ESE5700 Project 2 Report

Abhik Kumar abhikkum@seas.upenn.edu, **Kelly LAI** kwlaial@seas.upenn.edu
University of Pennsylvania https://github.com/abhikkum/TEST1

Abstract
This report details the design and optimization of a Configurable Logic Block (CLB) for the ESE 5700 Fall 2024 project, focusing on minimizing the figure of merit (FOM), defined as the product of area, average energy, and the inverse of maximum operating frequency. The CLB comprises a basic logic element (BLE) implemented with a 16:1 lookup table (LUT) based on SRAM cells, a multiplexer designed with pass-transistor logic, and control logic for data loading and operation. The design process began with constructing and verifying individual SRAM cells, ensuring robust read and write operations. Subsequently, these cells were integrated into the LUT, which was optimized for delay, energy, and area. The CLB was simulated under various test cases to validate logical correctness and performance, with results confirming functionality at maximum operating frequency. Design metrics were analyzed using Cadence simulations, and layouts were created to ensure area efficiency. Iterative optimization was employed to balance performance trade-offs, including alternative designs and variations to refine the final implementation. The report provides schematics, simulations, and supporting evidence for the optimization process. This project highlights a systematic approach to digital design, integrating analytical and practical methodologies to deliver a high-performance, energy-efficient, and compact CLB architecture.

Contents

1 Introduction

The Configurable Logic Block (CLB) is designed by integrating four key components: (1) a Serial-In Parallel-Out (SIPO) register, (2) an SRAM array, (3) a Lookup Table (LUT), and (4) a D flip-flop.

1.1 SIPO Register

The SIPO register facilitates the loading process by converting serial input data into a parallel format, enabling efficient storage. This reduces the number of input pins required and simplifies the data-loading process for the SRAM array.

1.2 SRAM Array

The SRAM array stores the 16-bit truth table required to define the logic functionality of the CLB. The SRAM cells ensure reliable read and write operations while maintaining minimal energy and area overhead. The stored data serves as the configuration for the LUT.

1.3 Lookup Table (LUT)

The LUT, implemented as a 16:1 multiplexer using pass-transistor logic, maps the 4-bit input address to the corresponding truth table values stored in the SRAM. The LUT operates efficiently, providing the desired output corresponding to the input address.

1.4 D Flip-Flop

The D flip-flop is integrated into the CLB to enable sequential logic operations, providing the functionality needed for state retention and clock synchronization.

- In this design, a falling-edge-triggered D flip-flop is used, allowing the CLB to process data efficiently and synchronize outputs with the system clock.
- A buffer is placed between the LUT and the D flip-flop. The buffer restores any threshold voltage (V_{th}) drop in the signal output from the LUT. This ensures that the input to the D flip-flop meets the necessary noise margin and voltage levels for reliable latching.
- Without the buffer, the signal degradation caused by the LUT's pass-transistor logic could lead to errors or unstable behavior in the D flip-flop.
- The flip-flop also enables the CLB to support sequential logic circuits, making it suitable for use in finite state machines or pipelines.

1.5 Working

Together, these four components work in tandem to create a highly efficient and compact CLB. The SIPO facilitates efficient data loading, the SRAM array stores configuration data, the LUT provides fast logic evaluation, and the D flip-flop ensures proper synchronization and state retention. The addition of a buffer between the LUT and the D flip-flop further enhances the design by improving signal integrity. This design is optimized for performance, energy efficiency, and compactness, making it ideal for use in modern programmable logic devices.

2 Components Schematics and Symbols

All the small components and symbols defined here will be used in later parts

2.1 NOR2

NOR2 schematics 1 and symbol 2 :

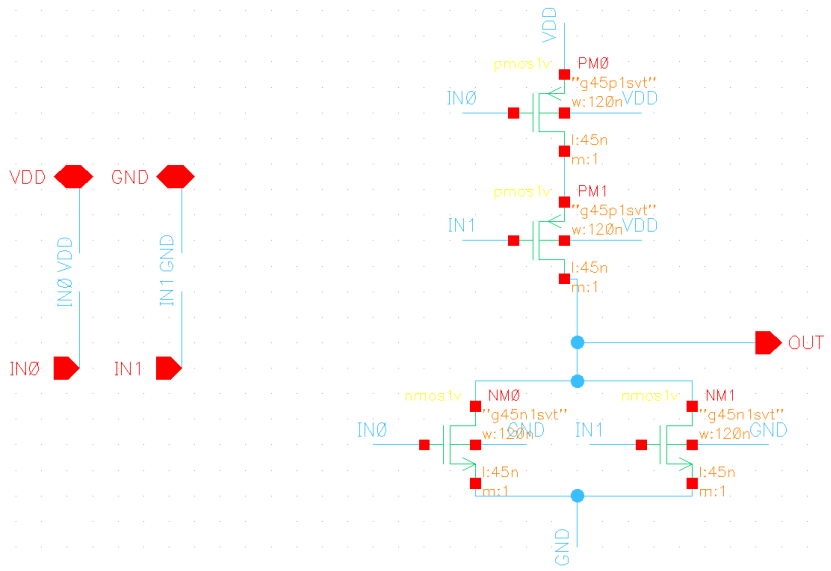


Figure 1: NOR2 Schematics

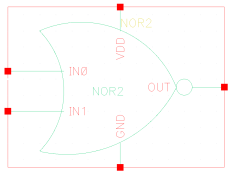


Figure 2: NOR2 Symbol

The truth table for a 2-input NOR gate is shown below:

Input A	Input B	Output (A NOR B)
0	0	1
0	1	0
1	0	0
1	1	0

2.2 NAND2

NAND2 schematics 3 and symbol 4:

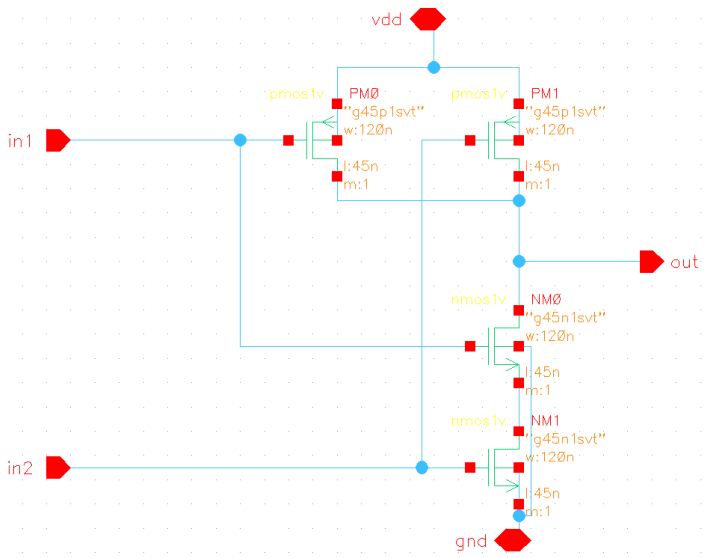


Figure 3: NAND2 Schematics

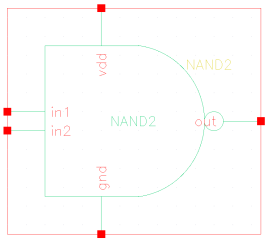


Figure 4: NAND2 Symbol

The truth table for a two-input NAND gate is shown below:

Input A	Input B	Output (A NAND B)
0	0	1
0	1	1
1	0	1
1	1	0

2.3 120n Width Minimum Inverter

Minimum inverter schematics 5 and symbol 6 :

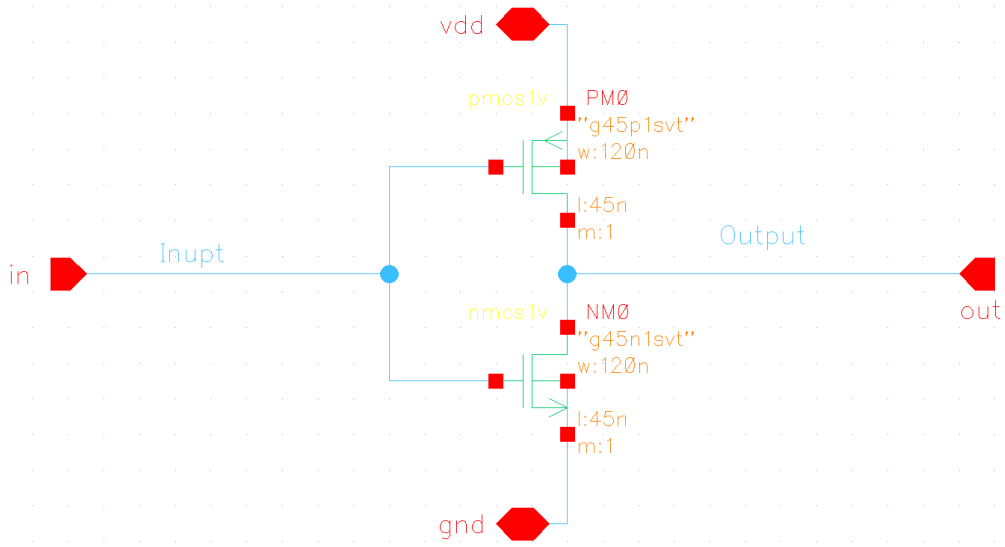


Figure 5: 120n Wdith Minimum Inverter Schematics

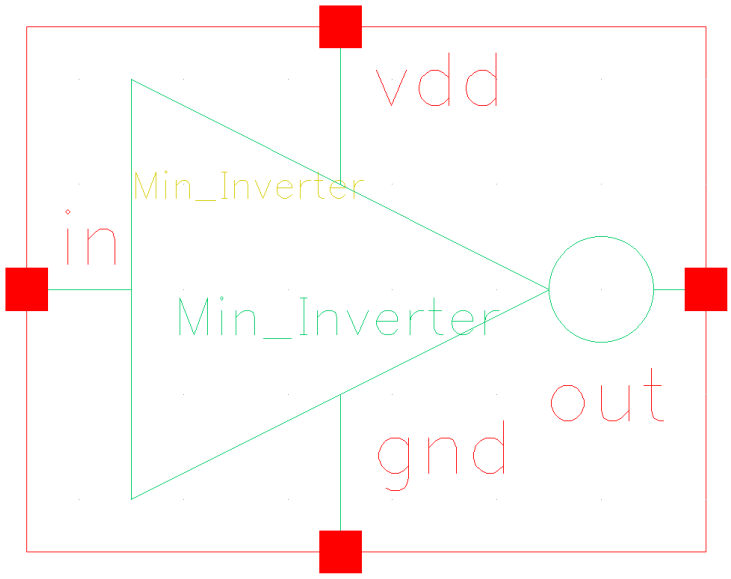


Figure 6: 120n Width Minimum Inverter Symbol

2.4 Buffer

Buffer schematics 7 and symbol 8 :

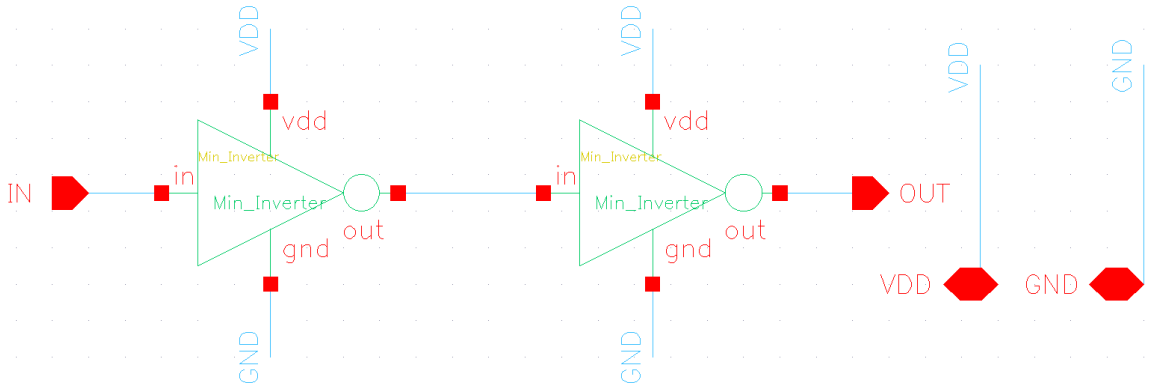


Figure 7: Buffer Schematics

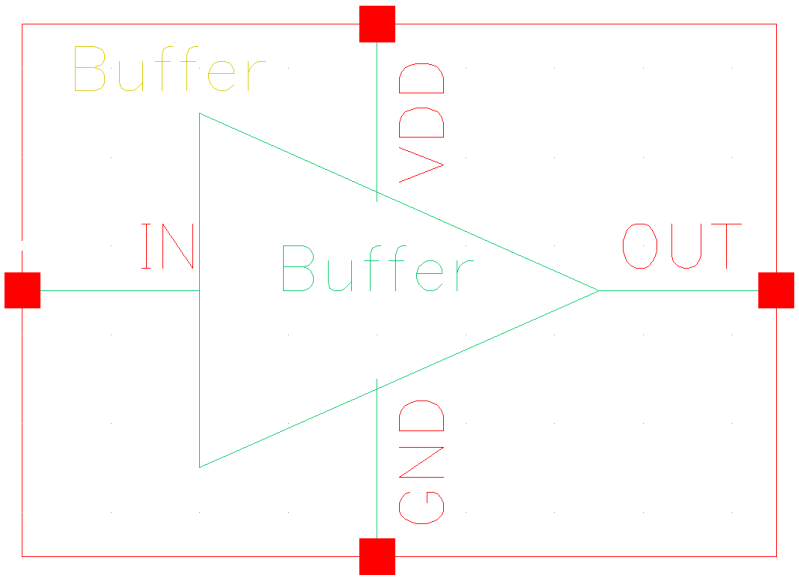


Figure 8: Buffer Symbol

A buffer using two inverters ensures signal integrity by restoring the original voltage levels of an input signal. The first inverter inverts the signal, and the second inverter restores it to its original state, effectively eliminating noise and degradation.

2.5 100Cg Inverter

100Cg Inverter schematics 9 and symbol 10:

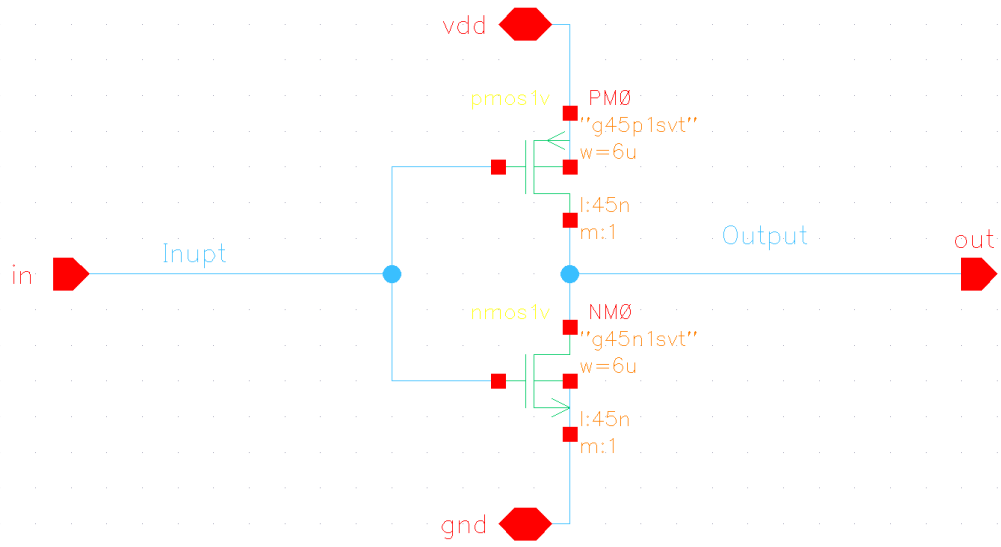


Figure 9: 100Cg Inverter Schematics

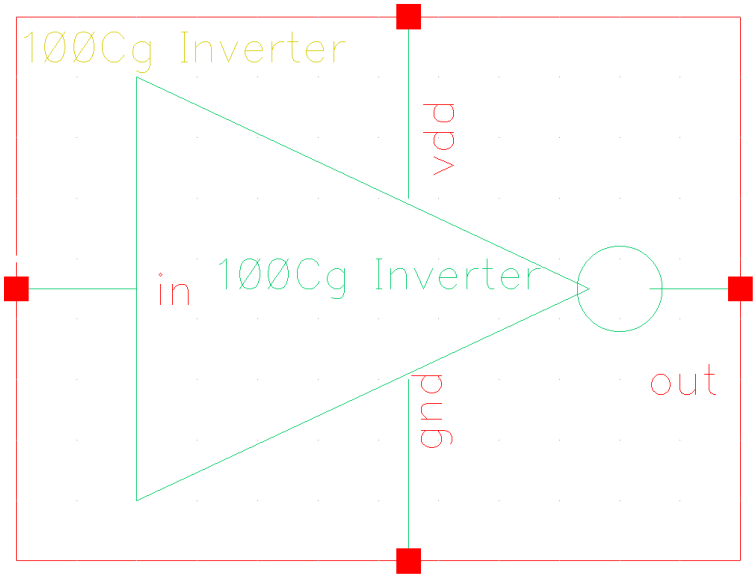


Figure 10: 100Cg Inverter Symbol

We have sized the NMOS to 50Cg and PMOS to 50Cg to achieve the 100Cg Inverter used for loading the output to simulate the real load.

2.6 W Inverter

W variable inverter schematics 11 and symbol 12:

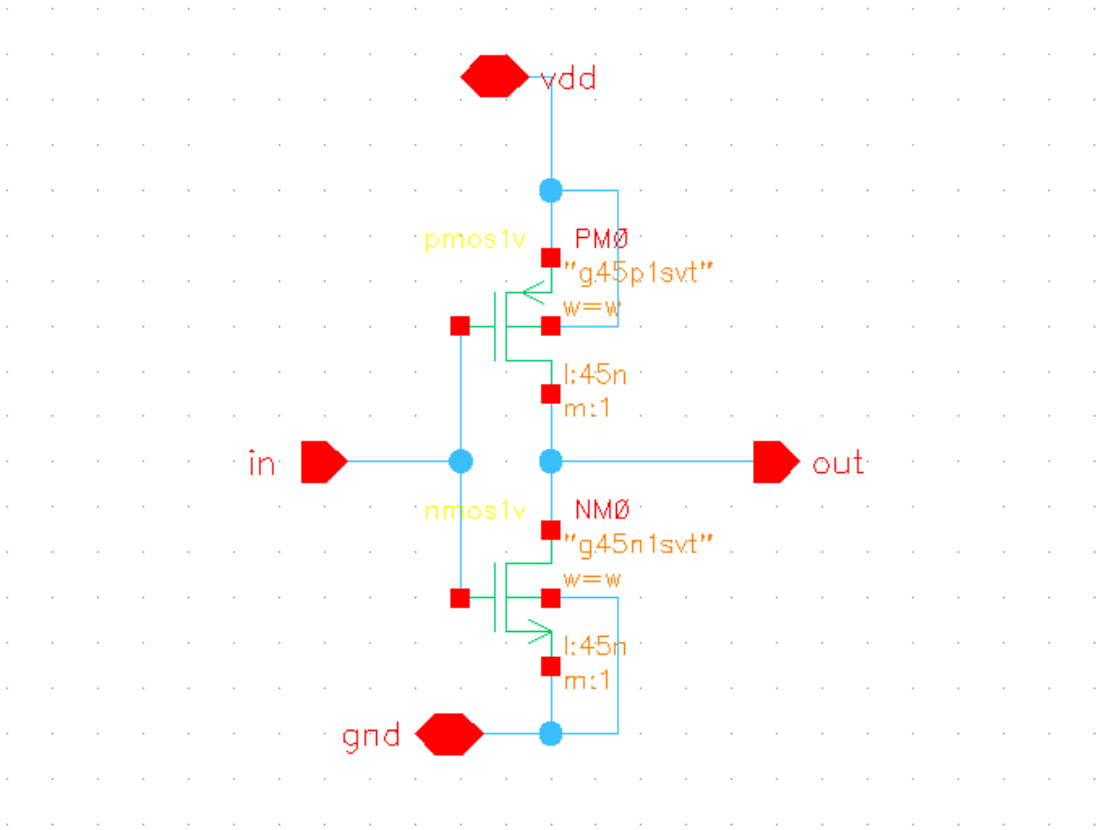


Figure 11: W Variable Inverter Schematics

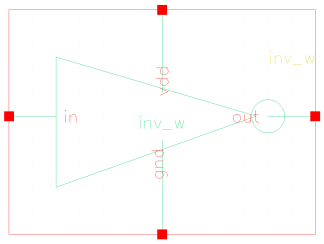


Figure 12: W Variable Inverter Symbol

2.7 Non-overlapping Clock

Non-overlapping clock schematics 13 and symbol 14:

A non-overlapping clock is used to generate two clock signals (ϕ_1 and ϕ_2) that are never high simultaneously, ensuring safe and glitch-free operation in sequential circuits. It prevents short-circuit conditions and improves timing reliability by introducing a gap (dead time) between the high phases of the two signals.

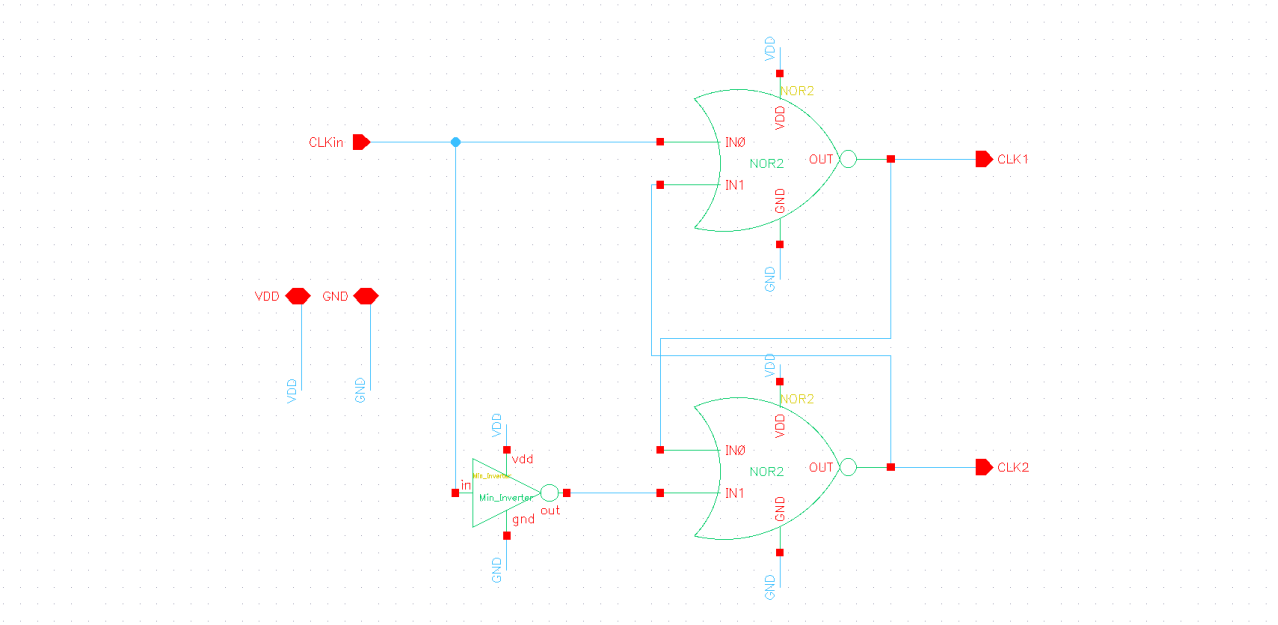


Figure 13: Non-overlapping Clock Schematics

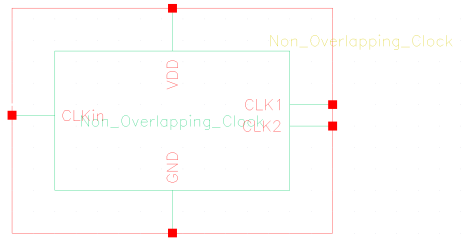


Figure 14: Non-overlapping Clock Symbol

Master Clock	ϕ_1	ϕ_2
0	0	0
1	1	0
0	0	0
1	0	1

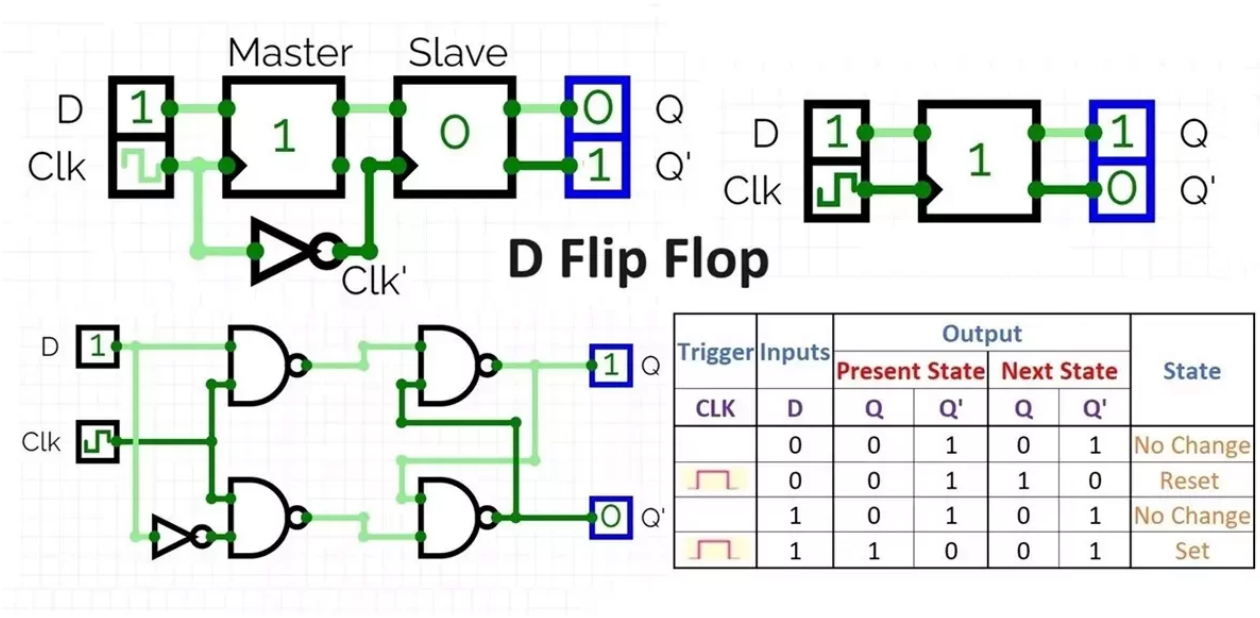
- The non-overlapping clock generator takes a **master clock** as input.
- It splits the clock cycle into two phases:
  - **Phase 1 (ϕ_1)**: Active during the high phase of the master clock but switches off before the falling edge.
  - **Phase 2 (ϕ_2)**: Active after the falling edge of the master clock, ensuring no overlap with ϕ_1.
- The gap between ϕ_1 and ϕ_2 ensures no simultaneous transitions in connected circuits, reducing timing conflicts and power consumption spikes.

3 Serial-In Parallel-Out (SIPO) Register

The SIPO16 register efficiently converts a 16-bit serial input into a parallel output, enabling data transfer to the SRAM.

3.1 Functional Description and Logic Operation

The **Serial-In-Parallel-Out (SIPO16)** register is a crucial component of the design, converting serial data input into parallel output. It is implemented using 16 D flip-flops, each storing a single bit of data. The SIPO16 register operates based on a clock signal, which controls the shifting of data through the register. The main idea behind this register is to serially load data into the first flip-flop , which then shifts the data bit by bit through the chain of flip-flops to produce the parallel output as shown in 15 below.



Trigger	Inputs	Output				State
		Present State		Next State		
CLK	D	Q	Q'	Q	Q'	
	0	0	1	0	1	No Change
⎍	0	0	1	1	0	Reset
	1	0	1	0	1	No Change
⎍	1	1	0	0	1	Set

Figure 15: D Flip Flop Truth Table, Circuit Diagram, Working and Applications [1]

3.2 D Flip-Flop Design

The SIPO16 module consists of 16 D flip-flops, adapted from the design used in Homework 8, as illustrated in 16 Unlike the original design from Homework 8, where the output Q was essential, it is not required in this implementation.

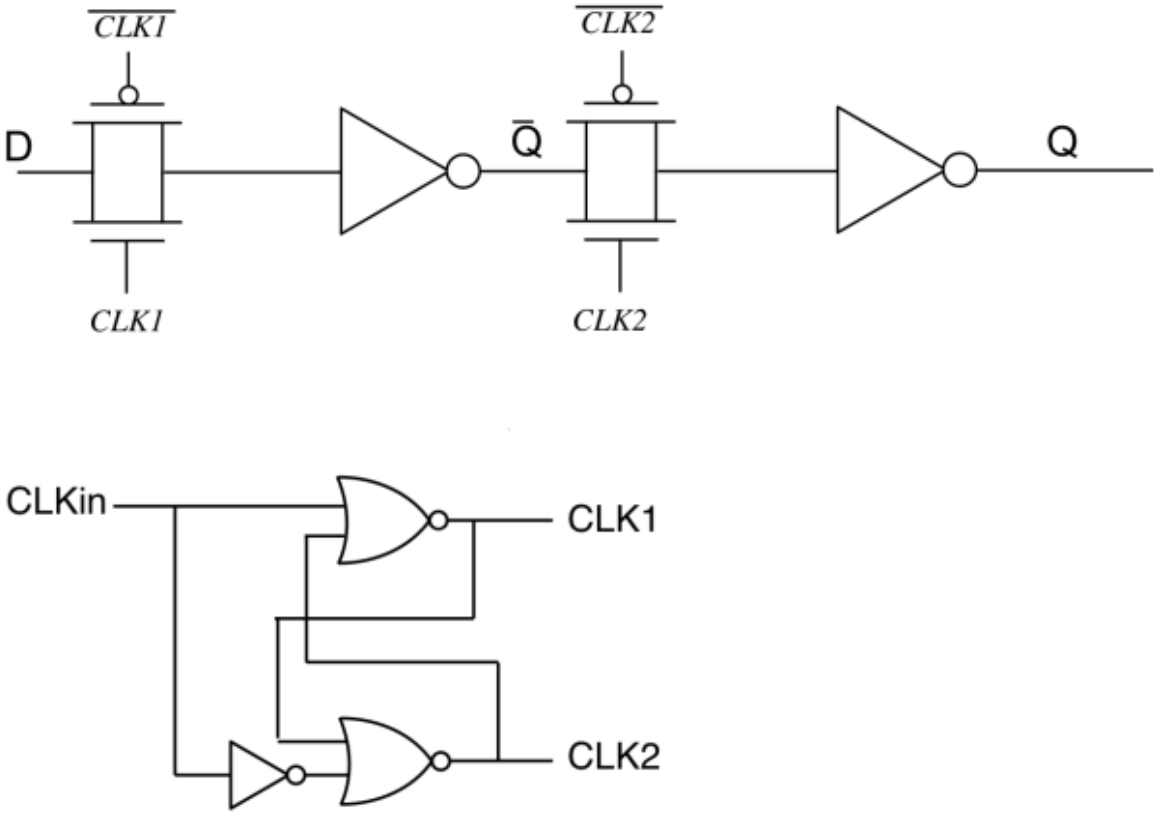


Figure 16: Schematics from HW8

The graph below shows the schematics 17 as well as the symbols 18 of D Flip-Flop from homework 8 and non-overlapping clock module: We have switched the triggering of the D-Flip-Flop to negative edge trigger for our setup.

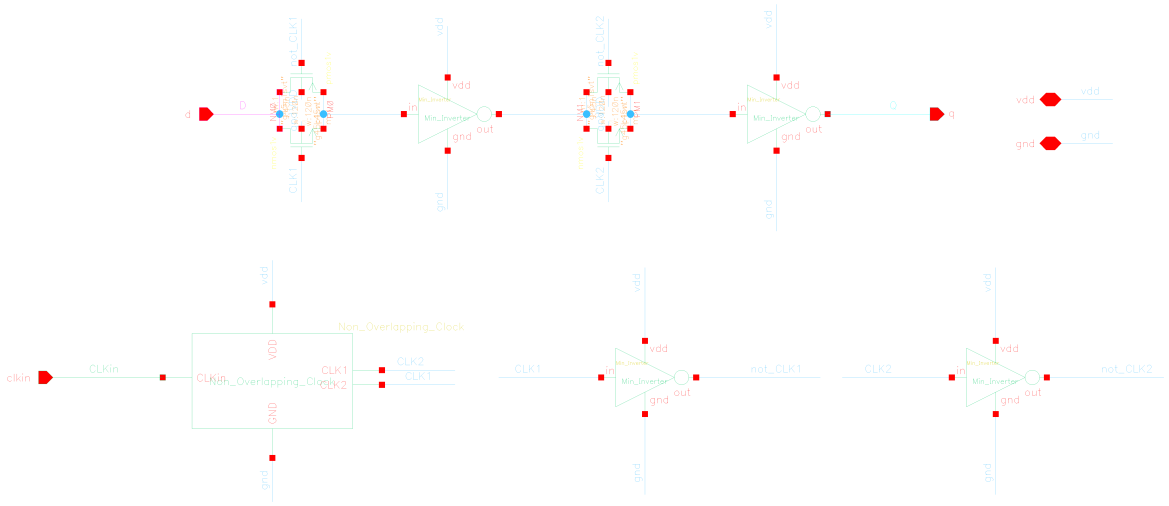


Figure 17: D Flip-Flop Schematics

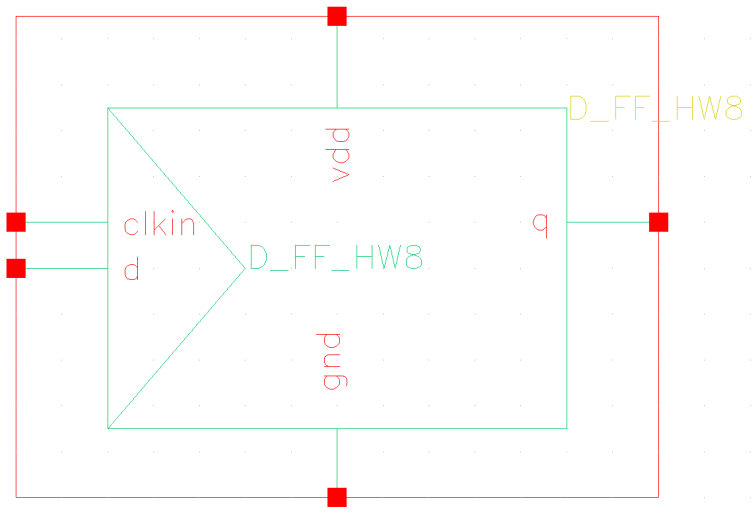


Figure 18: D Flip-Flop Symbol

3.3 D Flip-Flop Test

The test setup figure 19 is setting CLK with period 8ns while data with period 40ns with a 6ns delay figure20 and figure 21. The slower data rate (5 clock cycles per bit of data) allows the propagation of each bit through the register to be more clearly visible. This makes it easier to track how each bit shifts from one flip-flop to the next in sequence, validating the design's logical correctness step by step 23.

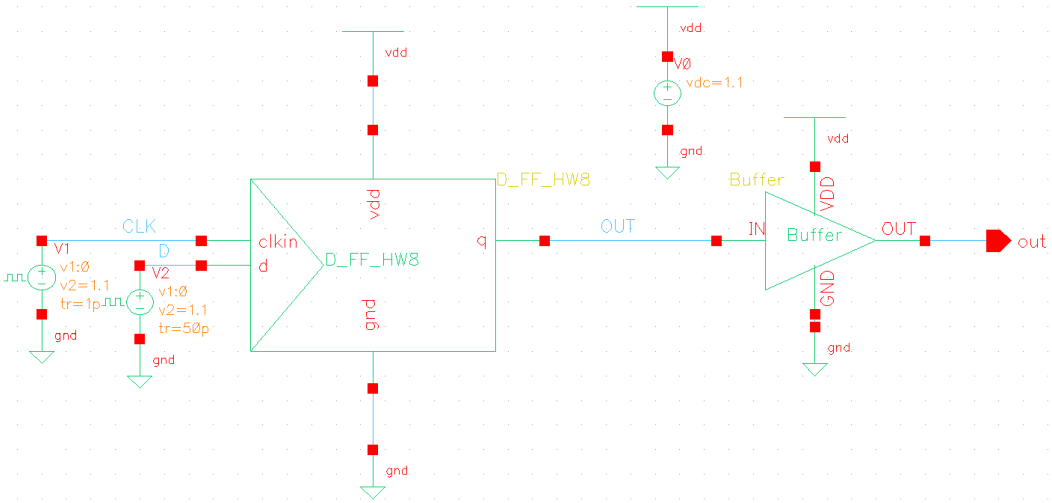


Figure 19: D Flip-Flop Test

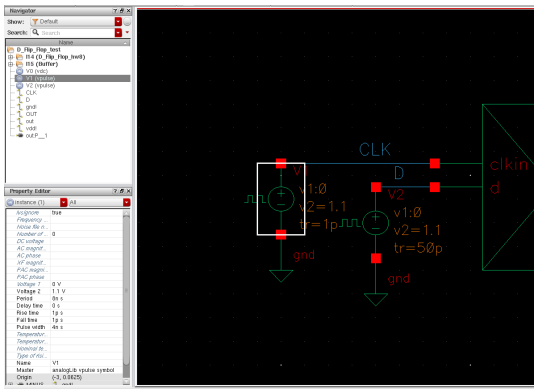


Figure 20: D Flip Flop Clock Setup

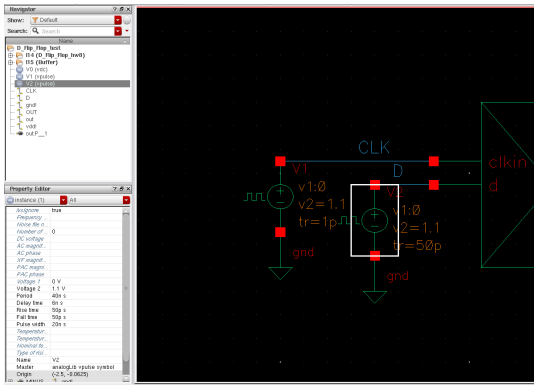


Figure 21: D Flip Flop Data Setup

The test result is performing what it should be like in the truth table: Our D Flip flop is a

falling edge flip flop that means the output data only switches at the falling edge of the clock cycle, if there is any change in the data.

Trigger	Inputs	Output				State
		Present State		Next State		
CLK	D	Q	Q'	Q	Q'	
	0	0	1	0	1	No Change
⎍	0	0	1	1	0	Reset
	1	0	1	0	1	No Change
⎍	1	1	0	0	1	Set

Figure 22: D Flip Flop Truth Table [1]

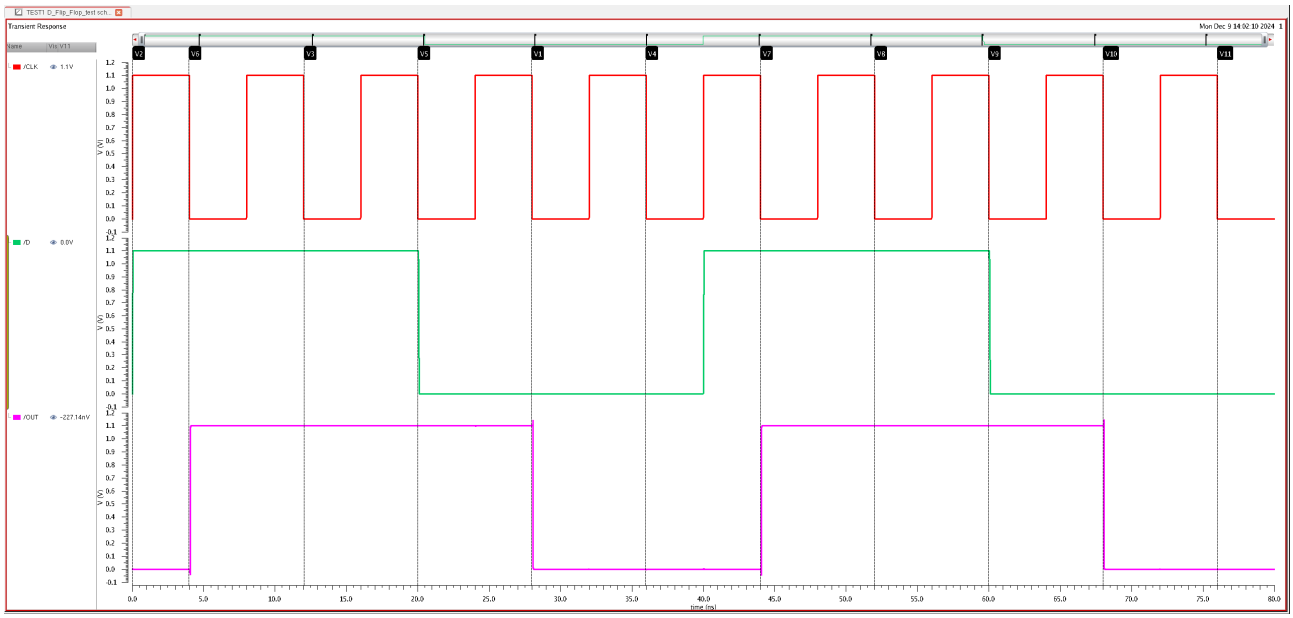


Figure 23: D Flip Flop Test Result

3.4 SIPO16 Design

Using the schematics from the D Flip-Flop design, we employ 16 D Flip-Flops for the SIPO16 module. The design is based on the one from Homework 8, with modifications as necessary. Each D Flip-Flop is connected in series to shift in data, with the output of each flip-flop feeding into the next one in the sequence shown in figure 24.

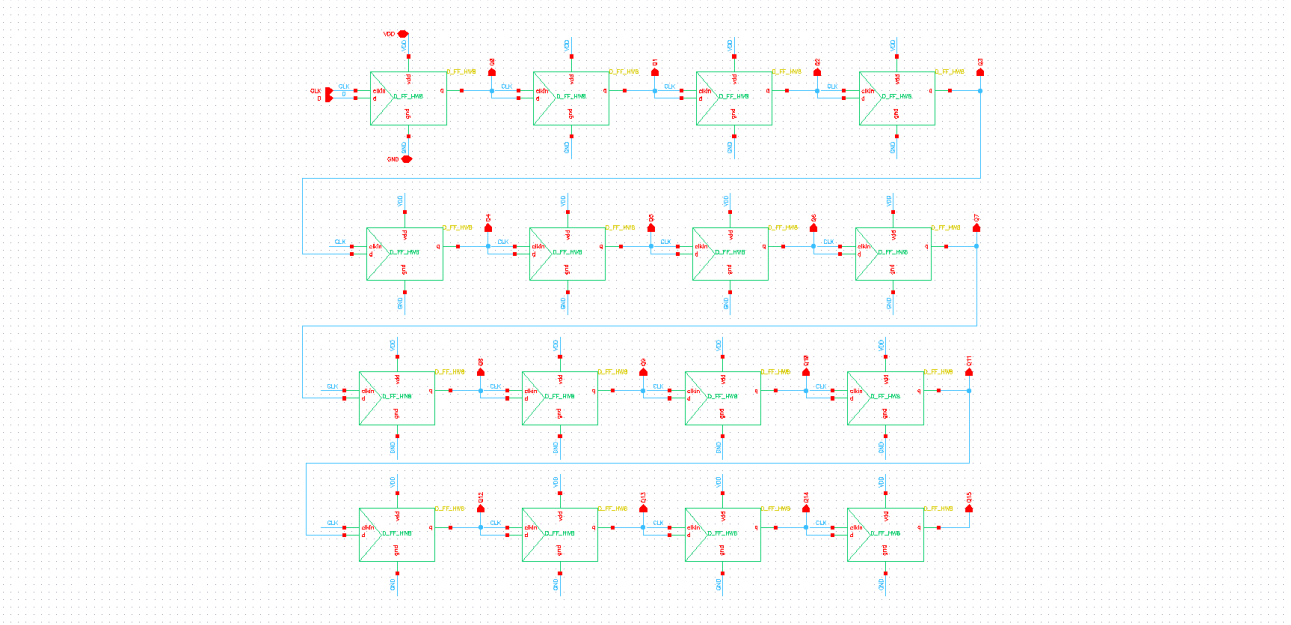


Figure 24: SIPO16 Schematics

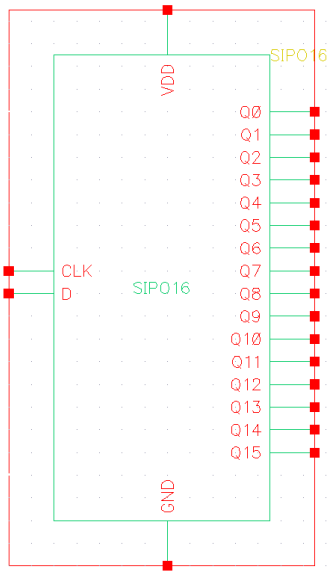


Figure 25: SIPO16 Symbol

3.5 SIPO16 Test

The setup for the CLK in figure 27 and Data in figure 28 input includes a period of 8ns with pattern:

`1110001010101001`

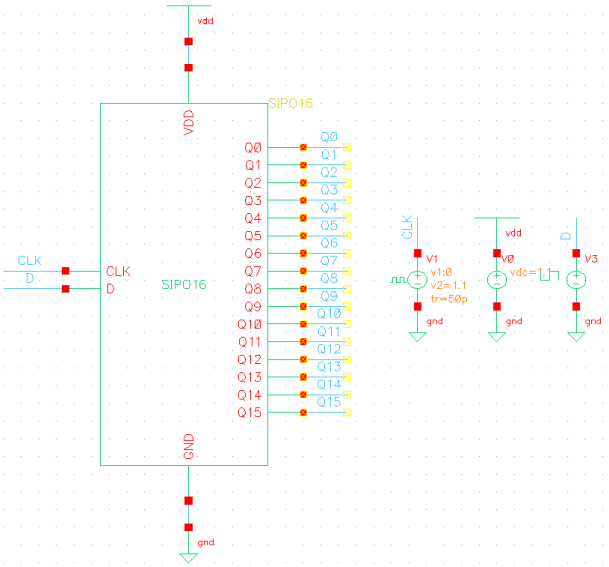


Figure 26: SIPO16 Test Schematics

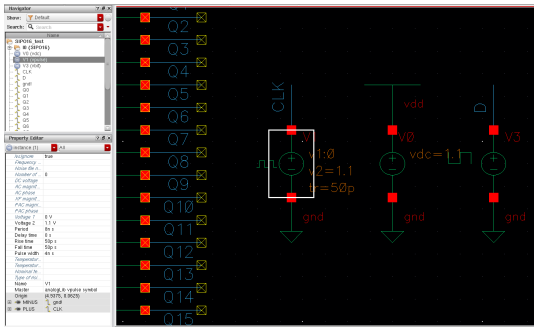


Figure 27: Clock Setup

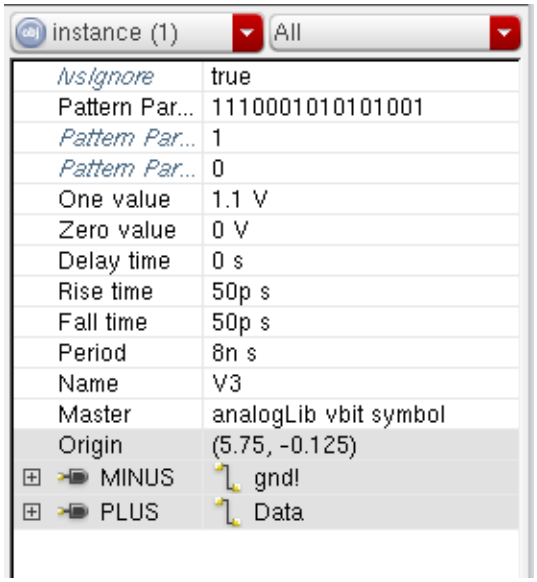


Figure 28: Data Setup

The test result: The waveform correctly demonstrates the behavior of the **SIPO16 register**, where each output Q[n] exhibits a **1-clock-cycle delay** relative to the input Data and the preceding outputs. As data enters the first D flip-flop at each rising clock edge, it shifts through the subsequent flip-flops, with each stage introducing a one-clock-cycle delay. This ensures that the data is converted from serial input to parallel output, with the final output Q[15] representing the data that entered the register 16 clock cycles earlier. The waveform confirms the correct operation of the register, validating the serial-to-parallel conversion and synchronous behavior with the clock signal.

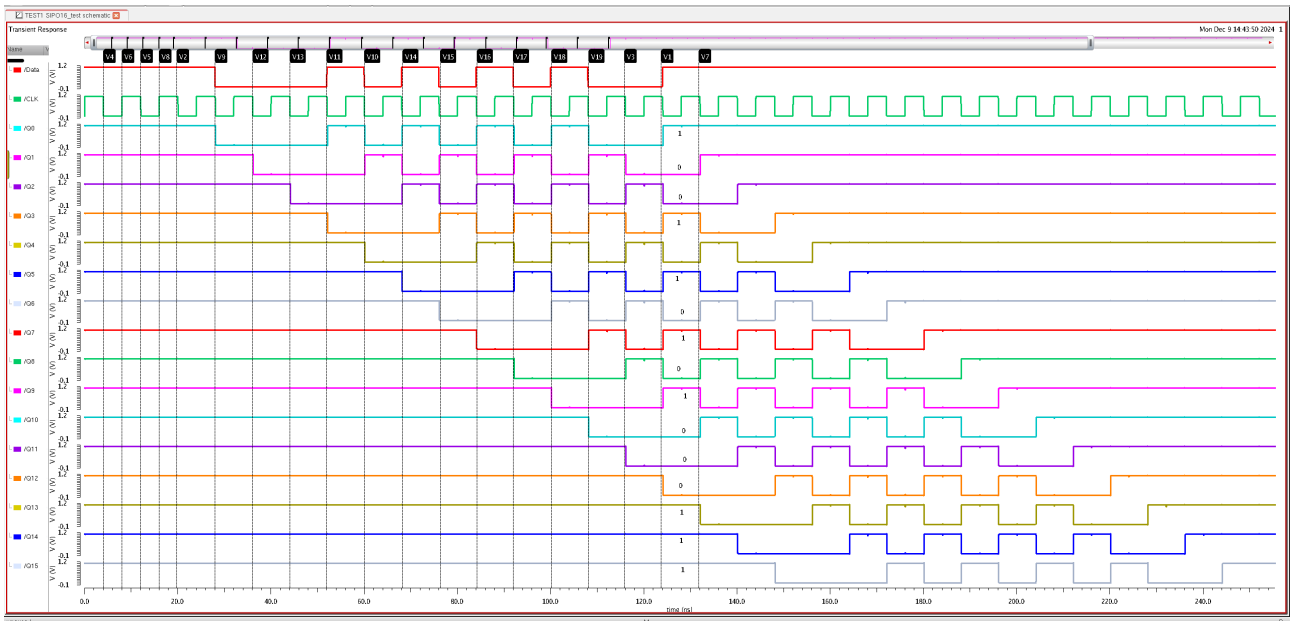


Figure 29: SIPO16 Test Result

4 Static Random Access Memory (SRAM)

SRAM is a fast, volatile memory that uses flip-flops to store data and does not require refreshing, making it ideal for cache and high-speed applications.

4.1 Functional Description and Logic Operation

Static Random Access Memory (SRAM) stores data using flip-flops composed of transistors, with the 6T design being the most common due to its balance of speed, stability, and compactness. In a 6T SRAM cell, two cross-coupled inverters store a single bit, with access controlled by word lines (WL) and write enable (Write_EN) signals. During write operations, Write_EN is high, and the bitline is driven with the desired data, while during read operations, Write_EN is low, allowing the data to be accessed through the bitline. The bitlines are precharged to a known state, typically high, by a precharge module to ensure proper data access. The SRAM's functionality is verified by reading and writing alternating data values, confirming that data integrity is maintained during read and write cycles.

4.2 SRAM Design

We are using the 6T SRAM design introduced in the references. Compared to other SRAM designs, the **6T SRAM** in figure (31) and (30) is often preferred for speed due to its balance between stability, compactness, and performance. When compared to **4T SRAM** designs, the 6T design provides better data stability and reliability, as the extra transistors (the second inverter) help maintain the stored data more reliably, especially under varying conditions like process variation or noise. While **8T SRAM** designs can offer better noise immunity and more robust read/write operations, they generally consume more area and power, making the 6T SRAM a more efficient choice for speed-sensitive applications. Additionally, the 6T SRAM provides faster access times than other complex SRAM designs due to its simpler structure, which enables quicker read and write cycles.

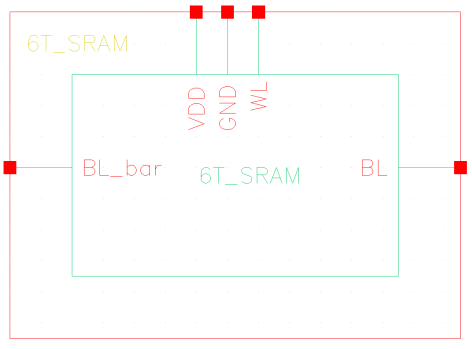


Figure 30: SRAM Symbol

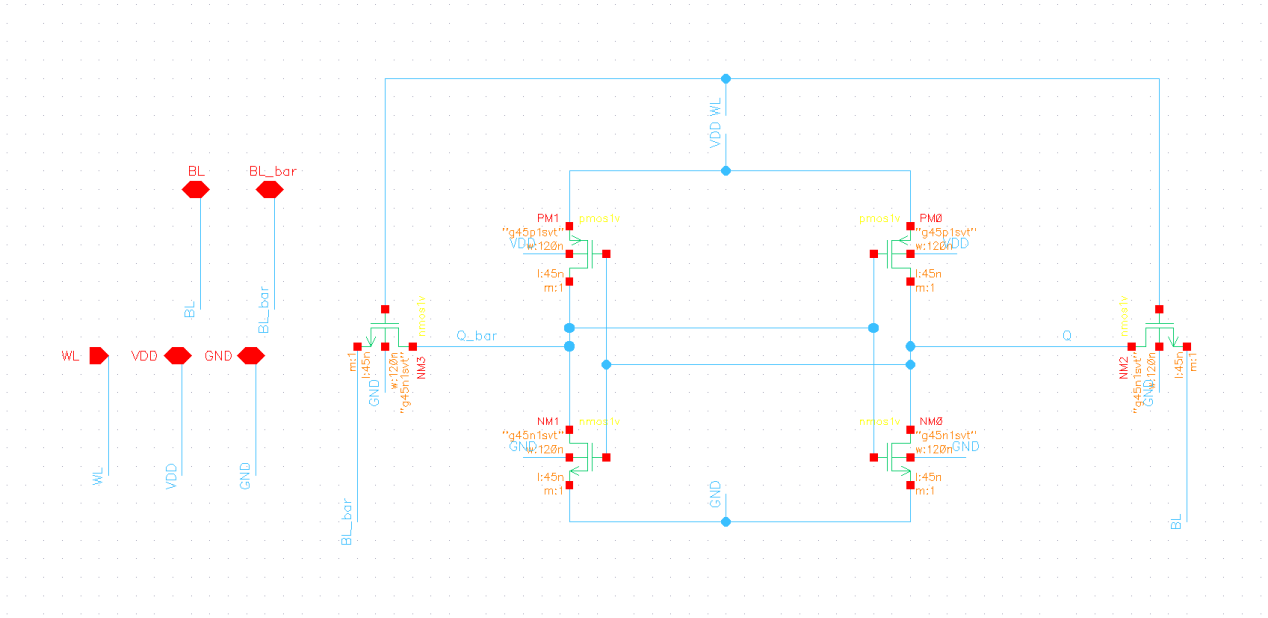


Figure 31: SRAM Schematics

4.3 SRAM Test

We created a buffer connecting corresponding D data input (ON BL and BL$_bar$), $CLKclock(WL)$, $andWrit$

Period: 8ns,

Rise time: 50ps,

Drop time: 50ps.

To test the correct operation of the SRAM, we first write a high value (1), then read it twice. Next, we write a low value (0) and read it twice as well. We repeat this process by writing a high value and reading once, writing a high value again and reading once more, then writing a low value and reading again, followed by writing low and reading once more. The purpose of this sequence is to verify that the read operation does not destroy the stored data. Writing the opposite bit (1 to 0 or 0 to 1) ensures that we can successfully write opposite values, while reading and writing the same bit (e.g., writing 1 and reading 1) checks the integrity of the data.

The sequence of operations is as follows: Write(1), Read(1), Read(1), Write(0), Read(0), Read(0), Write(1), Read(1), Write(1), Read(1), Write(0), Read(0), Write(0), Read(0).

Inserted data pattern in figure 33:

`10000010100000`

$High : 1 \quad Low : 0$

to VBIT. Clock VDC with pulse width 4ns. Write_EN input:

`10010010101010`

$Write : 1 \quad Read : 0$

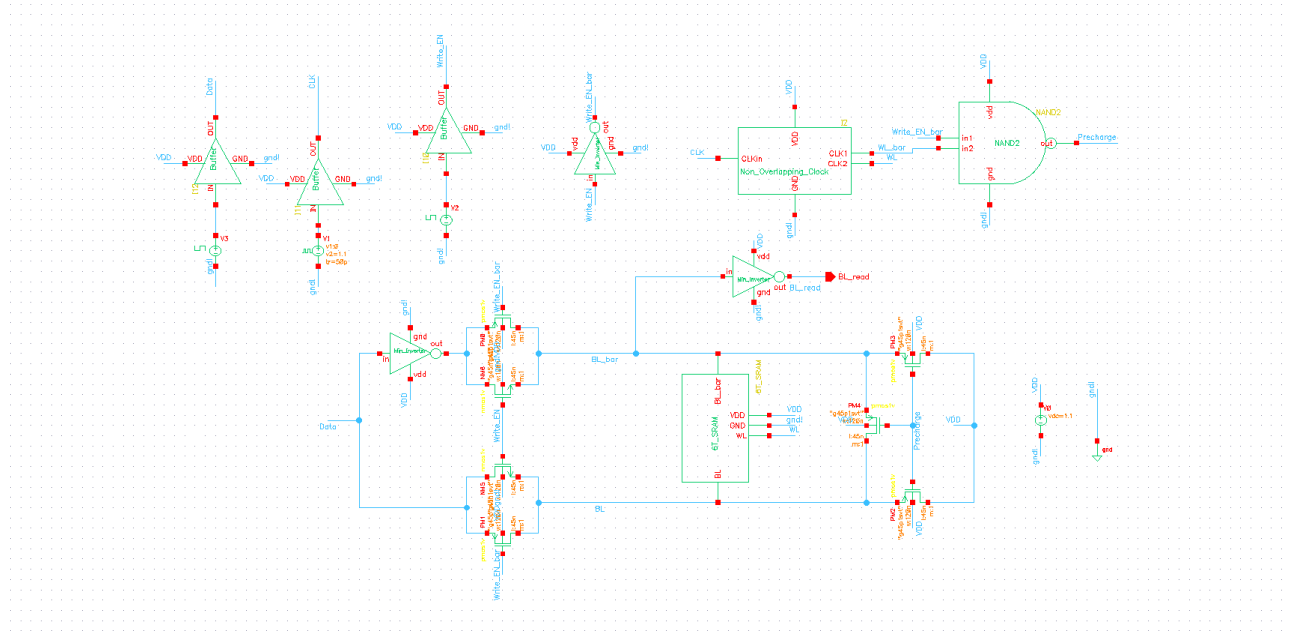


Figure 32: SRAM Test Schematics

Test Setup:

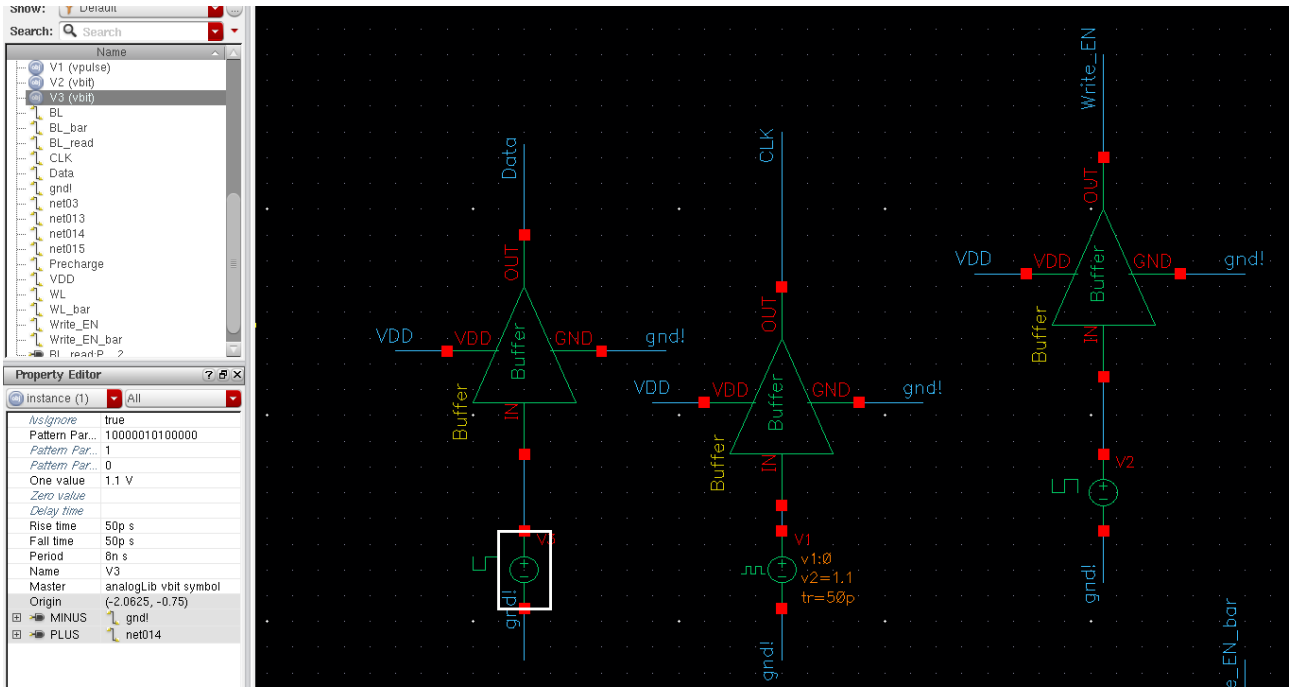


Figure 33: Data Setup

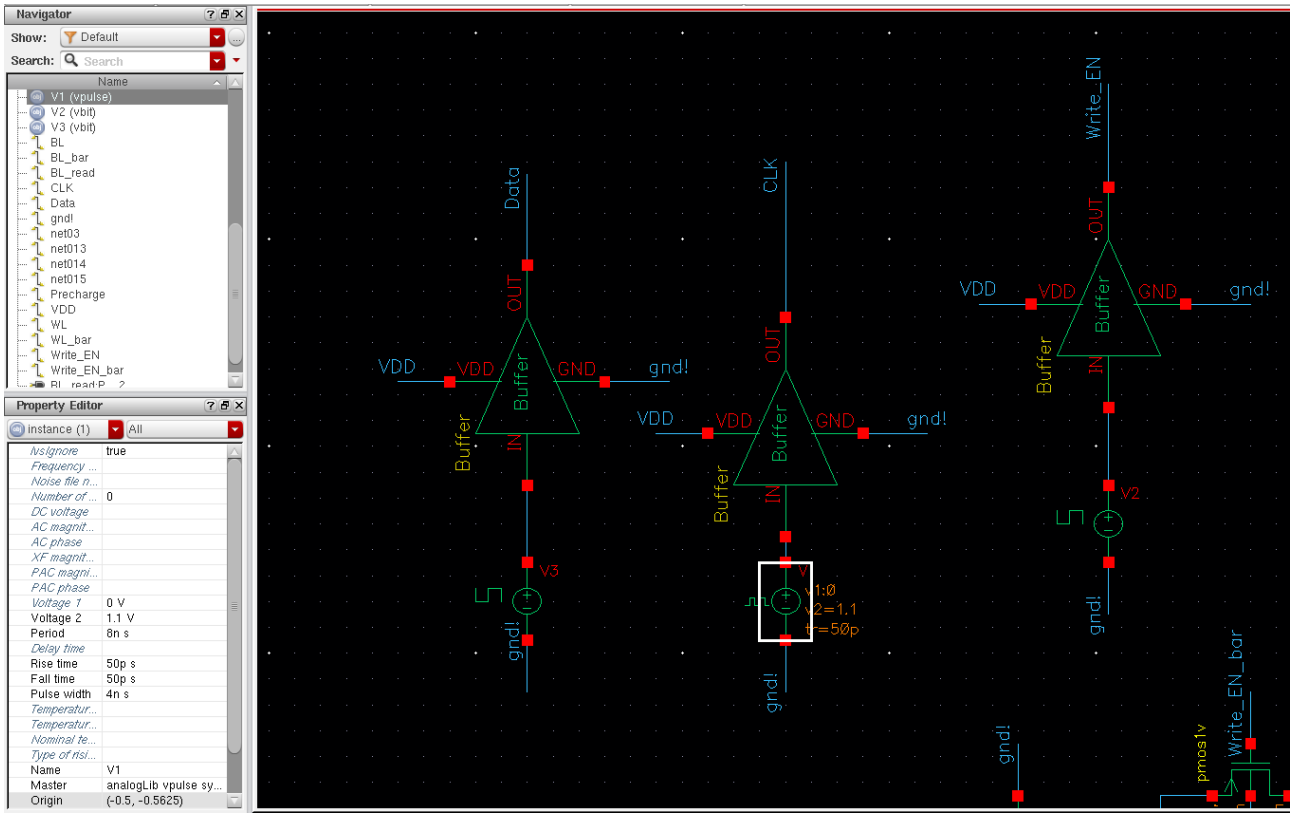


Figure 34: Clock Setup

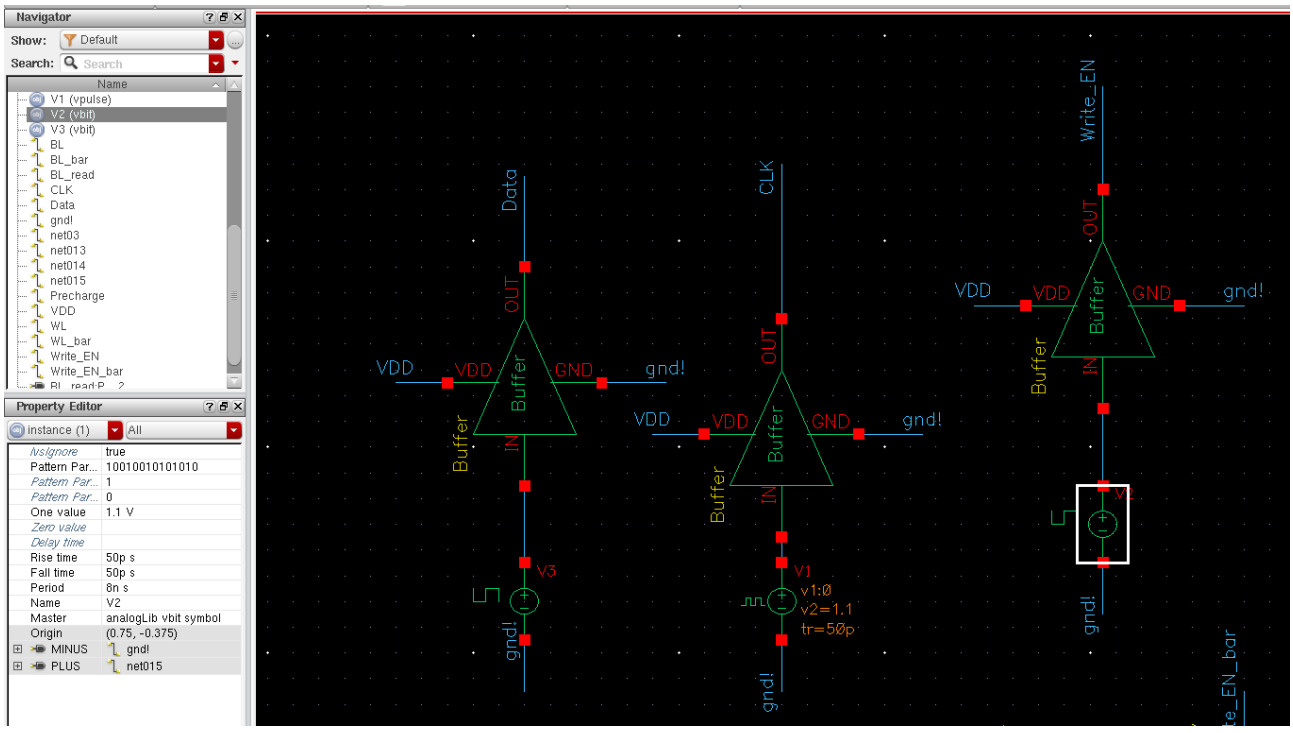


Figure 35: Write Enable Setup

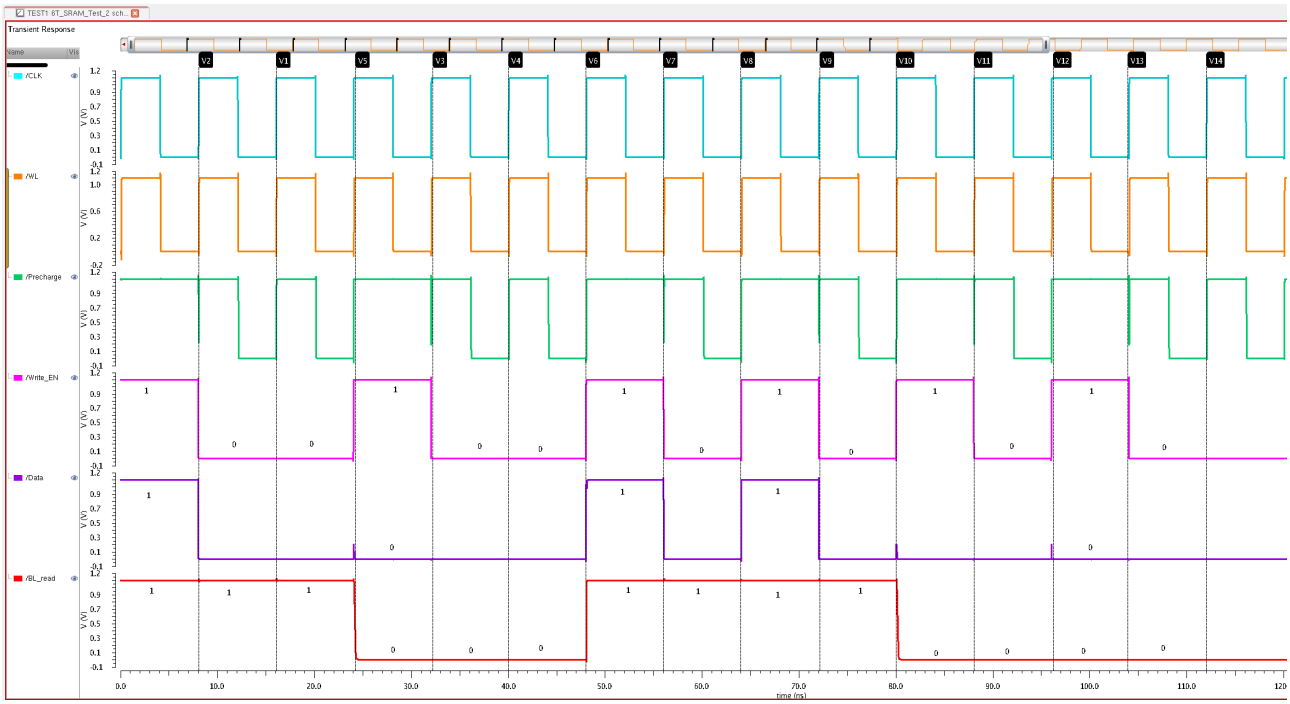


Figure 36: SRAM Test Result

WL	Write_EN	Precharge	BL_read
1	1	1	$ValidData$
1	0	1	$ValidData$
0	1	1	$ValidData$
0	0	0	$High - Z$

4.4 SRAM Array Design

The **SRAM array** requires a **precharge module** to ensure that the bitlines are properly initialized before any read or write operations take place. The precharge module is essential for setting the bitlines to a known state (typically high) prior to accessing the data stored in the memory cells.

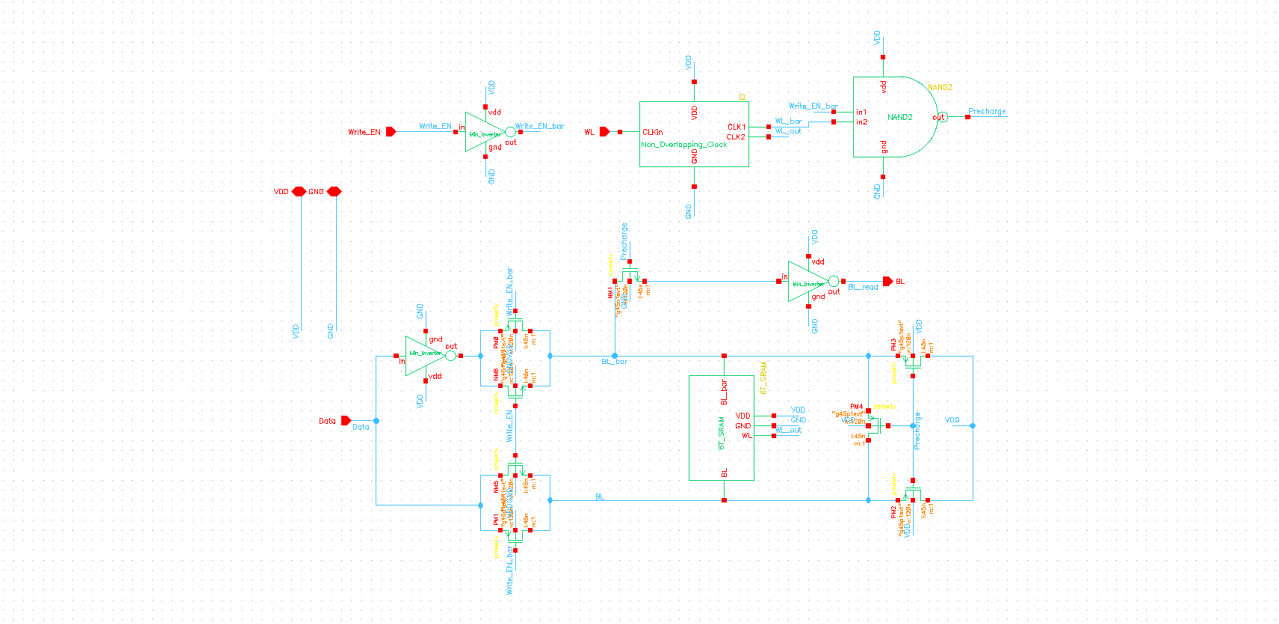


Figure 37: SRAM Precharge Schematics

For the precharge circuit shown in figure 37, we use a PMOS transistor to control the flow of VDD to the BitLine. The precharge operation is executed using an OR logic, as shown in the truth table. This same precharge circuit is also used to prevent the reading of the VDD signal during the precharge phase. To achieve this, we control the read access of the BitLine using an NMOS transistor behaving as transmission gate.

We make a module of SRAM with the precharge circuitry surrounding it for better modality of the Array. Then we shorted the wordline and Write_EN (Load) of all the 16 modules to build the array.

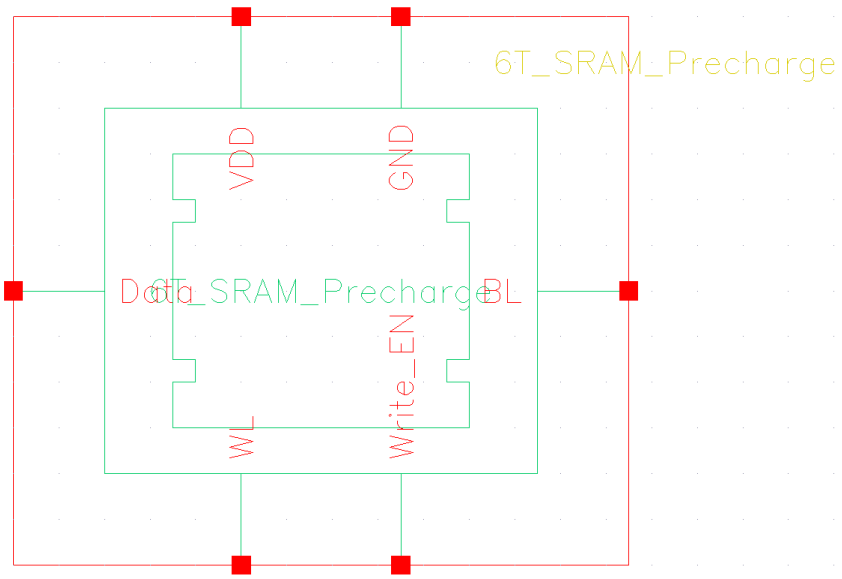


Figure 38: SRAM Precharge Symbol

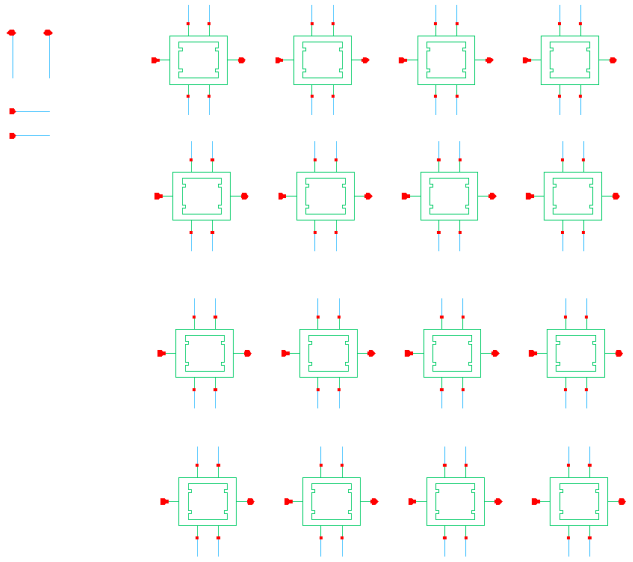


Figure 39: SRAM Array Schematics

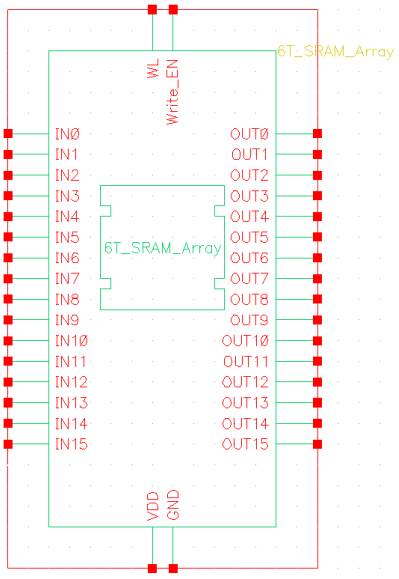


Figure 40: SRAM Array Symbol

4.5 SRAM Array Test

We created a buffer connecting corresponding D data input, CLK clock (WL), and Write_EN (Load) to test the functionality of the SRAM. Setting all the parameters with:

- Period: 10ns,
- Rise time: 50ps,
- Drop time: 50ps.

Inserted data pattern can also be seen in figure **??**:

$$1010101010101010$$

$$High : 1 \quad Low : 0$$

using a VBIT. Clock using Vpulse pulse width 5ns and period of 10ns. Write_EN input using the VBIT to control the writing in SRAM array and reading from SRAM Array can be seen in figure43 with the period of 10ns:

0 : Read 1: Write

$$00000000000000001000000000000000$$

$$Write : 1 \quad Read : 0$$

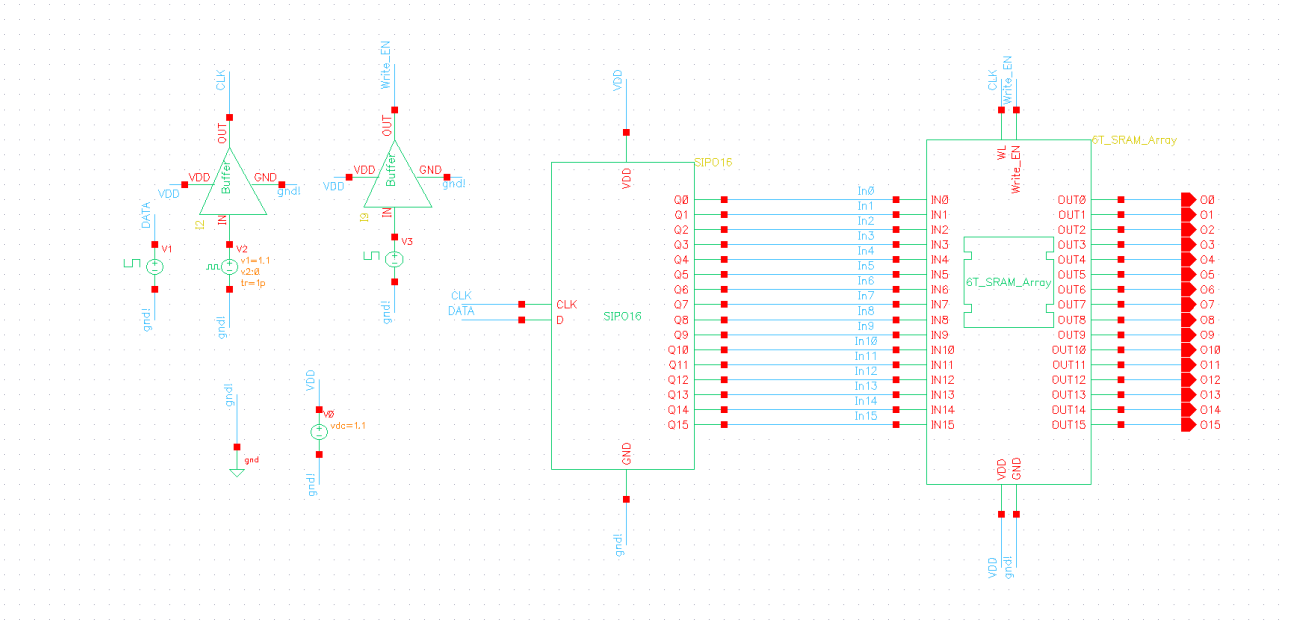
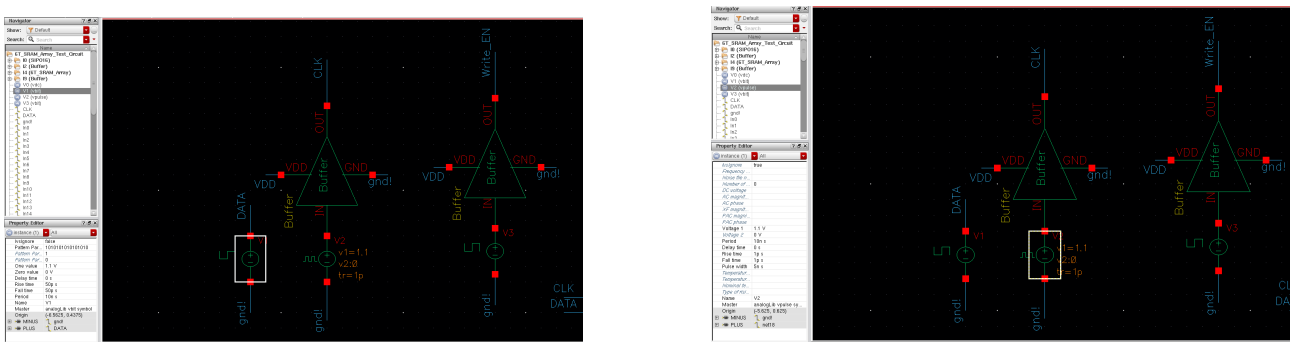


Figure 41: SRAM Test Schematics

Test Setup:



(a) Data Setup (b) Clock Setup

Figure 42: Data and Clock Setups for SRAM Array

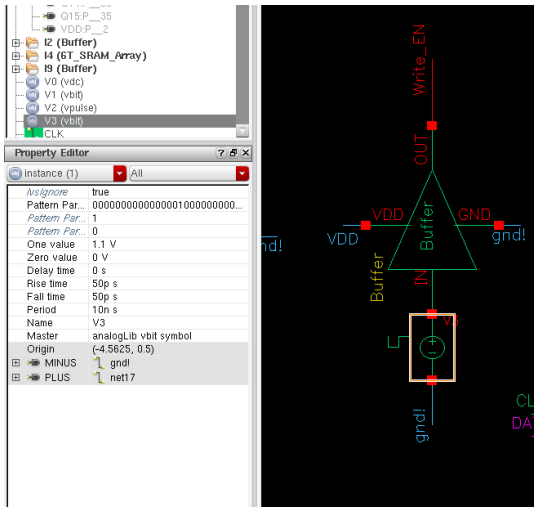


Figure 43: Write Enable Setup

Test Result:

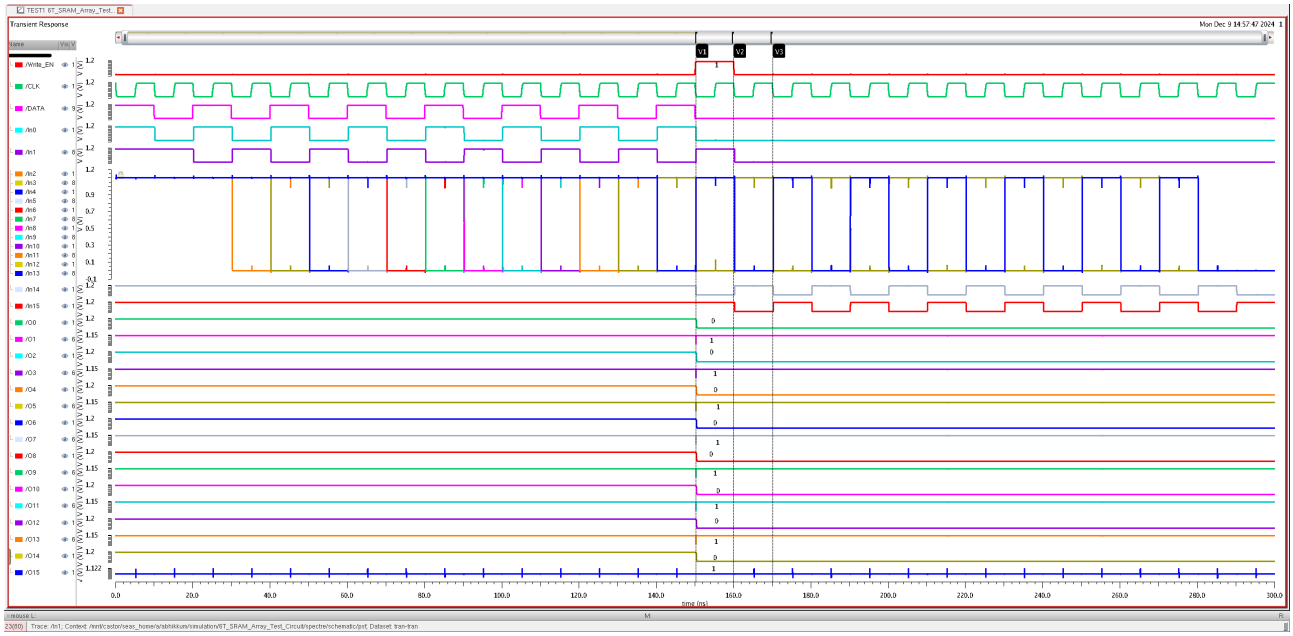


Figure 44: SRAM Array Test Result

We can observer in the 16th cycle we get all the 16bit data, At the end of the 15th cycle we switch on the Write_EN and all 16 bits data gets written in the SRAM Array in the 16th cycle and in the 17th cycle when Write_EN is close we are reading the data the MSB can be observed in Q15 and LSB in Q0.

The data is also annotated in the graph for better observation.

`1010101010101010`

5 Look-Up Table (LUT)

A Look-Up Table (LUT) is a memory element used in FPGAs to implement logic functions by storing pre-computed truth table values for fast evaluation.

5.1 Functional Description and Logic Operation

The Look-Up Table (LUT) in this design is used to implement logic functions within an FPGA by storing pre-computed truth table values for fast evaluation. The design incorporates an optimized 2-to-1 multiplexer (MUX), where the inverter is moved outside the cell to switch the entire column with a single inversion, improving speed. The LUT operates by using select signals (S0, S1, S2, S3) to determine the corresponding output value based on the truth table. The MUX design is modified for higher performance, with only one MUX cell retaining the original design, while the rest are optimized. The LUT is tested by inputting select signals in a sequence of time periods (8ns, 16ns, 32ns, 64ns) and applying alternating input values (e.g., 0101010101010101). The output is compared against the truth table to ensure correct operation, and the results align with the expected behavior.

5.2 Baseline 2-to-1 MUX Design

The design reused from homework 7:

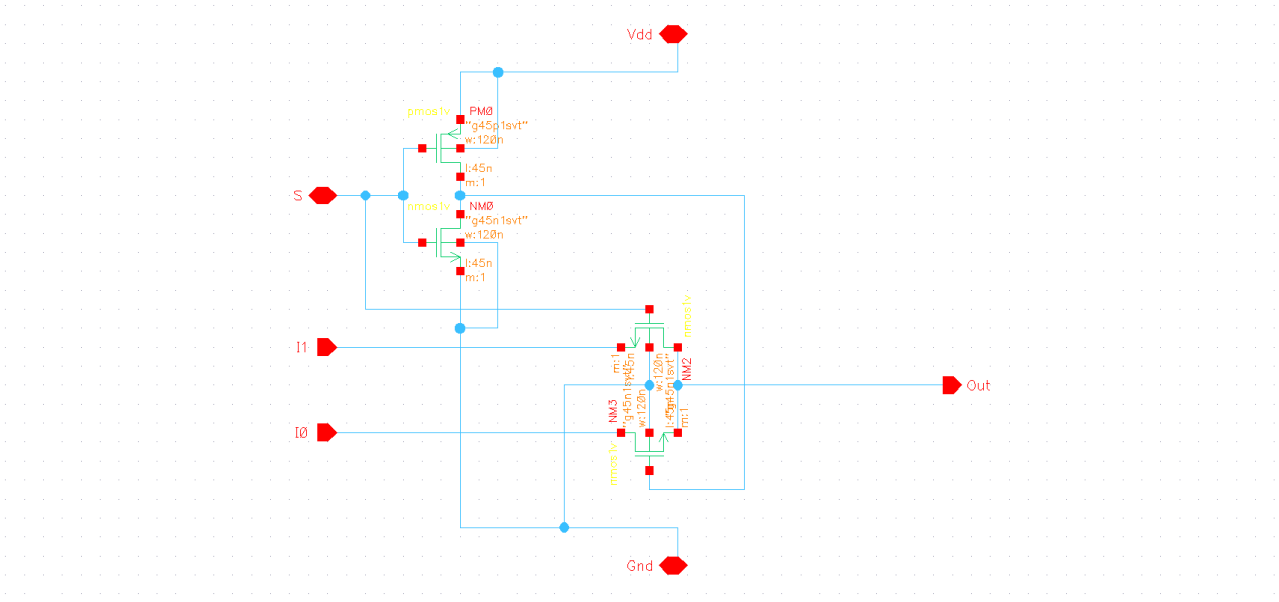


Figure 45: Homework 7 MUX Schematics

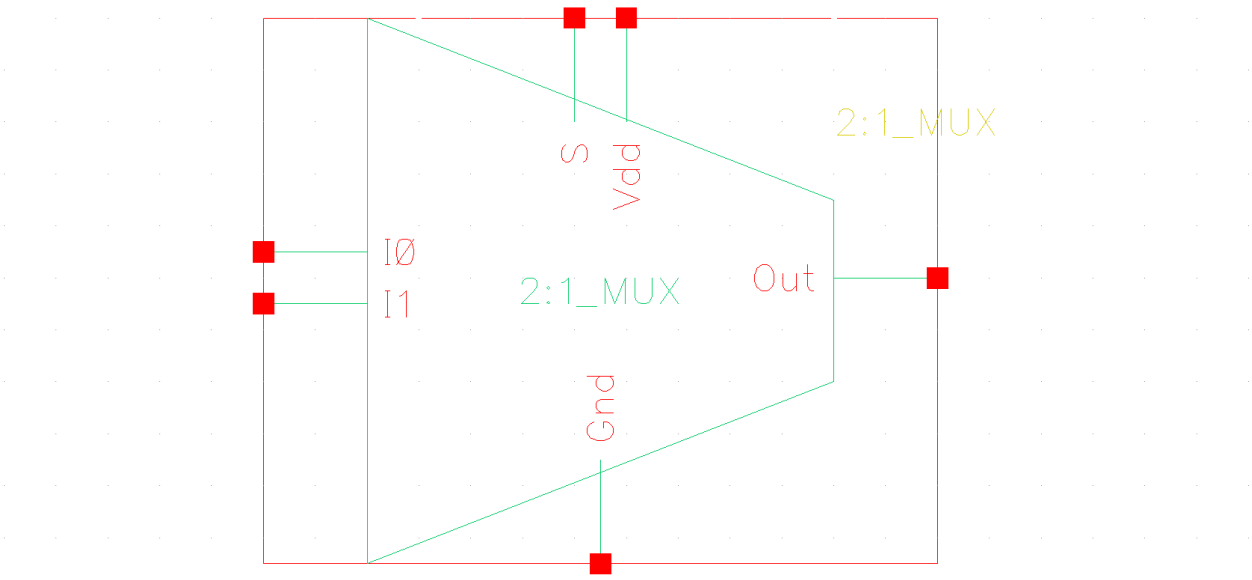


Figure 46: Homework 7 MUX Symbol

5.3 Baseline LUT Design

The design was reused from homework 7 and check its delay:

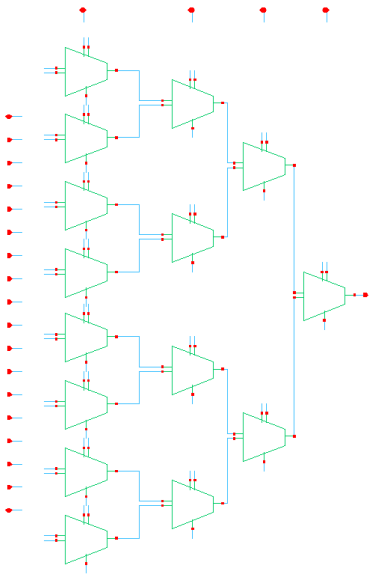


Figure 47: Homework 7 LUT Schematics

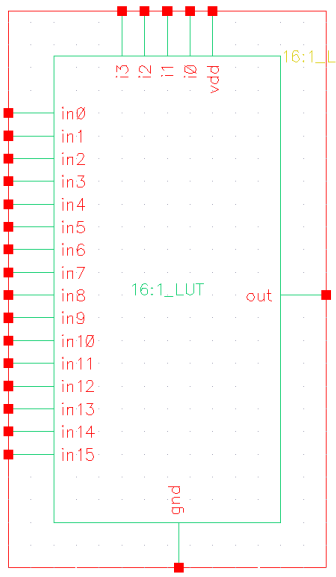


Figure 48: Homework 7 LUT Symbol

Delay Test Schematics:

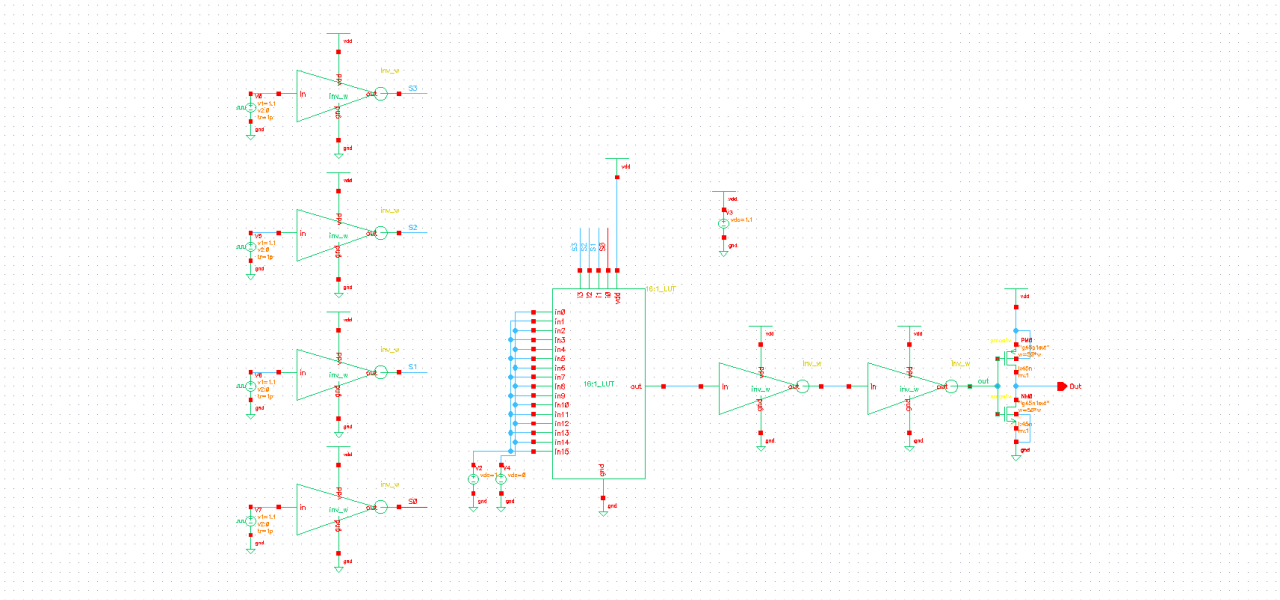


Figure 49: Delay Test Schematics

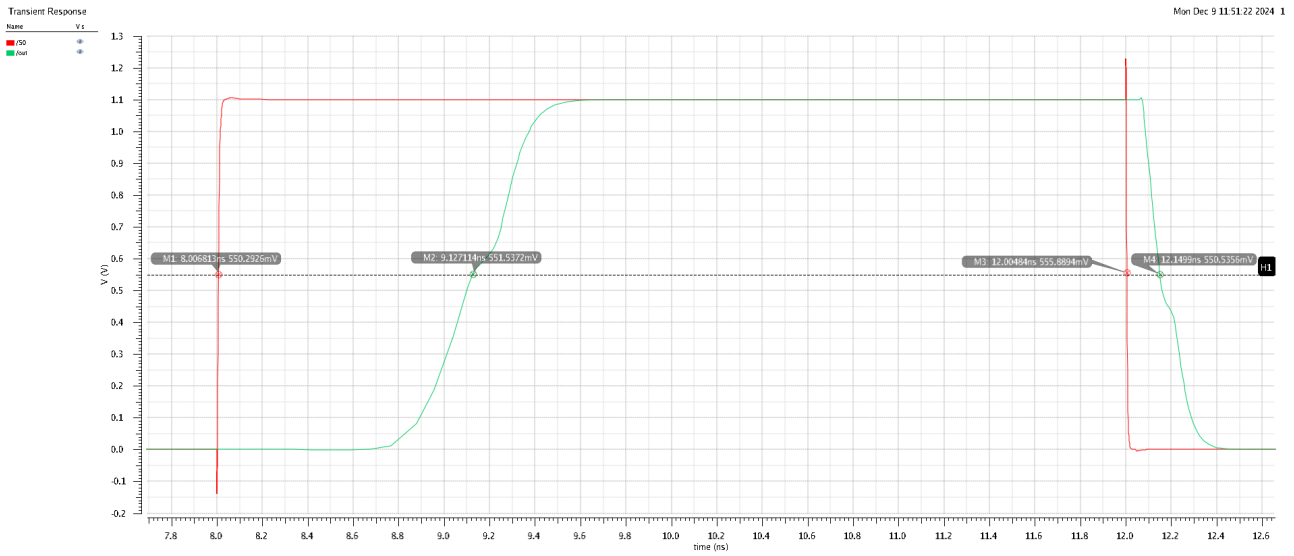


Figure 50: Delay Test Result

Timing Analysis

- T_{lh}: 1.12 ns
- T_{hl}: 0.145 ns
- T: $\frac{T_{lh}+T_{hl}}{2} = \frac{1.12+0.145}{2} = 0.6325\,ns$
- Worst case delay: 1.12 ns

5.4 Optimized Transistor Width

We sweep the width of the transistors from minimum 120nm to 240nm in 10 steps

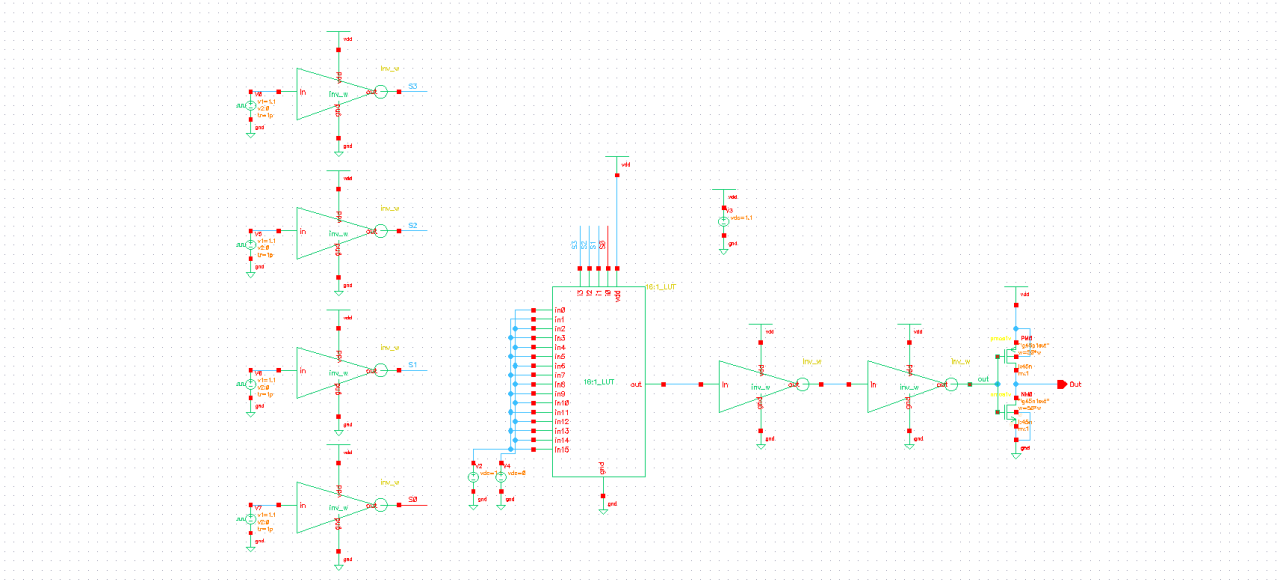


Figure 51: Delay Test Schematics

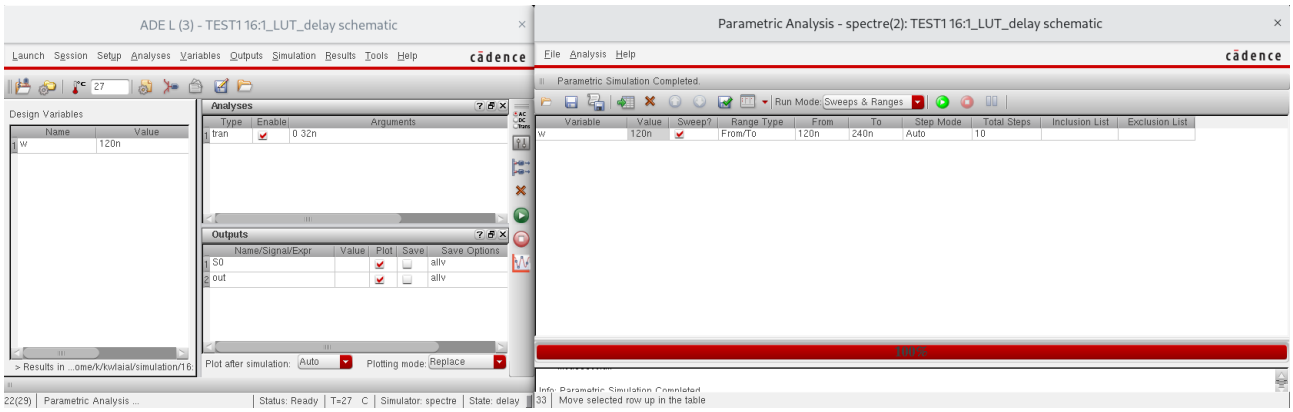


Figure 52: Parametric Setup of Transistor Width

From the analysis result, it shows that the minimum transistor of width 120nm has the best performance of lowest delay. Therefore, width 120nm as the minimal transistor is used.

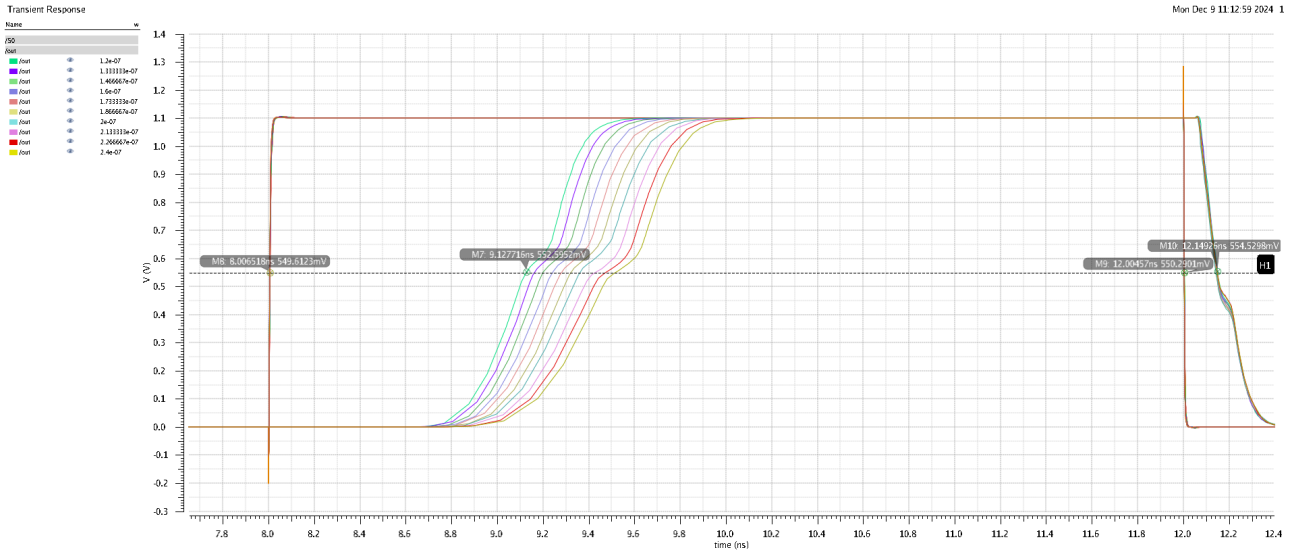


Figure 53: Parametric Analysis of Transistor Width

5.5 Optimised 2-to-1 MUX Design

To achieve higher speed, our optimized design relocates the inverter connecting the S gate to the outside of the cell, allowing a single inversion to switch the entire column.

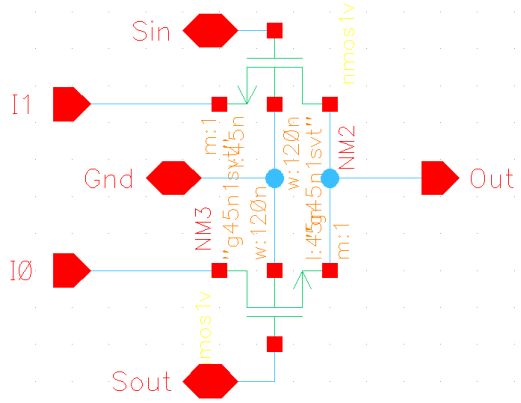


Figure 54: Optimised 2-to-1 MUX Schematics

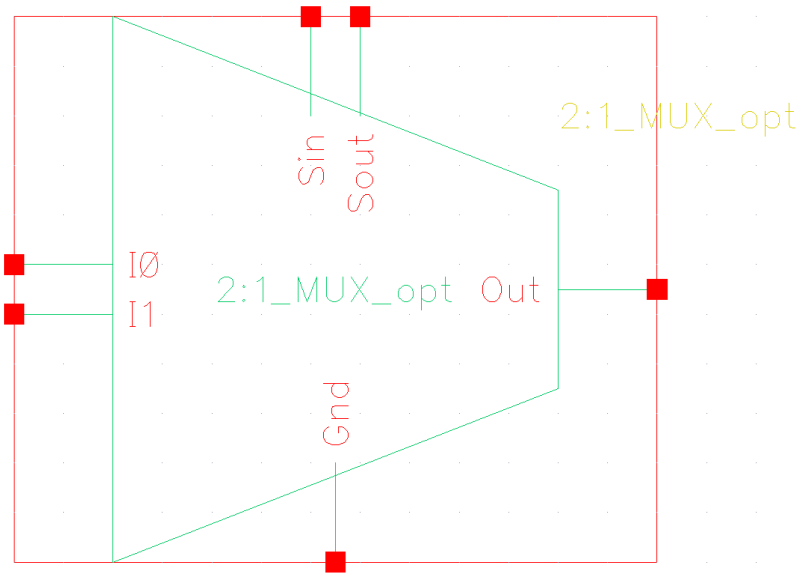


Figure 55: Optimised 2-to-1 MUX Schematics

5.6 Optimised 2-to-1 MUX Test

The test set the vdc periods of I0, I1 and S gates to be 8ns, 16ns and 32ns correspondingly to check the test results.

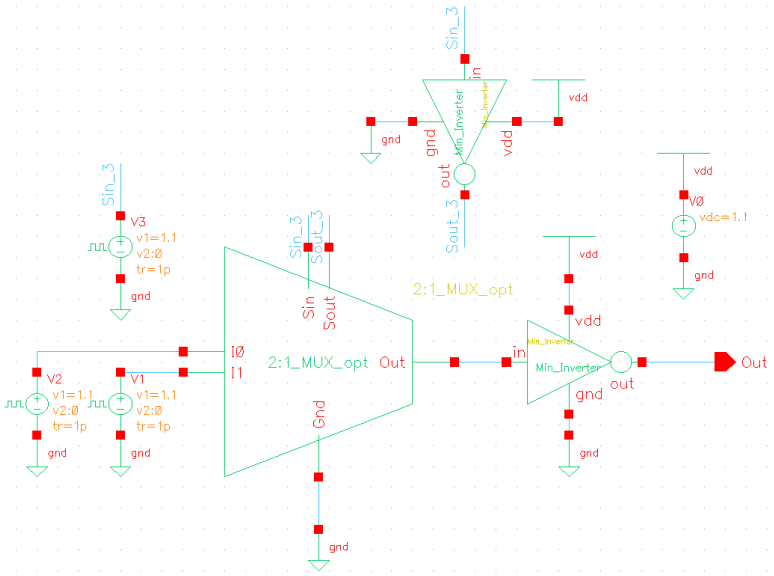
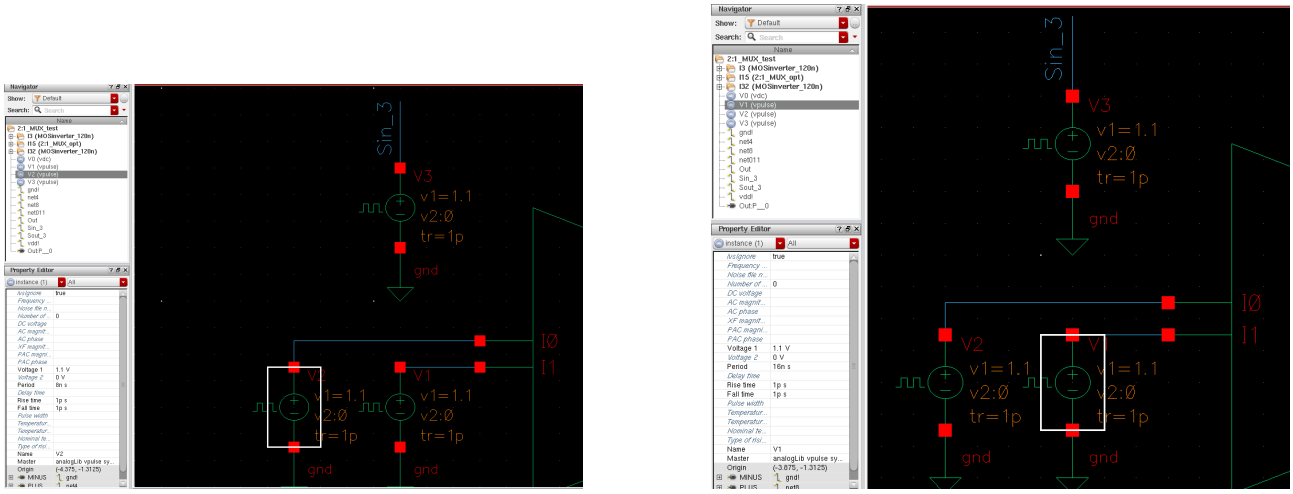


Figure 56: Optimised 2-to-1 MUX Test

Test Setup:



(a) I_0 Setup

(b) I_1 Setup

Figure 57: I_0 and I_1 Setups

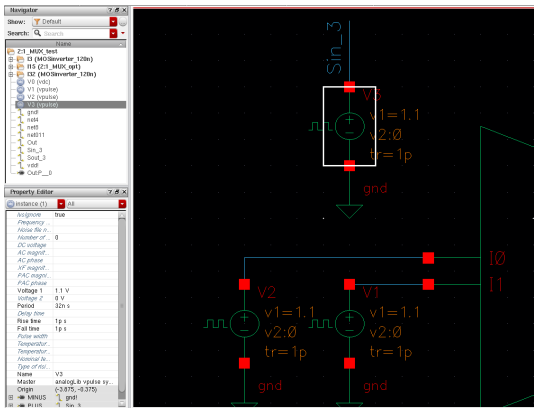


Figure 58: Select Setup

Test result: The result aligns with the MUX turth table

S	**I1**	**I0**	**MUX_Output**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

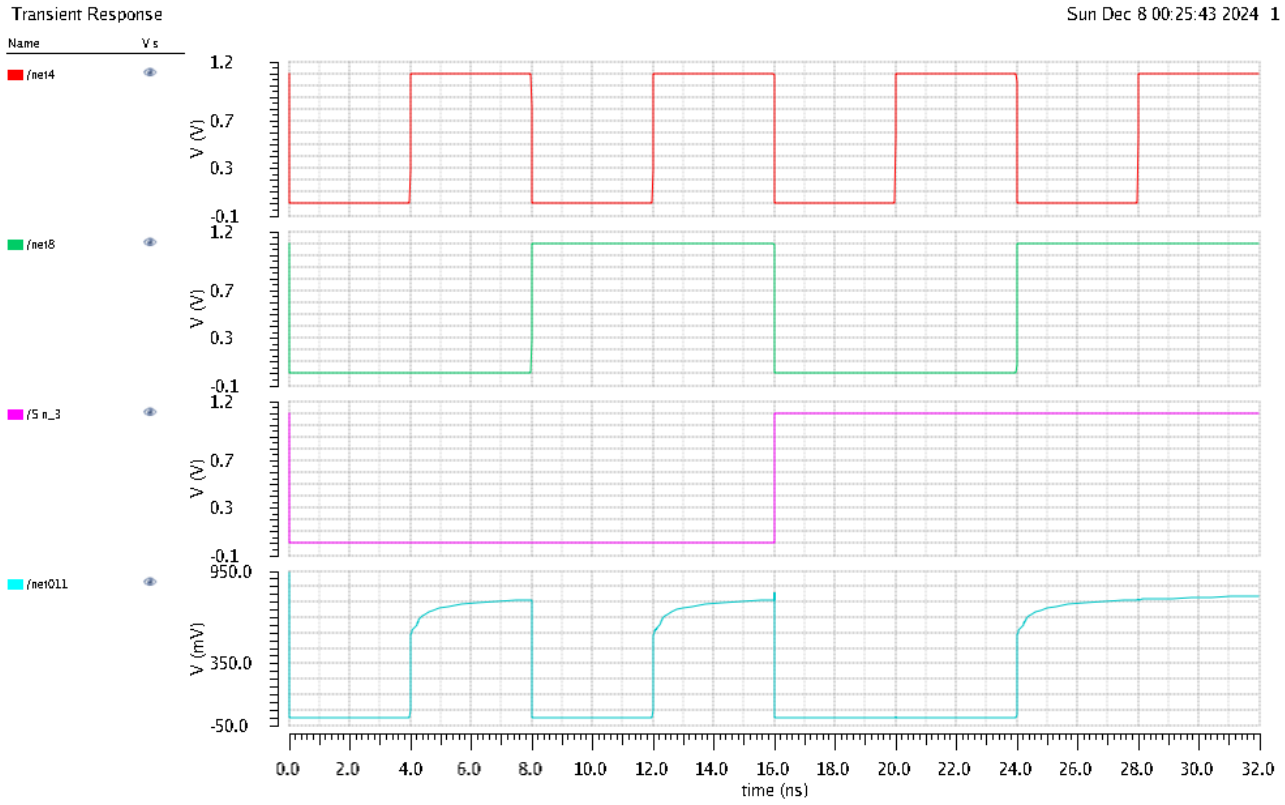


Figure 59: Optimised 2-to-1 MUX Test Result

5.7 Optimised LUT Design

From the optimized MUX design above, the inverter has been moved outside the cell to switch per select signal, while the column with only one MUX cell retains the original design, with the inverter embedded inside the MUX, as shown below.

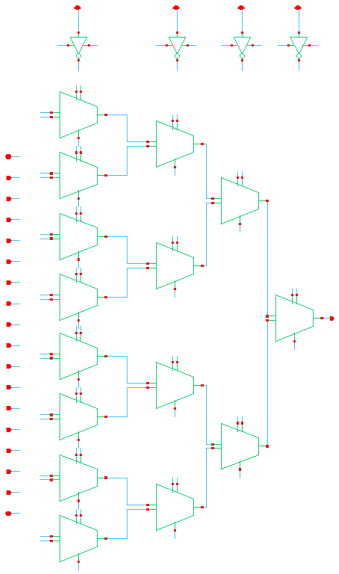


Figure 60: LUT Schematics

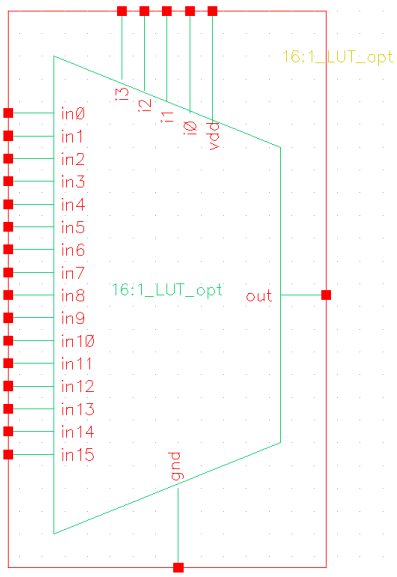


Figure 61: LUT Symbol

5.8 Optimised LUT Test

The selects S0, S1, S2, S3 are inputed in sequence 8n, 16n, 32n, 64n periods and set the inputs of 0101010101010101.

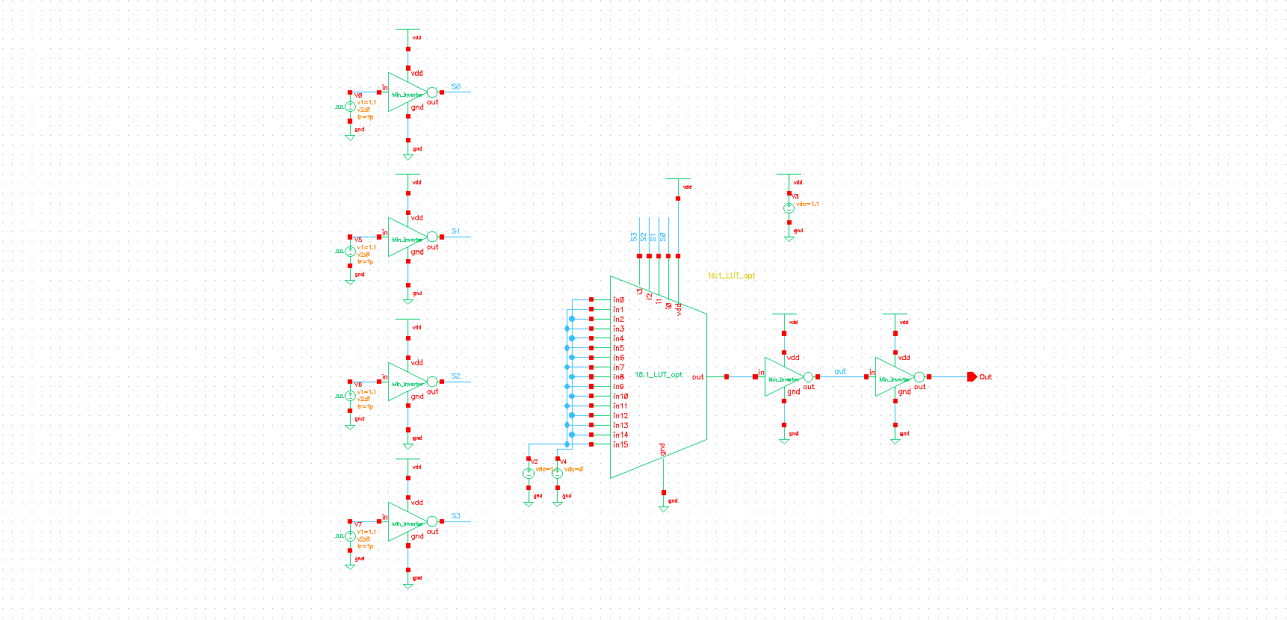
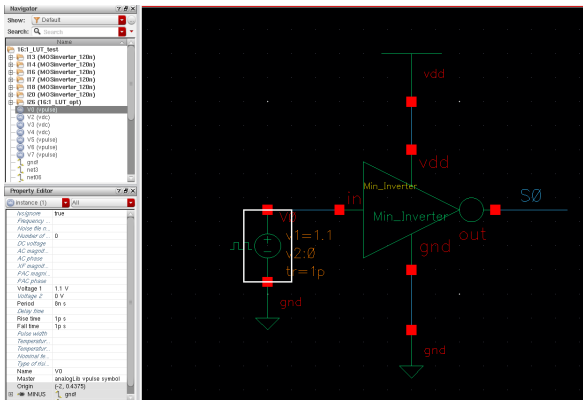
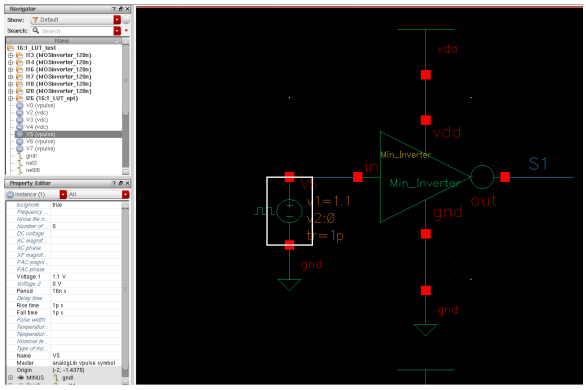


Figure 62: LUT Test Schematics

Test Setup:

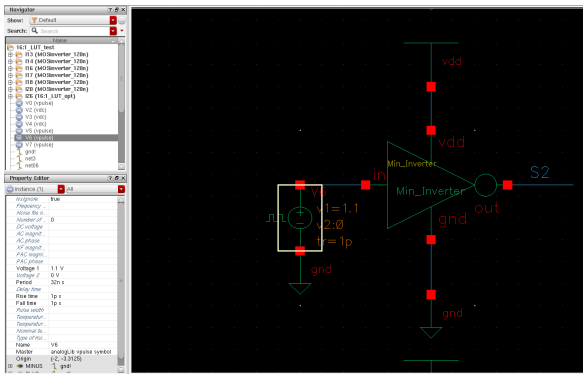


(a) S_0 Setup

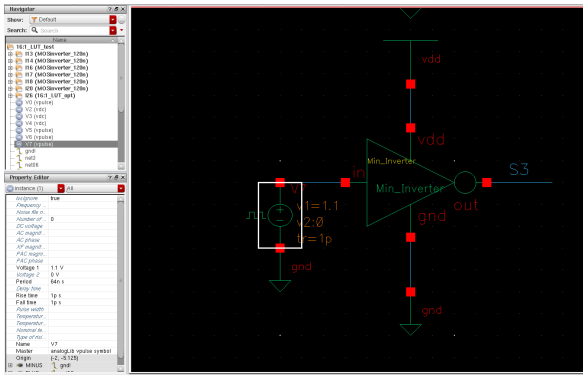


(b) S_1 Setup

Figure 63: S_0 and S_1 Setups



(a) S_2 Setup



(b) S_3 Setup

Figure 64: S_2 and S_3 Setups

Test Result:

S3	S2	S1	S0	Output
1	1	1	1	0
1	1	1	0	0
1	1	0	1	0
1	1	0	0	0
1	0	1	1	0
1	0	1	0	0
1	0	0	1	0
1	0	0	0	0
0	1	1	1	1
0	1	1	0	1
0	1	0	1	1
0	1	0	0	1
0	0	1	1	1
0	0	1	0	1
0	0	0	1	1
0	0	0	0	1

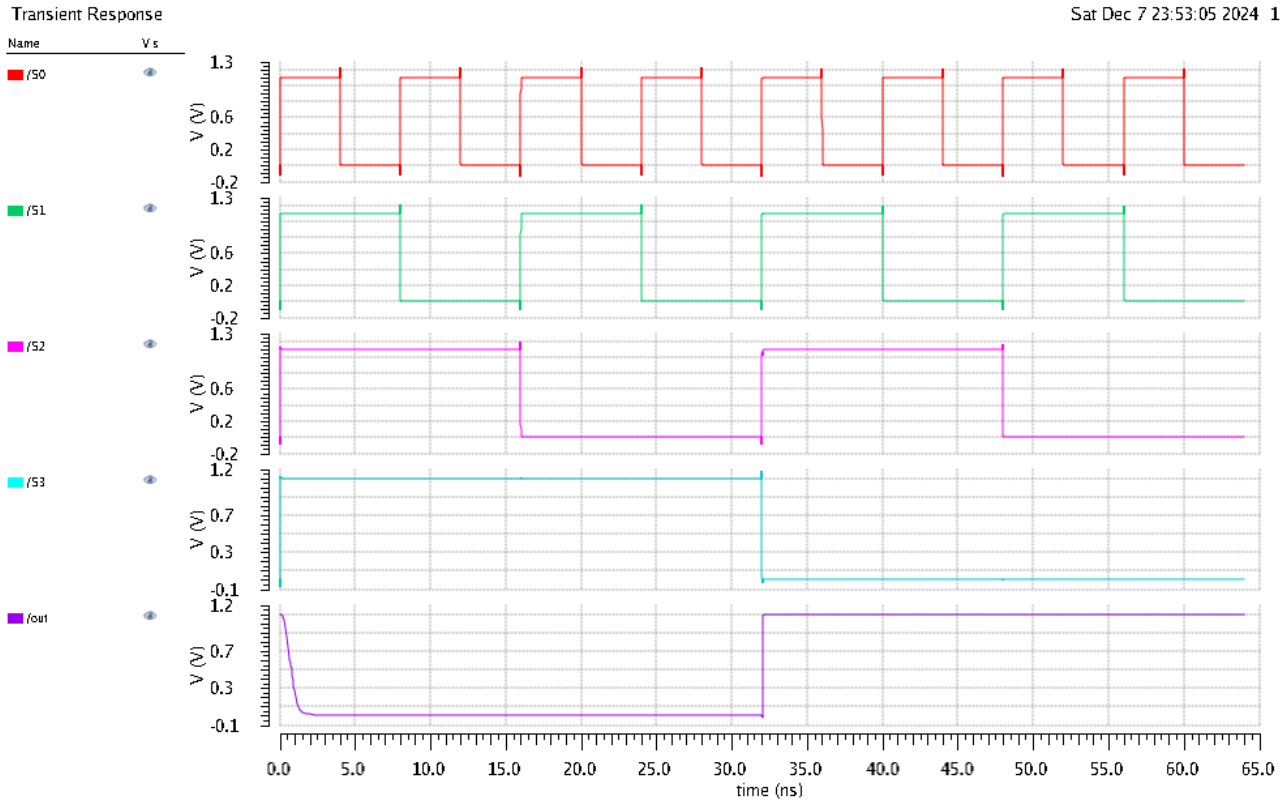


Figure 65: LUT Test Result

5.9 Optimised LUT Delay

Setting the width of the transistors be minimum: 120nm

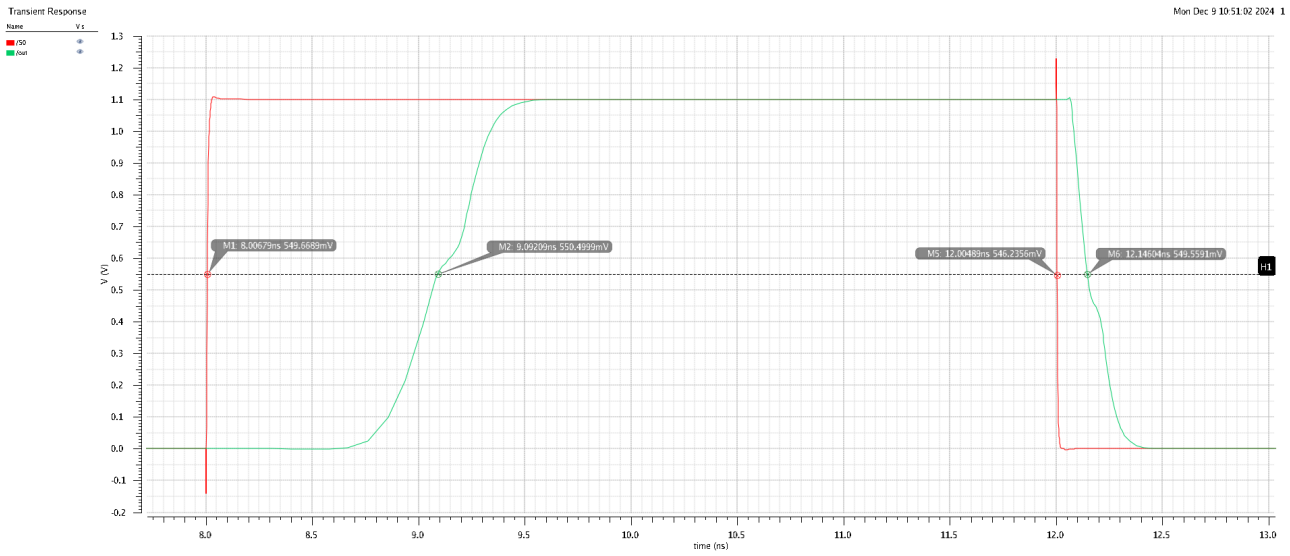


Figure 66: Optimised LUT Delay

Timing Analysis

- T_{lh}: 1.08 ns

- T_{hl}: 0.141 ns
- T: $\frac{T_{lh}+T_{hl}}{2} = \frac{1.08+0.141}{2} = 0.611\,ns$
- Worst case delay: 1.08 ns

Delay improvement compared with baseline:

$$\frac{(1.08 - 1.12)}{1.12} \times 100 = -3.57\%$$

6 Configurable Logic Block (CLB)

A Configurable Logic Block (CLB) is a fundamental FPGA component enabling flexible logic implementation through LUTs, flip-flops, and routing resources.

6.1 Functional Description and Logic Operation

Components of the CLB Unit

The CLB (Configurable Logic Block) unit consists of the following main components:

1. **SIPO (Serial-In-Parallel-Out) Shift Register:**
   - This shift register writes the input data serially into the **16 SRAM cells**.
   - The leftmost bit in the input data represents the **MSB (Most Significant Bit)**, while the rightmost bit represents the **LSB (Least Significant Bit)**.
   - It ensures efficient loading of data into the SRAM cells, one bit at a time, and outputs the data in parallel.
2. **16 SRAM Cells:**
   - These cells store the configuration data required for the logic operation.
   - The stored data can be selectively written into the **LUT (Look-Up Table)** during operation, acting as programmable memory.
3. **16-bit LUT (Look-Up Table):**
   - The LUT acts as a programmable truth table.
   - It selects one of the 16 SRAM cells based on the input address, which determines the logic function being implemented.
   - The selected SRAM cell's value is passed to the output.
4. **Buffer:**
   - This component restores any **threshold voltage drop ($\mathbf{V_{th}}$)** that may occur during data transfer, ensuring signal integrity.
   - It maintains a stable output voltage level for reliable logic operations.
5. **D Flip-Flop:**
   - The selected bit from the LUT output is stored in the D flip-flop.
   - This flip-flop can be used for sequential logic or to synchronize the output with a clock signal.
   - We are using the inverted input of clock for the D-Flip-FLop because it is an rising edge trigger device and by giving the inverted clock input we can

trigger the D-Flip-Flip half clock cycle earlier reducing the energy delay. (If the clock input is non inverted from the rest of the system the D-Flip-Flop captures the previous bit due the delay in switching signal as clk switches quite quickly compared to that and we have to discard that bit and then the correct operating takes place.)

- The **SIPO shift register** facilitates easy programming of the 16 SRAM cells by converting a serial data stream into parallel bits. This approach simplifies the design and reduces the number of input pins required.
- The **SRAM cells** act as non-volatile memory for the LUT, enabling the CLB to be configured to implement different logic functions. We are only writing in the array when all the data are available to reduced the energy consumption during the continuous write.
- The **LUT** is the heart of the CLB, as it allows for flexible implementation of logic functions. With 4 switching bits, the LUT can implement any 4-variable Boolean function by referencing its 16 possible outputs.
- The **buffer** ensures that the output signal strength is maintained, even if there are losses due to the capacitive or resistive effects in the circuit.
- Finally, the **D flip-flop** is critical for sequential logic operations, enabling the CLB to function as a part of larger finite state machines or clocked systems.

6.2 CLB Design

Finally, we put all the components together to construct a CLB unit and put the unit inside a new testbench to test its frequency and energy consumption. All the inputs are driven by buffer to simulate the driving load.

6.2.1 CLB without D-Flip Flop Verification

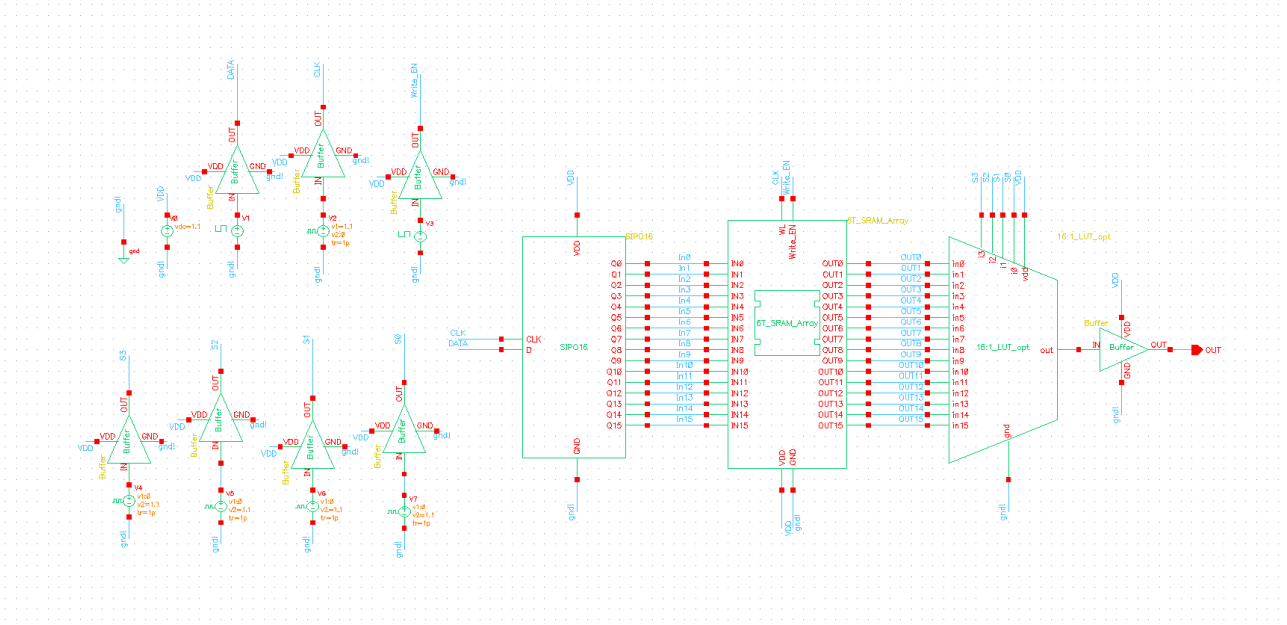


Figure 67: CLB Without D Flip-Flop

Result

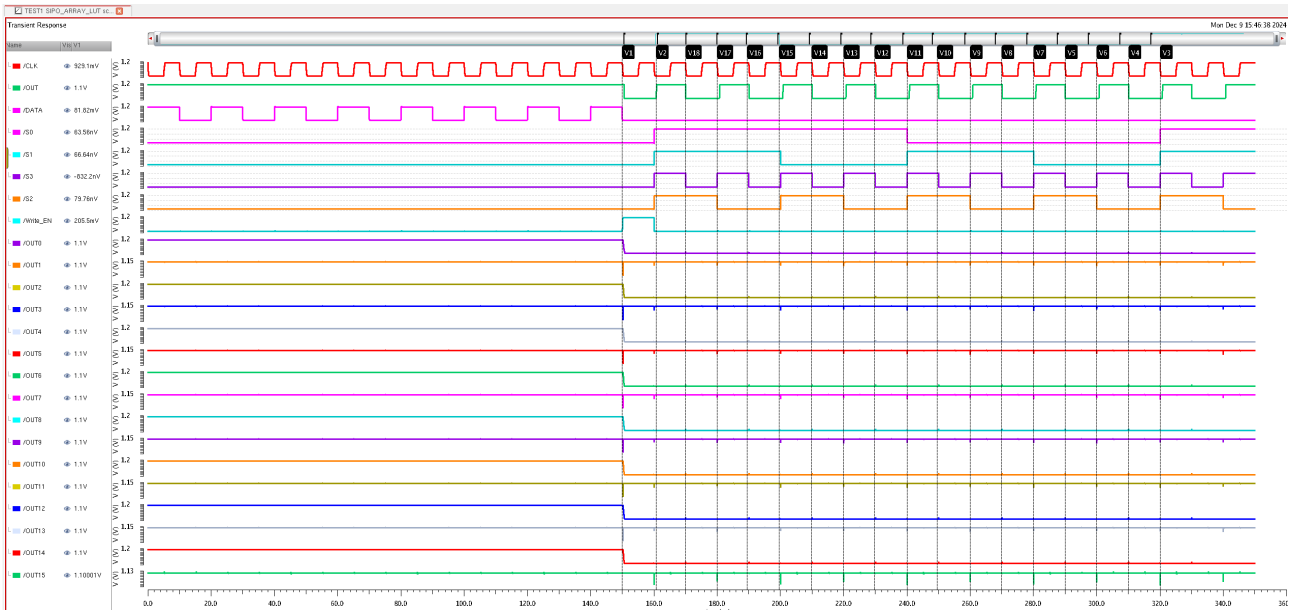


Figure 68: CLB Without D Flip-Flop Output

In the graph shown in figure (68) we can observe that the Write_EN is switched during the 16th clock cycle and we start the read operation in the 17th cycle using the LUT. Our MSB (OUT15) is 1 and LSB (OUT0) is 0 and other bits are also correct, which can be seen in figure (68) 1010101010101010

6.2.2 CLB Module

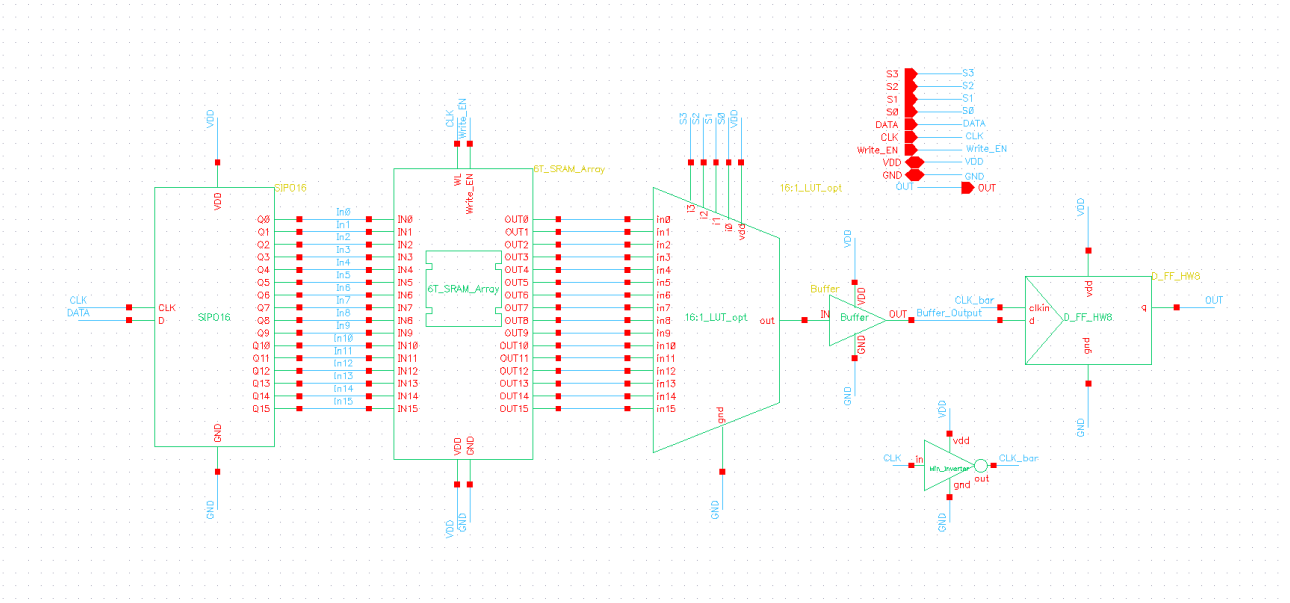


Figure 69: CLB Schematics

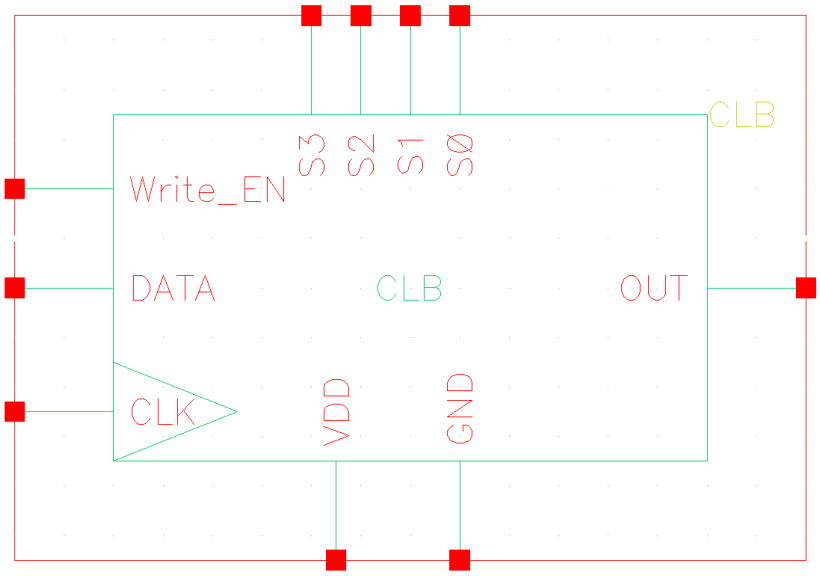


Figure 70: CLB Symbol

6.3 CLB Test

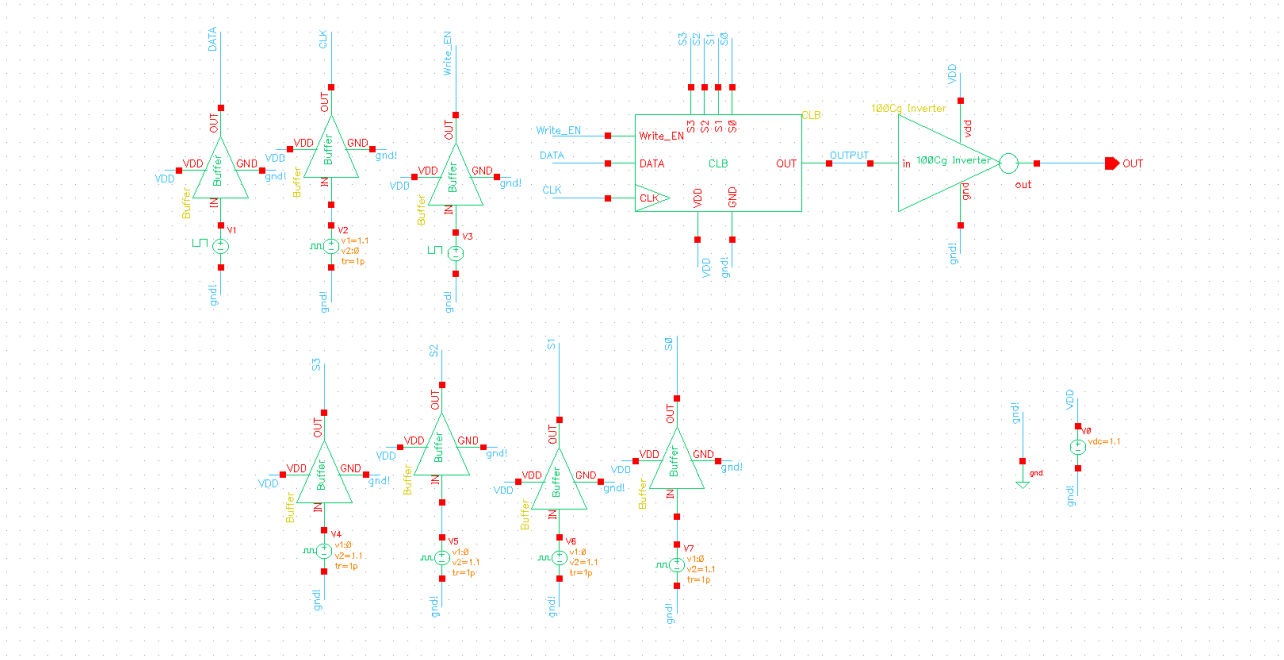
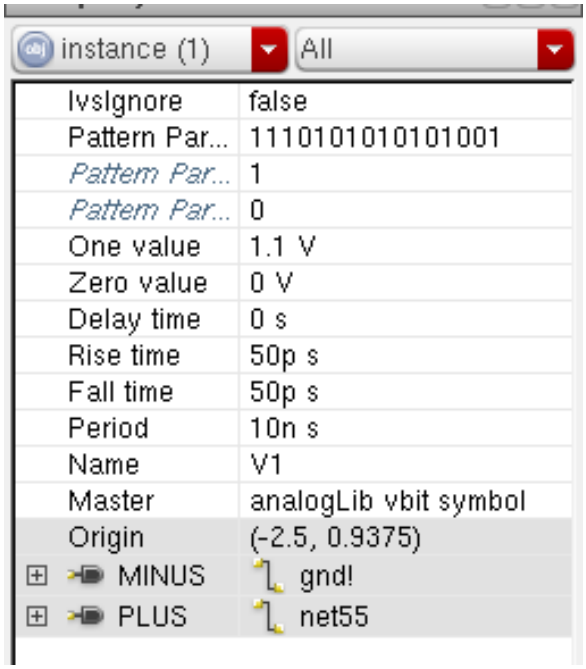
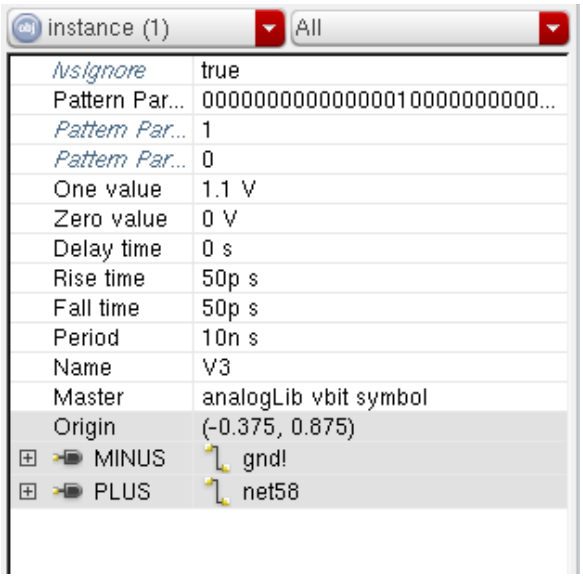


Figure 71: CLB Test Schematics

Test Setup:

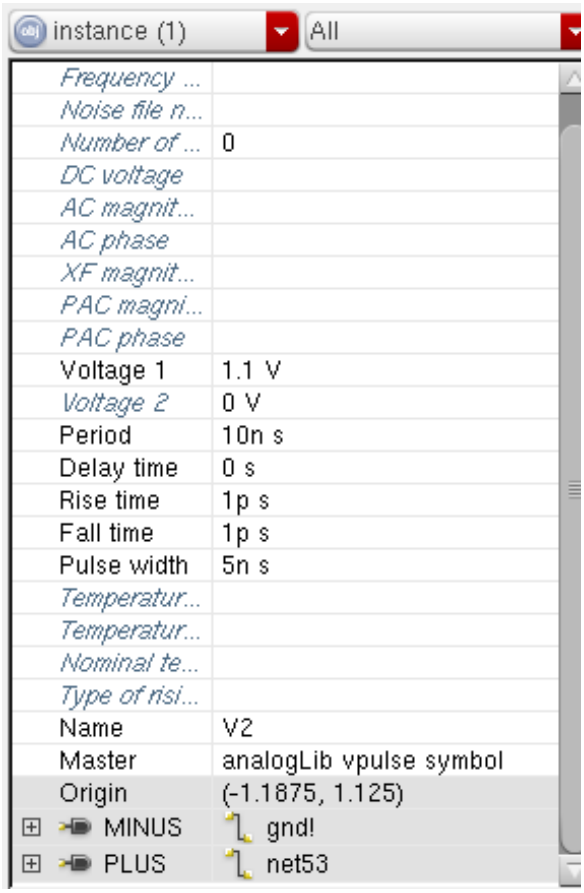


(a) Data Setup

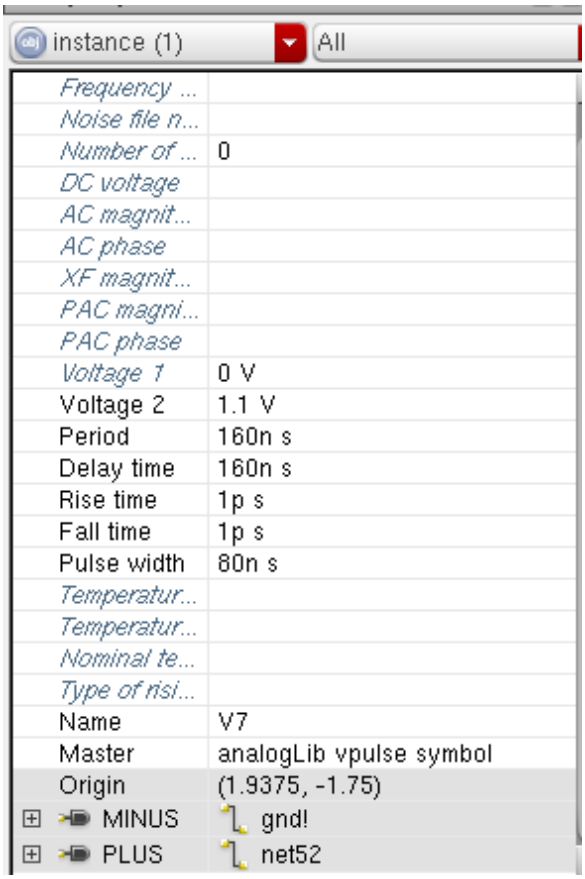


(b) Write Enable Setup

Figure 72: Data and Write Enable Setups



(a) Clock Setup



(b) S_0 Setup

Figure 73: Clock and S_0 Setups

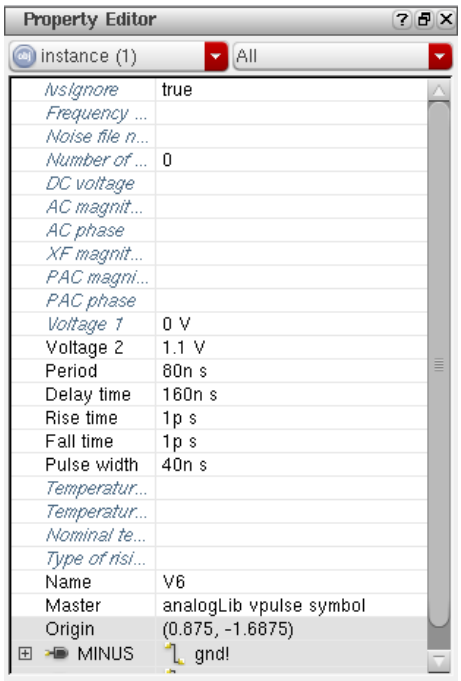
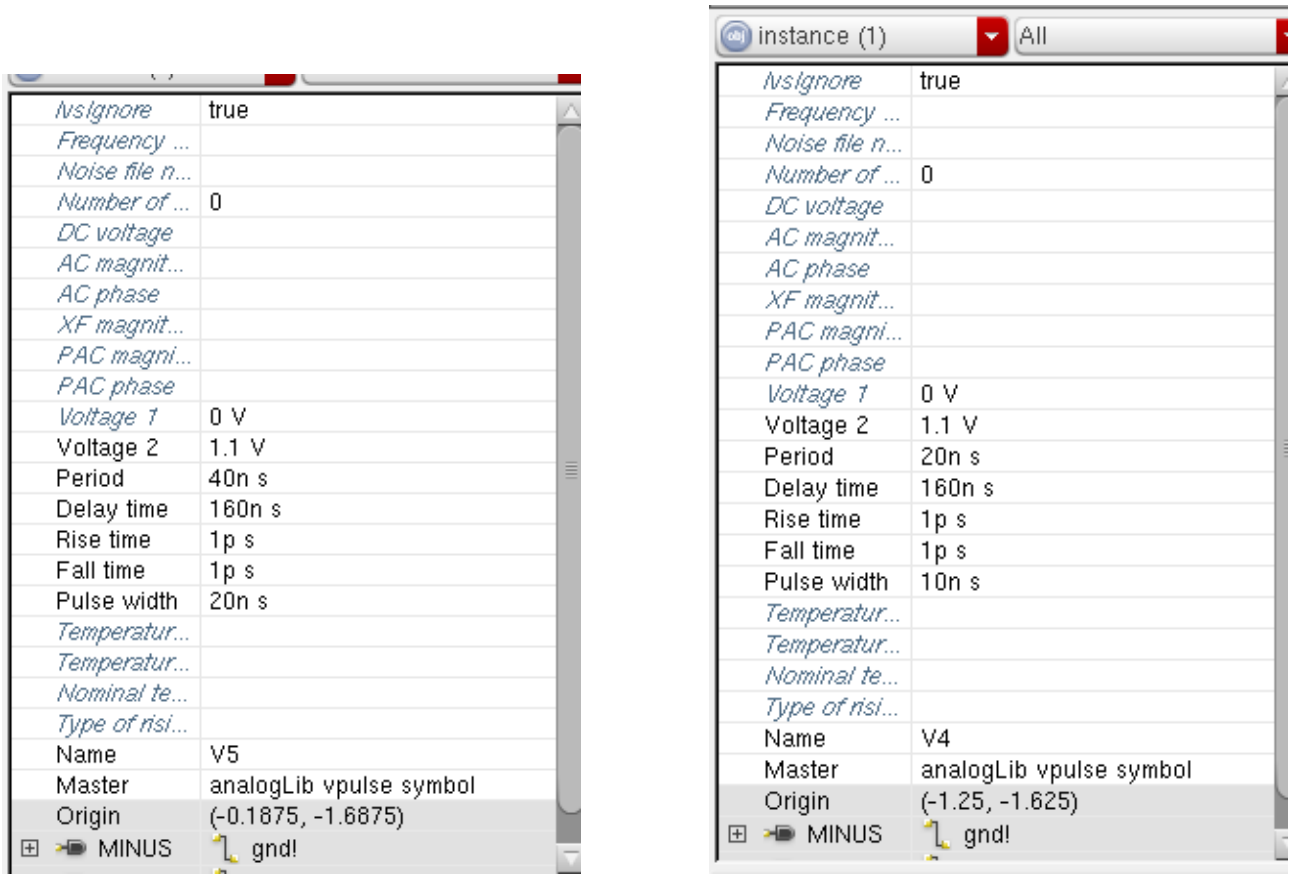


Figure 74: S1 Setup



(a) S_2 Setup (b) S_3 Setup

Figure 75: S_2 and S_3 Setups

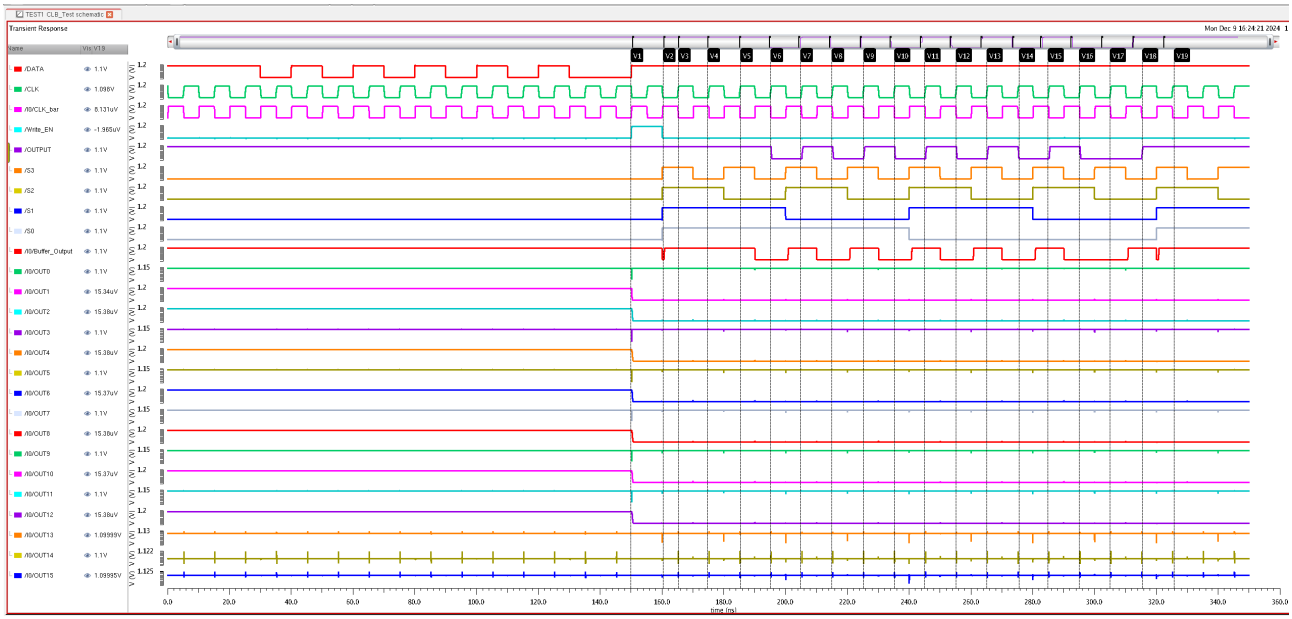


Figure 76: CLB Test Result

We feed the data (1110101010101001) into the Serial-in Parallel-out (SIPO) block, along with the clock signal. Once the SIPO has received all the data, we control the Write_Enable

(LOAD) to write the data into the SRAM. After the data is written and stabilized in the SRAM, we use the Lookup Table (LUT) to read the data from the SRAM array (160ns in our case). The switching input is fed through a Vpulse signal with periods that are factors of 2, covering all the bits (Period of S0 = 10ns, S1 = 20ns, S2 = 40ns, S3 = 80ns).

The table below shows the periods of the signals S_0 to S_4 and the clock signal (CLK).

Signal	**Period (Time in ns)**
CLK	T
S_0	$2T$
S_1	$4T$
S_2	$8T$
S_3	$16T$

Since we are using a D-Flip-Flop with a falling edge trigger, the read data passes through a buffer to restore any Vth loss in the LUT. With the falling edge (CLK_bar) of the clock, the data is reflected in the D-Flip-Flop (after 160ns + the next falling clock edge = 165ns). In the initial phase, all the switching inputs are high (MSB), and we observe a high output in the D-Flip-Flop, which matches the input data. As the LSB input data is high and the switching controls of the LUT are low, we continue to observe a high output, which is consistent with the input data. The outputs for other bits also match the input data, based on the switching control.

6.4 Maximum Operating Frequency

To get the maximum frequency, we have ran multiple parametric analysis and after a certain frequency the D-Flip-flop setup time exceeds the data hold time and D-Flip-Flop stops given the correct output data. And our CLB output doesn't fully restored to specified noise margin.

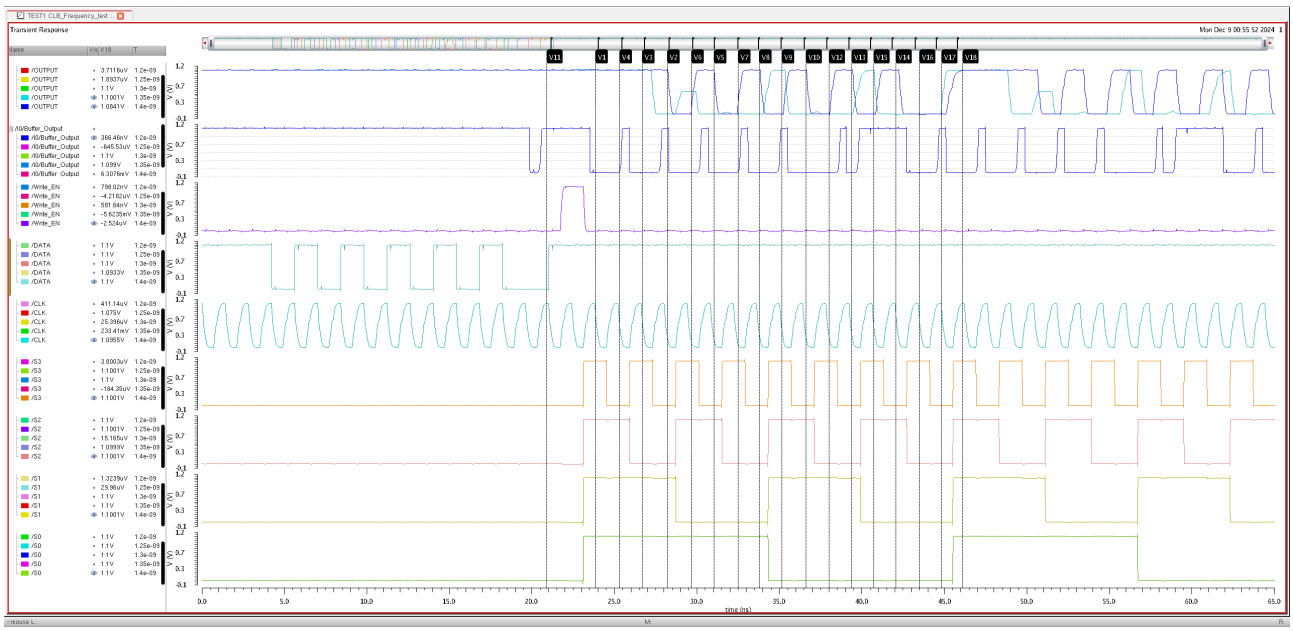


Figure 77: CLB Frequency Analysis (1.2ns - 1.4ns)

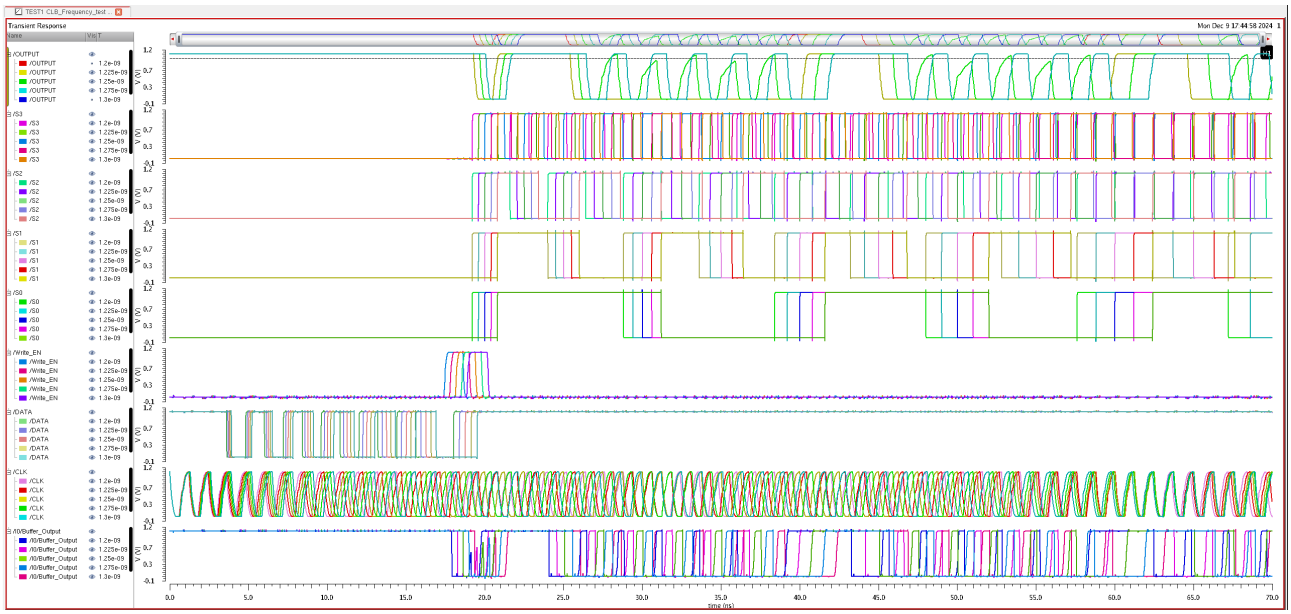


Figure 78: CLB Frequency Analysis (1.2ns - 1.3ns)

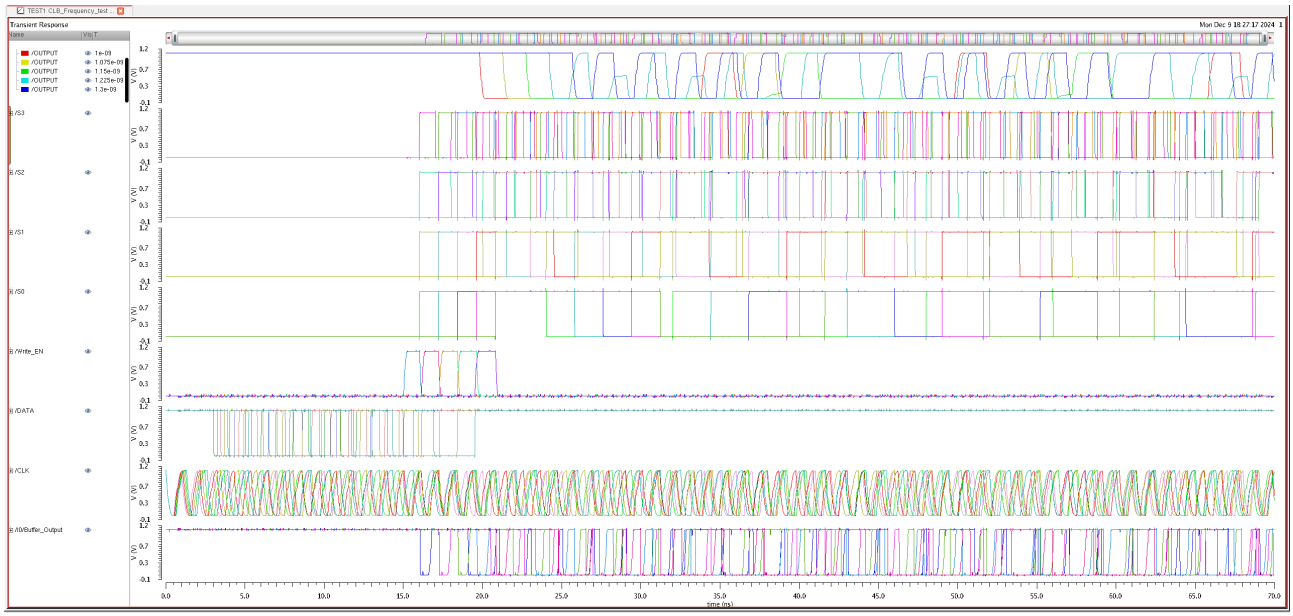


Figure 79: CLB Frequency Analysis (1ns - 1.3ns)

For our design the max frequency we get is at 1.275ns of 7843137254.9 Hz (**784.313 Mhz**) if we further increase the frequency the data is fully observed by the D-Flip-Flop. As our D-Flip-Flop is rising edge triggered and we are feeding it the clock Bar the input data 1110101010101001 can be observed at the output when the data from the restoring buffer sees the falling clock of the CLK_bar out output node of the D-flip-flop shows the correct output 1110101010101001. Which can be observed in the figure 80.

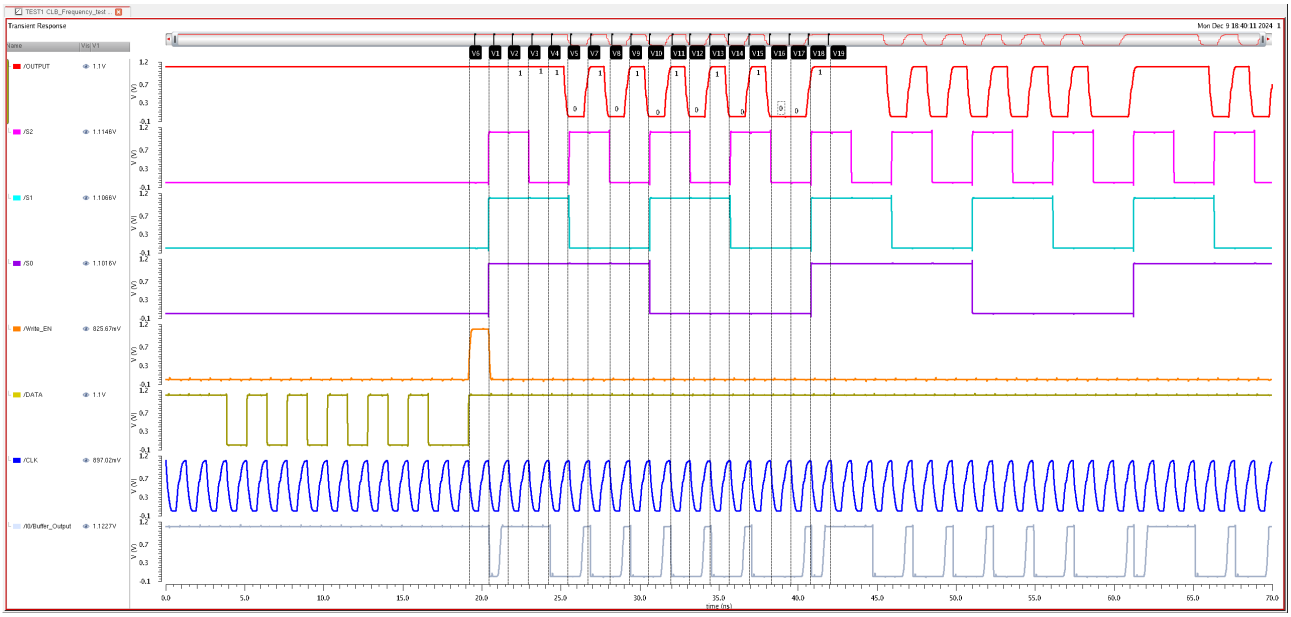


Figure 80: CLB Frequency Period 1.275ns

We can also observe the data is sampled at the falling edge of the D-Flip-FLop which takes the CLK_bar as the input clock resulting in shift of period by the final Flip Flop.

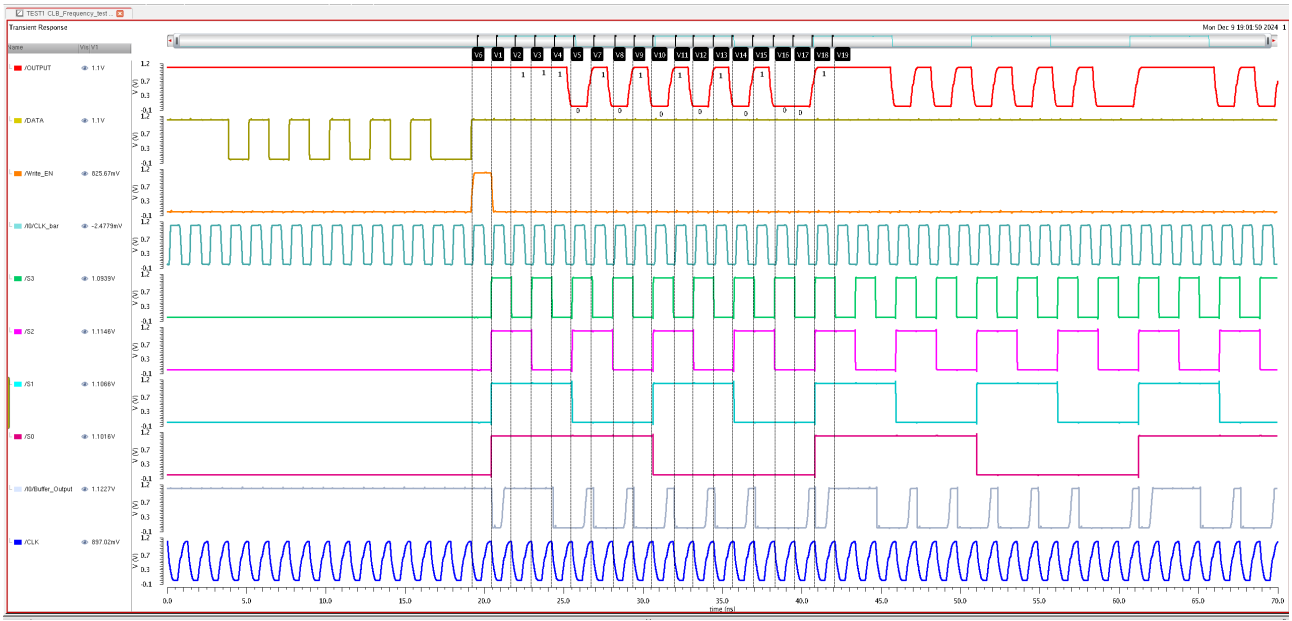


Figure 81: CLB Frequency Period 1.275ns

This show the corrected operation of the CLB at 784.313 Mhz.

6.5 Average Energy

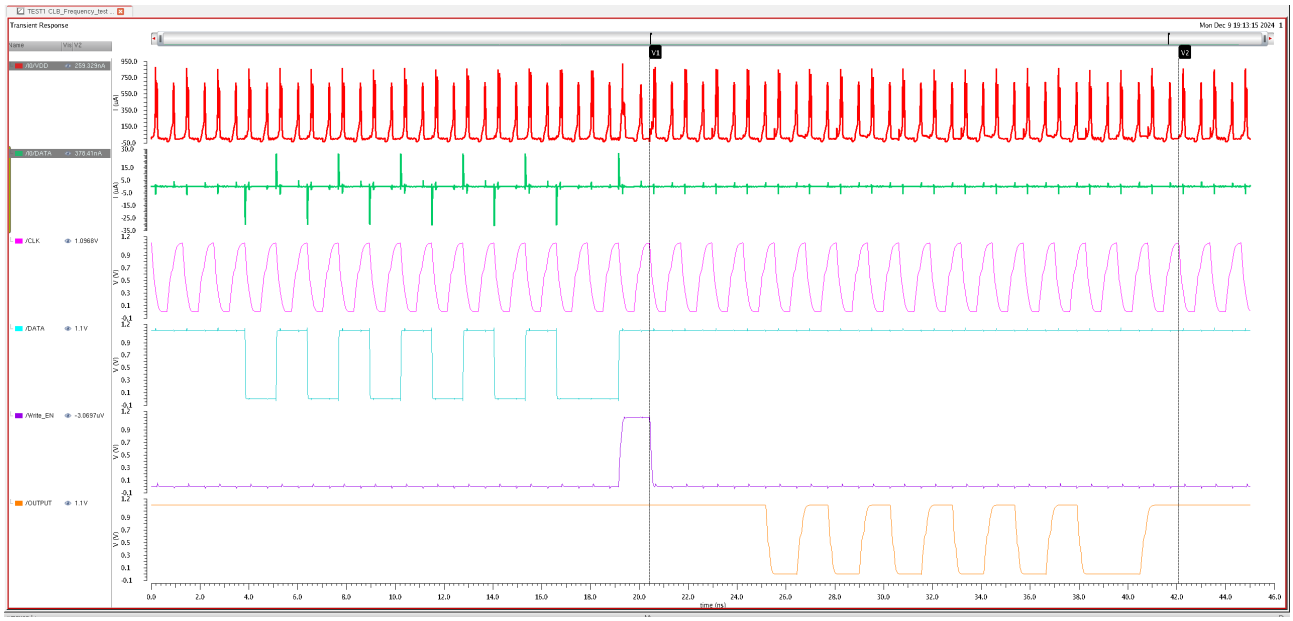


Figure 82: Current Graph

Assume 20% loading energy and 80% active energy estimate the average energy

6.5.1 Loading Energy

Measure the maximum energy for loading data into the SRAM.

- The loading energy is measured as the maximum energy consumed when the SIPO shift register writes a full set of 16 bits into the SRAM array.
- The energy is determined primarily by:
  - *Capacitive charging* of the bitlines to appropriate levels.
  - *Switching activity* of the transistors involved in the write operation.
- The loading energy can be calculated using the formula:

$$E_{load} = T_{write} \cdot I_{abs} \cdot V_{DD}$$

  where:
  - T_{write}: Time period taken to write the 16 bits into the SRAM array.
  - I_{avg}: Absolute current during the write operation.
  - V_{DD}: Supply voltage.

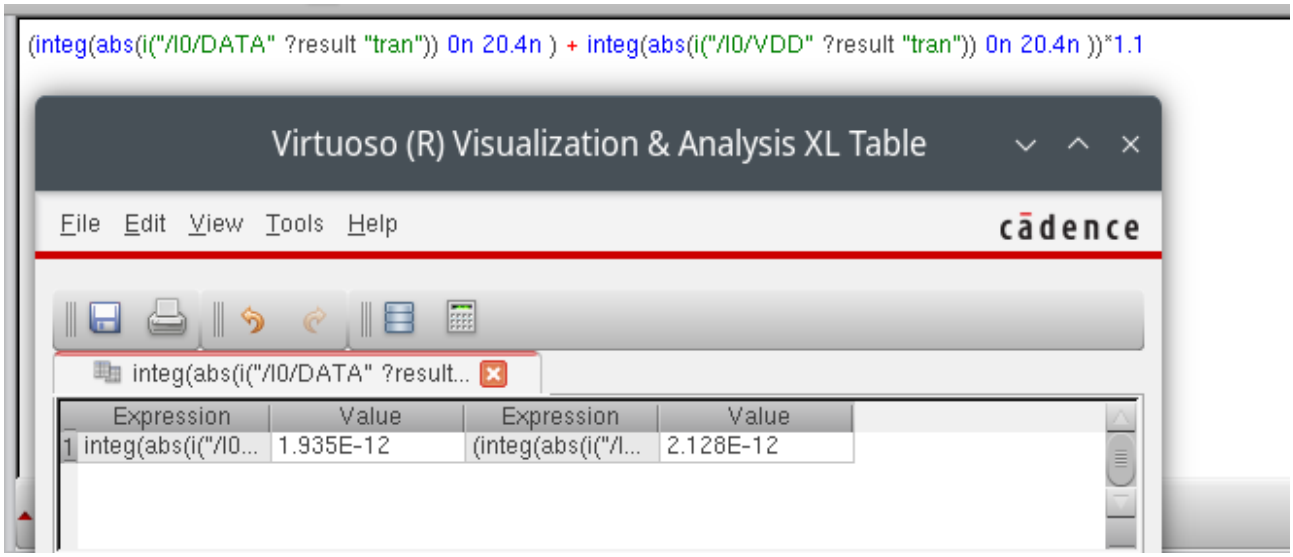


Figure 83: Loading Energy

For our design the loading period is 16 x clock period = 16 x 1.275ns = 20.4ns. Starting from 0 to 20.4ns The loading energy we get is 2.128 $\cdot 10^{-12} J$.

6.5.2 Active Energy

Measure the energy for cycling through all address bits from 0 to 15 at max operating frequency. Active energy refers to the energy consumed while cycling through all possible addresses of the SRAM (from 0 to 15) at the maximum operating frequency. This includes the dynamic switching of wordlines, bitlines, sense amplifiers, and address decoders.

Measurement:

- Active energy is measured when the LUT (Look-Up Table) is accessed sequentially for all 16 addresses.
- The process involves:
  - Activation of the *decoder circuit* to select the corresponding SRAM cell.
  - Reading the stored data from the bitlines via the *sense amplifier*.
  - Propagation of the output through the *D flip-flop*.
  - Switching activity of the address lines and wordlines.
- The active energy can be calculated using the formula:

$$E_{active} = T_{read} \cdot I_{abs} \cdot V_{DD}$$

  where:
  - T_{read}: Time period taken to sequentially read the entire data from the SRAM till the output through the D flip-flop.
  - I_{avg}: Absolute current during the read operation.
  - V_{DD}: Supply voltage.

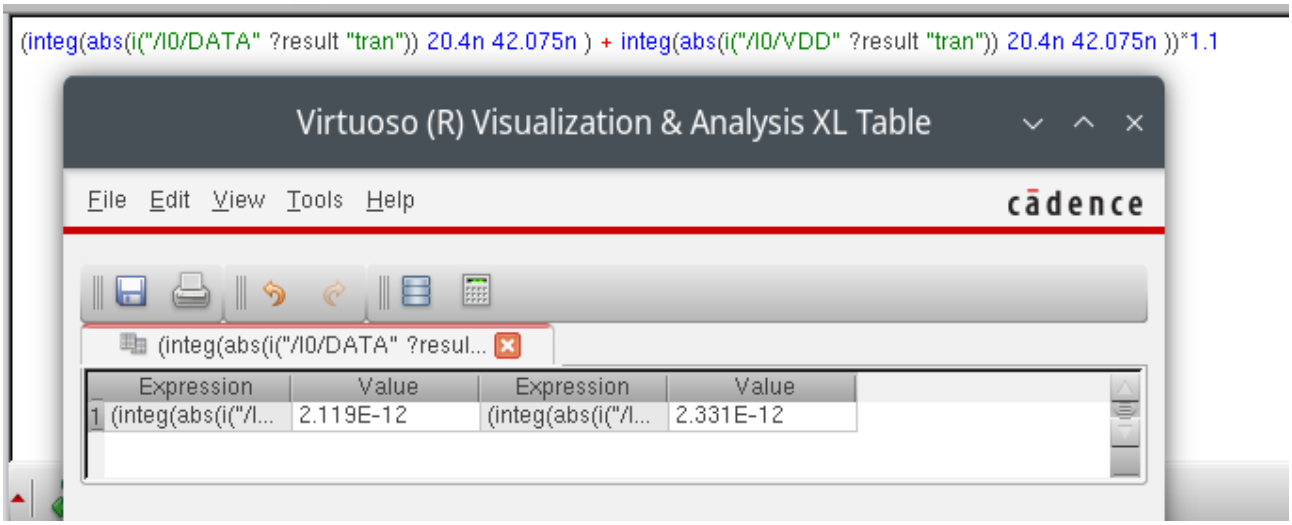


Figure 84: Active Energy

For our design the Active period is 17 x clock period = 17 x 1.275ns = 21.675ns. Starting from 20.4ns to 42.075ns

The active energy we get is 2.331 $\cdot 10^{-12} J$.

6.6 Average Energy

Assumption: The average energy is a weighted combination of loading energy and active energy, assuming that the system spends 20% of the time on loading and 80% of the time on active operations.

Formula:

$$E_{avg} = 0.2 \cdot E_{loading} + 0.8 \cdot E_{active}$$

$$E_{avg} = (0.2 \cdot 2.128 \cdot 10^{-12}) + (0.8 \cdot 2.331 \cdot 10^{-12})$$

$$E_{avg} = (0.2 \cdot 2.128 \cdot 10^{-12}) + (0.8 \cdot 2.331 \cdot 10^{-12})$$

$$E_{avg} = 0.4256 \cdot 10^{-12} + 1.8648 \cdot 10^{-12}$$

$$E_{avg} = 2.2904 \cdot 10^{-12}\, Joules$$

$$E_{avg} = 2.2904 \cdot 10^{-12}\, J$$

Observation

- **Loading energy:** Dominated by the write operation, which charges the bitlines and drives the wordlines for storing data.
- **Active energy:** Includes address decoding and data read operations, which involve cycling through all addresses and activating the sense amplifier.

6.7 Area of LUT and SRAM Array

Since our optimization method is to pull the inverter outside the MUX as shown in fiugure 54. The correct area calculation for our case needs to include the inverters outside the MUX as shown in figure 60. While for the SRAM cell we are using 6T as shown in figure 31. The updated formula listed below:

$$area_{cell} = \sum_{i \in Tx} W_i \quad and \quad area_{2:1mux} = \sum_{j \in Tx} W_j$$

$$area = 15 \cdot area_{2:1mux} + 16 \cdot area_{cell} + 4 \cdot area_{inv}$$

$$area_{2:1mux} = 2 \cdot 120\,nm = 240\,nm$$

$$area_{cell} = 6 \cdot 120\,nm = 720\,nm$$

$$area_{inv} = 2 \cdot 120\,nm = 240\,nm$$

$$area = 15 \cdot 240nm + 16 \cdot 720nm + 4 \cdot 240nm = 16080\,nm$$

Now, multiplying by length $45\,nm$ to get the final area:

$$area = 16080\,nm \cdot 45\,nm$$

$$area = 723600\,nm^2$$

Final Answer:

$$area = 723600\,nm^2$$

6.8 FOM

$$FOM = area \cdot averageEnergy \cdot \frac{1}{maxFrequency}$$

$$area = 723600 \cdot 10^{-18} m^2$$

$$E_{avg} = 2.2904 \cdot 10^{-12}\,J$$

$$F_{Max} = 784313725.49\,Hz$$

$$FOM = \left(723600 \cdot 10^{-18}\right) \cdot \left(2.2904 \cdot 10^{-12}\right) \cdot \frac{1}{784313725.49}$$

First, calculate the product of area and energy:

$$723600 \cdot 10^{-18} \cdot 2.2904 \cdot 10^{-12} = 1.655260544 \cdot 10^{-24}$$

Now divide by F_{Max}:

$$FOM = \frac{1.655260544 \cdot 10^{-24}}{784313725.49}$$

$$FOM = 2.111 \cdot 10^{-33}\, m^2 \cdot J \cdot s$$

Final Answer:

$$FOM = 2.111 \cdot 10^{-33}\, m^2 \cdot J \cdot s$$

In Ed Discuss it mentioned that the unit should be in unit of length m, this is to show that we understand the concept, and here stated the result without x45nm:

$$FOM \div (45 \cdot 10^{-9})\, m$$

$$Value = 2.111 \cdot 10^{-33} \div (45 \cdot 10^{-9}) = 4.6911 \cdot 10^{-26}\, m \cdot J \cdot s$$

6.9 Summary Table

The summary of key parameters highlights the importance of the maximum operating frequency, energy consumption, area, and figure of merit (FOM) in the design. The maximum frequency defines the circuit's speed, with higher values typically leading to better performance but potentially increased power consumption and area. Energy consumption is split into loading and active energy, with active energy being slightly higher, indicating more energy is used during computation. Average energy is a weighted sum of the two, emphasizing the higher power usage during computation. The area of the LUT and SRAM array reflects the physical space the design occupies, with larger areas potentially increasing costs and reducing integration efficiency. Finally, the FOM provides an overall measure of design efficiency, with lower values indicating better optimization across frequency, energy, and area.

Parameter	**Value**	**Equation**
Maximum Operating Frequency	784.313 MHz	$f_{max} = \frac{1}{Clockperiod} = \frac{1}{1.275\,ns} = 784.313\,MHz$
Loading Energy	2.128×10^{-12} Joules	$E_{load} = V_{DD} \cdot I_{load} \cdot t_{load} = 2.128 \times 10^{-12}\,J$
Active Energy	2.331×10^{-12} Joules	$E_{active} = V_{DD} \cdot I_{active} \cdot t_{active} = 2.331 \times 10^{-12}\,J$
Average Energy	2.2904×10^{-12} Joules	$E_{avg} = (E_{load} \times 0.2) + (E_{active} \times 0.8) = 2.2904 \times 10^{-12}\,J$
Area of LUT and SRAM Array	723600 nm^2	$A_{total} = A_{LUT} + A_{SRAM} = 723600\,nm^2$
Figure of Merit (FOM)	2.111×10^{-33} m^2·J·s	$FOM = \frac{A_{total} \cdot E_{avg}}{f_{max}} = 2.111 \times 10^{-33}\,m^2 \cdot J \cdot s$
LUT Worst Case Delay	1.08 ns	Obtained from graph

Table 1: Summary of key parameters with equations

7 Optimization Process

7.1 Step 1: Component-Level Optimizations

7.1.1 SIPO (Serial-In-Parallel-Out) Register

- **Initial Design:** Used 16 D flip-flops, each responsible for storing a single bit of data.
- **Optimization:**
  - Adapted D flip-flops from Homework 8.
  - Implemented negative-edge triggering to improve energy efficiency by reducing delay.

7.1.2 SRAM Design

- **Initial Design:** Standard 6T SRAM design.
- **Optimization:**
  - Selected 6T SRAM over 4T and 8T for a balance between speed, stability, and area.
  - Added a precharge module for bitline initialization to improve read/write reliability.
  - we tried increasing the size of the SRAM but it doesn't improve the performance and SRAM only counts towards less than 20% in the delay of the CLB.

7.1.3 Look-Up Table (LUT)

- **Baseline Design:** 16:1 multiplexer using pass-transistor logic with internal inverters.
- **Optimization:**
  - Externalized inverters in all MUX cell, allowing a single inversion to switch entire columns.
  - Optimized transistor width to 120 nm after parametric analysis for minimum delay.
  - The optimized LUT redced the dely by 3% and also reduced the area for the LUT.

7.2 Step 2: Subsystem Testing

7.2.1 SIPO Testing

- Tested with varying clock periods and data patterns to verify serial-to-parallel conversion.

7.2.2 SRAM Testing

- Tested read/write integrity using alternating data values.
- Validated bitline precharge operation to ensure data stability.

7.2.3 LUT Testing

- Verified correct output for all truth table combinations using sequential select signals (S_0 to S_3).

7.3 Step 3: Integration into the CLB

7.3.1 Functional Design

- Integrated the SIPO, SRAM array, and LUT into the CLB.
- Added a buffer to restore any threshold voltage drop.
- Included a D flip-flop for sequential logic operations, using a falling-edge trigger to improve timing.

7.3.2 CLB Testing

- Verified complete functionality with test patterns (1110101010101001) through the SIPO, SRAM, and LUT.

7.4 Step 4: Performance Metrics

7.4.1 Maximum Frequency

- Conducted parametric frequency analysis to determine the maximum operating frequency:

$$f_{max} = 784.313\,MHz$$

- Verified the CLB outputs remained stable at this frequency.

7.4.2 Energy Measurements

- **Loading Energy:** $E_{load} = 2.128 \times 10^{-12}\,J$.
- **Active Energy:** $E_{active} = 2.331 \times 10^{-12}\,J$.
- **Average Energy:**

$$E_{avg} = 0.2 \cdot E_{load} + 0.8 \cdot E_{active} = 2.2904 \times 10^{-12}\,J$$

7.5 Area Optimization

- Calculated total area as:

$$Area = 15 \cdot area_{2:1mux} + 16 \cdot area_{SRAM} + 4 \cdot area_{inv}$$

Result:

$$Area = 723600\,nm^2$$

7.6 Step 6: Figure of Merit (FOM)

- Defined as:
$$FOM = Area \cdot AverageEnergy \cdot \frac{1}{MaxFrequency}$$
- Calculated as:
$$FOM = 2.111 \times 10^{-33}\, m^2 \cdot J \cdot s$$

7.7 Conclusion

- **Improvements Achieved:**
  - Reduced delay by 3.57% through transistor width optimization.
  - Reduced energy consumption by balancing loading and active energy contributions.
  - Minimized area by redesigning the LUT with external inverters.
- This systematic process optimized the CLB for performance, energy efficiency, and compactness.