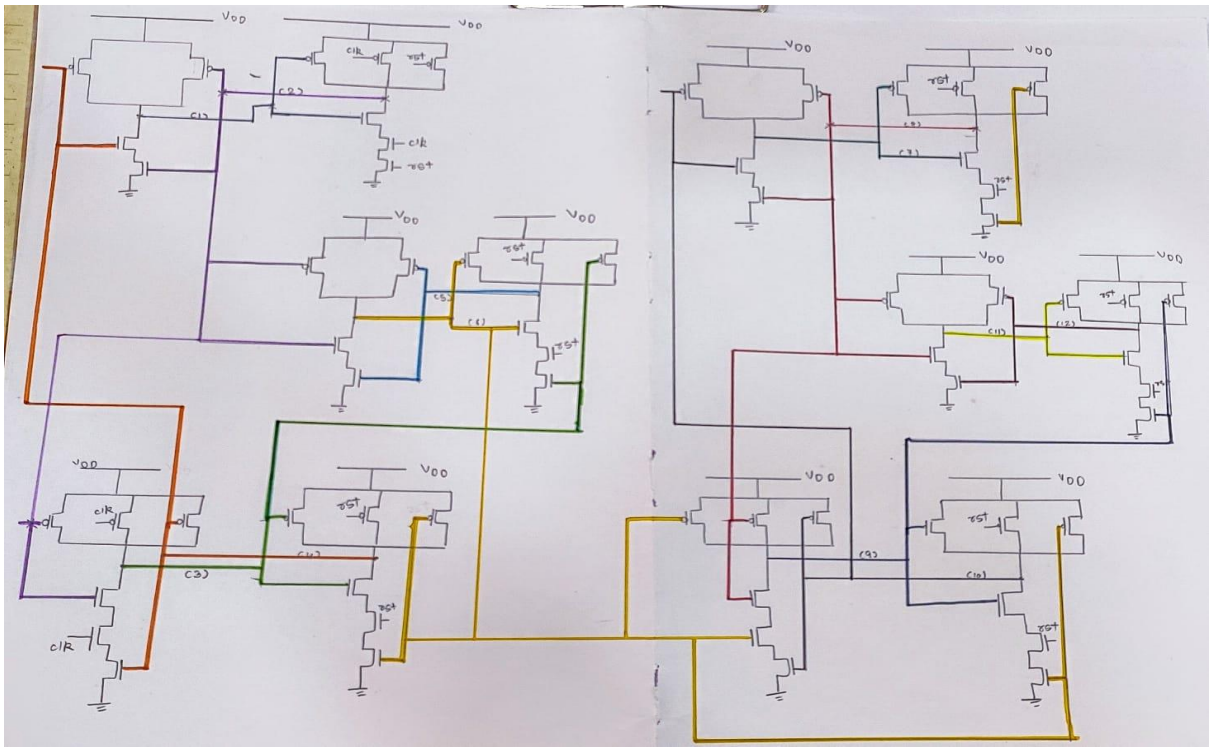

Fig9. The transistor-level CMOS circuit for the down counter

6. Stick a diagram for the above implementation level using the proper color code

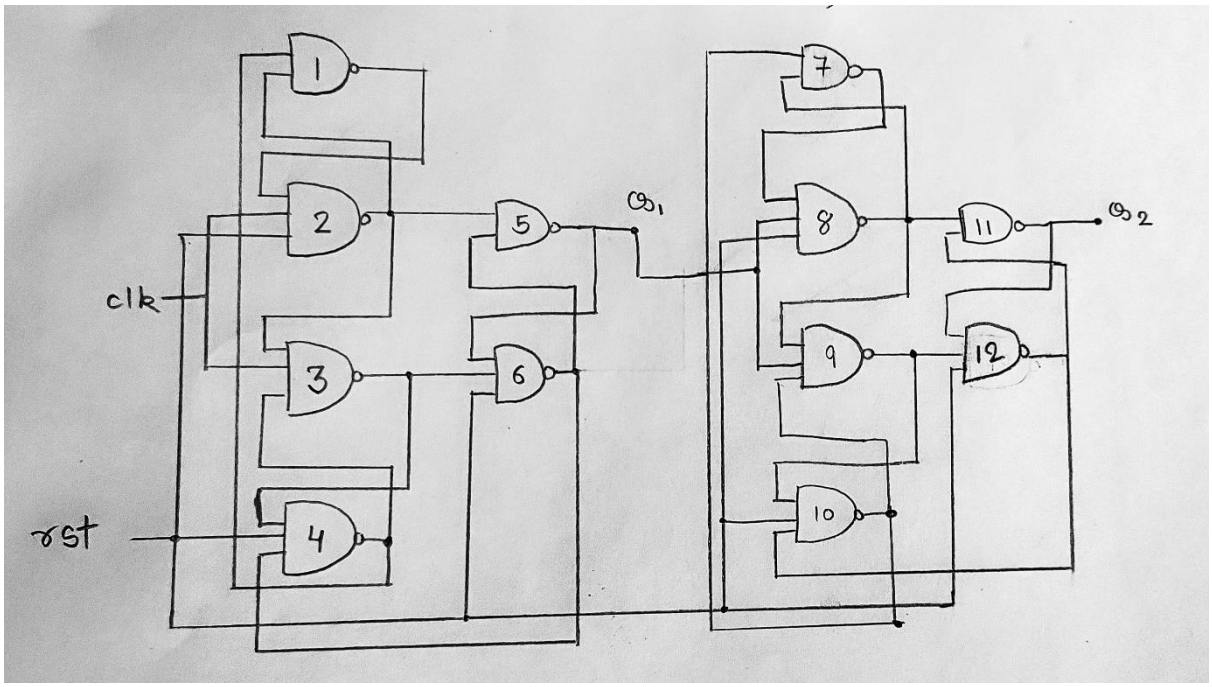

Fig10. Circuit to understand the stick diagram

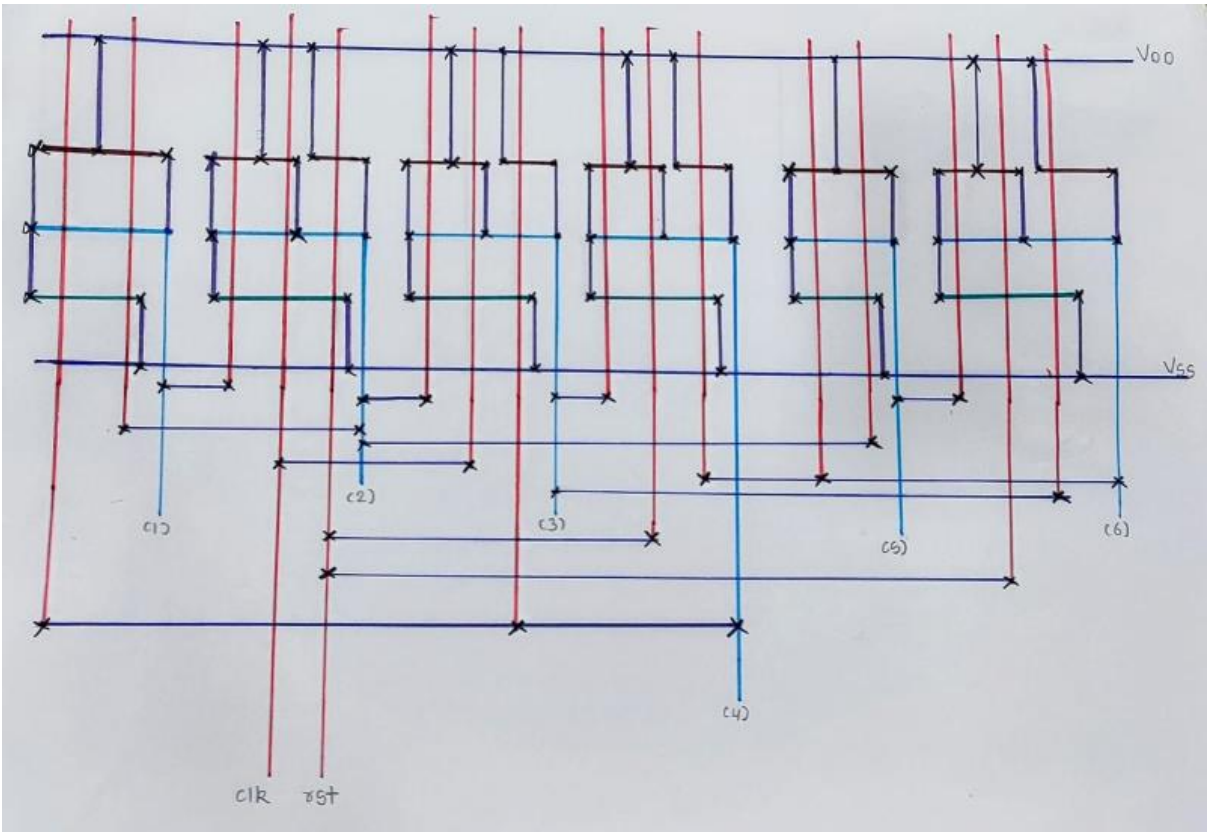

Fig11. Stick diagram

In this stick diagram, we haven't implemented the whole circuit as it was not feasible to understand hence we have implemented a single D flip flop. Now as we know, in aa asynchronous counter the inverted output of the first flip-flop will be the clock of the next flip-flop. Hence in the whole circuit, we made this stick diagram and the only change will be in the second we provide the Q1 bar in the clock.

7. Prepare the layout using micro-wind:

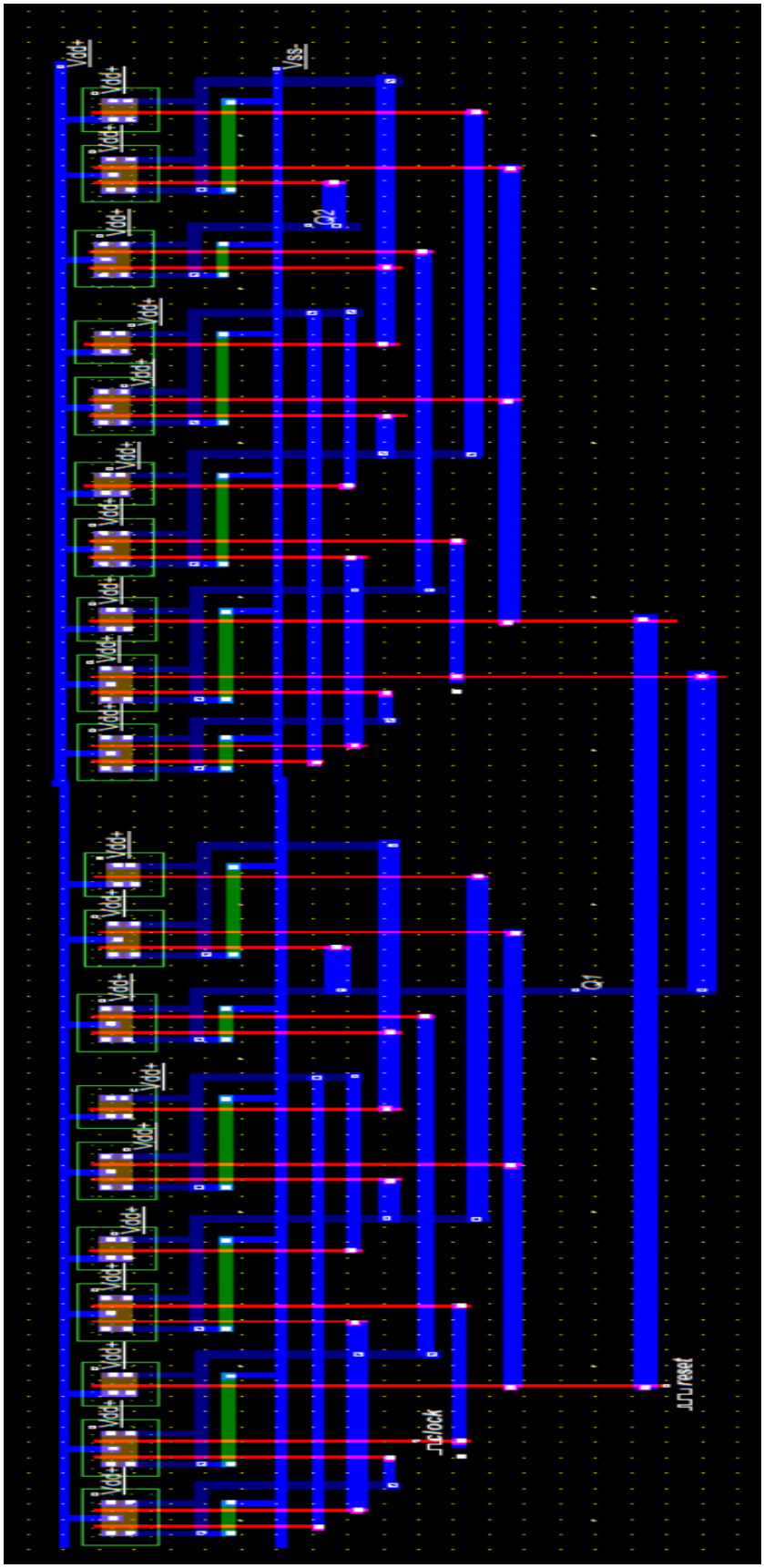

Fig12. The layout of the 2-bit down counter(manually)

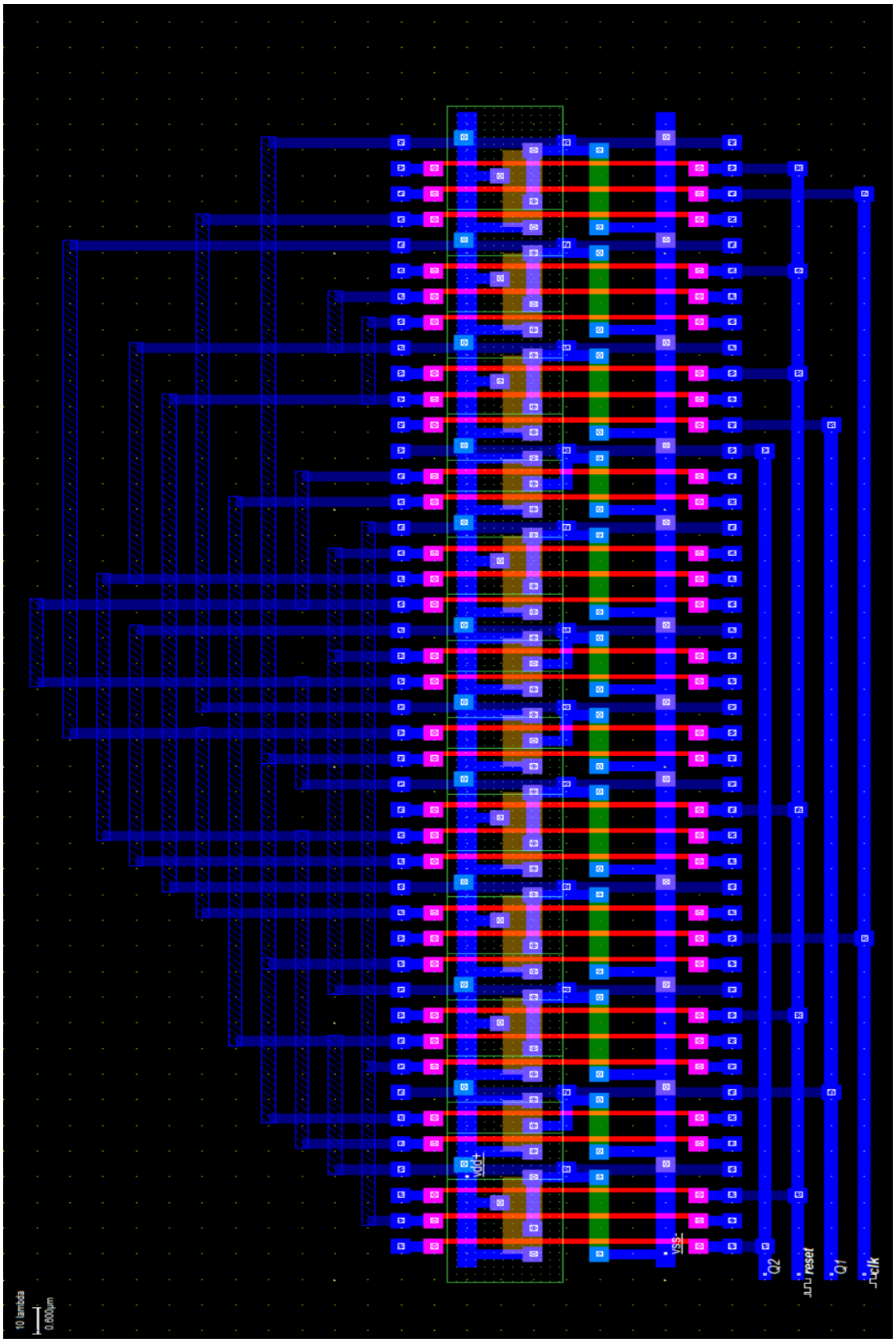

Fig13. The layout of the 2-bit up counter using the Verilog file

8. Simulation of the layout:

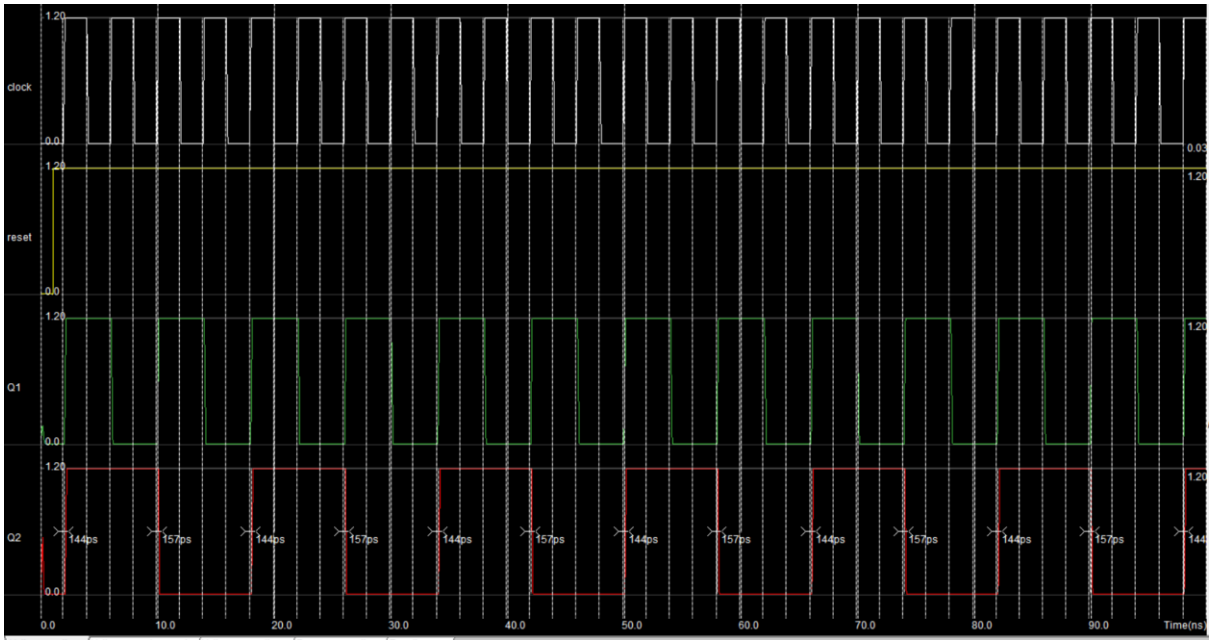

Fig14. The result of the down counter

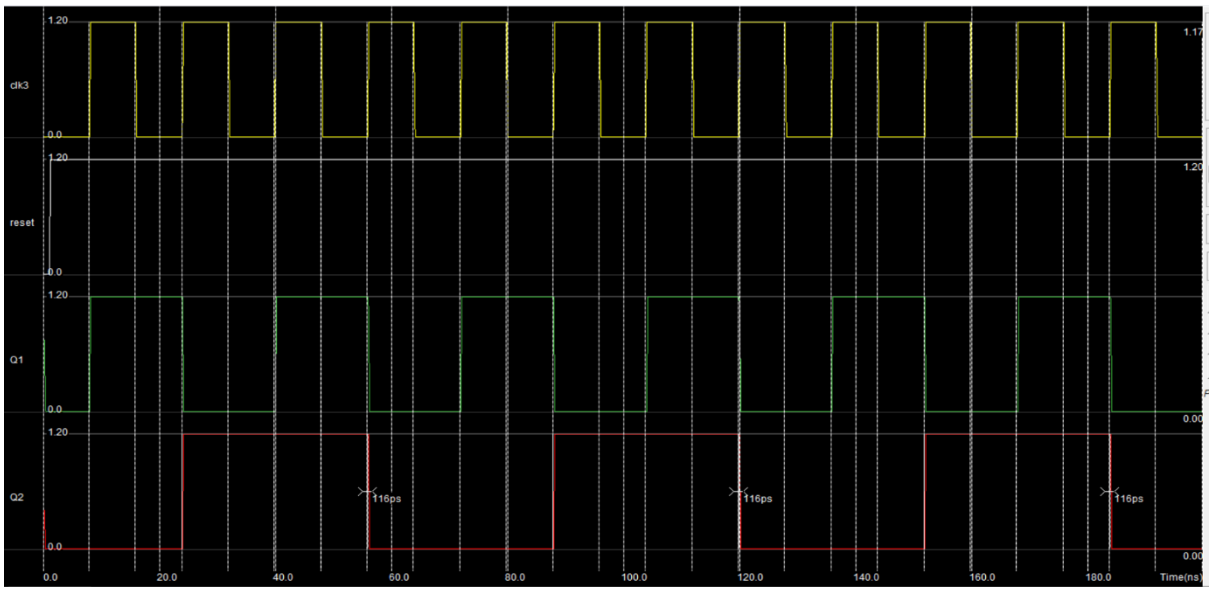

Fig15. The result of the upcounter

9. Measure the rise time, fall time, propagation delay and other parameter

This is the reading of the down counter
For the 0.12um technology
In this case, the Voh is equal to 1.2V

Flip flop output	Tplh	Tphl
Q1	-	-
Q2	144ps	157ps

Fig7. Propagation delay

Input	Rise time	Fall time
CLK	0.025ps	0.025ps
Reset	0.1ps	0.1ps

Fig8. Rise and Fall time

For the 0.18um technology
In this case, the Voh is equal to 2V

Flip flop output	Tplh	Tphl
Q1	-	-
Q2	356ps	393ps

Fig9. Propagation delay

For the 0.35um technology
In this case, the Voh is equal to 3.5V

Flip flop output	Tplh	Tphl
Q1	-	-
Q2	972ps	1116ps

Fig9. Propagation delay

This is the reading of up counter
For the 0.12um technology
In this case, the Voh is equal to 1.2V

Flip flop output	Tplh	Tphl
Q1	-	-
Q2	960ps	1220ps

Fig7. Propagation delay

Input	Rise time	Fall time
CLK	0.025ps	0.025ps
Reset	0.1ps	0.1ps

Fig8. Rise and Fall time

For the 0.18um technology
In this case, the Voh is equal to 2V

Flip flop output	Tplh	Tphl
Q1	-	-
Q2	271ps	298ps

Fig9. Propagation delay

For the 0.35um technology
In this case, the Voh is equal to 3.5V

Flip flop output	Tplh	Tphl
Q1	-	-
Q2	710ps	790ps

Fig9. Propagation delay

10. Conclusion

In conclusion, a 2-bit asynchronous counter was implemented using a NAND3 edge-triggered D flip-flop. The use of a NAND-based SR flip-flop as a D flip-flop proved impractical due to high delays, while a transmission gate (TG) D flip-flop was not suitable due to its level-triggered nature. Although a master-slave configuration could be an option, setup time violations impacted its reliability in counter applications, leading to the selection of the NAND3 edge-triggered flip-flop for improved performance.

The down counter was manually implemented, and the up counter was synthesized through a Verilog file. In a 120 nm technology process, the manually implemented counter exhibited lower delay than the Verilog-based counter. In contrast, at 180 nm and 350 nm technology nodes, the delay in the manually implemented counter increased relative to the compiled Verilog counter, indicating that finer technology processes favor manual implementations in terms of speed.

11. References

[1]. CMOS digital integrated circuit by Sung-Mo Kang
[2]. An Overview of a 4-Bit Synchronous Up Counter using DSCH v3.5
[3]. IEEE transactions on very large scale integration (vlsi) systems, vol. 20, no. 9, september 2012.

12.Verilog code

```
// DSCH 3.5
// 30-10-2024 11:12:31
// C:\Users\yash patel\Desktop\nirma univeristy\nirma sem
5\vlsi\project\counterusingnand3.sch

module counterusingnand3( clock,reset,Q2,Q1);
 input clock,reset;
 output Q2,Q1;
 wire w5,w7,w8,w9,w10,w11,w12,w13;
 wire w14,w15;
 nand #(2) nand3_1(w7,Q2,w5,reset);
 nand #(2) nand2_2(Q1,w8,w9);
 nand #(2) nand3_3(w10,w5,w7,reset);
 nand #(2) nand3_4(w12,w9,clock,w11);
 nand #(2) nand3_5(w14,w13,w8,reset);
 nand #(2) nand2_6(w15,w9,w11);
 nand #(2) nand2_7(w13,w14,w10);
 nand #(2) nand3_8(w5,w14,w8,w10);
 nand #(2) nand2_9(Q2,w7,w14);
 nand #(2) nand3_10(w8,Q1,w12,reset);
 nand #(2) nand3_11(w11,w12,w8,reset);
 nand #(2) nand3_12(w9,w15,clock,reset);
```

```
endmodule

// Simulation parameters in Verilog Format
always
#64000 clock=~clock;
#200 reset=~reset;

// Simulation parameters
// clock CLK 320 320
// reset CLK 1 1
```