University of Pennsylvania
Department of Electrical and System Engineering

ESE 5700 - Digital Integrated Circuits and VLSI Fundamentals

Project: Configurable Logic Block

Submitted by –
Adithya Rajeev (adithyar)
Deepak Shankar (deesh)

Github link -
https://github.com/adithyarUpenn/ConfigurableLogicBlock

Contents

Preface 8
Introduction 10
What is a Configurable Logic Block (CLB) 10
2x1 Multiplexer (2x1 MUX) 12
Design Schematic 14
Symbol 15
Testing Circuit 15
Simulation Plot 16
16x1 Multiplexer (16x1 MUX) 17
Design Schematic 18
Symbol 19
Testing Circuit 19
Simulation Plot 21
Worst case delay 21
Parametric Sweep 22
Delay plot 22
Clock 24
Schematic 25
Symbol 25
Testing Circuit 26
Simulation Plot 26
Layout 27
DRC Proof 27
D Flip flop 28
Design Schematic 29
Symbol 30
Testing Circuit 30
Simulation Plot 31
Layout 31
DRC Proof 32
Serial In Parallel Out (SIPO) Register 33

Design Schematic 34
Symbol 34
Testing Circuit 35
Simulation Plot 36
Layout 36
DRC Proof 37
Static Random Access Memory (SRAM) 38
Design Schematic 39
Symbol 40
Testing Circuit 41
Storing 1 41
Storing 0 41
Simulation Plot 42
Storing 1 42
Storing 0 42
Layout 43
DRC Proof 43
SRAM Array 44
Design Schematic 44
Symbol 45
Testing Circuit 46
Simulation Plot 46
Layout 47
DRC Proof 47
Configurable Logic Block (CLB) 48
Design Schematic 48
Symbol 49
Testing Circuit 50
Simulation Plot 50
Layout 51
DRC Proof 52
Design Metrics 53

Energy 53
Loading Energy 53
Active Energy 55
Average Energy 56
Maximum Operating Frequency 56
Figure Of Merit (FOM) 57
Summary Table 57
References 58

Figures

Figure 1 Design 8
Figure 2 2x1 Symbol 12
Figure 3 2x1 Truth Table 12
Figure 4 2x1 MUX using Pass Transistor Logic 13
Figure 5 2x1 MUX Schematic 14
Figure 6 2x1 MUX Symbol 15
Figure 7 2x1 MUX Test Schematic 15
Figure 8 2x1 MUX Plot 16
Figure 9 16x1 MUX using 2x1 MUX 17
Figure 10 16x1 MUX Truth Table 18
Figure 11 16x1 MUX Schematic 18
Figure 12 16x1 MUX Symbol 19
Figure 13 16x1MUX_TestSchematic 20
Figure 14 16x1 MUX Plot 21
Figure 15 16x1 MUX Parametric Sweep 22
Figure 16 16x1 MUX Delay plot 22
Figure 17 Clock Logic Diagram 24
Figure 18 Clock Schematic 25
Figure 19 Clock Symbol 25
Figure 20 Clock Test Schematic 26
Figure 21 Clock Plot 26
Figure 22 Clock Layout 27
Figure 23 Clock DRC Proof 27
Figure 24 D Flip flop Logic Diagram 28
Figure 25 D Flip flop Truth Table 28
Figure 26 D Flip flop using Pass Transistor Logic 29
Figure 27 D Flip flop Schematic 29
Figure 28 D Flip flop symbol 30
Figure 29 D Flipflop Test Schematic 30
Figure 30 D Flipflop Plot 31
Figure 31 D Flipflop Layout 31
Figure 32 D Flipflop DRC Proof 32
Figure 33 SIPO Block Diagram 33
Figure 34 SIPO Schematic 34
Figure 35 SIPO Symbol 34
Figure 36 SIPO Test Schematic 35
Figure 37 SIPO Plot 36
Figure 38 SIPO Layout 36
Figure 39 SIPO DRC Proof 37
Figure 40 SRAM Block Diagram 38

Figure 41 SRAM Schematic 39
Figure 42 SRAM Symbol 40
Figure 43 SRAM Test Schematic storing 1 41
Figure 44 SRAM Test Schematic Storing 0 41
Figure 45 SRAM for Storing 1 42
Figure 46 SRAM for Storing 0 42
Figure 47 SRAM Layout 43
Figure 48 SRAM DRC Proof 43
Figure 49 SRAM Array Schematic 44
Figure 50 SRAM Array Symbol 45
Figure 51 SRAM Array Test Schematic 46
Figure 52 SRAM Array Plot 46
Figure 53 SRAM Array Layout 47
Figure 54 SRAM Array DRC Proof 47
Figure 55 CLB Schematic 48
Figure 56 CLB Symbol 49
Figure 57 CLB Test Schematic 50
Figure 58 CLB Plot 50
Figure 59 CLB Layout 51
Figure 60 CLB DRC Proof 52
Figure 61 CLB Loading Energy Test Schematic 53
Figure 62 Loading Energy Plot 54
Figure 63 Loading Energy Calculation 54
Figure 64 CLB Active Energy Test Schematic 55
Figure 65 CLB Active Energy Plot 55
Figure 66 Current Integral Values 56

Preface

Design Problem:
To design the circuit-level implementation of a Configurable Logic Block (CLB) comprised of a single Basic Logic Element (BLE) using a Look-up Table (LUT).

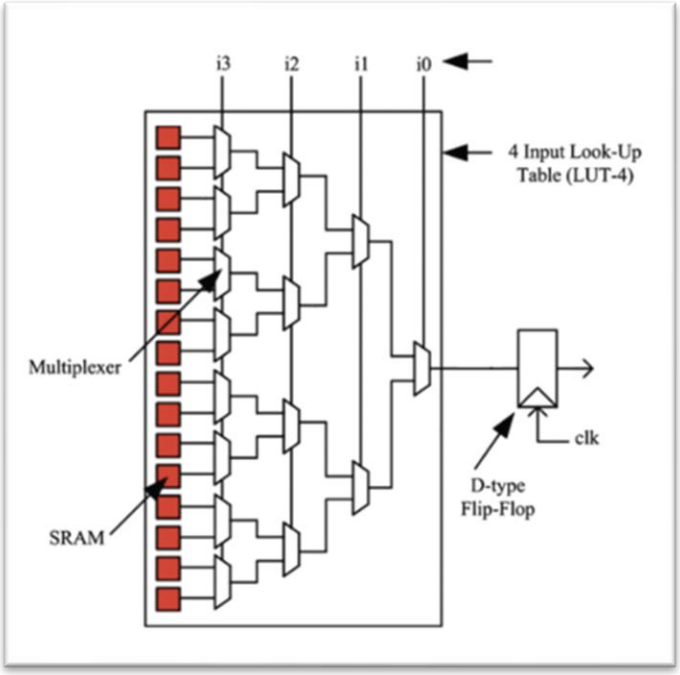


Figure 1 Design

The aim is to create a single design of a CLB with inputs –

- Clock - Clk
- Load (high when loading SRAM data)
- Data (serial input of bitstream to load into SRAM)
- LUT select <3:0>
- Output: Out

The CLB in an FPGA holds a custom truth table, which is loaded when the chip is powered up and implements the functionality of a Multiplexer (MUX). The CLB should be designed along with the control logic to load the SRAM cells with the desired truth table. The LUT inputs act as the address lines for a corresponding one-bit-wide RAM cell.

The LUT will have the following inputs –

- data<15:0>
- addr< 3:0>
- output: Out.

When an FPGA is configured, the data bits of the LUT are loaded with ones or zeros based on what the desired truth table would be.

Design Metrics:
The design must be measured with the following design metrics -

- **Frequency**: Maximum frequency of input switching which results in correct operation of the design. This should be related to your worst-case delay of the LUT and the flip-flop timing constraints. Load output will be 100 times the minimum size gate capacitance (100Cg)
- **Average energy**: Assume 20% loading energy and 80% active energy
    - **Loading energy**: The maximum energy for loading data into the SRAM.
    - **Active energy**: The energy for cycling through all address bits from 0 to 15 at max operating frequency.
- **Area**: Layout a single SRAM cell and single 2x1 sub-MUX. Include your DRC and LVS results. Report the area as area = 15 · areaMUX + 16 · areacell
- Figure Of Merit (FOM): Report the FOM = area * averageEnergy / maxFrequency

Introduction

What is a Configurable Logic Block (CLB) –

A Configurable Logic Block (CLB) stands as a foundational unit in the architecture of Field-Programmable Gate Arrays (FPGAs), playing a central role in the development of digital circuits. Its importance lies in providing designers with a dynamic and versatile platform to configure and tailor digital logic, making FPGAs well-suited for a wide range of applications.

Components of a Configurable Logic Block (CLB):

- **Look-Up Tables (LUTs):**

At the heart of the CLB is the Look-Up Table (LUT), a compact and configurable memory unit that houses truth tables for various logic functions. The LUT's size, determined by the number of inputs, allows for the implementation of diverse combinational logic functions.

2. Flip-Flops or Registers:
In support of sequential logic and data storage, CLBs typically integrate flip-flops or registers. These elements facilitate the creation of state machines and enable the synchronized storage of signals within the design.

3. Multiplexers and Routing Resources:
CLBs encompass programmable multiplexers and routing resources, facilitating the flexible interconnection of signals. This capability empowers designers to route signals between different logic elements within the CLB and interconnect multiple CLBs, enhancing overall configurability.

4. Carry Chains:
To efficiently handle arithmetic functions, CLBs may incorporate resources such as carry chains. These prove invaluable for tasks like addition and multiplication, contributing to the FPGA's capability to execute diverse computational operations.

5. Cascading and Carry Logic:
CLBs typically offer resources for cascading and managing carry logic, enabling seamless connections between multiple CLBs to implement more intricate functions. This ensures the efficient handling of carry signals, particularly in arithmetic operations.

6. Additional Features:
Tailored to specific FPGA architectures, CLBs may provide additional features, such as specialized processing units, memory blocks, or application-specific resources. These augment the CLB's versatility, addressing a broad spectrum of design requirements.

Design Flow:
In FPGA design, designers commonly utilize Hardware Description Languages (HDL) such as Verilog or VHDL to articulate the digital logic. Synthesis tools subsequently translate this description onto the CLBs and routing resources of the FPGA. This procedural approach empowers designers to craft bespoke digital designs without the need for custom silicon.

To summarize, the Configurable Logic Block is a pivotal element in FPGA architecture, offering designers a flexible and adaptable platform for digital circuit implementation. Its modular and configurable nature facilitates the creation of diverse and customized designs, positioning FPGAs as powerful solutions across applications, from digital signal processing to embedded systems.

This report demonstrates the design of a simple Configurable Logic Block (CLB) using the following basic building blocks –

- Clock
- 16-bit SIPO
- SRAM
- 16-bit Look-up Table (LUT)
- D Flip-flop

The report will summarize the functionality of each block and provide simulation proofs for the same.

2x1 Multiplexer (2x1 MUX)

At the heart of the 16 bit Look-up Table (LUT) is the 2x1 MUX which is cascaded to design a 16x1 Multiplexer which acts as the LUT.

Multiplexer in digital circuit has many inputs (m) with only one output and it is denoted by m:1 MUX where m is the number of inputs. It selects only one input (one input is connected to the output) based on the select line. We can have 2X1 MUX with 1 select line, 4X1 MUX with 2 select line, 8x1 MUX with 3 select lines, 16X1 MUX with 4 select lines and so on.

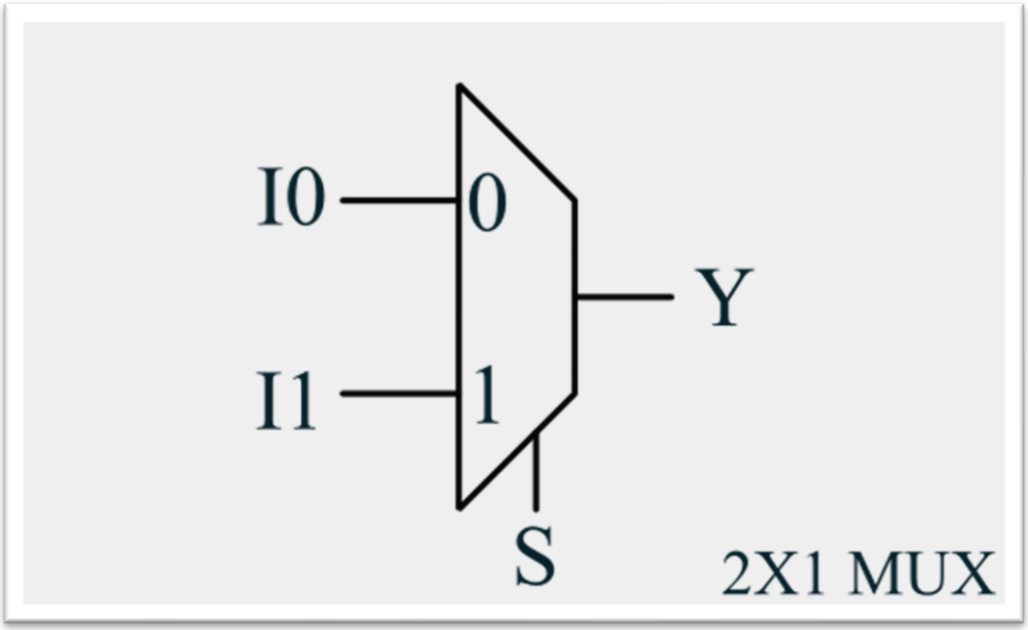


Figure 2 2x1 Symbol

S	I0	I1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Figure 3 2x1 Truth Table

A 2x1 MUX can be represent using the following equation –

$$Y = \bar{S}I0 + SI1$$

In this design, we have used Pass Transistor Logic to implement a 2x1 MUX. Pass Transistor design of a MUX uses lesser Transistors compred to the other MUX design which reduces the area required significantly as the design get bigger and uses several 2x1 MUX.

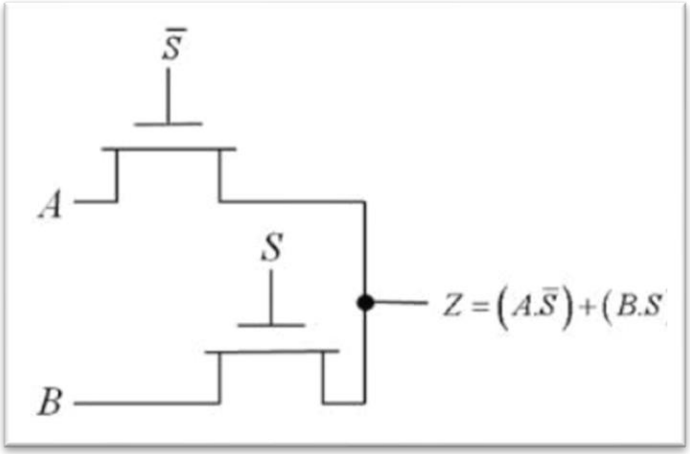


Figure 4 2x1 MUX using Pass Transistor Logic

Advantages of the Pass Transistor design –

Minimized Signal Delay
Pass transistors exhibit lower ON resistance compared to traditional NAND or NOR gates, resulting in decreased signal delay as the signal traverses through with reduced impedance.

Enhanced Signal Integrity
The low ON resistance of pass transistors, when activated, establishes a path with minimal impedance between input and output, elevating overall signal integrity. This characteristic is especially advantageous in applications with high-speed or high-frequency requirements.

Space-Efficient Design
Pass transistor implementations often demonstrate superior space efficiency when contrasted with equivalent designs employing multiple gates. This efficiency stems from a potential reduction in the number of transistors, contributing to a more compact overall design.

Simplified Circuitry
Designs based on pass transistors tend to simplify overall circuitry compared to alternatives utilizing various gates. This simplicity in design can facilitate the layout process and potentially lead to a reduction in power consumption.

Diminished Power Consumption:
Pass transistors, when in the OFF state, typically consume minimal power. This differs from static gates, which may draw power even without signal transitions. In scenarios emphasizing power efficiency, pass transistors may emerge as a favorable choice.

Design Schematic

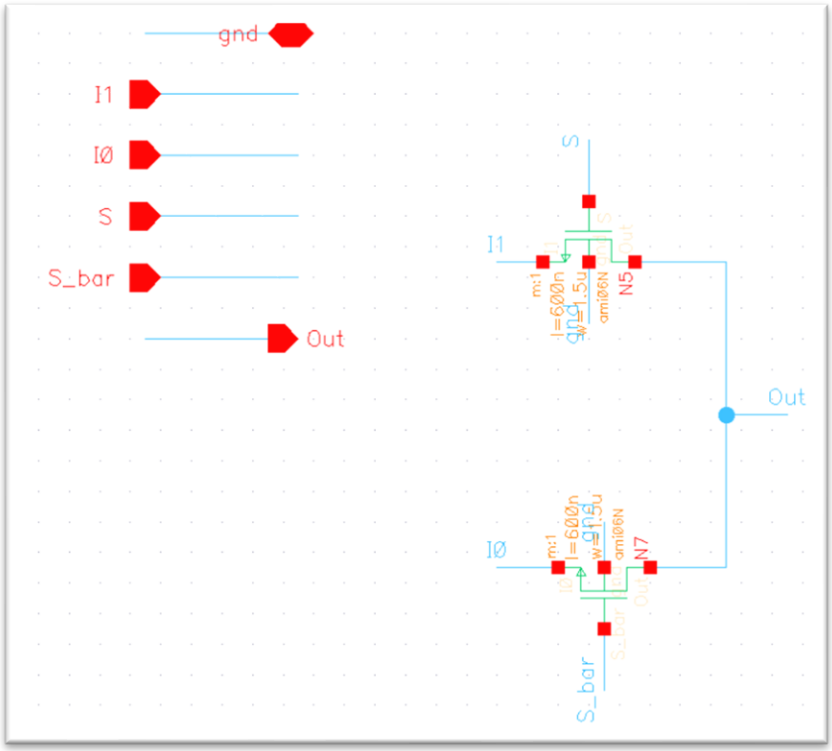


Figure 5 2x1 MUX Schematic

We eliminated the inverter from the select line of the 2x1 MUX and shared it across the select lines for each of the four stages in our 16x1 MUX design. This modification allows us to reduce the number of transistors in our 16x1 MUX. As a result, our foundational design is now more efficiently optimized in terms of area, offering improvements over the alternative of having an inverter in each of the 2x1 MUX units.

Symbol

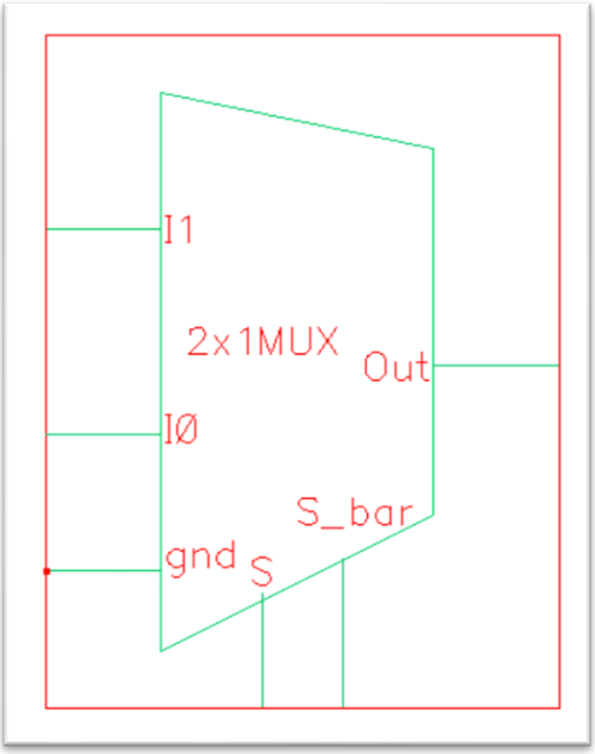


Figure 6 2x1 MUX Symbol

Testing Circuit

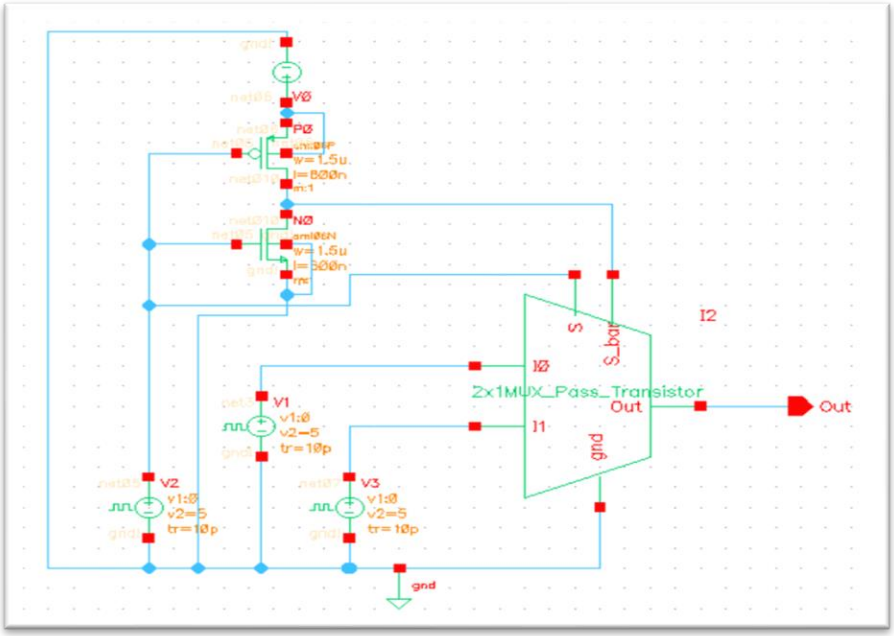


Figure 7 2x1 MUX Test Schematic

The test schematic was configured to validate the functionality of our 2x1 MUX. To assess our design, we systematically cycled through the address bit, causing the output to track the input according to the state of the address bit. Two distinct Pulse sources are supplying the inputs, with one having a period that is twice that of the other. This arrangement enables us to thoroughly test various scenarios and cases.

Simulation Plot

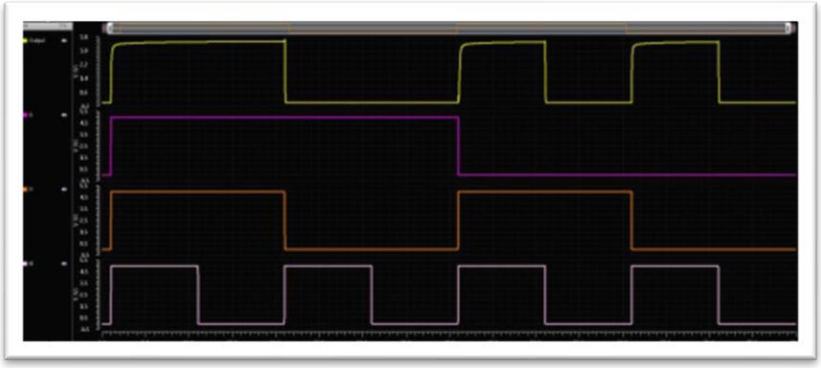


Figure 8 2x1 MUX Plot

16x1 Multiplexer (16x1 MUX)

A 16x1 Multiplexer (MUX) is a digital circuit that selects one of 16 input lines and directs it to the output based on a control signal. The term "16x1" denotes that there are 16 inputs and one output. The control lines (S0 to S3) determine which input is selected. The binary value represented by these control lines determines the selected input.

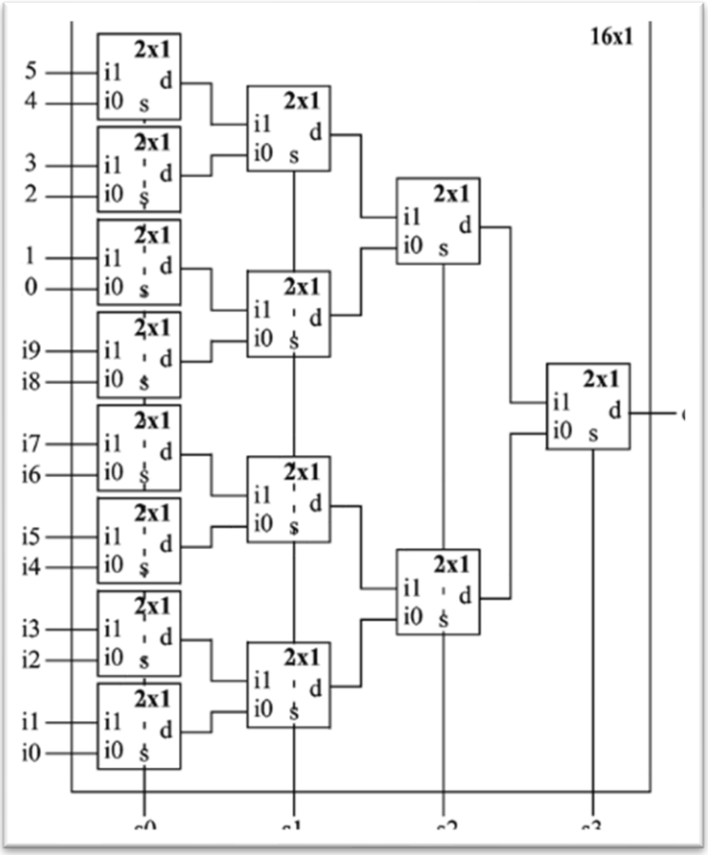


Figure 9 16x1 MUX using 2x1 MUX

In our design of the 16x1 MUX, an inverter has been applied to the select lines in each of the four stages. This decision was made to circumvent the necessity of having inverters in each of the 2x1 MUX units, where such inverters would have been redundant within the context of the 16x1 MUX design.

Selection Input				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5
0	1	1	0	I_6
0	1	1	1	I_7
1	0	0	0	I_8
1	0	0	1	I_9
1	0	1	0	I_{10}
1	0	1	1	I_{11}
1	1	0	0	I_{12}
1	1	0	1	I_{13}
1	1	1	0	I_{14}
1	1	1	1	I_{15}

Figure 10 16x1 MUX Truth Table

Design Schematic

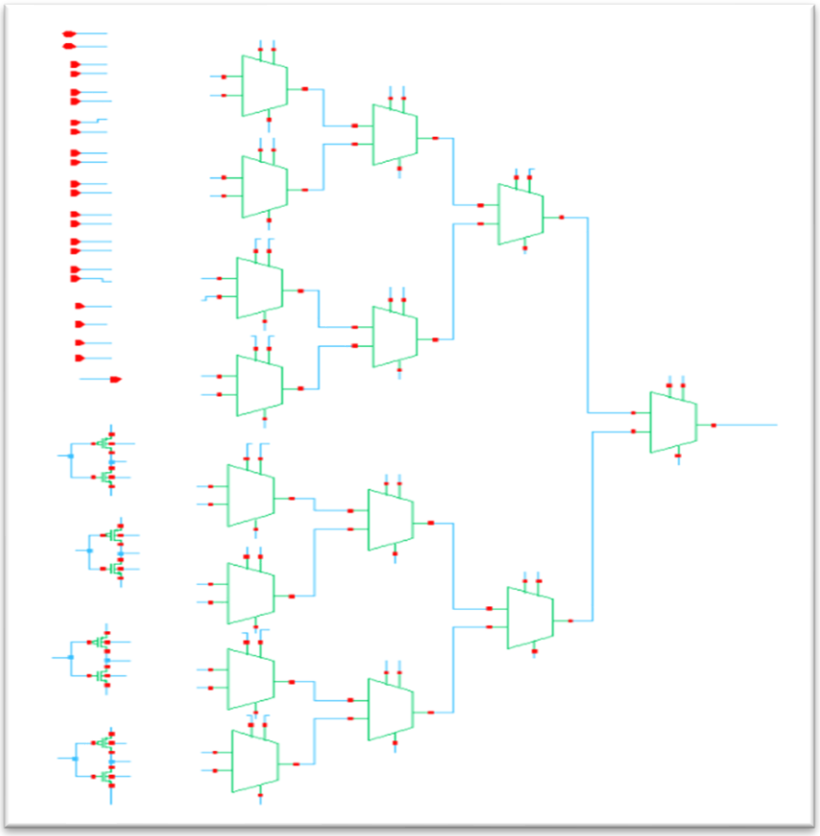


Figure 11 16x1 MUX Schematic

Symbol

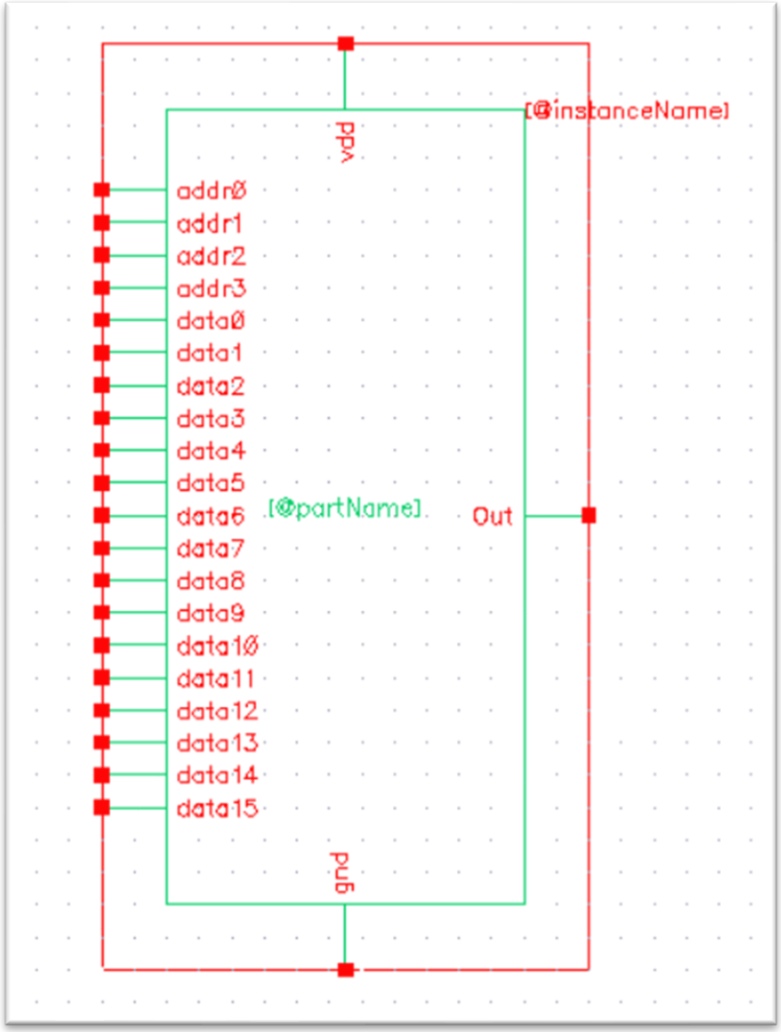


Figure 12 16x1 MUX Symbol

Testing Circuit

In the test schematic, we have arranged a setup to validate the functionality of our 16x1 MUX design. To achieve this, we've connected the alternate bits from I15 to I0 to a 5V supply. This configuration allows us to systematically test switching input cases and ascertain whether our design aligns with all possible input scenarios.

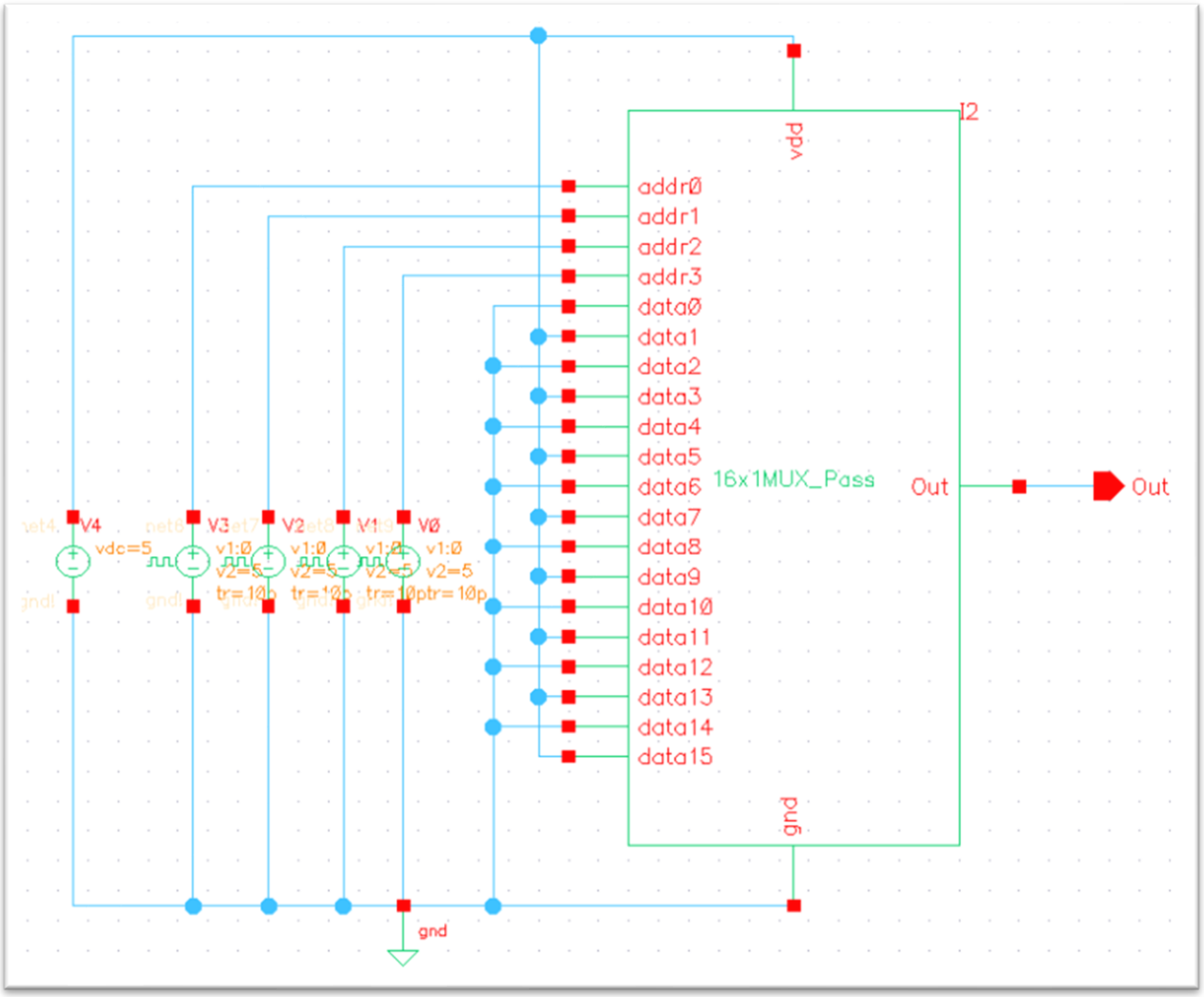


Figure 13 16x1MUX_TestSchematic

Simulation Plot

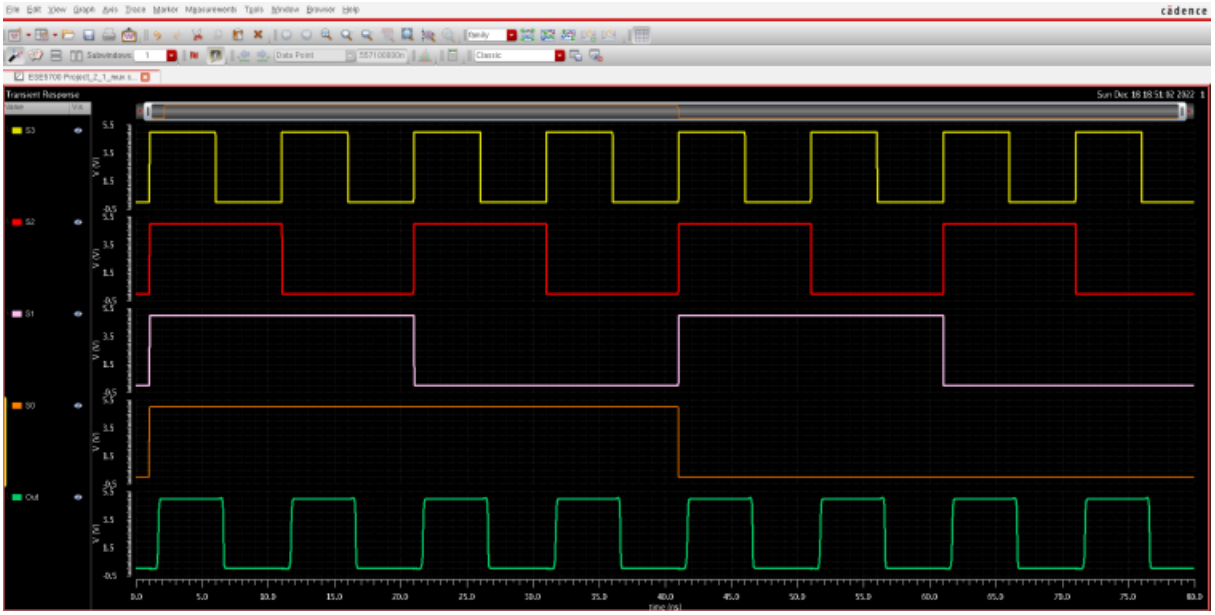


Figure 14 16x1 MUX Plot

Worst case delay

We measure the delay between the S3 select line and the output, pre-load. In our configuration, the S3 select line (corresponding to the leftmost stage) serves as the least significant bit (LSB), while the S0 select line (for the rightmost stage) functions as the most significant bit (MSB). We've identified a scenario that contributes to the maximum worst-case delay: when S3 is transitioning, and the other three select lines are maintained at a logical high (VDD).

This situation results in the output toggling between In14 and In15, given that the select lines are either 1110 (In14) or 1111 (In15). We intentionally switch S3 because this select line drives the largest number of 2x1 MUXs, leading to a higher parasitic capacitance compared to the other three stages. Consequently, it makes sense to switch the select line (S3) associated with the highest load and capacitance.

All select lines receive their inputs through inverters, and the output pin of the 2x1 MUX is connected to In15 of the 16x1 MUX. This configuration allows us to assess the delay, taking into account the specific conditions that contribute to the maximum delay scenario.

Parametric Sweep

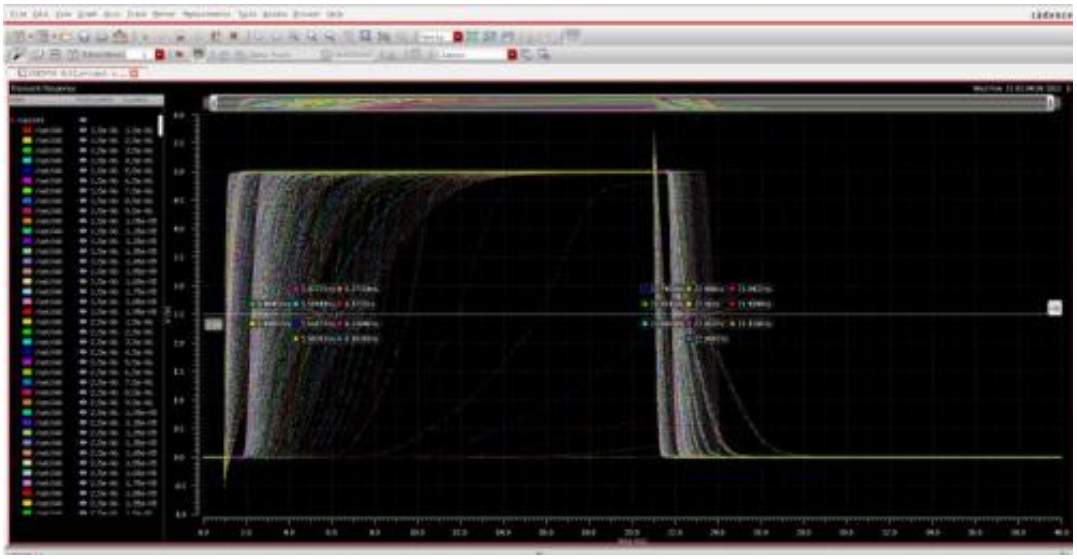


Figure 15 16x1 MUX Parametric Sweep

Delay plot

Figure 16 16x1 MUX Delay plot

τP LH = 2.3581ns − 1.2459ns = 1.1122ns
τPHL = 21.9177ns − 21.1171ns = 0.8006ns
So, the worst-case delay comes out to be 1.1122ns.

We carefully determined the optimized dimensions as follows:

PMOS Width: 12.5 µm
NMOS Width: 11.5 µm

For the inverter steering our 16x1 multiplexer in the scenario with the maximum delay, as well as the transistors for the 2x1 multiplexer and the buffer employed for restoration just before the load, the dimensions align with those specified above. To ensure consistency, the inverters propelling the select lines of our Device Under Test (DUT) must possess the same drive strength as a 2x1 multiplexer. Similarly, the buffer used for restoration must be sized identically, considering it is driven by a single 2x1 multiplexer.

The transistors within the load inverters are dimensioned to function as a 100Cg load.

PMOS Width in the load inverter: 70 µm
NMOS Width in the load inverter: 70 µm

While it was possible to further decrease the worst-case delay by enlarging the PMOS and NMOS sizes, we opted against exceeding a certain point to avoid an overall increase in the total area. Therefore, we decided to maintain the dimensions mentioned above for our design.

Clock

In our design, we have using two phase non-overlapping clocks for synchronization of the CLB.
A two-phase non-overlapping clock consists of two complementary clock signals with non-overlapping transitions. In this context, "non-overlapping" means that the rising edge of one clock signal occurs precisely when the falling edge of the other clock signal occurs, and vice versa. This arrangement helps prevent any overlap in the active state of the two clock signals.

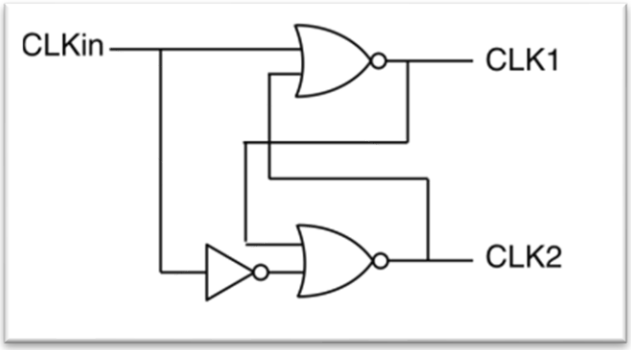


Figure 17 Clock Logic Diagram

The duty cycle of each individual clock signal (CLK1 or CLK2) is 50%, but when considered together, the overall duty cycle is less than 50% due to the non-overlapping nature.
The non-overlapping characteristic is useful in controlling the timing of digital circuits, particularly in scenarios where avoiding simultaneous switching of certain components is critical.

Schematic

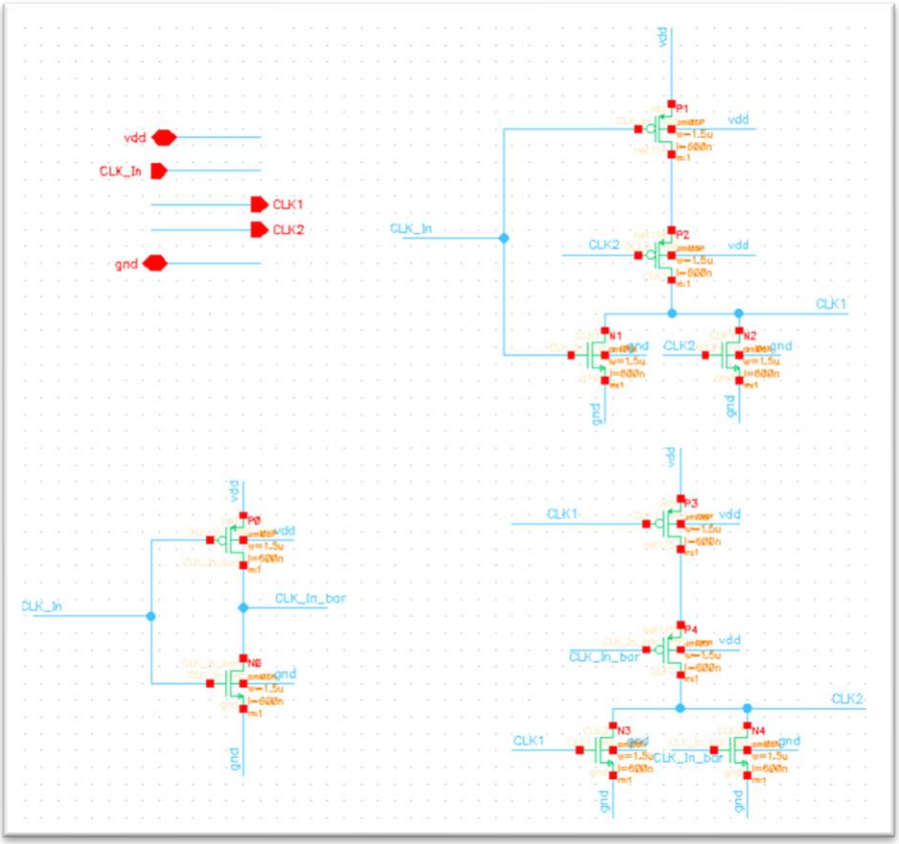


Figure 18 Clock Schematic

Symbol

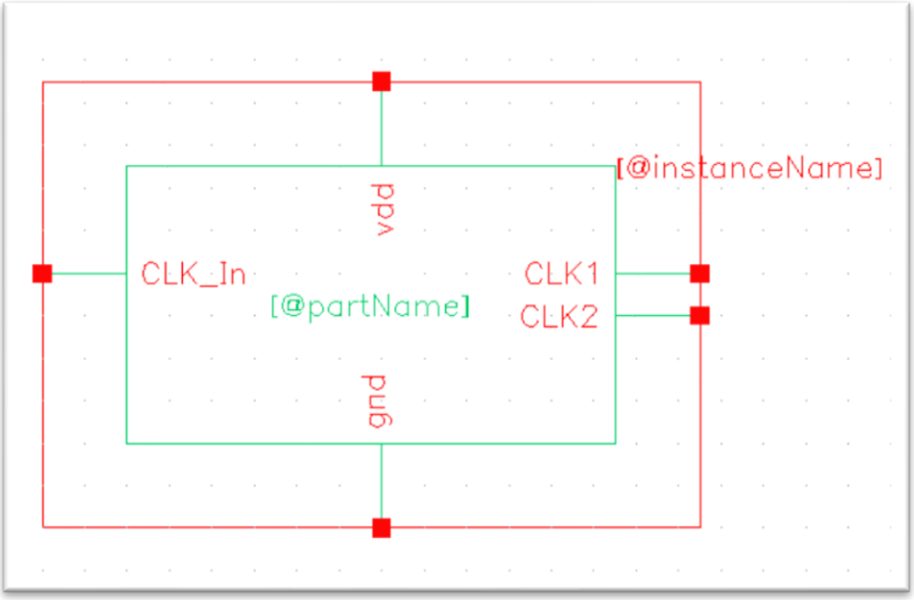


Figure 19 Clock Symbol

Testing Circuit

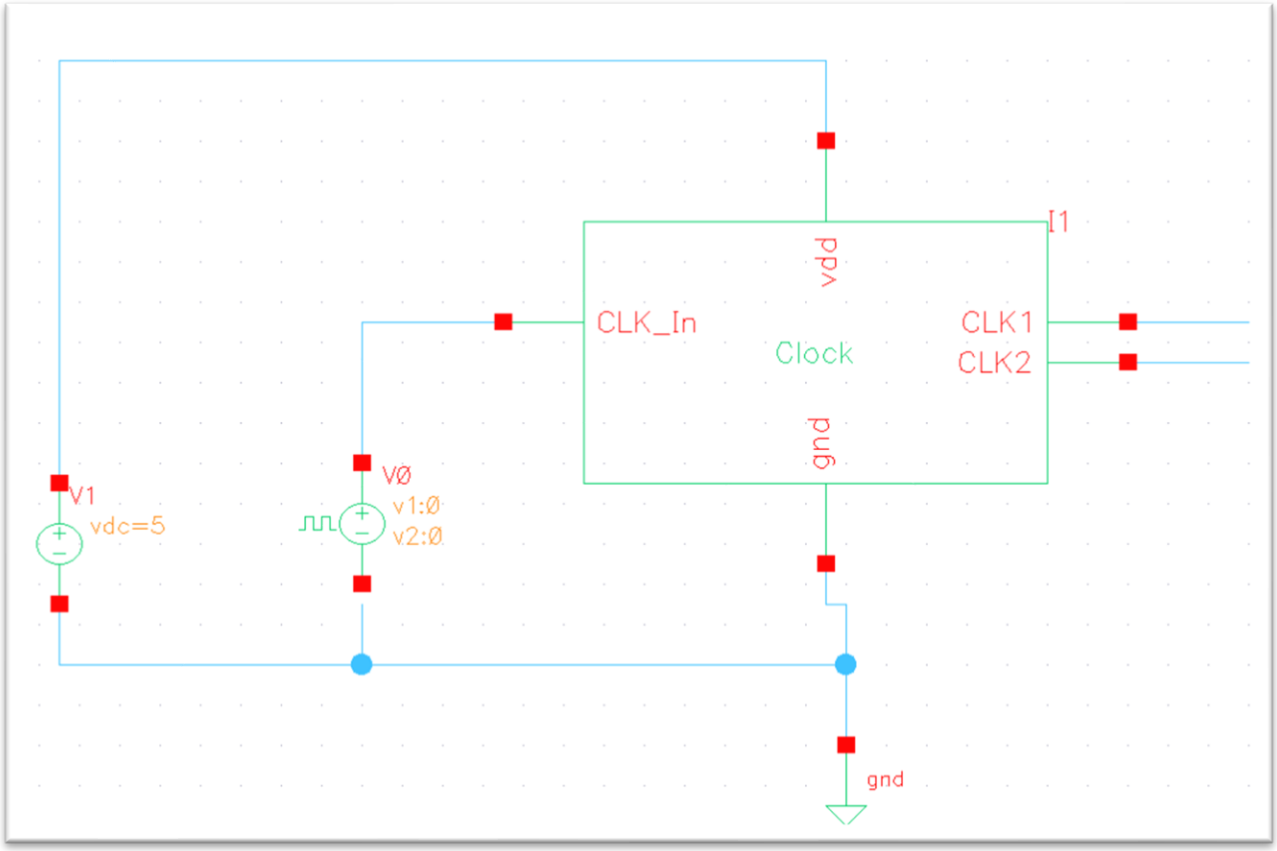


Figure 20 Clock Test Schematic

Simulation Plot

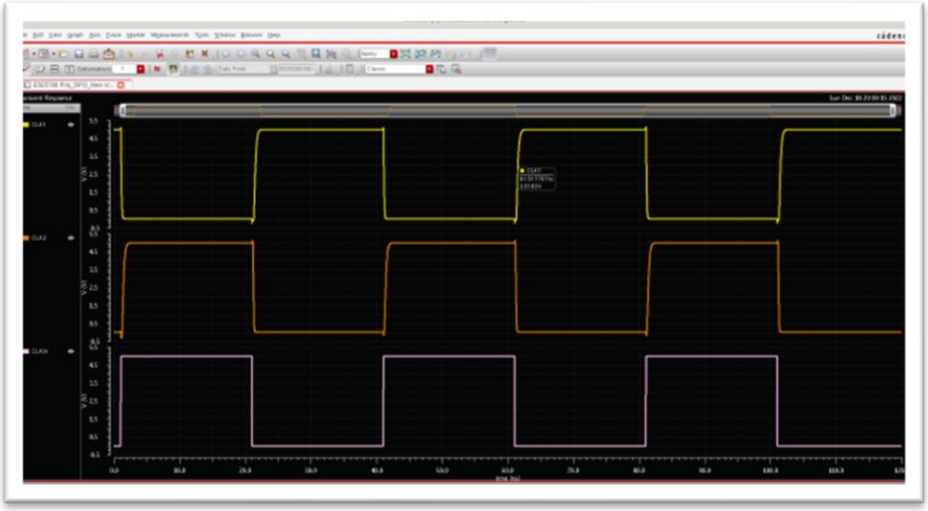


Figure 21 Clock Plot

Layout

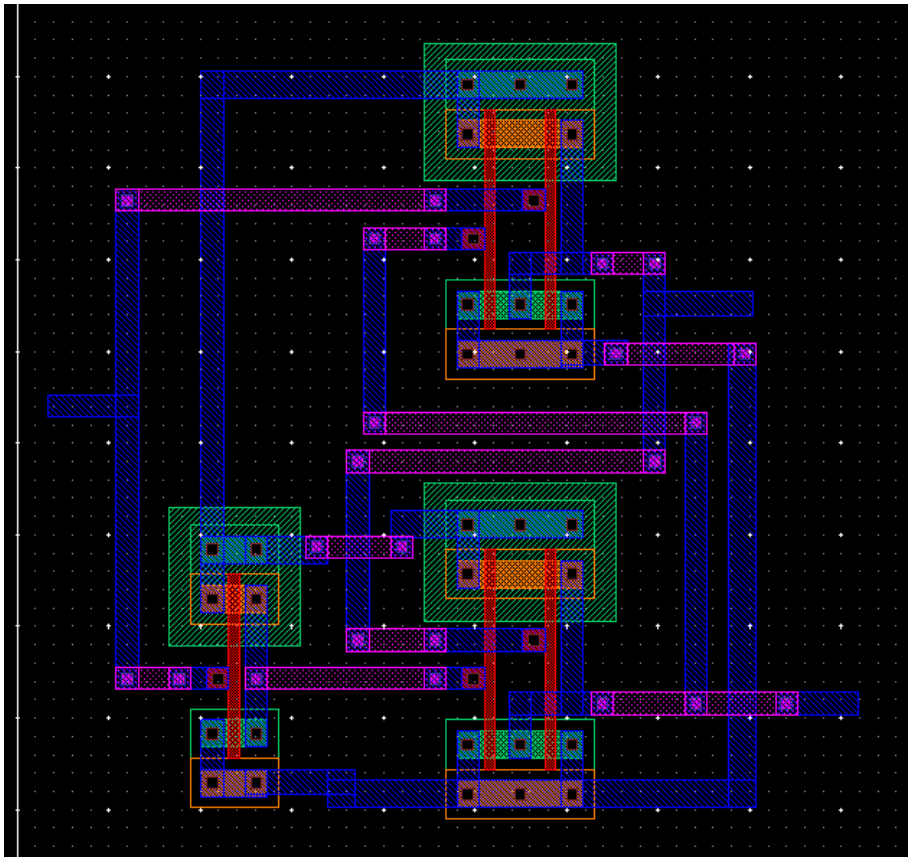


Figure 22 Clock Layout

DRC Proof

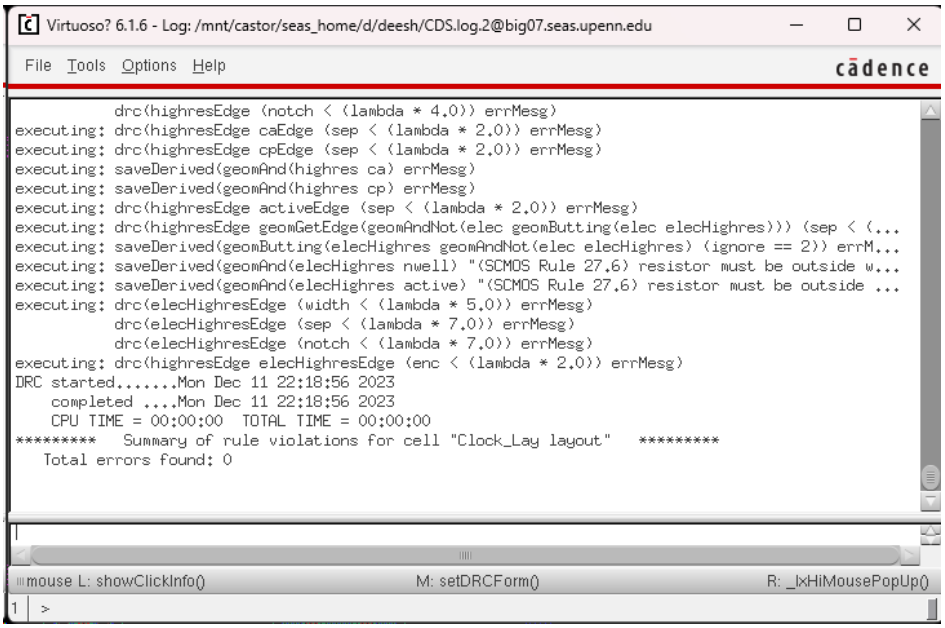


Figure 23 Clock DRC Proof

D Flip flop

A D flip-flop, short for Data or Delay flip-flop, is a type of digital storage element in digital circuits. It is a bistable multivibrator, meaning it can exist in one of two stable states—either high or low. The D flip-flop is widely used in digital systems for its ability to store a single bit of data and transfer it to the output based on a clock signal.

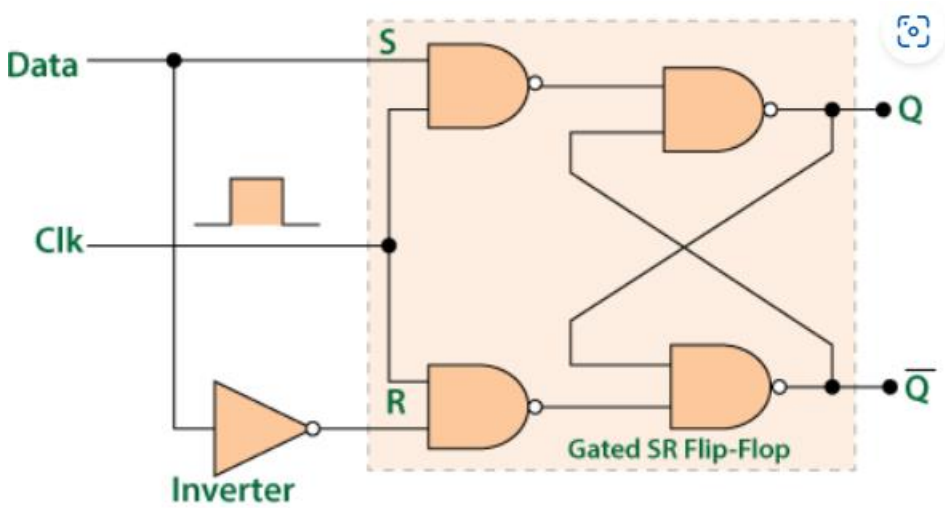


Figure 24 D Flip flop Logic Diagram

CLK	D	Q	Q'
↑	0	0	1
↑	1	1	0

Figure 25 D Flip flop Truth Table

We have designed a Positive edge triggered D Flip Flop using Pass transistors and inverters. We went with this design because Pass-transistor-based flip-flops often have lower power consumption compared to other flip-flop types.

This D flip-flop follows the Master-Slave configuration, and all the transistors utilized in its design are minimized in size.

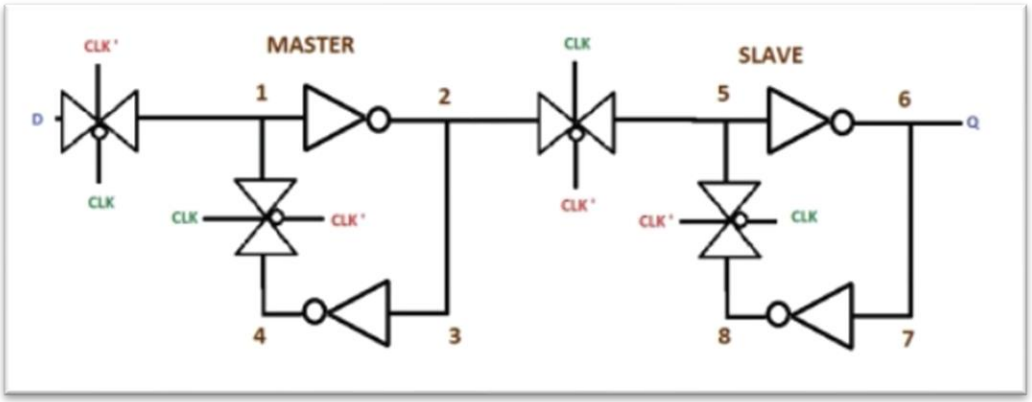


Figure 26 D Flip flop using Pass Transistor Logic

Design Schematic

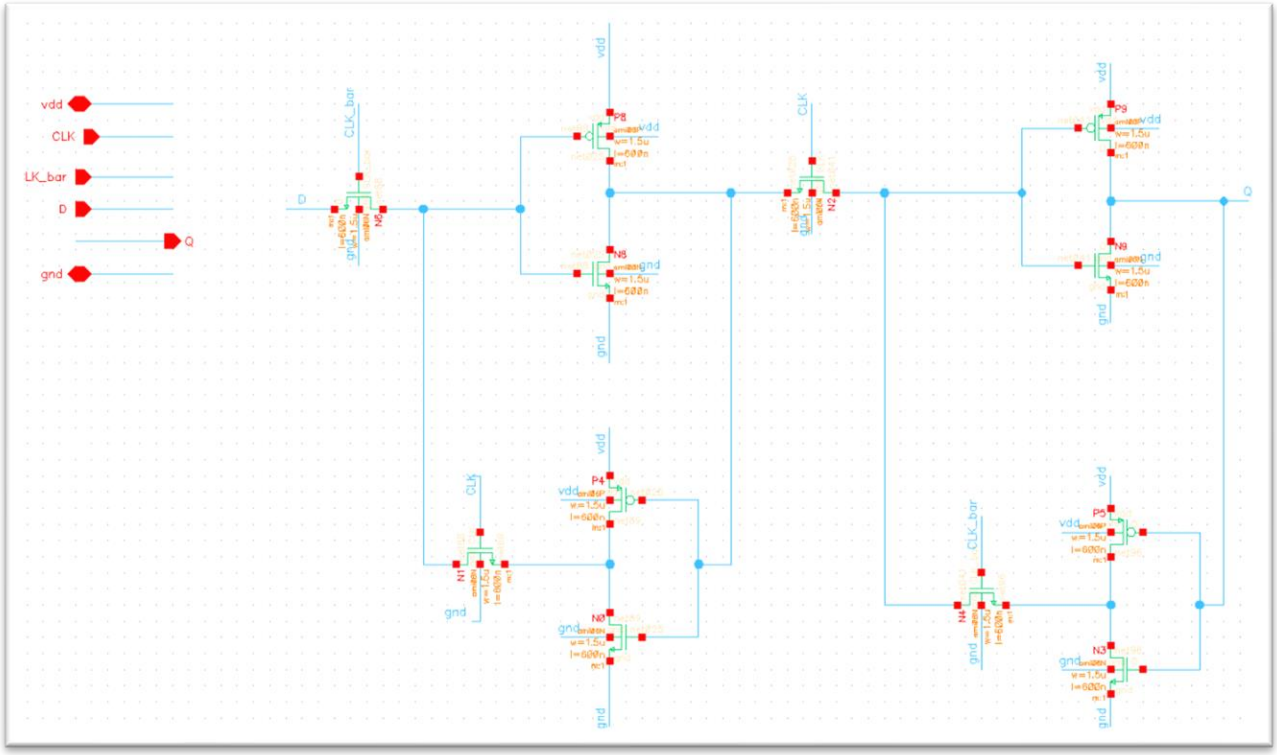


Figure 27 D Flip flop Schematic

Symbol

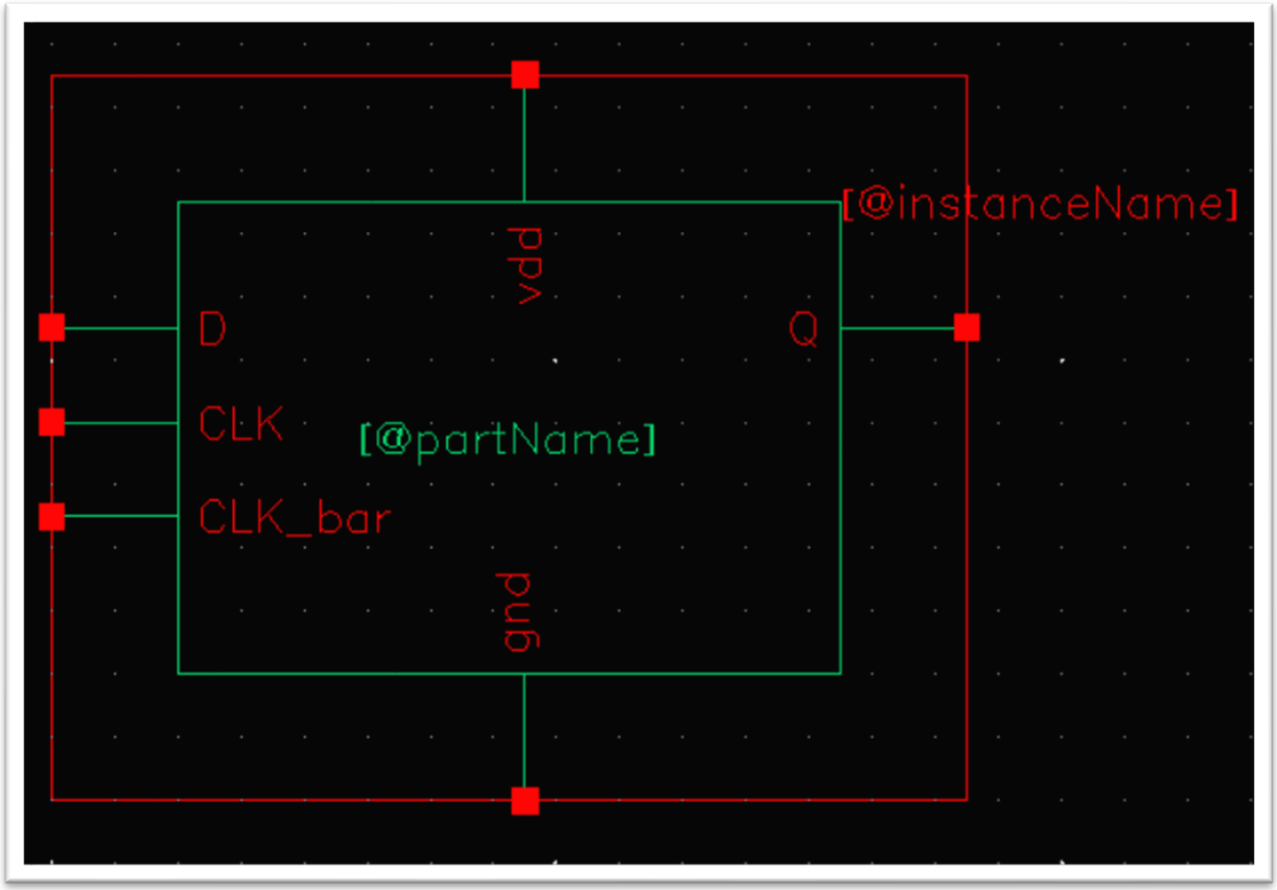


Figure 28 D Flip flop symbol

Testing Circuit

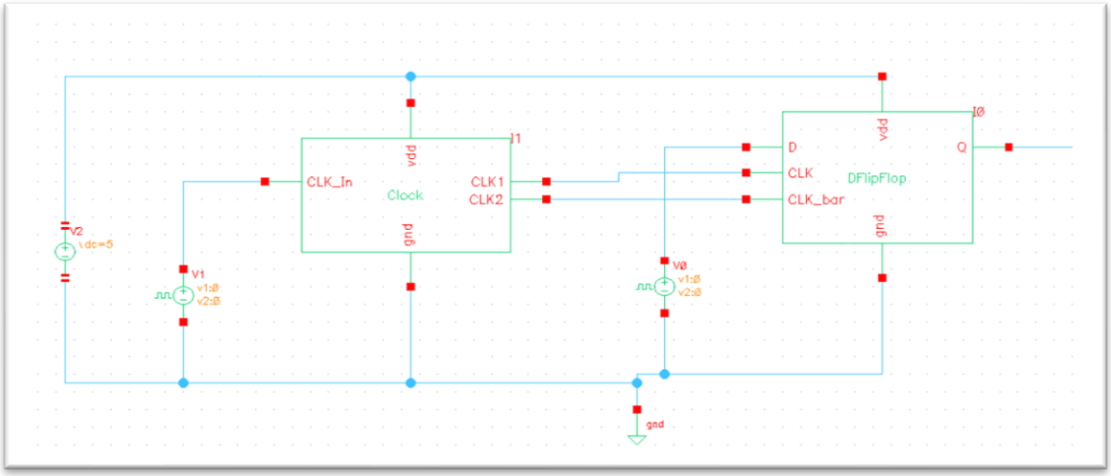


Figure 29 D Flipflop Test Schematic

Simulation Plot

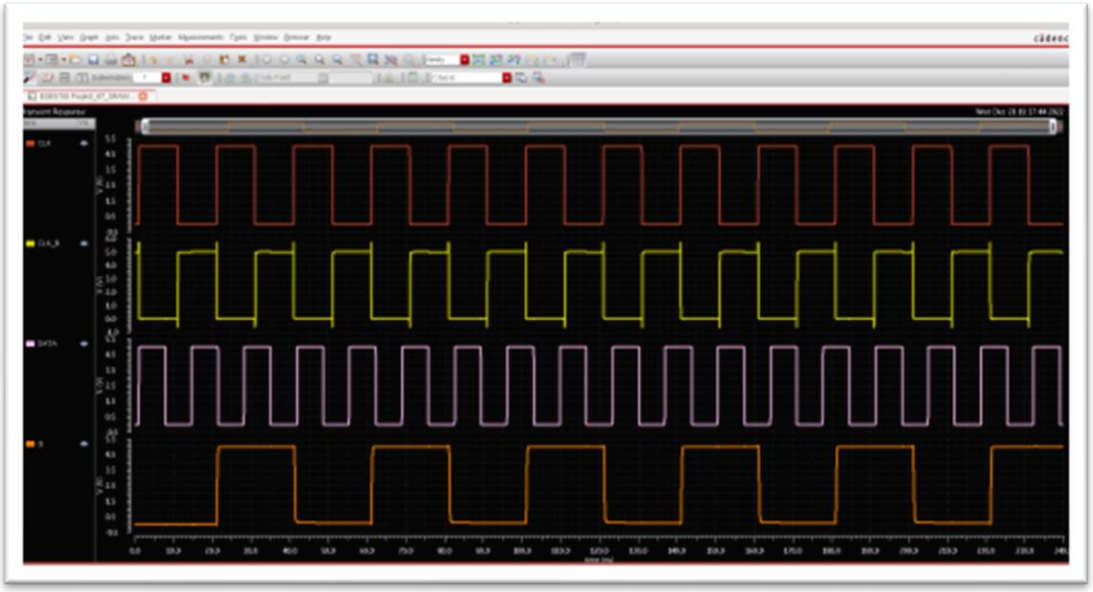


Figure 30 D Flipflop Plot

Layout

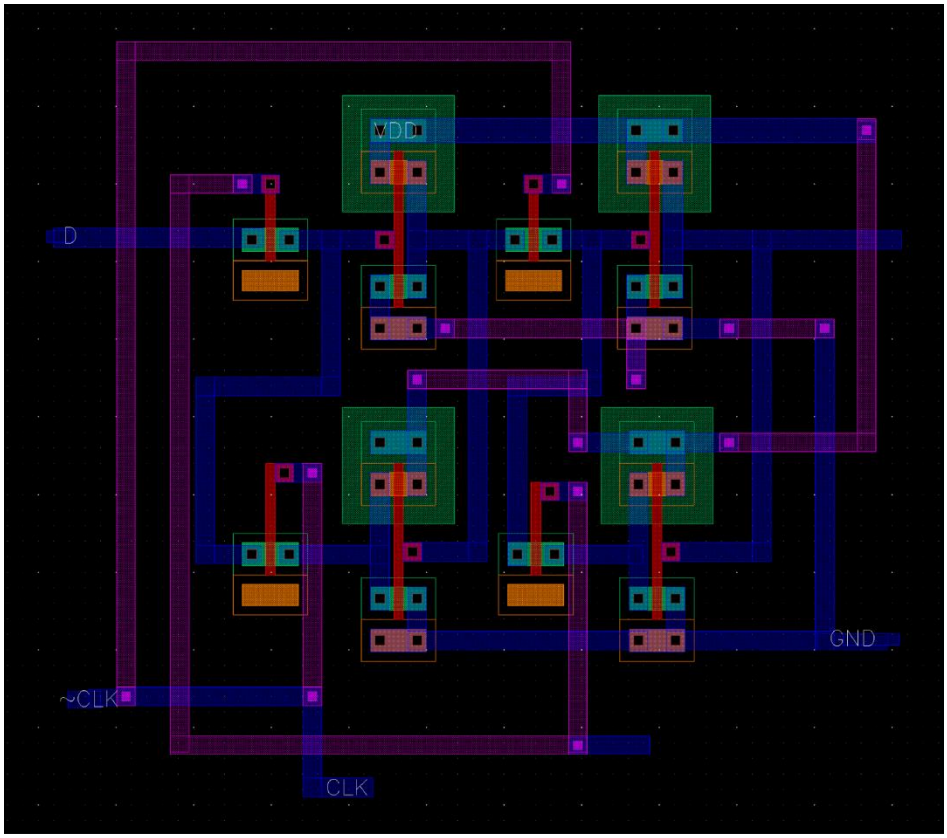


Figure 31 D Flipflop Layout

DRC Proof

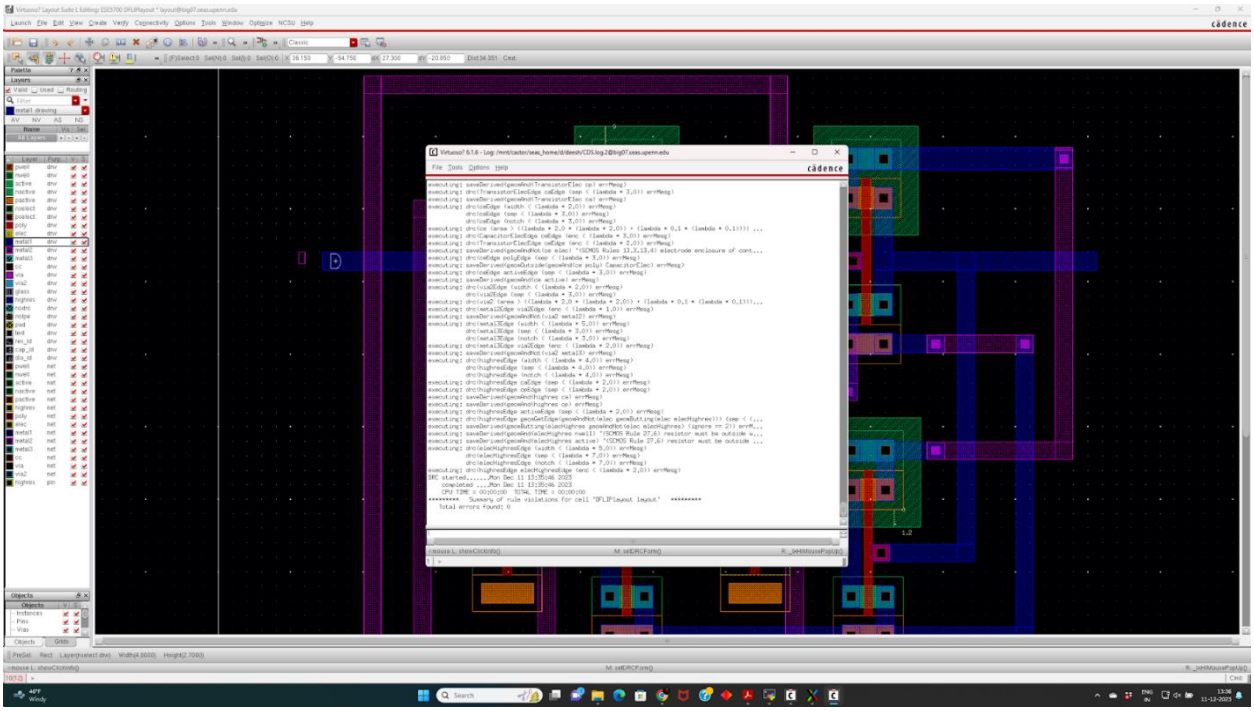


Figure 32 D Flipflop DRC Proof

Serial In Parallel Out (SIPO) Register

A Serial-In Parallel-Out shift register is a sequential logic device that can store and shift data bits. It consists of a chain of flip-flops connected in series, with data input and output terminals. The data is shifted from one flip-flop to the next, either in a serial or parallel fashion, depending on the mode of operation.

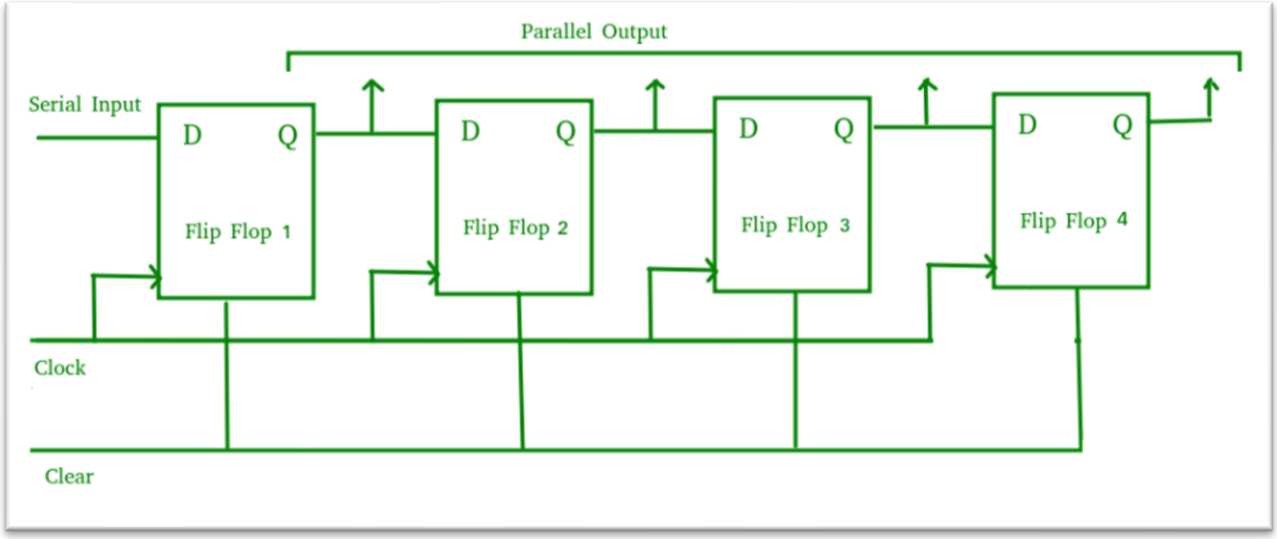


Figure 33 SIPO Block Diagram

The basic operation of a SIPO shift register involves the sequential transfer of data bits through a series of flip-flops. The register has one input line called the serial input (SI) and parallel output lines (Q0, Q1, Q2, etc.) corresponding to each flip-flop. The clock signal (CLK) controls the shifting of data.

We employ a serial-in, parallel-out shift register, taking in a serial bitstream. The resulting output from this shift register is directed to the SRAM array. This connection is established through the sequential arrangement of 16 D flip-flops in series. At each stage, the parallel output is fed into the SRAM array, effectively cascading 16 SRAMs together.

Design Schematic

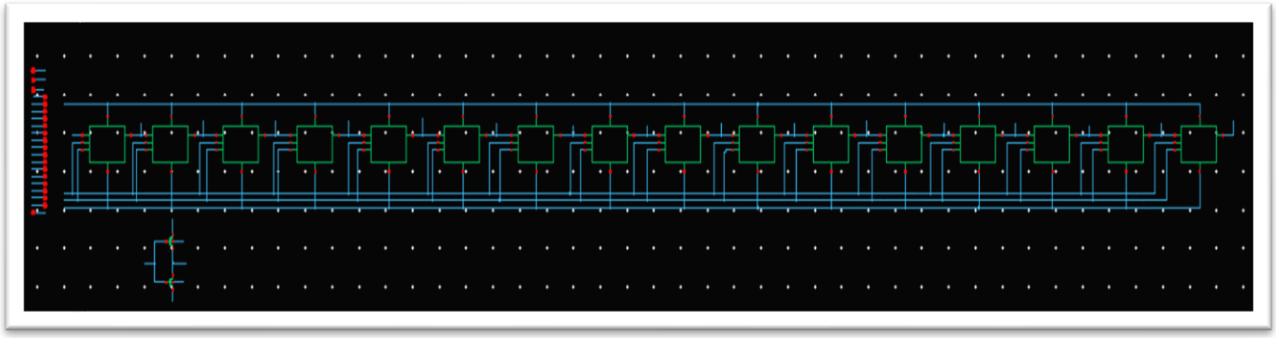


Figure 34 SIPO Schematic

Symbol

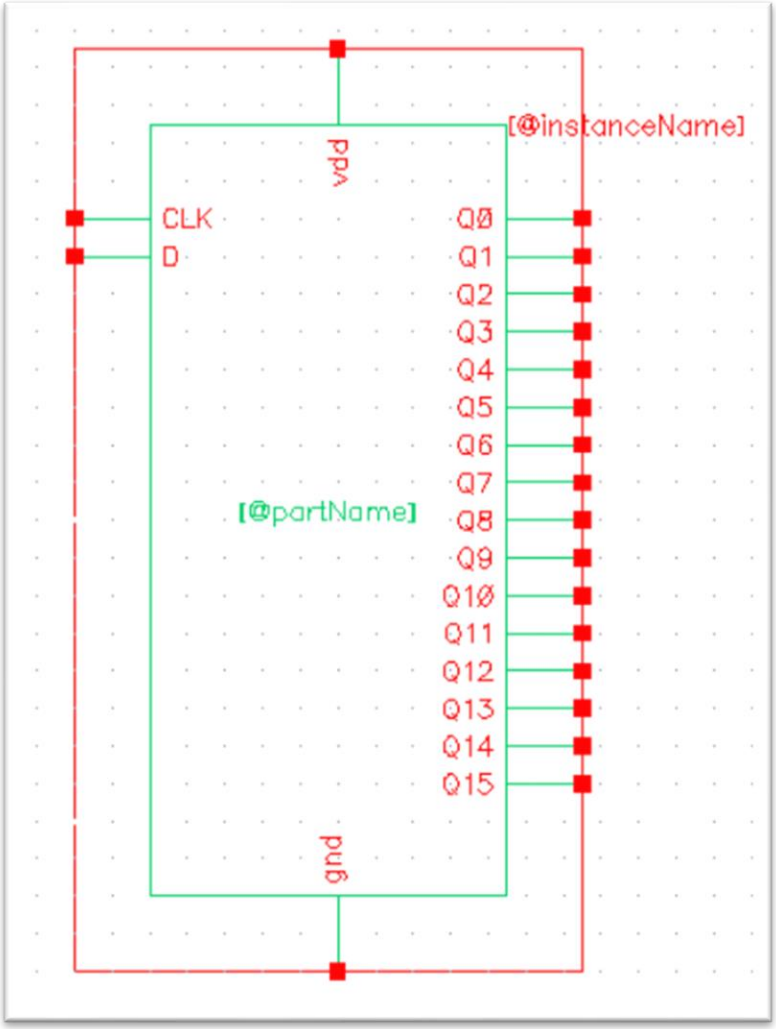


Figure 35 SIPO Symbol

Testing Circuit

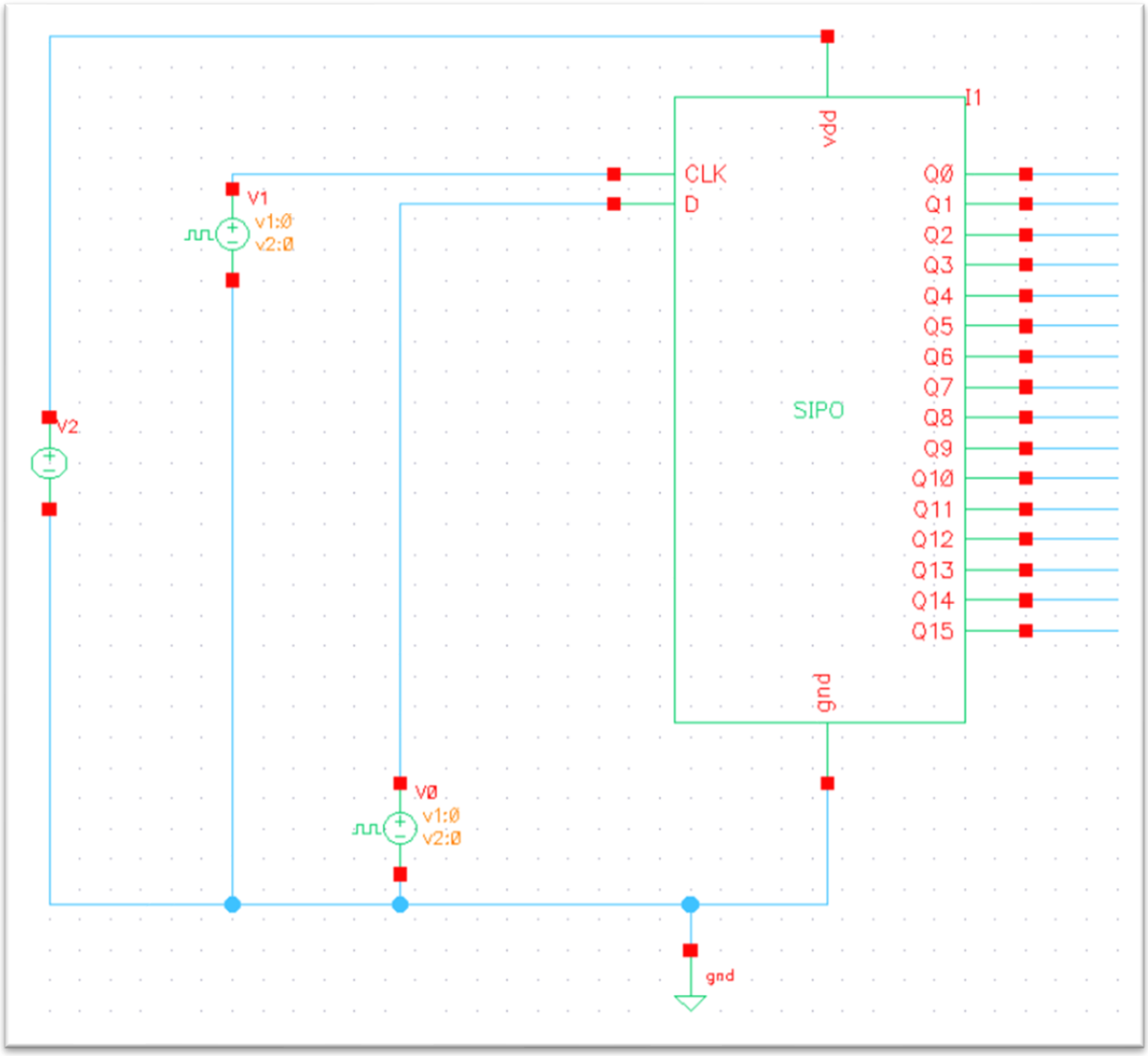


Figure 36 SIPO Test Schematic

Simulation Plot

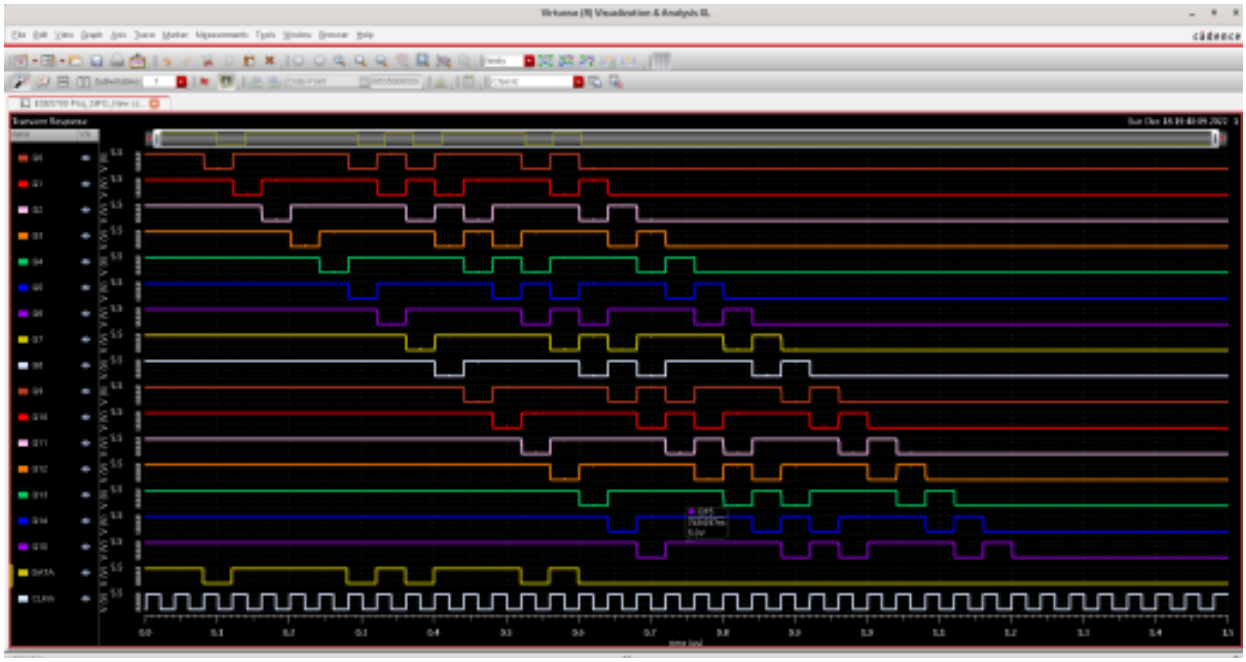


Figure 37 SIPO Plot

Layout

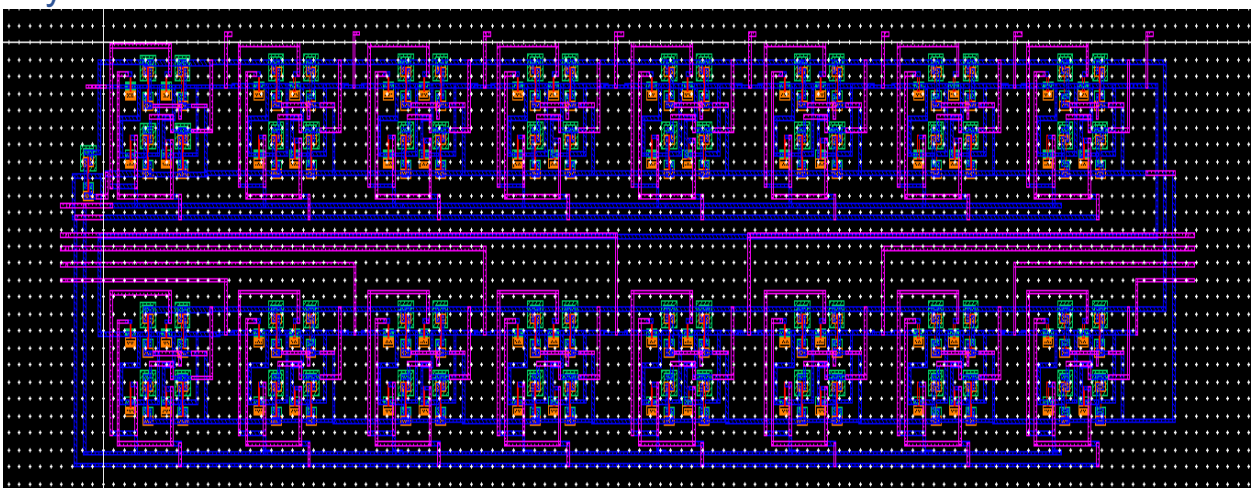


Figure 38 SIPO Layout

DRC Proof

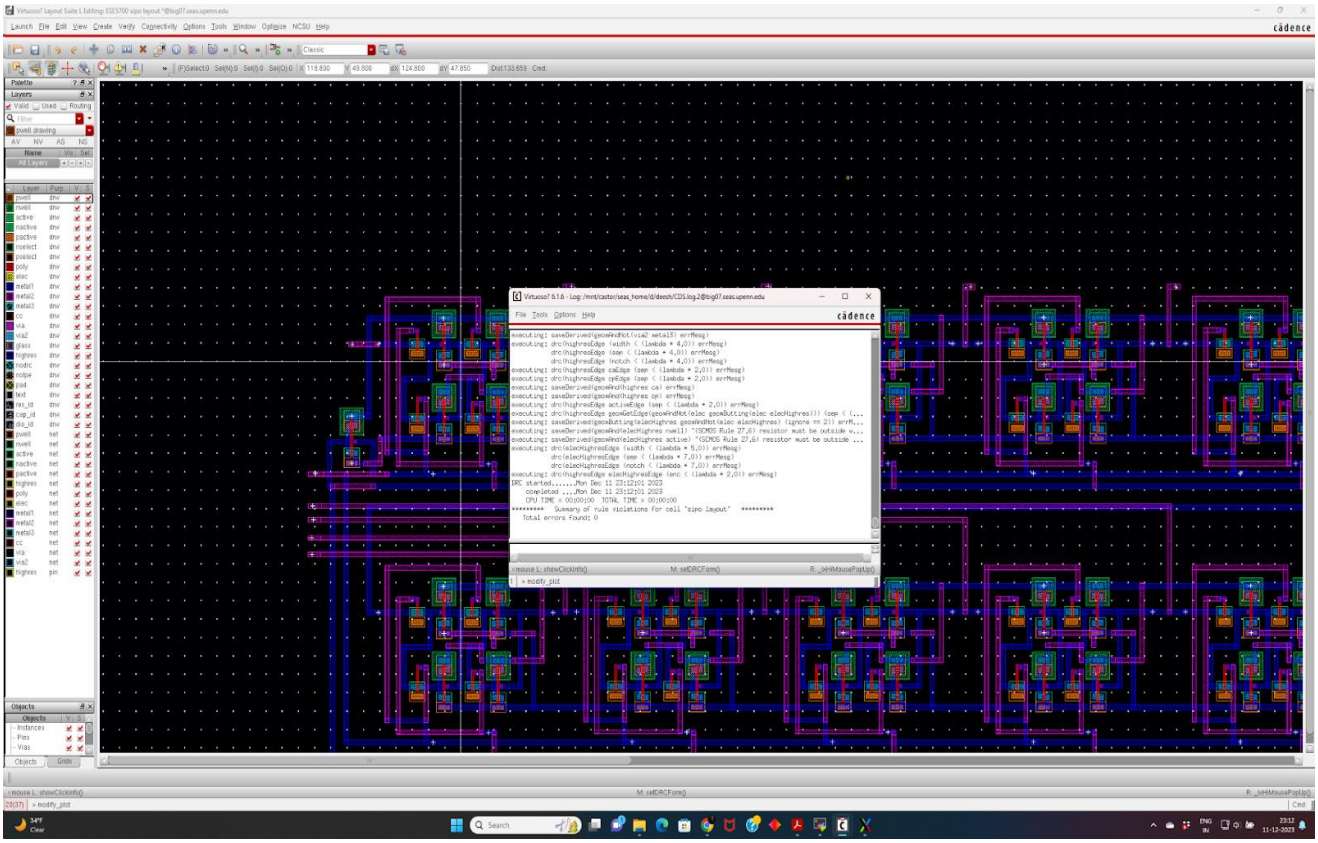


Figure 39 SIPO DRC Proof

Static Random Access Memory (SRAM)

Static Random-Access Memory (SRAM) is an essential component in Field-Programmable Gate Arrays (FPGAs), contributing to the programmable and reconfigurable nature of these devices. In FPGAs, SRAM is primarily used for configuring the programmable logic and for storing temporary data.

In FPGAs, SRAM is often used as configuration memory. Configuration cells in the SRAM store the logic states of each programmable element (look-up tables, flip-flops, etc.) in the FPGA. During the configuration process, the FPGA is programmed by loading the desired configuration data into the SRAM cells. This process determines the functionality of the FPGA.

SRAM-based FPGAs are highly flexible due to their re-programmability. Designers can iterate and experiment with different configurations for their digital circuits. Block RAMs in FPGAs provide high-speed access to stored data, making them suitable for applications with stringent performance requirements.

SRAM plays a crucial role in the configuration and operation of FPGAs, providing flexibility, adaptability, and high-speed data storage capabilities. Designers leverage the reconfigurability of SRAM to implement a wide range of digital circuits in FPGA applications.

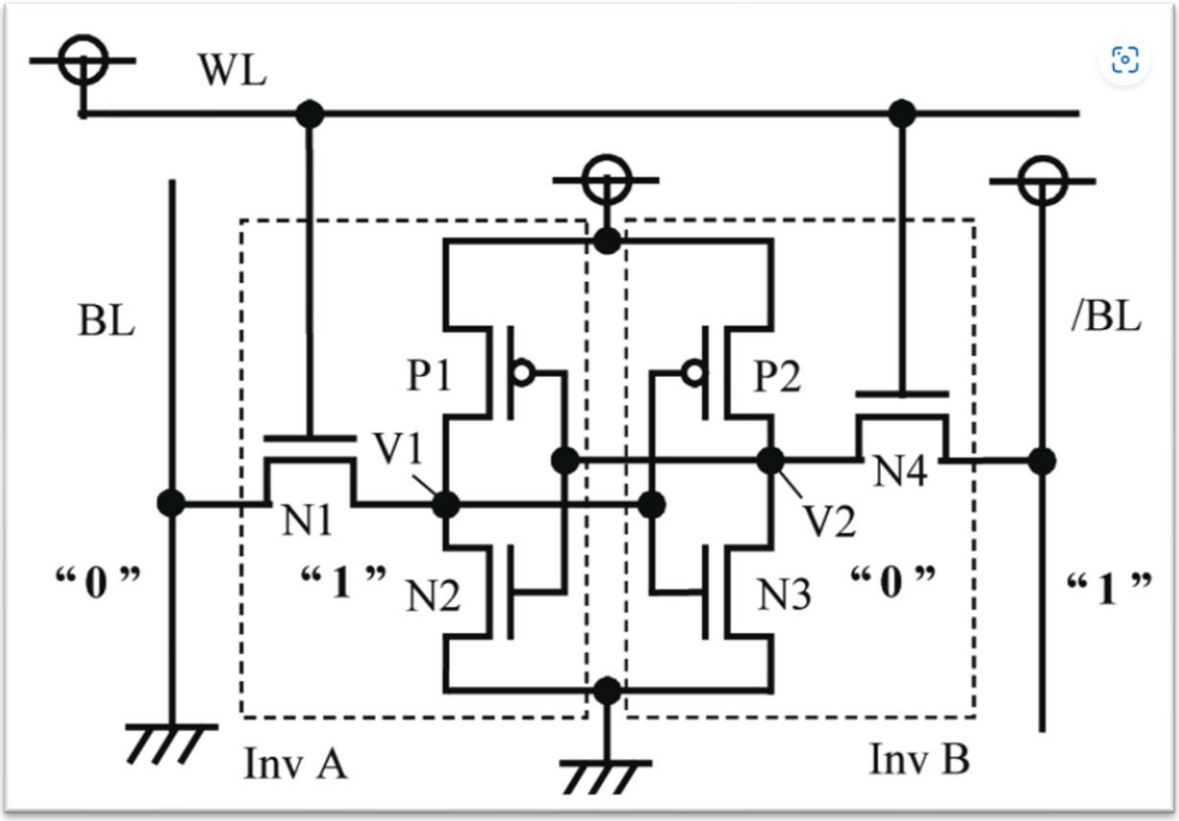


Figure 40 SRAM Block Diagram

We have devised a conventional 6-transistor (6-T) SRAM cell, configuring the size of the NMOS transistors in the cross-coupled inverters to 3μm. Specifically, we established the sizes of NMOS N2 and N3 at 3μm, emphasizing the importance of NMOS N2 being stronger than N1. During the discharge phase, the bitline bar discharges through both N1 and N2. Hence, we have opted for a larger size for N2 NMOS compared to NMOS N1. This same principle applies to NMOS N3 and N4, where N3 is significantly larger than N4 to ensure read stability.

The decision not to increase the sizes of the NMOS transistors in the cross-coupled inverter beyond 3μm is grounded in the principle that all bit cells within a single SRAM cell ideally should remain compact, preventing the memory cell from becoming excessively bulky.

For the NMOS transistors in the write circuitry, we have set their sizes to 4μm to guarantee write stability. It's worth noting that these sizes were not determined through parametric analysis but rather by adhering to ratios recommended for the general functionality of a 6-T SRAM cell.

Design Schematic

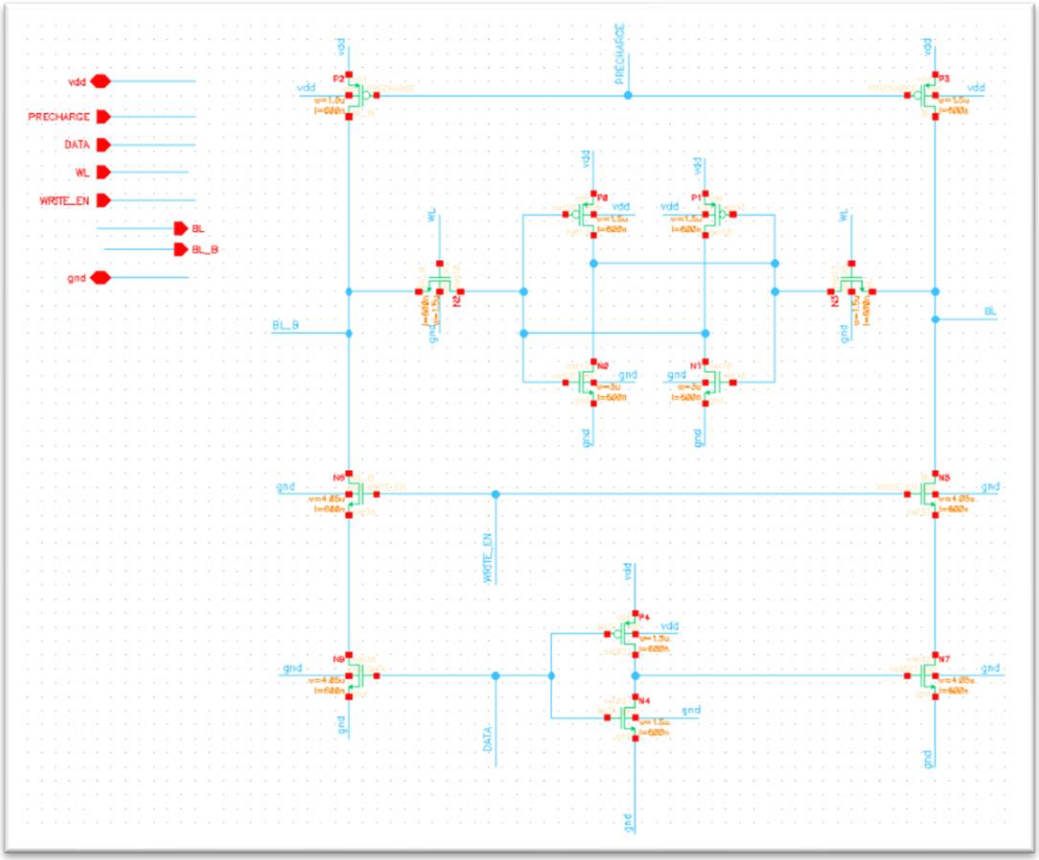


Figure 41 SRAM Schematic

Symbol

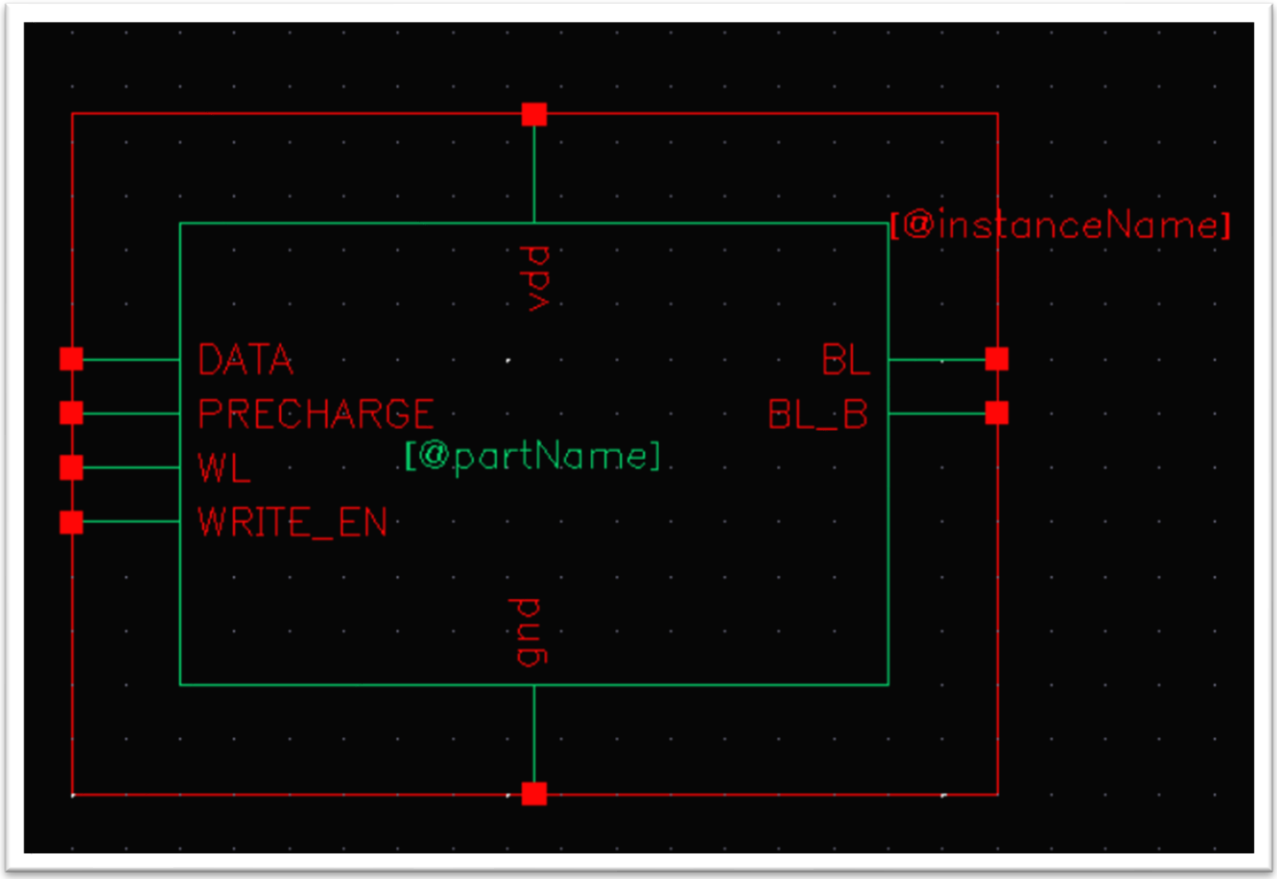


Figure 42 SRAM Symbol

Testing Circuit

Storing 1

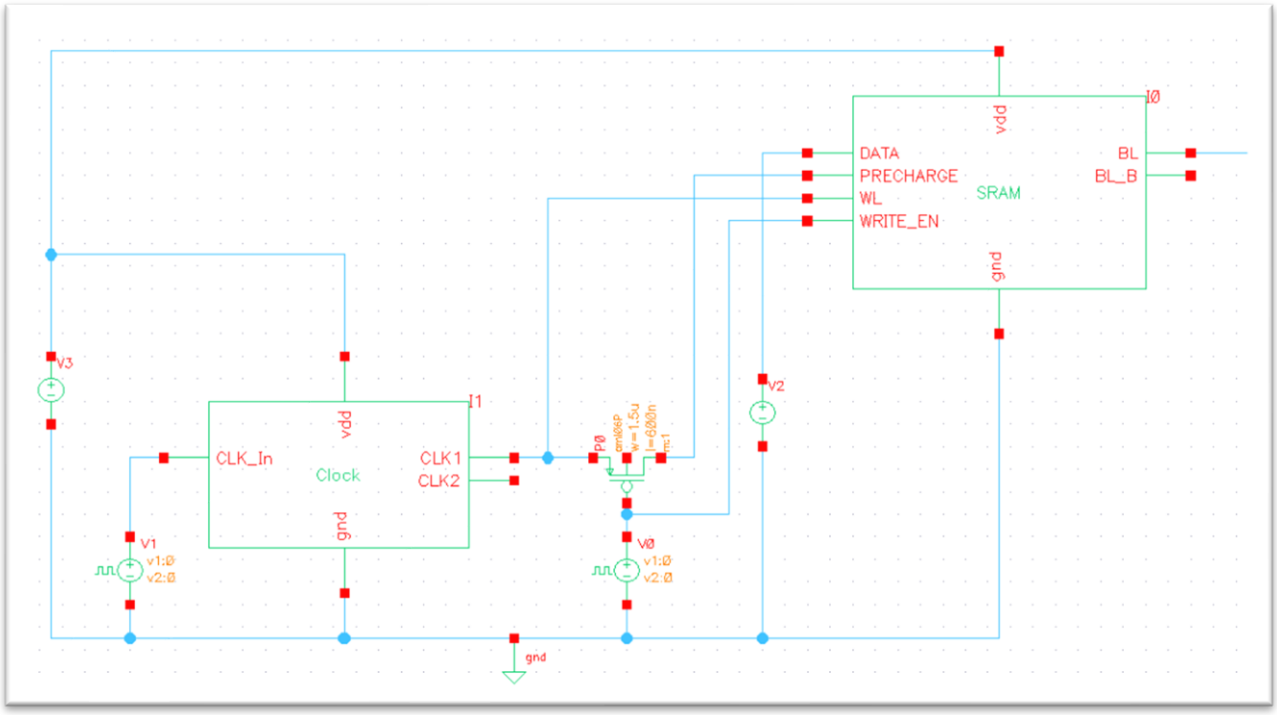


Figure 43 SRAM Test Schematic storing 1

Storing 0

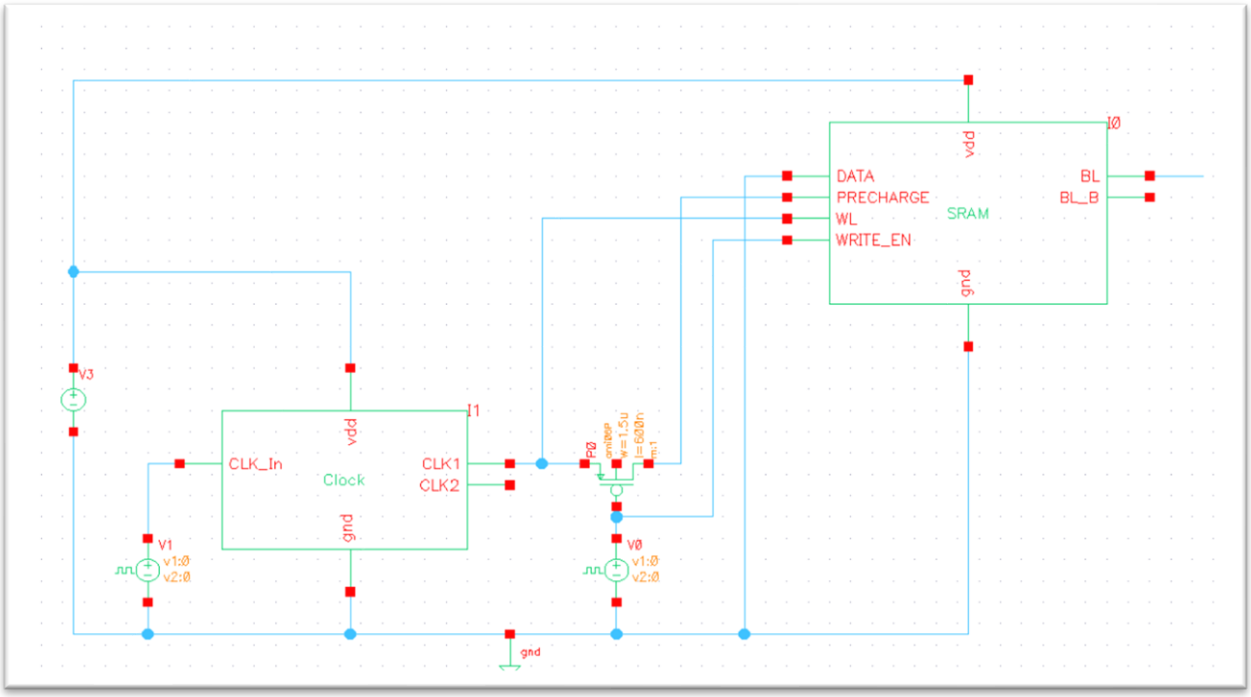


Figure 44 SRAM Test Schematic Storing 0

Simulation Plot

Storing 1

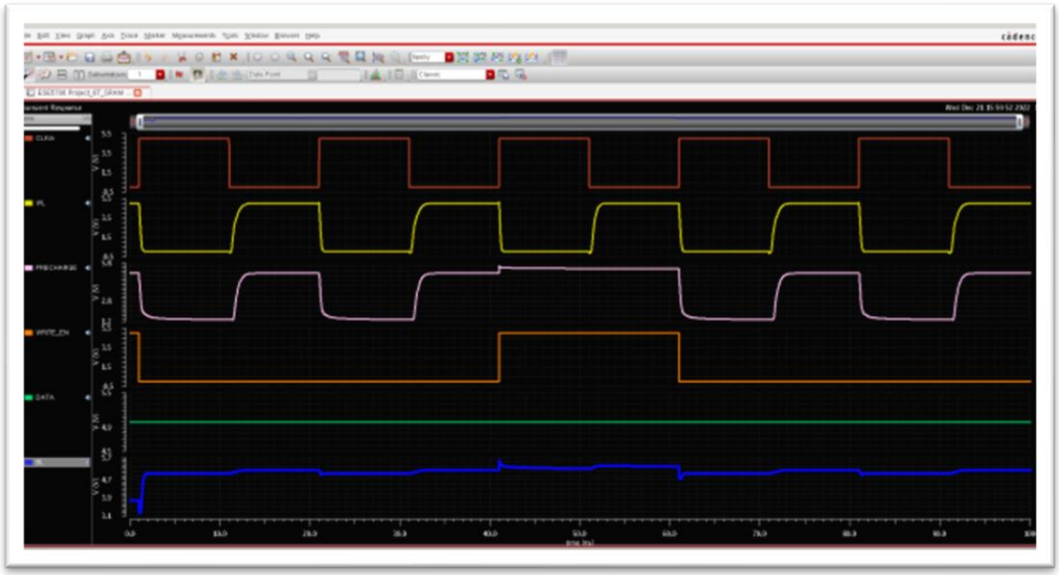


Figure 45 SRAM for Storing 1

Storing 0

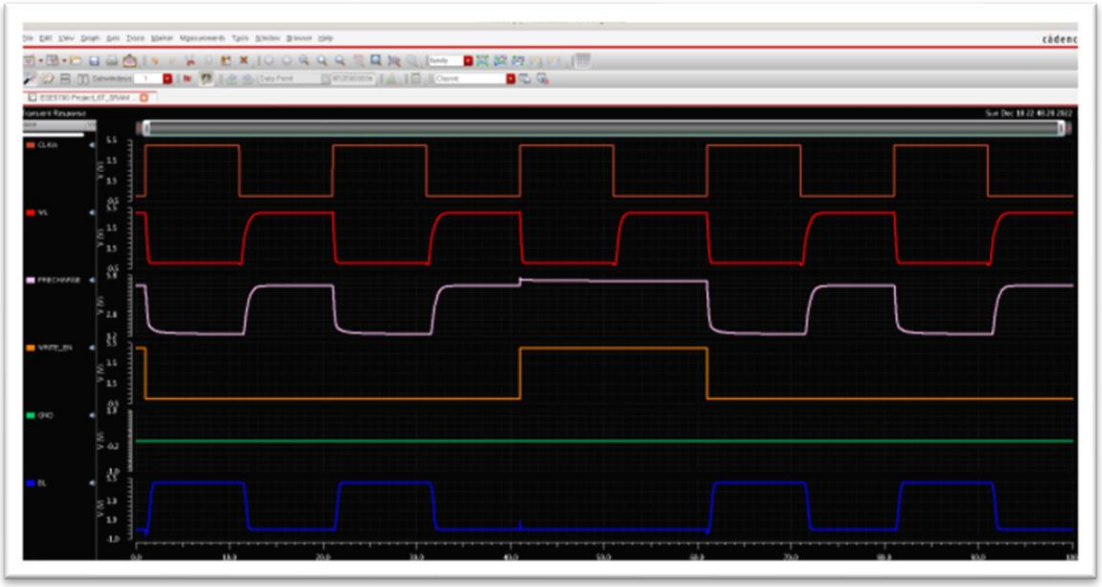


Figure 46 SRAM for Storing 0

Layout

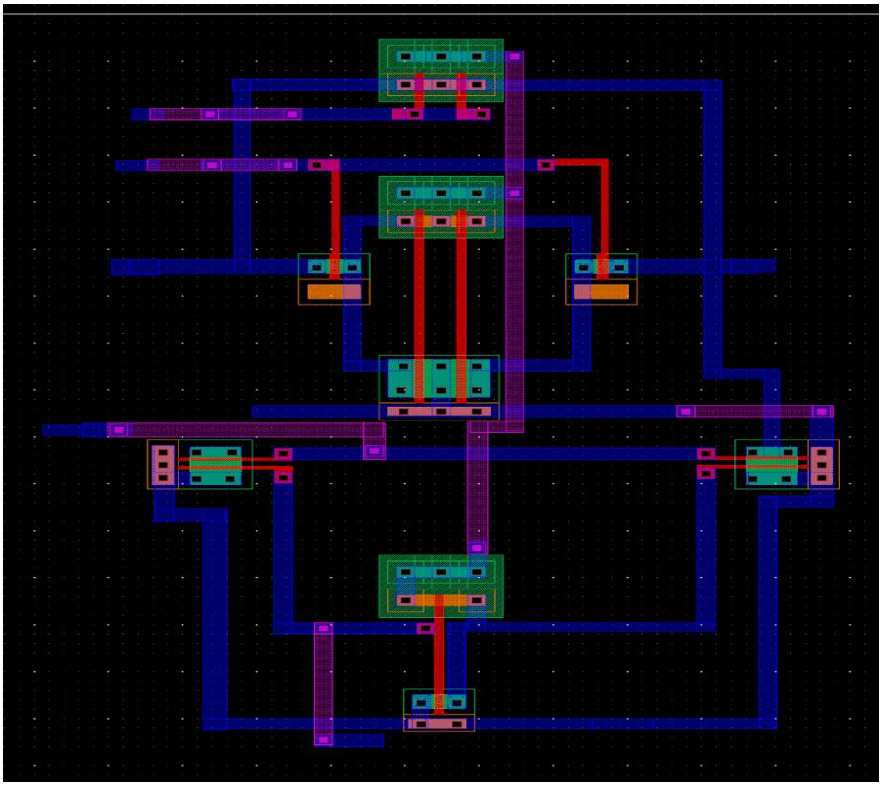
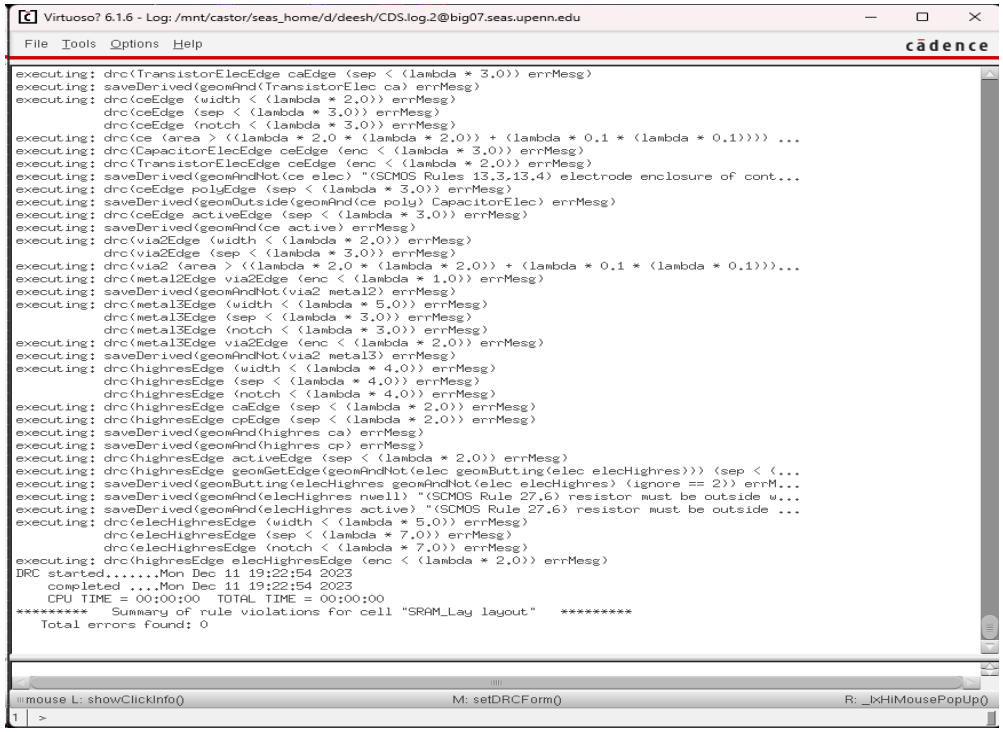


Figure 47 SRAM Layout

DRC Proof



```
Virtuoso? 6.1.6 - Log: /mnt/castor/seas_home/d/deesh/CDS.log.2@big07.seas.upenn.edu
File  Tools  Options  Help                                                    cādence
executing: drc(TransistorElecEdge caEdge (sep < (lambda * 3.0)) errMesg)
executing: saveDerived(geomAnd(TransistorElec ca) errMesg)
executing: drc(ceEdge (width < (lambda * 2.0)) errMesg)
           drc(ceEdge (sep < (lambda * 3.0)) errMesg)
           drc(ceEdge (notch < (lambda * 3.0)) errMesg)
executing: drc(ce (area > ((lambda * 2.0 * (lambda * 2.0)) + (lambda * 0.1 * (lambda * 0.1)))) ...
executing: drc(CapacitorElecEdge ceEdge (enc < (lambda * 3.0)) errMesg)
executing: drc(TransistorElecEdge ceEdge (enc < (lambda * 2.0)) errMesg)
executing: saveDerived(geomAndNot(ce elec) "(SCMOS Rules 13.3,13.4) electrode enclosure of cont...
executing: drc(ceEdge polyEdge (sep < (lambda * 3.0)) errMesg)
executing: saveDerived(geomOutside(geomAnd(ce poly) CapacitorElec) errMesg)
executing: drc(ceEdge activeEdge (sep < (lambda * 3.0)) errMesg)
executing: saveDerived(geomAnd(ce active) errMesg)
executing: drc(via2Edge (width < (lambda * 2.0)) errMesg)
           drc(via2Edge (sep < (lambda * 3.0)) errMesg)
executing: drc(via2 (area > ((lambda * 2.0 * (lambda * 2.0)) + (lambda * 0.1 * (lambda * 0.1)))...
executing: drc(metal2Edge via2Edge (enc < (lambda * 1.0)) errMesg)
executing: saveDerived(geomAndNot(via2 metal2) errMesg)
executing: drc(metal3Edge (width < (lambda * 5.0)) errMesg)
           drc(metal3Edge (sep < (lambda * 3.0)) errMesg)
           drc(metal3Edge (notch < (lambda * 3.0)) errMesg)
executing: drc(metal3Edge via2Edge (enc < (lambda * 2.0)) errMesg)
executing: saveDerived(geomAndNot(via2 metal3) errMesg)
executing: drc(highresEdge (width < (lambda * 4.0)) errMesg)
           drc(highresEdge (sep < (lambda * 4.0)) errMesg)
           drc(highresEdge (notch < (lambda * 4.0)) errMesg)
executing: drc(highresEdge caEdge (sep < (lambda * 2.0)) errMesg)
executing: drc(highresEdge cpEdge (sep < (lambda * 2.0)) errMesg)
executing: saveDerived(geomAnd(highres ca) errMesg)
executing: saveDerived(geomAnd(highres cp) errMesg)
executing: drc(highresEdge activeEdge (sep < (lambda * 2.0)) errMesg)
executing: drc(highresEdge geomGetEdge(geomAndNot(elec geomButting(elec elecHighres))) (sep < (...
executing: saveDerived(geomButting(elecHighres geomAndNot(elec elecHighres) (ignore == 2)) errM...
executing: saveDerived(geomAnd(elecHighres nwell) "(SCMOS Rule 27.6) resistor must be outside w...
executing: saveDerived(geomAnd(elecHighres active) "(SCMOS Rule 27.6) resistor must be outside ...
executing: drc(elecHighresEdge (width < (lambda * 5.0)) errMesg)
           drc(elecHighresEdge (sep < (lambda * 7.0)) errMesg)
           drc(elecHighresEdge (notch < (lambda * 7.0)) errMesg)
executing: drc(highresEdge elecHighresEdge (enc < (lambda * 2.0)) errMesg)
DRC started.......Mon Dec 11 19:22:54 2023
    completed ....Mon Dec 11 19:22:54 2023
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*********   Summary of rule violations for cell "SRAM_Lay layout"   *********
   Total errors found: 0
mouse L: showClickInfo()        M: setDRCForm()        R: _lxHiMousePopUp()
1  >
```

Figure 48 SRAM DRC Proof

SRAM Array

We've created an SRAM array composed of 16 SRAM cells, systematically linked together to provide inputs to a 16x1 multiplexer. The output bitlines (BL) from each individual SRAM cell are interconnected and directed to the inputs of the 16x1 MUX.

Design Schematic

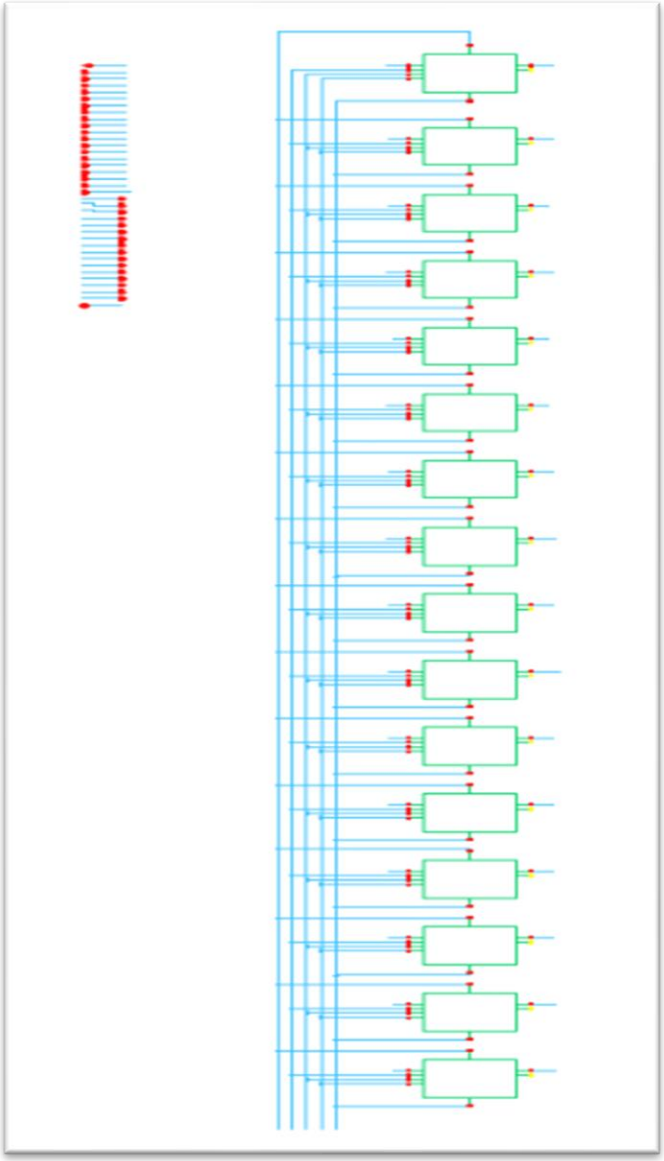


Figure 49 SRAM Array Schematic

Symbol

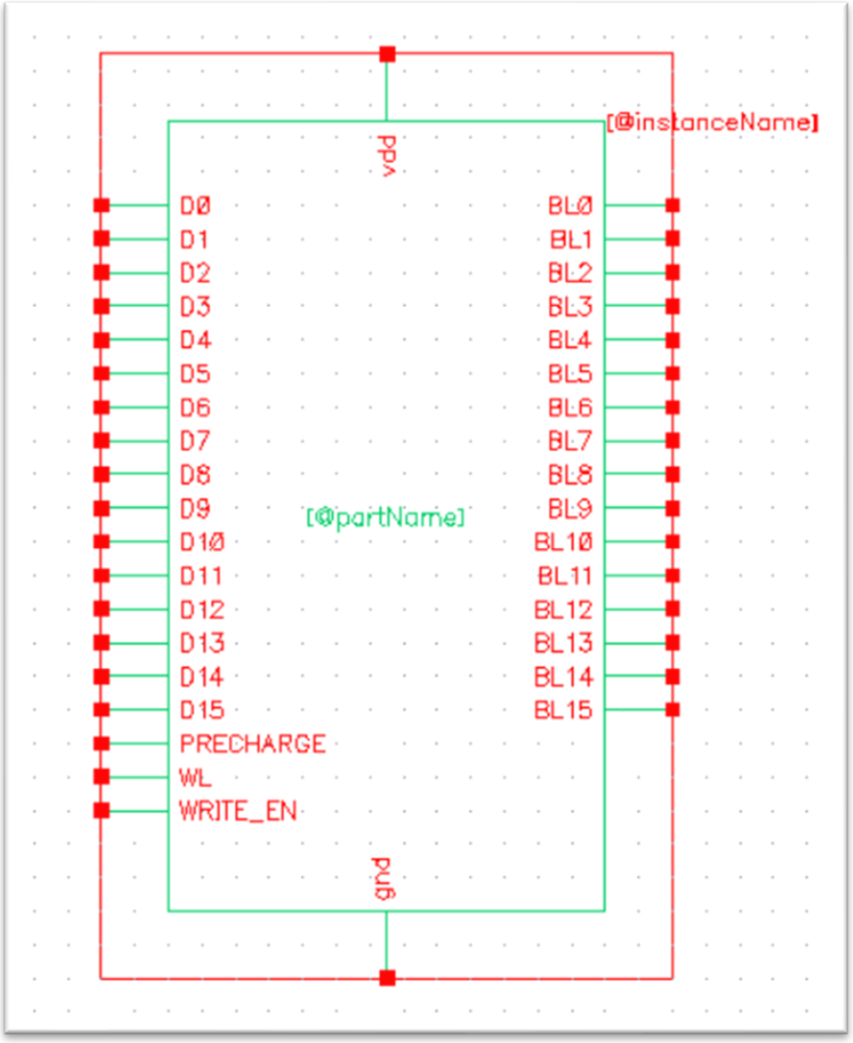


Figure 50 SRAM Array Symbol

Testing Circuit

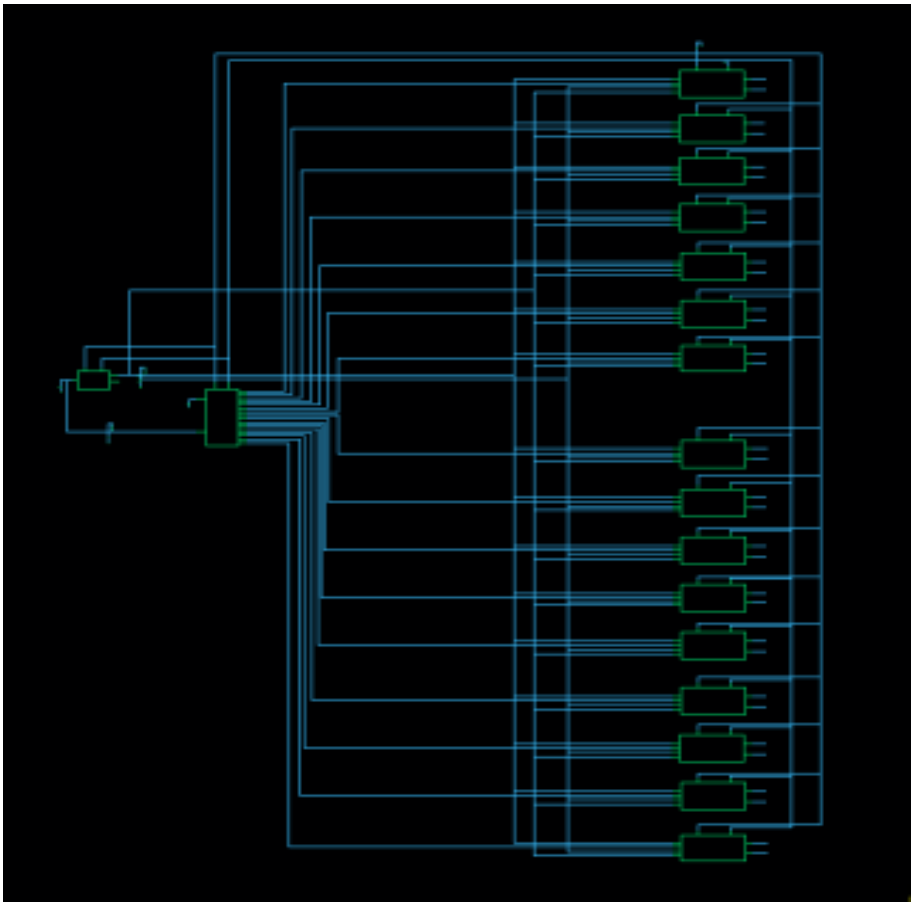


Figure 51 SRAM Array Test Schematic

Simulation Plot

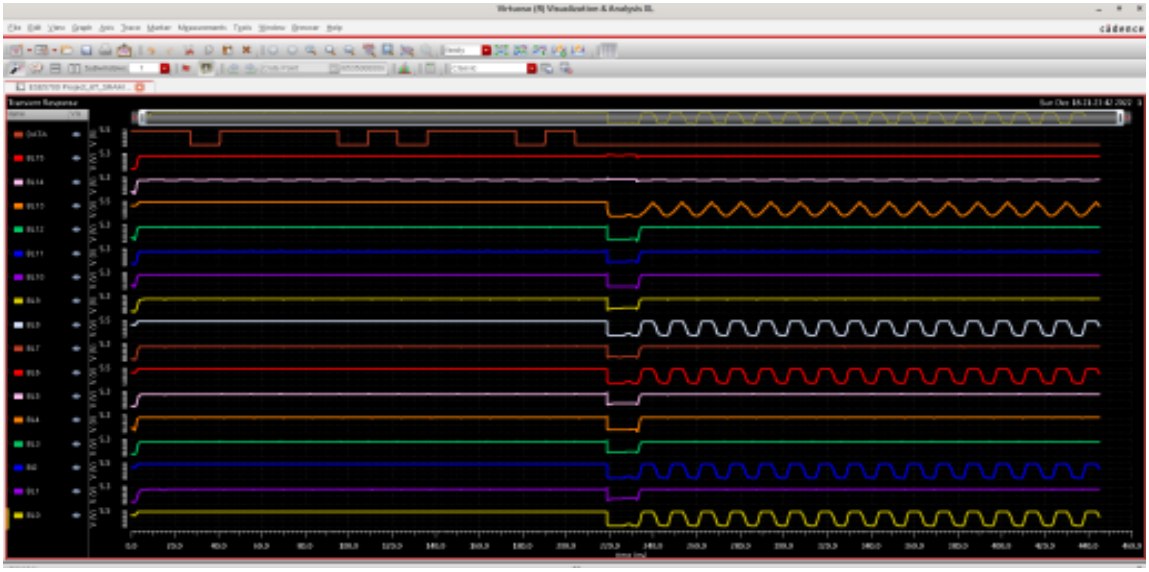


Figure 52 SRAM Array Plot

Layout

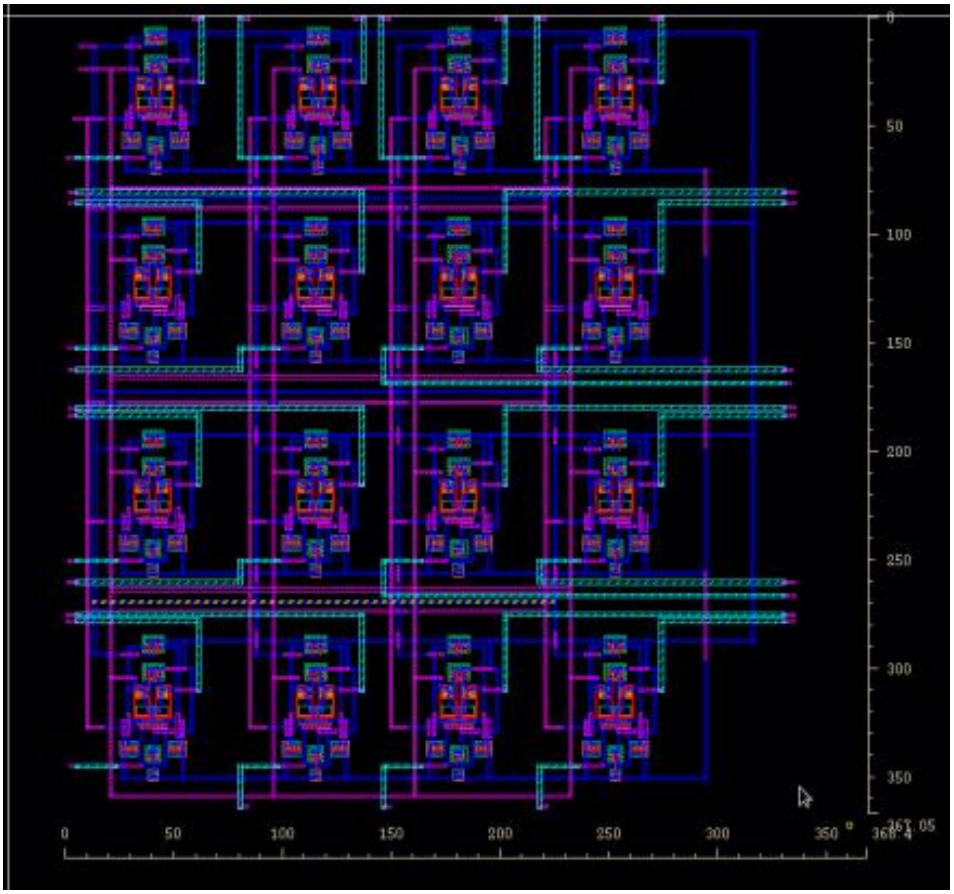


Figure 53 SRAM Array Layout

DRC Proof

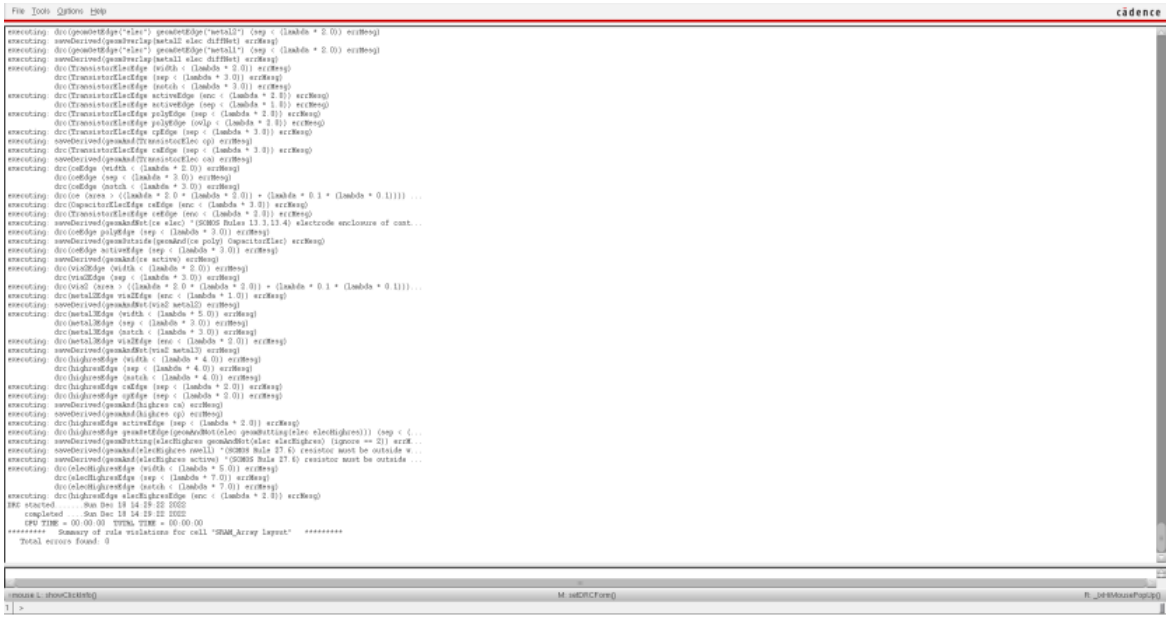


Figure 54 SRAM Array DRC Proof

Configurable Logic Block (CLB)

A Configurable Logic Block (CLB) is a fundamental building block within a Field-Programmable Gate Array (FPGA). It is a versatile and reconfigurable unit that enables the implementation of various digital logic functions. The CLB consists of a combination of look-up tables (LUTs), flip-flops, multiplexers, and other configurable elements.
The primary function of a CLB is to implement combinational logic. With the inclusion of flip-flops, CLBs also support the implementation of sequential logic, enabling the creation of registers and memory elements. CLBs provide a network of programmable interconnects, allowing signals to be routed between various logic elements within the FPGA.

Design Schematic

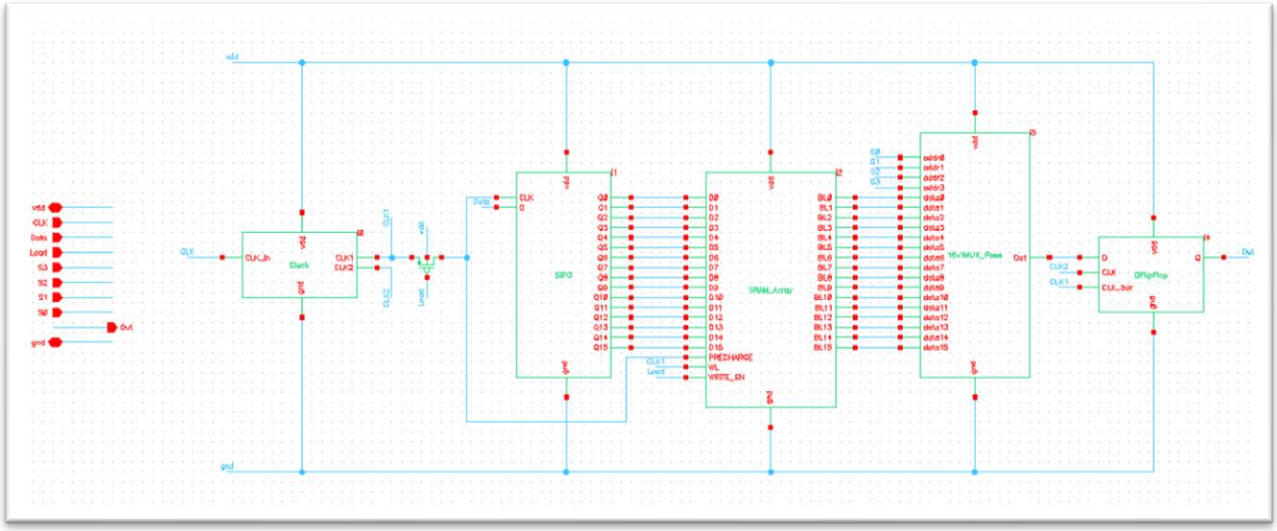


Figure 55 CLB Schematic

In our Configurable Logic Block design, the clock module is linked to the Serial-In Parallel-Out (SIPO) register, where a serial bitstream is introduced. A LOAD signal governs the SIPO register, allowing us to load data in 16-bit increments, thereby altering the state of the SIPO register.
Subsequently, the output from the SIPO register is directed to our SRAM array, responsible for storing the received data. With the data now stored in the SRAM array, any specific cell is read based on the select inputs of our 16:1 Multiplexer (Mux). Finally, to mitigate transients, a D Flip Flop is positioned at the output.
All the pass transistors utilized in this design are set to minimum size.

Symbol

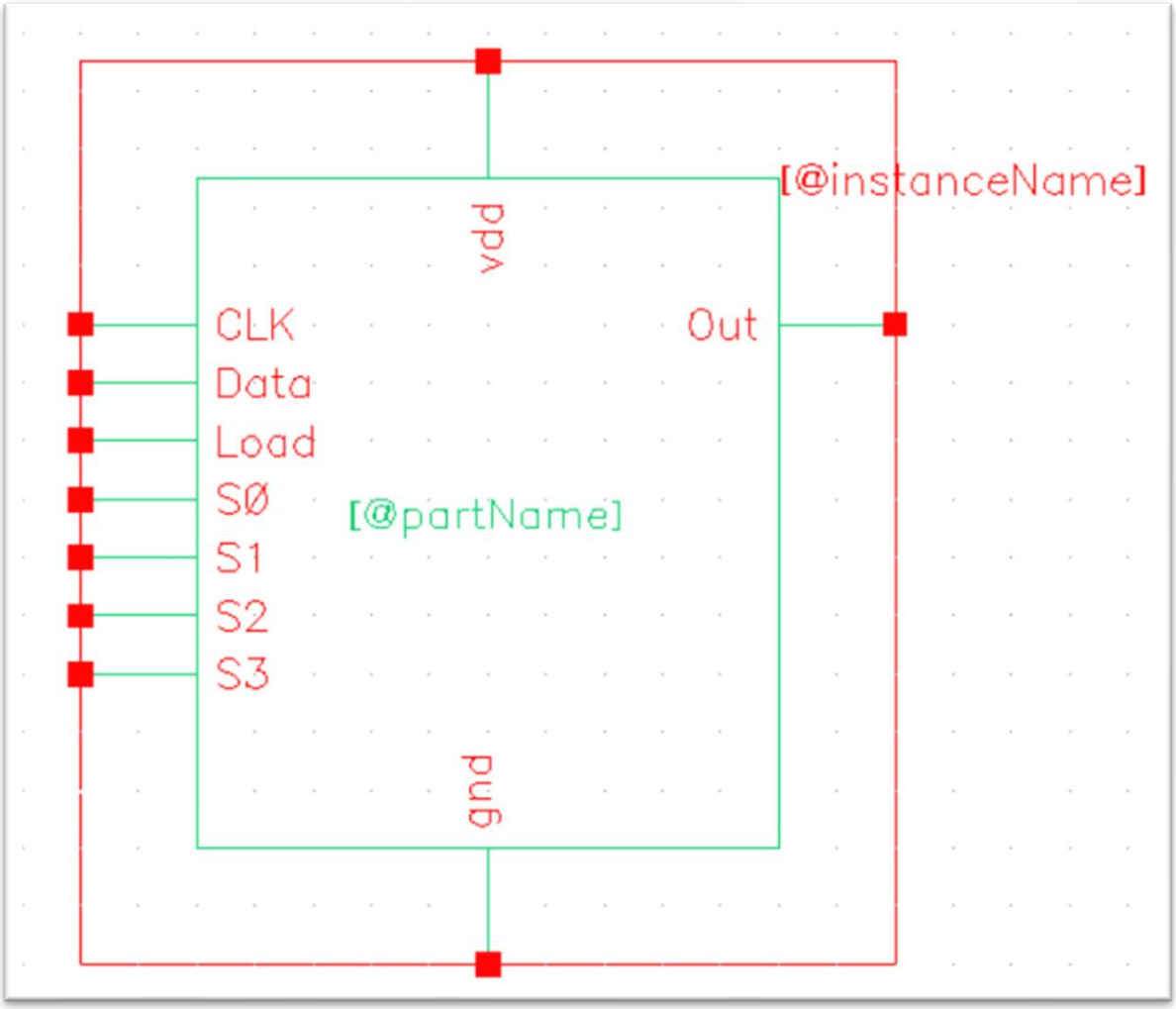


Figure 56 CLB Symbol

Testing Circuit

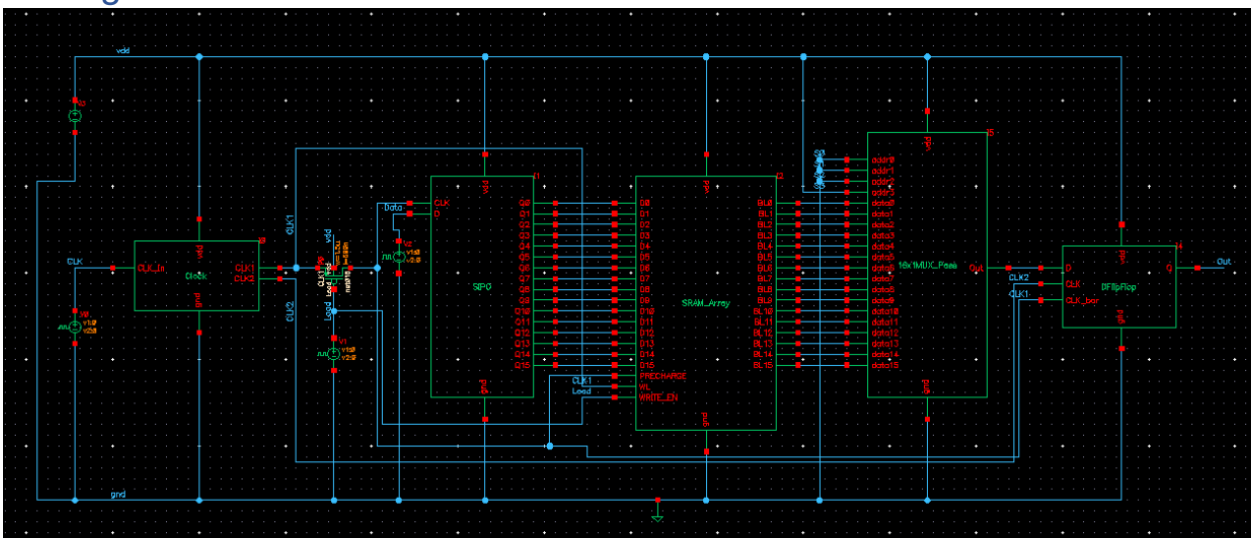


Figure 57 CLB Test Schematic

Simulation Plot

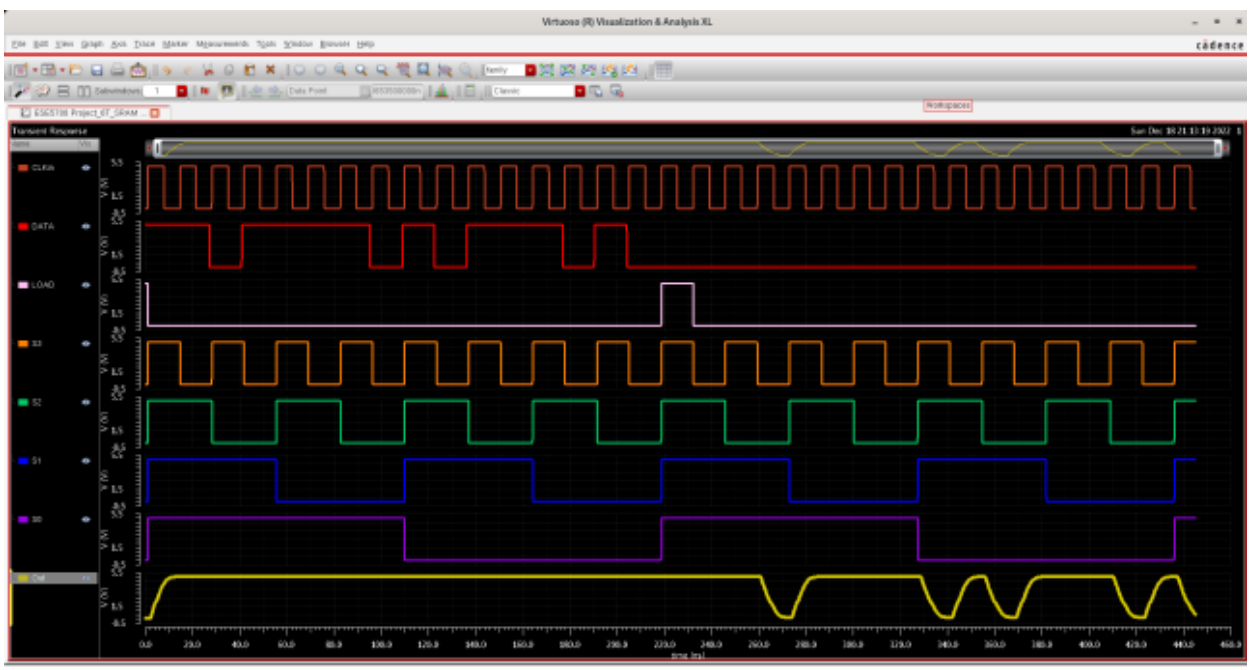


Figure 58 CLB Plot

The simulation plot illustrates that the output aligns with the input data generated from a vbit source, contingent on the state of the select lines of our 16:1 Mux. The address bits for the Look-Up Table (LUT) undergo cycling through vpulse sources with different periods to ensure testing of every possible combination of the select lines.

During testing of the Configurable Logic Block, we explored various periods of CLKin. However, it exhibited failure to operate at periods below 16.4ns. Consequently, we had

to settle on a time period of 16.4ns for our main clock. This signifies that the maximum operating frequency for our Configurable Logic Block is 60.975 MHz.

As our SRAM and 16:1 Mux are not extensively sized, our maximum operating frequency is moderately low. However, this was a deliberate design decision, considering the area constraints of our Configurable Logic Block.

Layout

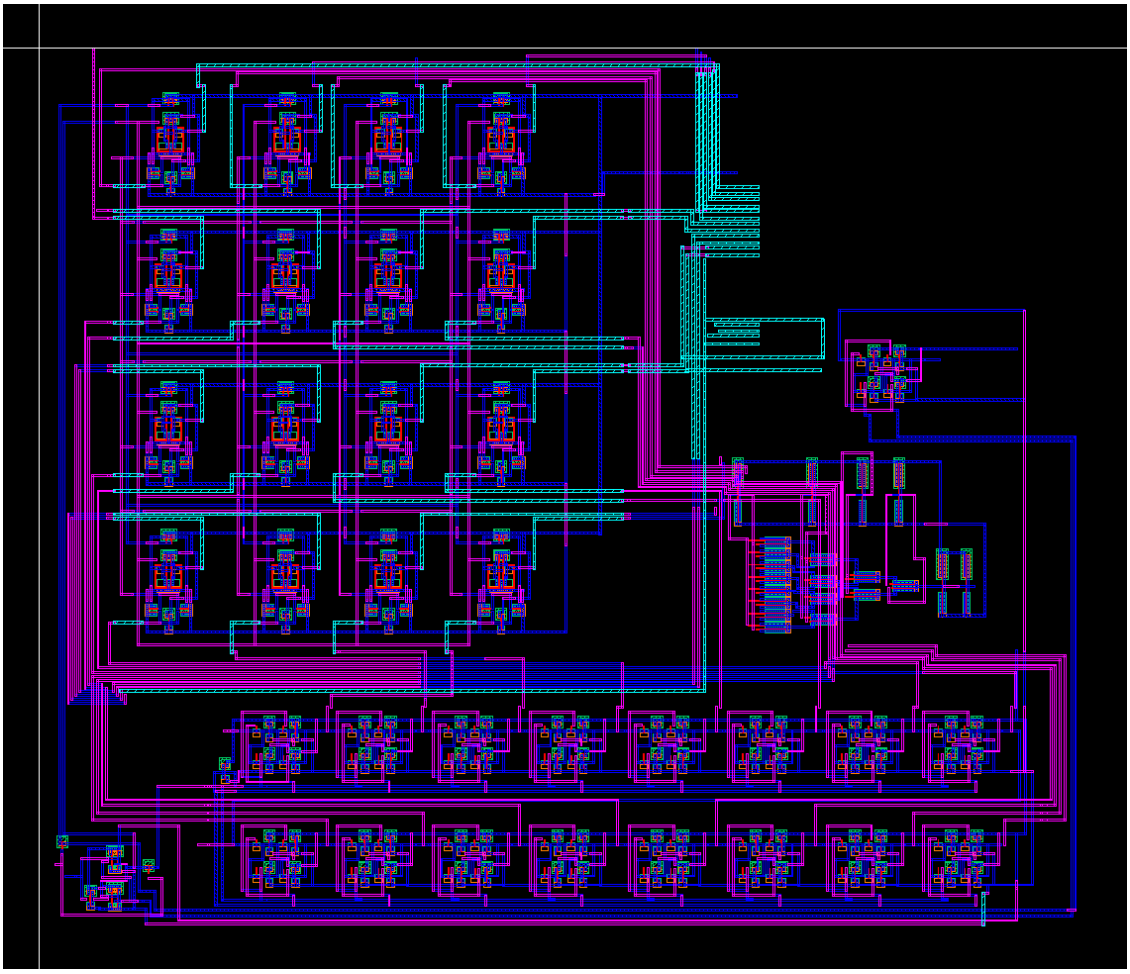


Figure 59 CLB Layout

Area = (672.8 * 672.8) μm^2

DRC Proof

```
Virtuoso? 6.1.6 - Log: /mnt/castor/seas_home/d/deesh/CDS.log.2@big07.seas.upenn.edu
File  Tools  Options  Help                                                    cādence

executing: saveDerived(geomAndNot(via2 metal3) errMesg)
executing: drc(highresEdge (width < (lambda * 4.0)) errMesg)
           drc(highresEdge (sep < (lambda * 4.0)) errMesg)
           drc(highresEdge (notch < (lambda * 4.0)) errMesg)
executing: drc(highresEdge caEdge (sep < (lambda * 2.0)) errMesg)
executing: drc(highresEdge cpEdge (sep < (lambda * 2.0)) errMesg)
executing: saveDerived(geomAnd(highres ca) errMesg)
executing: saveDerived(geomAnd(highres cp) errMesg)
executing: drc(highresEdge activeEdge (sep < (lambda * 2.0)) errMesg)
executing: drc(highresEdge geomGetEdge(geomAndNot(elec geomButting(elec elecHighres))) (sep < (...
executing: saveDerived(geomButting(elecHighres geomAndNot(elec elecHighres) (ignore == 2)) errM...
executing: saveDerived(geomAnd(elecHighres nwell) "(SCMOS Rule 27.6) resistor must be outside w...
executing: saveDerived(geomAnd(elecHighres active) "(SCMOS Rule 27.6) resistor must be outside ...
executing: drc(elecHighresEdge (width < (lambda * 5.0)) errMesg)
           drc(elecHighresEdge (sep < (lambda * 7.0)) errMesg)
           drc(elecHighresEdge (notch < (lambda * 7.0)) errMesg)
executing: drc(highresEdge elecHighresEdge (enc < (lambda * 2.0)) errMesg)
DRC started.......Mon Dec 11 23:41:11 2023
    completed ....Mon Dec 11 23:41:13 2023
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:02
*********   Summary of rule violations for cell "CLB_FINAL_LAYOUT layout"   *********
   Total errors found: 0

mouse L: showClickInfo()        M: setDRCForm()        R: _lxHiMousePopUp()
1  > modify_plot
```

Figure 60 CLB DRC Proof

Design Metrics

Energy

Loading Energy

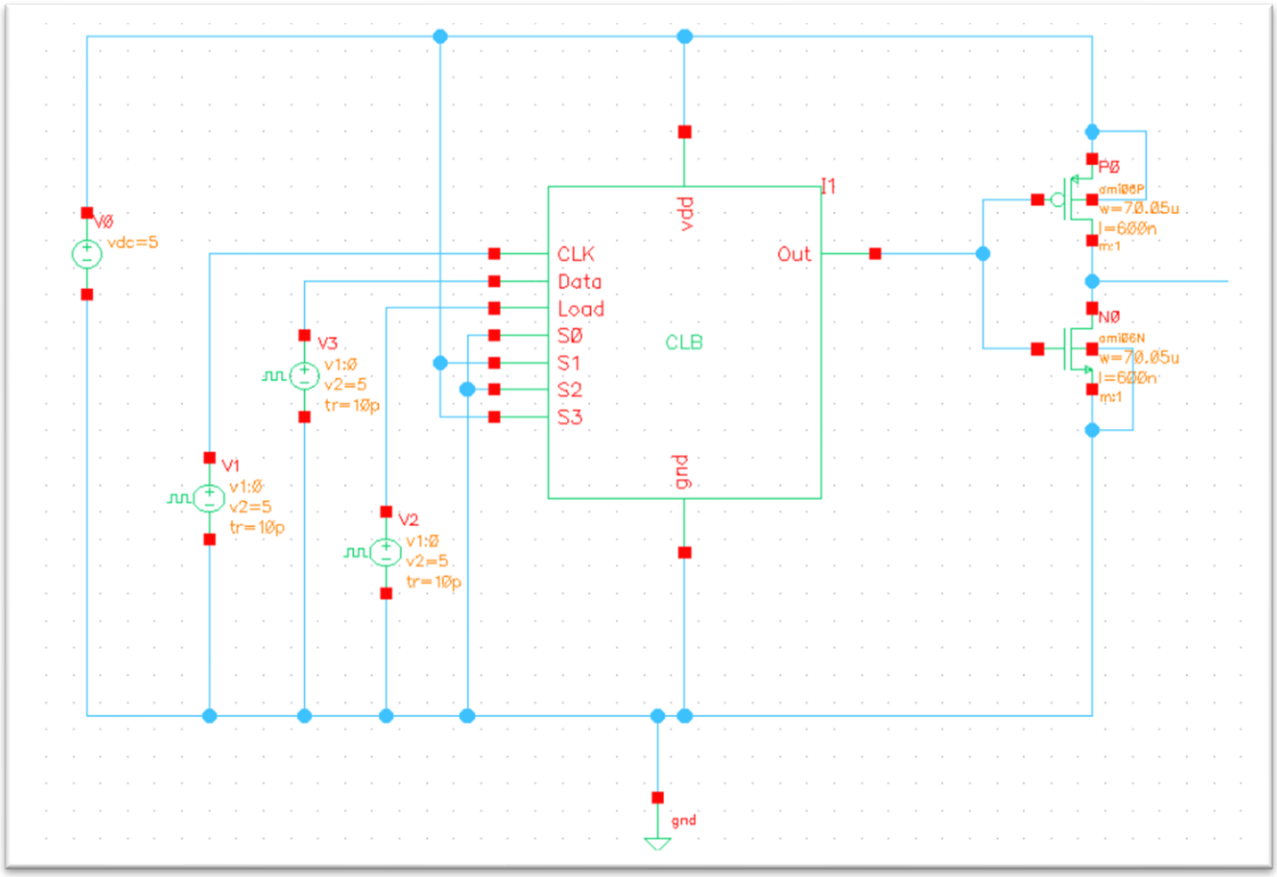


Figure 61 CLB Loading Energy Test Schematic

To measure Loading Energy, we probed the VDD pin of both the Serial In Parallel Out register and the SRAM array. Subsequently, the current derived from the graph was integrated over a 17-cycle time span (16 cycles for loading data into the Serial In Parallel Out register and an additional cycle for writing data into the SRAM array). The resulting energy values from both plots were then aggregated to determine the total loading energy. It's worth noting that the output is loaded with 100Cg, as specified in the handout, and we achieved this using an inverter with PMOS and NMOS widths set to 70µm.

Figure 62 Loading Energy Plot

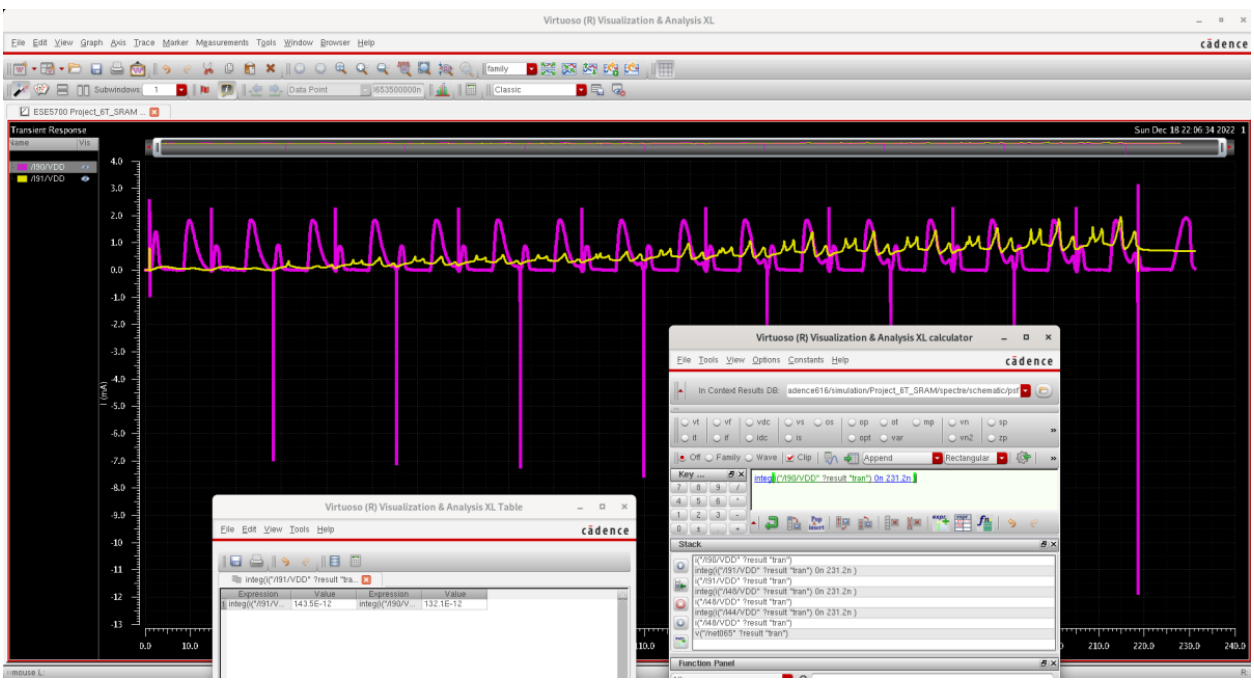


Figure 63 Loading Energy Calculation

The Loading Energy for our Design is given by,

$$\boldsymbol{Loading\ Energy} = \int \boldsymbol{I * Vdd} = (\mathbf{143.5}\ \boldsymbol{pA} + \mathbf{132.1}\boldsymbol{pA}) * \mathbf{5}\boldsymbol{V} = \mathbf{1378}\ \boldsymbol{pJ}$$

Active Energy

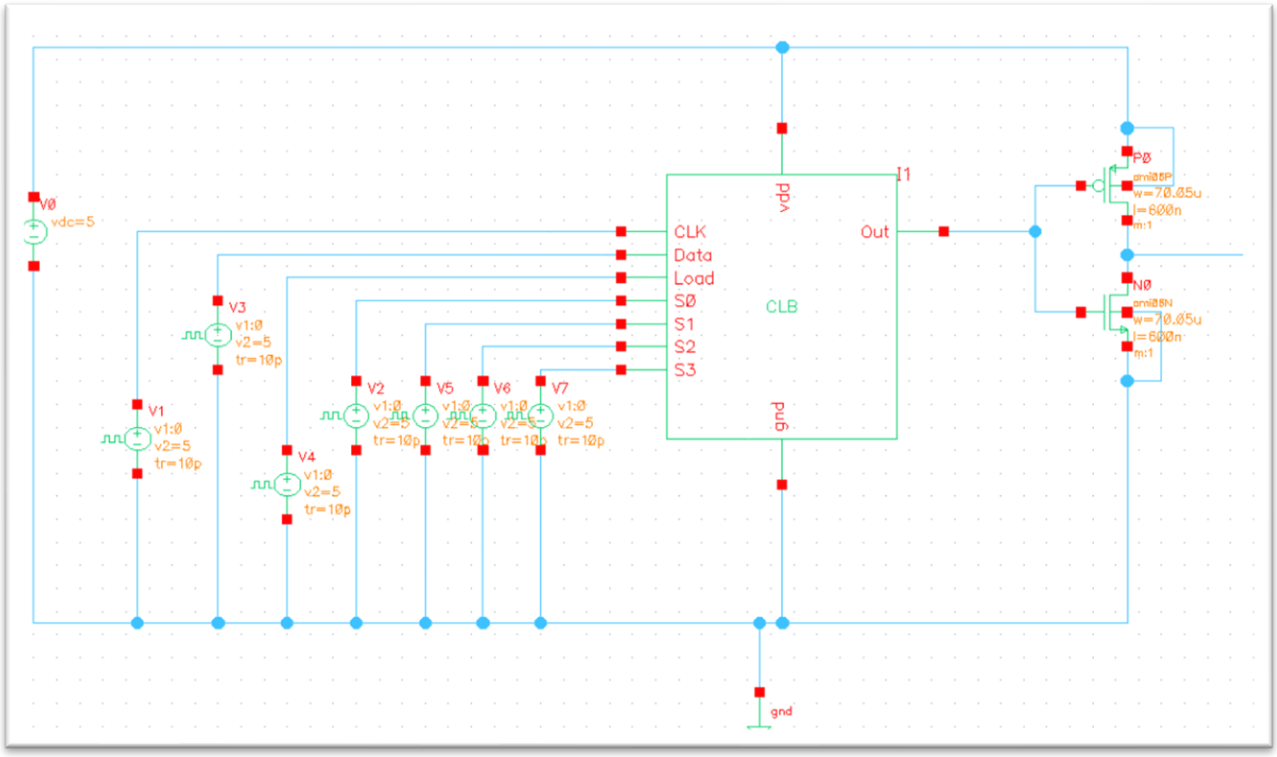


Figure 64 CLB Active Energy Test Schematic

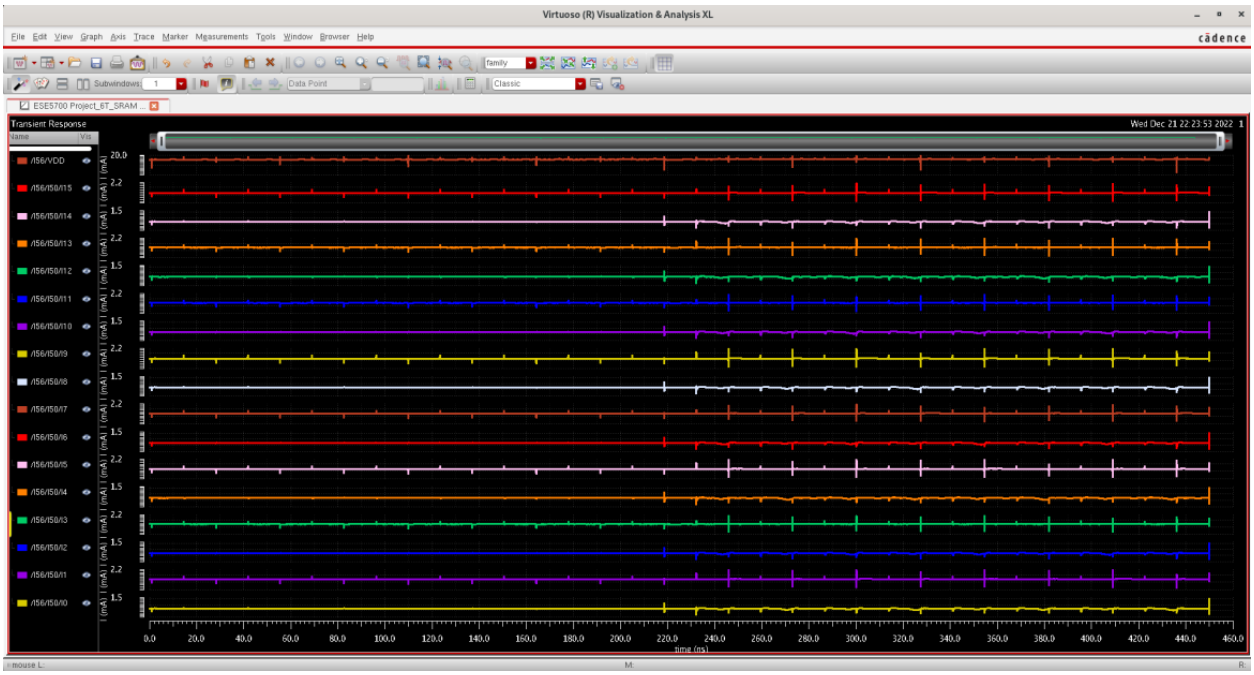


Figure 65 CLB Active Energy Plot

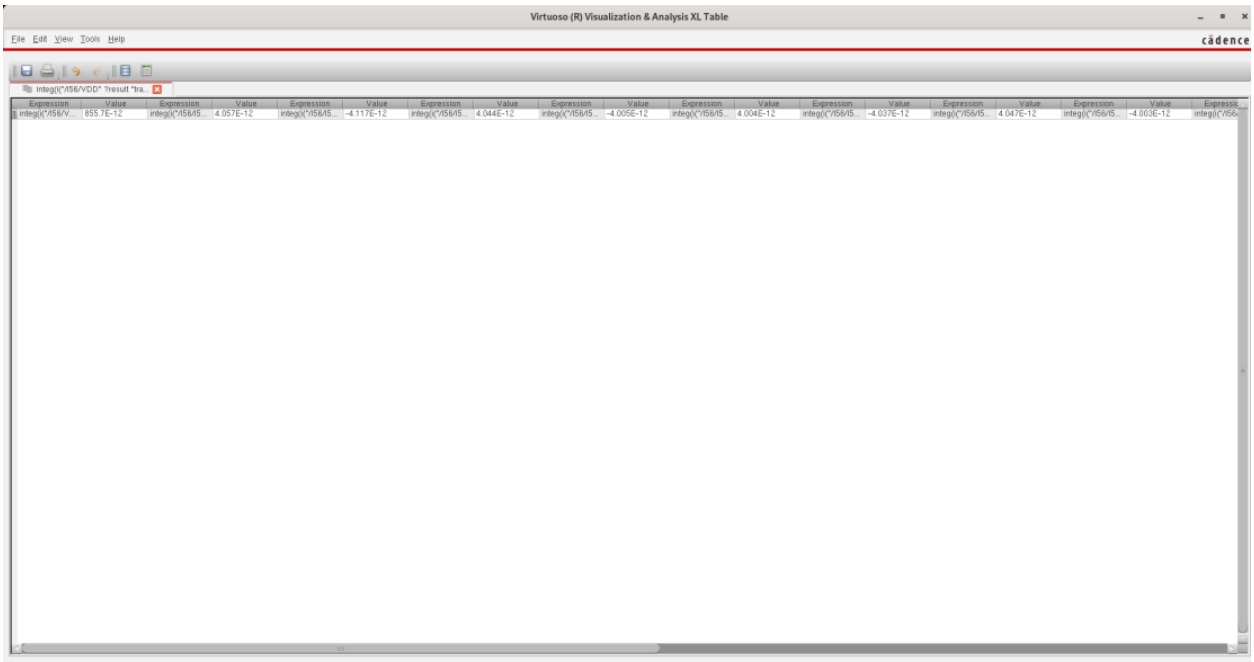


Figure 66 Current Integral Values

$$\boldsymbol{Active\ Energy = \int I * Vdd = (974.432pA) * 5V = 4872.16\ pJ}$$

Average Energy

$$Average\ Energy = (0.8 * Active\ Energy) + (0.2 * Loading\ Energy)$$
$$Average\ Energy = (0.8 * 4872.16pJ) + (0.2 * 1378pJ)$$
$$\boldsymbol{Average\ Energy = 4173.328\ pJ}$$

Maximum Operating Frequency

$$Maximum\ Operating\ Frequency = \frac{1}{Minimum\ Time\ Period}$$

$$Maximum\ Operating\ Frequency = \frac{1}{16.4\text{ns}}$$

$$\boldsymbol{Maximum\ Operating\ Frequency = 60.975\ \text{MHz}}$$

Figure Of Merit (FOM)

$$FOM = Area * Average\ Energy * \frac{1}{Maximum\ Operating\ Frequency}$$

Area = (672.8 * 672.8) µm2

$$FOM = Area * Average\ Energy * \frac{1}{Maximum\ Operating\ Frequency}$$

$$FOM = (672.8\ *\ 672.8)\ \mu m^2 * 4173.328\ pJ * \frac{1}{60.975\ MHz}$$

$$\boldsymbol{FOM = 3.09 * 10^{-23}\ m^2 Js}$$

Summary Table

Loading Energy	Active Energy	Average Energy	Area	Maximum Operating Frequency	Figure of Merit
1378 pJ	$\mathbf{4872.16\ pJ}$	$\mathbf{4173.328\ pJ}$	$\mathbf{(672.8\ *\ 672.8)\ \mu m^2}$	**60.975 MHz**	$\mathbf{3.09 * 10^{-23}\ m^2 Js}$

References

- Multiplexer In Digital Circuit - Circuit Fever
- Virtual lab (vlabs.ac.in)
- Digital Circuits - Multiplexers (tutorialspoint.com)
- Solved: Chapter 5 Problem 51E Solution | Digital Design With Rtl Design, Verilog And Vhdl 2nd Edition | Chegg.com
- Solved: Chapter 5 Problem 51E Solution | Digital Design With Rtl Design, Verilog And Vhdl 2nd Edition | Chegg.com
- D Flip Flop in Digital Electronics - Javatpoint
- D Flip Flop Operation – Positive Edge Triggered | allthingsvlsi (wordpress.com)
- Serial In Parallel Out (SIPO) Shift Register - GeeksforGeeks
- Diagram of the SRAM cell circuit of the write operation. | Download Scientific Diagram (researchgate.net)
- What is SRAM Memory: Static RAM » Electronics Notes (electronics-notes.com)