VLSI SYSTEMS DESIGN PROJECTS - CE 392

NON-VOLATILE LOGIC-IN-MEMORY FPGA DESIGN WITH MTJs

FINAL REPORT

TEAM 1: CAN AFACAN, NANO GOLDMAN, MARK WINICK

ADVISOR: PROFESSOR DAVID ZARETSKY

DATE SUBMITTED: 06/06/2025

ABSTRACT

SRAM-based FPGAs lose their configuration when the power is cut, so they require an external memory, increasing the area, and have a fixed latency upon reset. We propose to design a nonvolatile logic-in-memory LUT composed of two input cells using magnetic tunnel junction technology so that the logic block retains its state when the power is cut. The goal is to simulate a working logic block with nonvolatile LUTs in Cadence Virtuoso and demonstrate in example applications how it could help cut power and improve responsiveness in von-Neumann-based FPGA systems.

Keywords: FPGA, Non-Volatility, LUT, MTJ, Logic-In-Memory

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

The project explores a novel field-programmable gate array (FPGA) architecture utilizing non-volatile logic-in-memory look-up tables (LUTs) based on spin-transfer torque magnetic tunnel junctions (STT-MTJs). Traditional SRAM-based FPGAs that follow the classical von-Neumann architecture require external memory to retain logic configurations, which results in increased area, higher power consumption, and fixed startup latency. In contrast, this design integrates STT-MTJs directly into configurable logic blocks (CLBs) to preserve logical states even when powered off, thereby eliminating the need for separate memory and enhancing energy efficiency.

Our approach centers on developing 2-input and 3-input LUTs using MRAMs with MTJ devices modeled in Verilog-A and simulated in Cadence Virtuoso. We constructed interconnected CLB matrices, including three 9-CLB arrays and one 16-CLB array, to form a modular and scalable FPGA logic fabric. The system was tested in applications such as arithmetic computation and pattern recognition. Directional switch boxes were used to enable flexible signal routing between CLBs, and the architecture supported both synchronous and asynchronous operations.

The design achieves key benefits: reduction in power consumption by eliminating static leakage in SRAM (which constantly draws power to maintain its state, unlike MRAMs, which retain data without continuous power as it stores data as magnetic orientation), improved startup responsiveness due to non-volatility, and layout efficiency through the use of NMOS-based CLBs. Engineering standards such as IEEE 1800.2 (UVM), IPC-2221, and GDSII were followed to ensure simulation reliability and fabrication compatibility. However, challenges include the increased complexity of write operations in MTJs, susceptibility to magnetic probing, and ethical concerns around material sourcing.

In summary, this project demonstrates the viability of a non-volatile logic-in-memory FPGA architecture using MTJs, offering meaningful advancements in power efficiency and architectural simplification, with further opportunities for scalability and real-world deployment through PDK-level fabrication and larger matrix integration.

INTRODUCTION

In this project, we developed and implemented a non-volatile logic-in-memory FPGA architecture that integrates magnetic tunnel junctions directly into the logic fabric. Our primary focus was on designing CLBs whose look-up tables retain their programmed states using spin-transfer torque MTJs, removing the dependence on SRAM and external configuration memory.

Our implementation process began with the modeling of STT-MTJs in Verilog-A (Appendices Section 1), enabling accurate circuit-level simulation of their magnetoresistive behavior. These models were incorporated into both 2-input and 3-input LUT configurations, which were then embedded within custom-designed CLBs. Each CLB was developed to support synchronized logic via embedded D flip-flops and multiplexers, and asynchronous operation when needed. Simulations were conducted in Cadence Virtuoso to validate logic functionality, switching behavior, and integration feasibility.

To evaluate system-level performance and showcase the scalability of our design, we constructed a logic fabric composed of three 9-CLB matrices and a single 16-CLB matrix. 4:4 switch boxes were incorporated into 9-CLB matrices, allowing flexible signal steering between logic blocks. All CLB matrices were synthesized with full schematic and layout support, and their interconnections were designed to be reconfigurable, enabling a wide range of logic functionalities.

Our layout design followed Stockmeyer's Algorithm for hierarchical placement to minimize area while optimizing connectivity between logic modules. In contrast to traditional Manhattan-style routing, we employed Euclidean routing paths in selected segments of the design to minimize parasitic inductance and capacitance that could compromise signal integrity in high-density logic arrangements. This deviation from orthogonal wiring was necessary to maintain timing reliability, especially across critical inter-CLB signal paths.

We chose to showcase a pattern recognition application in our final demonstration, specifically a Hamming distance calculator operating on two 12-bit input numbers. This use case was selected deliberately to ensure that every programmable element in our design, CLBs, switch boxes, and control logic, was utilized in a meaningful way. This demonstration served not only as a validation of circuit correctness but also as a test of the design's flexibility and reconfigurability in a higher-order logic task.

Post-layout parasitic extraction (PEX) was performed on the final assembled matrices to examine the impact of routing and component placement on performance. Parasitic values of capacitance, inductance, and resistance across signal nets were measured and analyzed. High-capacitance nets were identified and investigated, with potential causes traced to densely packed routing segments and suboptimal via placements. These observations guided our suggestions for future iterations,

which include improved routing strategies, enhanced shielding on sensitive nets, and the integration of automated parasitic-aware placement tools.

In summary, our design effort combined custom MTJ modeling, hierarchical circuit design, parasitic-aware layout strategies, and real-world demonstration in a unified FPGA framework. The system supports energy-efficient, non-volatile logic execution and offers a path forward for integrating emerging memory technologies directly into programmable hardware fabrics.

BROADER CONSIDERATIONS

From a commercial standpoint, the feasibility of widespread adoption hinges on both technological readiness and the compatibility of MTJ-based logic with existing semiconductor fabrication ecosystems. Presently, most commercial semiconductor fabrication facilities (FABs) do not support MTJ integration within their standard production lines. MRAM devices that utilize MTJs are typically manufactured in specialized environments, often in isolated production steps. This is primarily due to the use of ferromagnetic metals that include cobalt, iron, ruthenium, etc., which pose contamination risks to conventional CMOS processing, particularly during plasma etching, lithography, and other stages sensitive to metallic impurities. Even trace contamination can compromise yield and device reliability across entire wafer batches, which would essentially make the integration of magnetic materials into mainstream processes a significant logistical and economic challenge. [5]

If our architecture were to be scaled for practical deployment, initial feasibility exists within niche markets where memory capacity demands are modest and the benefits of instant-on functionality and low idle power consumption outweigh memory density concerns. In current MRAM manufacturing, practical integration has reached megabyte-scale capacities. For this reason, logic-in-memory FPGAs of our design could become commercially competitive in domains such as aerospace, embedded control systems, defense electronics, and ultra-low-power edge computing: applications that benefit from rugged non-volatility, compact design, and fast wake-up behavior.

However, to rival existing high-performance FPGAs that follow the von Neumann architecture, particularly those that are used in data centers, artificial intelligence accelerators, and scientific computation, the design would need to scale internal non-volatile memory to the gigabyte level. Achieving such density using MTJs within a logic-in-memory paradigm would necessitate breakthroughs in integration techniques, interconnect routing, and MTJ scaling. Moreover, the overhead introduced by write latency and the energy cost of switching MTJs would need to be minimized or amortized by architectural advantages at larger scales.

In terms of industrial adoption, our design offers potential long-term value as part of a broader shift toward compute-in-memory architectures. As memory bottlenecks and data movement energy costs become dominant in modern workloads, especially in AI and machine learning, the decoupling of logic and memory is increasingly seen as a limiting factor. Non-volatile logic-in-memory architectures like ours present a conceptual alternative that reduces the need for frequent memory fetches, potentially leading to major gains in throughput and efficiency for data-intensive tasks. If integrated successfully, our design could contribute to a paradigm shift in reconfigurable computing, enabling a new generation of FPGAs optimized for power-aware, always-on, and data-centric applications.

On the ethical and regulatory front, several considerations arise from the use of rare and geopolitically sensitive materials. Elements such as cobalt and tantalum, frequently used in MTJ fabrication, are sourced from regions with histories of exploitative labor practices and environmental degradation, such as the Democratic Republic of the Congo. Responsible sourcing, full material traceability, and adherence to international ethical supply chain standards will be crucial for any commercial scaling of MTJ-based technologies. In addition, the non-volatility of MTJs introduces unique security implications. Since configuration data can persist even after power is removed, unauthorized data extraction through physical probing or reverse-engineering techniques could pose risks in sensitive applications. Hardware-level encryption, zeroization circuits, or magnetic shielding may be necessary to mitigate such vulnerabilities.

In conclusion, while our design is conceptually promising and demonstrates significant potential for architectural innovation, its broader impact depends on solving complex manufacturing, scaling, and ethical challenges. Future advancements in magnetic material integration, supply chain transparency, and logic-in-memory standardization will determine whether this class of FPGAs can transition from academic exploration to a disruptive force in commercial computing.

DESIGN CONSTRAINTS AND REQUIREMENTS

The foundation of this MTJ-based architecture lies in the construction of look-up tables (LUTs). Instead of employing SRAM cells, binary values are encoded in MTJs through their resistance states: high resistance for the antiparallel configuration (R_{AP}) and low resistance for the parallel configuration (R_P). This approach significantly reduces static power consumption and yields a saving of 5n - 1 transistors per n MTJs when compared to SRAM cells.

The read and write operations bear similarity to those in conventional LUTs; however, they operate via a mechanism known as spin-transfer torque (STT). When a current flows through the thicker fixed layer, it becomes spin-polarized. Directed into the thinner free layer, this polarized current transfers angular momentum, causing the free layer's magnetic orientation to switch. The CoFeB/MgO/CoFeB stack shown in Figure 1 enables perpendicular magnetic anisotropy, which stabilizes the magnetic states and enhances switching efficiency between the parallel and antiparallel configurations essential for data storage and retrieval [1].

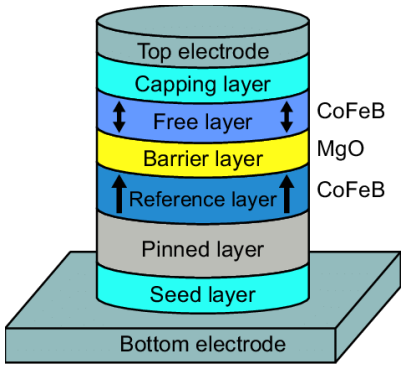


Figure 1: *Diagram of the MTJ stack structure utilizing interfacial perpendicular magnetic anisotropy (PMA) in a CoFeB/MgO/CoFeB configuration. [3]*

One of the foremost challenges in the implementation of spin-transfer torque magnetic tunnel junctions (STT-MTJs) within logic-in-memory systems pertains to the fundamental mechanism of read and write operations. STT-MTJs encode binary states via parallel (P) and anti-parallel (AP) magnetic configurations as described above, which correspond to distinct resistance levels. While read operations are conducted by passing a minimal sensing current through the device, sufficient only to observe the resistance without altering the magnetic state, write operations necessitate the application of a more substantial current over an extended duration to overcome the magnetic energy barrier and induce state switching. This introduces a non-negligible temporal overhead due to the inherent resistance hysteresis and switching stochasticity, which must be meticulously captured in Verilog-A models. Accordingly, the project required a rigorous literature review and precision-driven modelling to simulate the true magneto-electronic

dynamics and possible resistance values of STT-MTJs. Though the increased write latency represents a notable drawback, it is mitigated by the architectural advantage of eliminating the traditional von Neumann bottleneck between logic and memory domains. Furthermore, read operations remain effectively instantaneous, constrained only by the intrinsic RC time constants of the transistors, provided that the sensing current remains well below the critical threshold for unintended switching.

Another considerable challenge is the attainment of reliable and functional simulations for the entire 9-CLB matrix (Figure 2), including its associated switch boxes. The complexity of routing, combinatorial timing paths, and the management of inter-CLB logic propagation required precise structural and behavioural modelling. Ensuring that each CLB's internal logic, including its LUT, DFF, and multiplexing subsystems, performed as intended under variable input vectors demanded numerous iterative refinements to both the circuit topology and simulation stimuli. Moreover, synchronising the switch box logic to dynamically steer data paths across the matrix introduced additional design-layer dependencies that had to be tightly coordinated with the CLB logic states.

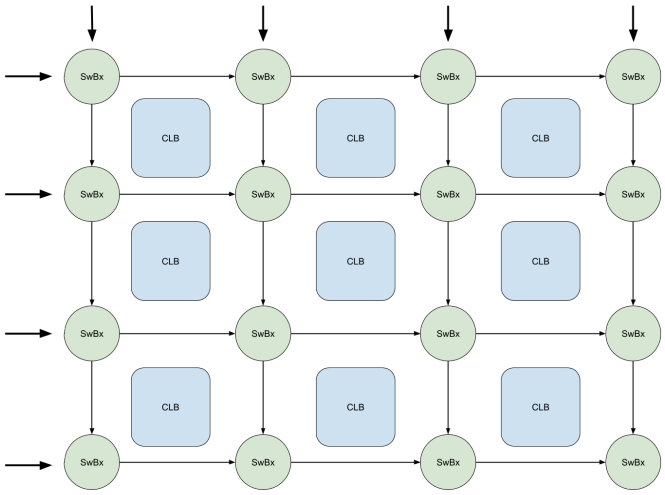


Figure 2: *Block diagram of a 9-CLB, 16 switch box matrix*

The expansion of 3:1 CLBs to a full 16-CLB matrix, and connecting 3 9-CLB matrices with a 16-CLB matrix (Figure 3), incorporates a higher degree of interconnectivity and a greater number of switch boxes, further compounds the simulation difficulty. This larger architecture requires extensive netlist organisation, as well as enhanced simulation runtime management, to handle the elevated scale of logic operations and signal propagation. Each additional CLB introduced exponential growth in the complexity of verification, necessitating a hierarchical simulation strategy and incremental validation of logic islands before integrating them into the global matrix. Achieving convergence and stability in these simulations, especially with the

MTJ-based resistive states embedded within the logic fabric, proved to be a resource-intensive but essential phase of the project.

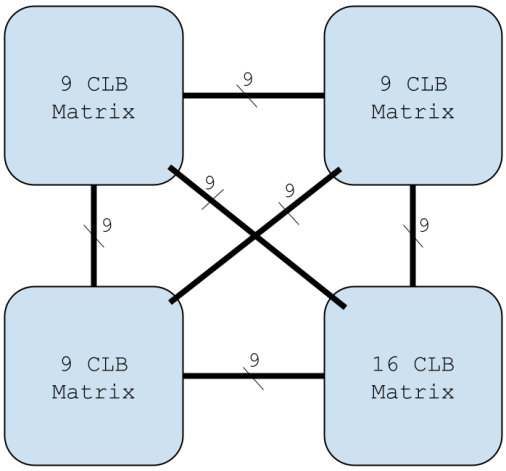


Figure 3: *Block diagram of the final 4-matrix architecture, including 3 9-CLB matrices and one 16-CLB matrix*

A final and critical aspect of the simulation process involves the planning and structuring of input vectors, referred to here as the vbit stream, as our Virtuoso simulation schematics use vbit components to encode discrete input signals. For the system to be meaningfully evaluated, testbench scenarios had to be crafted with foresight into logic dependencies, CLB interconnections, and MTJ state transitions. This required an understanding of the digital logic behaviour and also the sequential timing constraints imposed by the magneto-resistive elements. Developing a semi-automated analog input device using Verilog-A to generate and apply these inputs allowed for more efficient exploration of the FPGA's operational space and facilitated the demonstration of target applications such as arithmetic computation and pattern recognition.

In order to control and refine the overall power consumption of our device, each subcomponent (e.g., LUTs, switch boxes, DFF, etc.) will have its overall current probed immediately downstream from its voltage source over periods of switching activity and computation cycles using scripts to not disturb the existing voltage probes. This will be done to analyze the power consumption of different discrete operations. Transistor sizing and logic arrangements will then be adjusted and optimized to decrease this transitory power consumption while maintaining adequate timings and voltage thresholds.

DESIGN DESCRIPTION

The system architecture comprises a fabric of interconnected CLB matrices designed to support configurable logic operations. Specifically, two primary matrix configurations are implemented: a 3x3 array of 2-input CLBs (referred to as a 9-CLB matrix, Figure 4) and a 4x4 array of 3-input CLBs (referred to as a 16-CLB matrix). 9-CLB matrix is coupled with corresponding sixteen switch boxes to enable flexible routing and logic reconfiguration. These matrices are integrated into a larger structure to form a unified logic fabric capable of demonstrating complex operations such as arithmetic computation and pattern recognition. The architecture supports both sequential and asynchronous operation modes through peripheral control circuitry embedded within the matrix [4].

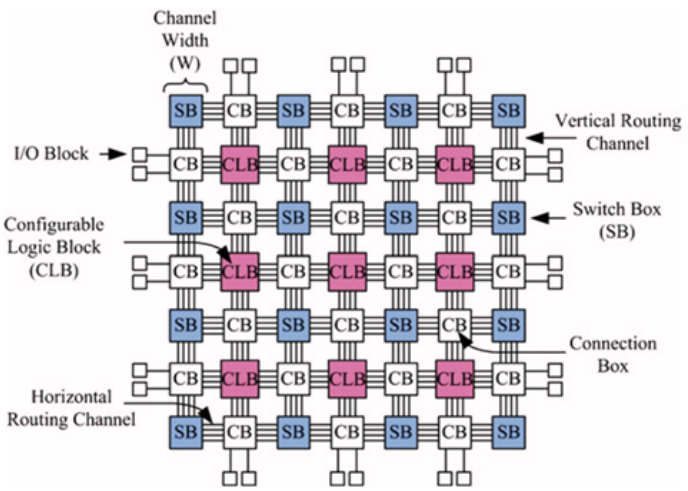


Figure 4: *Block diagram of conventional 9-CLB, 16 switch box matrix architecture (I/O blocks indicate the multiplexer selection inputs that go to the switch boxes as shown in Figure 5, connection boxes are abstracted wiring routes) [2]*

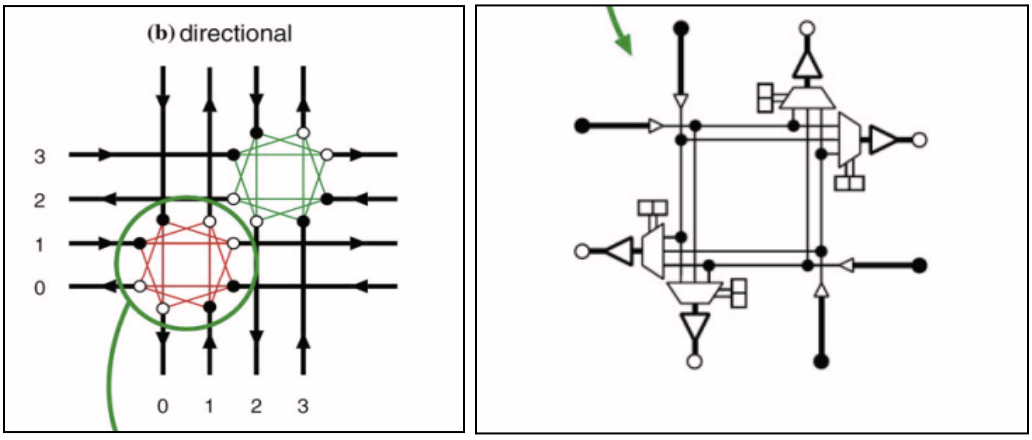


Figure 5: *Switch box design and block diagram set in full matrix [2]*

The switch box functions by using control bits, in our case S0 and S1, to control at any given time which inputs are passed to which output rails. The purpose of such a system is to route signals from one CLB to another to create complex systems without changing the function of a CLB. In a 4-input:4-output switch box, each input is connected to three possible output tracks via three input MUX blocks, preventing a signal from being passed back to the location it was just transmitted from. In our design, we will use 4-input:4-output switch boxes in the 2:1 CLB matrix.

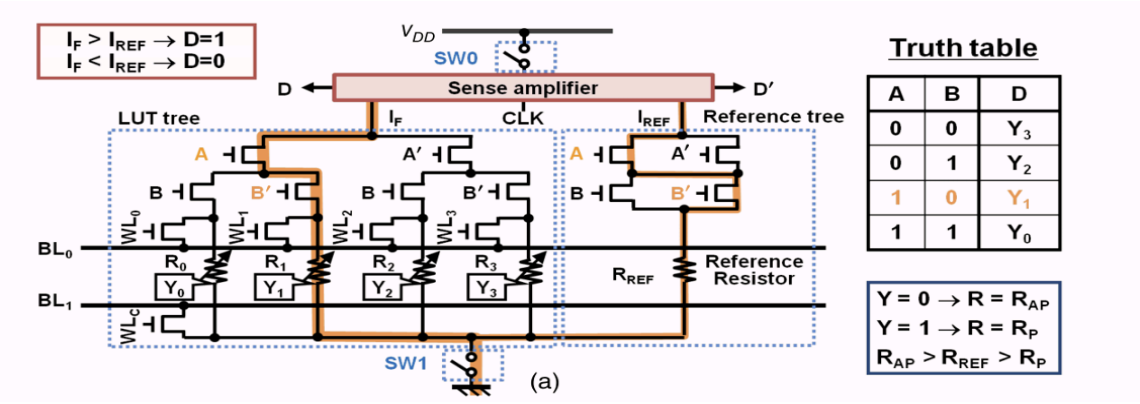


Figure 6: *An annotated schematic showing the MTJ's integration into a 2-input LUT block [1]*

Figure 6 illustrates the transistor-level schematic of a simplified 2-input LUT cell intended to demonstrate the basic working mechanism.

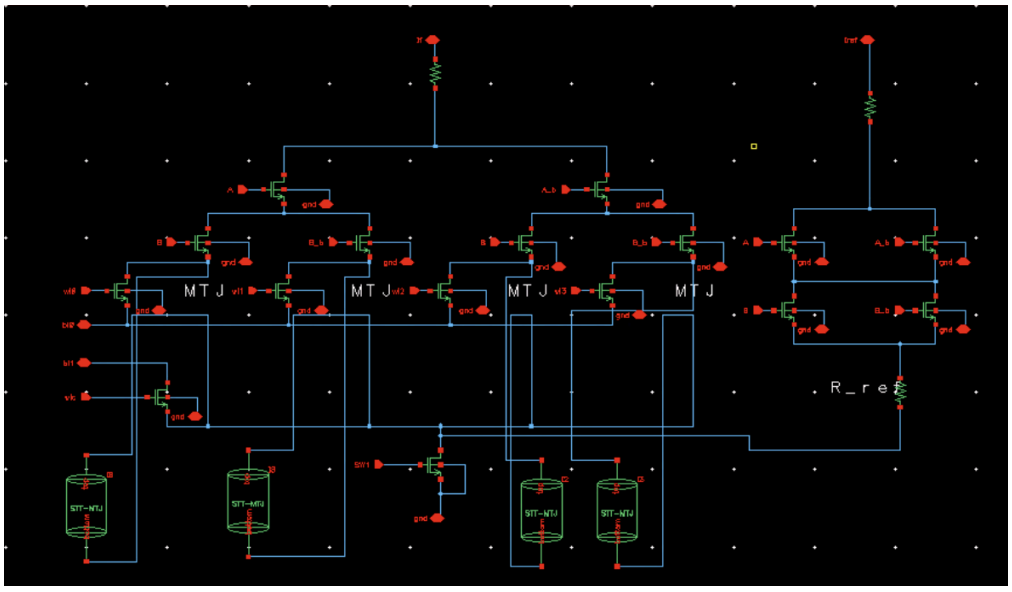


Figure 7: *2:1 STT-MTJ LUT schematic*

Figure 7 is the realization of the schematic of Figure 6 in Cadence Virtuoso, without the added sense amplifiers.

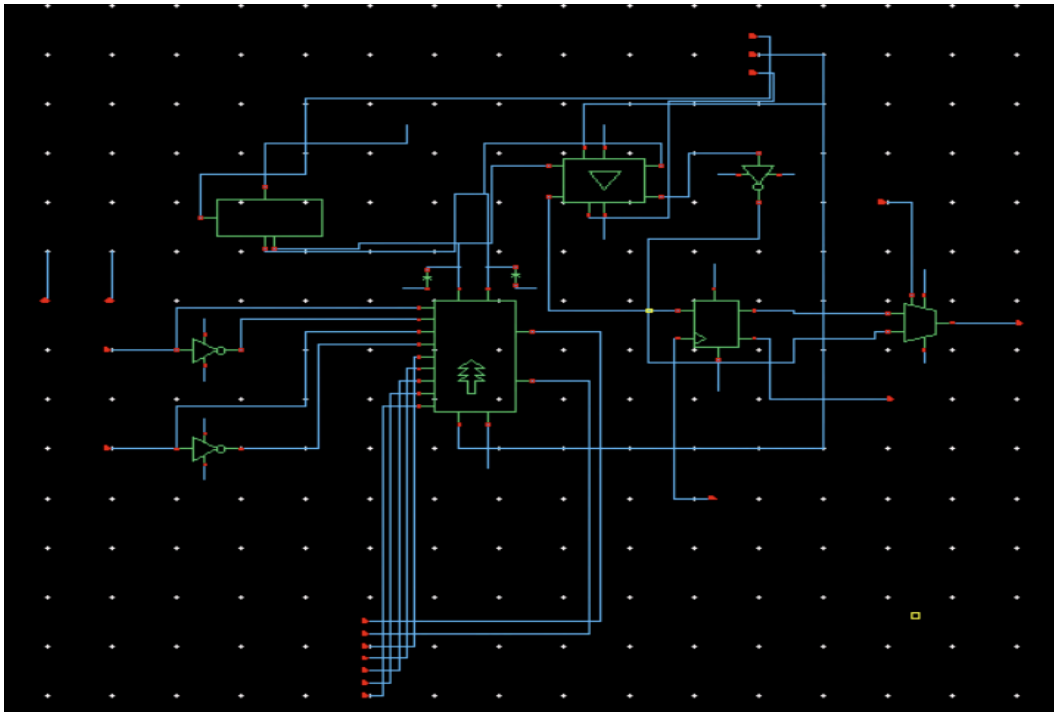


Figure 8: *2:1 CLB design*

In Figure 8, the top left box is a bitline conditioner, the two inverters for inverting the necessary 2 inputs where needed in the 2:1 LUT, the DFF and 2:1 MUX for controlling synchronicity, and a sense amplifier pictured at the top with an arrow label for detecting differences in voltages of the bitlines through which the reference and path finding currents (the path through the tree which corresponds with a particular truth table output) are compared. A multiplexer is added at the end of the path so that the users will have the chance to choose between previous and current inputs based on their synchronization needs in their circuits.

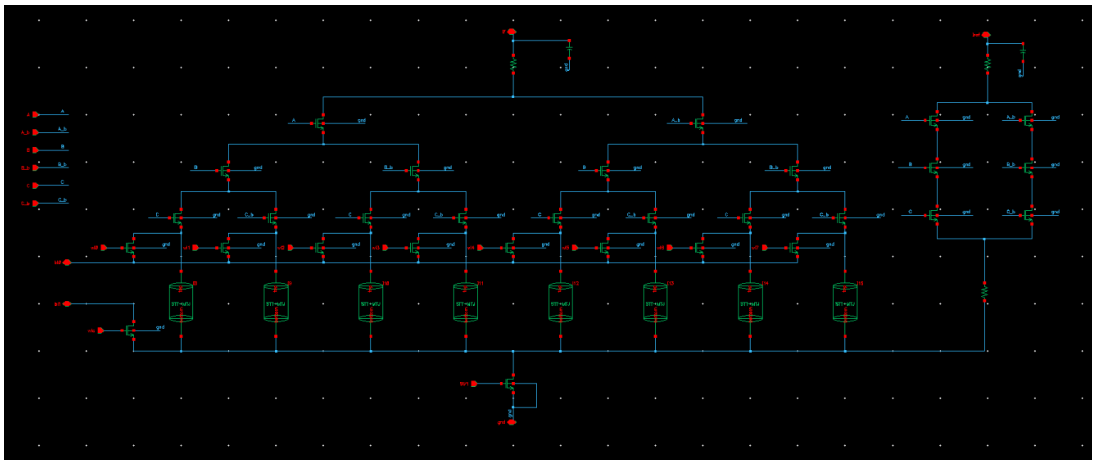


Figure 9: *3:1 STT-MTJ LUT schematic*

As shown in Figure 9, the 3:1 LUT design is an extension of the 2:1 LUT, with transistor widths increased to mitigate the higher resistance associated with an added layer of CMOS transistors

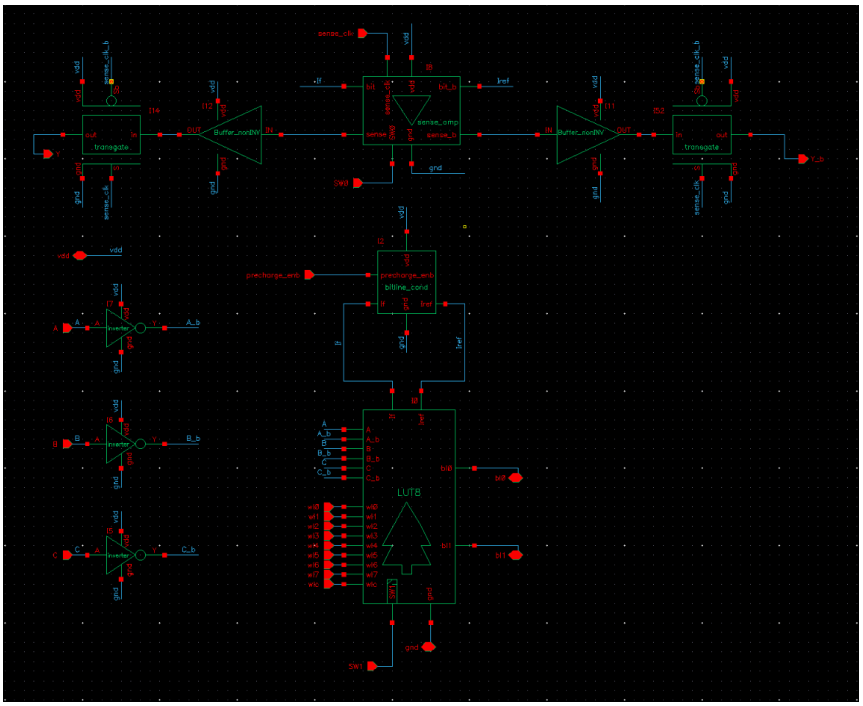


Figure 10: *3:1 'NVLUT' design*

3:1 NVLUT design (Figure 10): 3 Inverters on the left included for their necessary application in the LUT, the LUT (pictured bottom center), a bitline condition (in the center), and a sense amplifier at the top. The added features, which resemble SRAM in structure, are necessary for generating a reference current, I_{ref}, with which to compare against the current stream, I_f, fanning out to each of the MTJs and generating the associated truth table response from the highest current pathway (truth table match for the set input). **Note: the naming convention for "NVLUT" is that it includes the basic "LUT" as well as read and write circuitry. Both are nonvolatile by their nature of using MTJs; the adjustments of names are just to separate and modularize functionality.** Additionally, after refinements during the midway presentation, the sense-amplifier's bitline outputs were loaded equally with non-inverting buffers and transmission gates so as to stabilize the output and ensure the output signals are not affected by unequal or too large of a load.

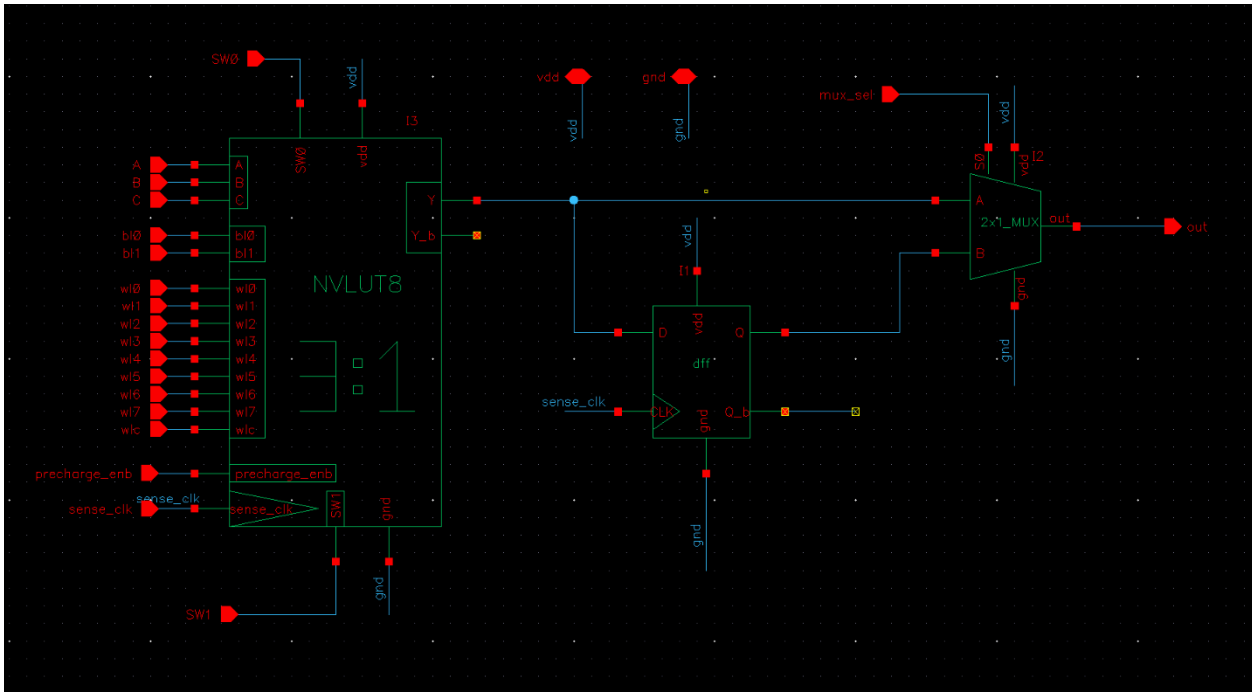


Figure 11: *Full 3:1 CLB schematic with NVLUT, DFF, and 2:1 MUX*

Figures 12 and 13 show the schematic and layout of the 9-CLB matrix composed of 2:1 CLBs and 4:4 switch boxes. The schematic captures the logical interconnections between CLBs through directional switch boxes for configurable routing. The layout implements this structure with proper floorplanning, pin accessibility, and spatial separation between logic and routing blocks. This matrix served as the base for early functionality tests and layout-level parasitic extraction. It uses both Manhattan and Euclidean style routings for parasitic and area minimization. The components stacked on the rightmost side of Figure 13 are the 2:1 LUTs, and the components ordered on the left, with a routing bus space in between, are the switch boxes.

Figures 14 and 15 present the schematic and layout of a 16-bit ripple-carry adder built using 3:1 CLBs. This adder was used in our final demonstration. Figure 16 shows the top-level layout representing the components used in our final demonstration, which combines three 9-CLB matrices and the 16-bit CLB-adder matrix. Placement followed Stockmeyer's algorithm, with outer pins treated as fixed boundary blocks, and routing arranged to isolate magnetic components.

Layouts of individual sub-components are provided in the Appendices Section 4.

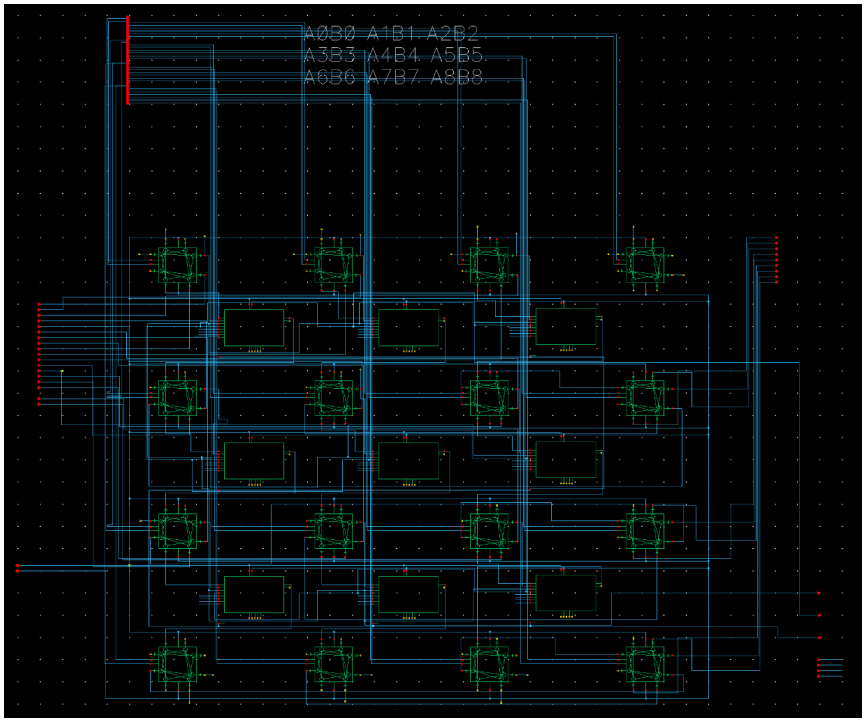


Figure 12: *Schematic of 9-CLB Matrix with 2:1 CLBs and Switch Boxes*

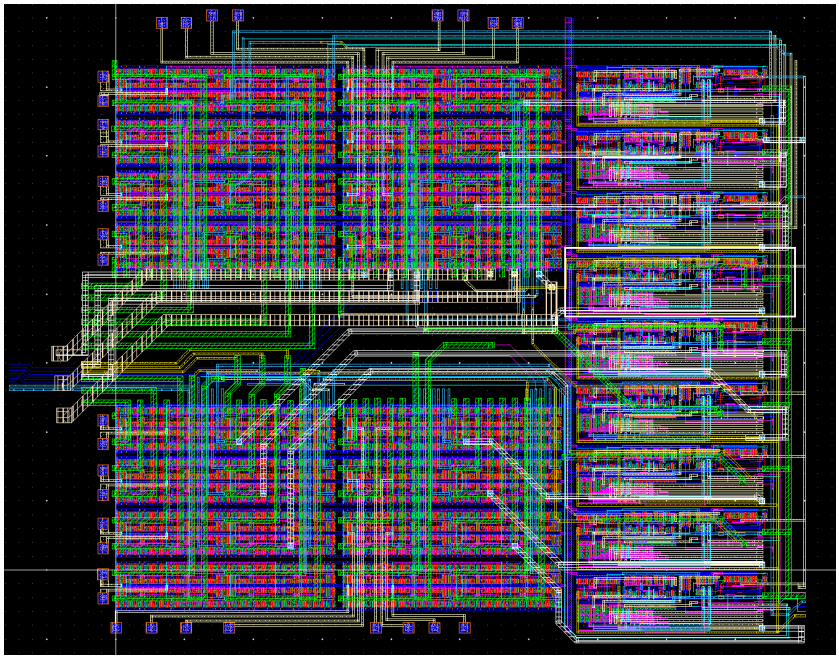


Figure 13: *Layout of 9-CLB Matrix with 2:1 CLBs and Switch Boxes*

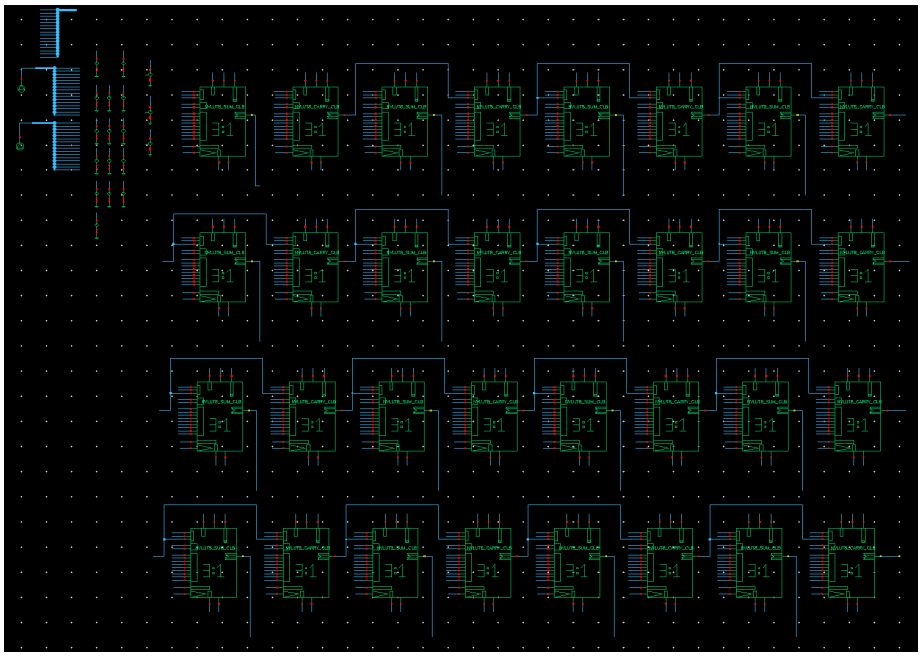


Figure 14: *Schematic of a 16-bit Adder with 3:1 CLBs*

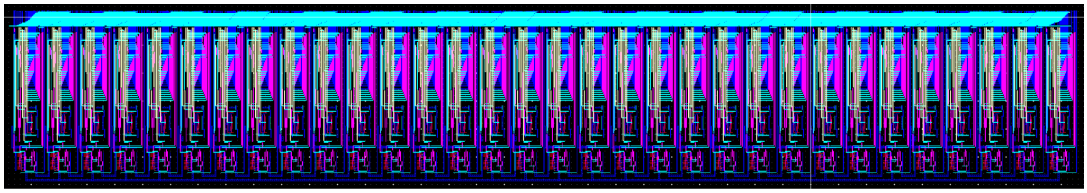


Figure 15: *Layout of 16-bit CLB-based Adder with 3:1 CLBs*

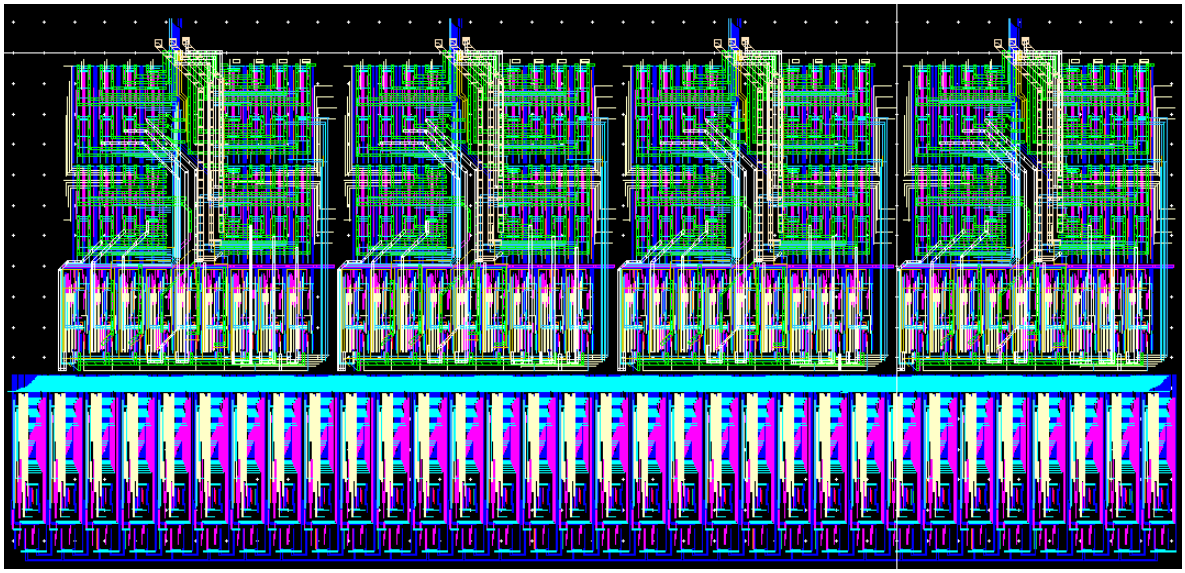


Figure 16: *Layout of Final Simulation Design with 9-CLB Matrices and 16-bit CLB-based Adder*

DESIGN OPTIMIZATIONS

With such a complex final implementation, design optimization was necessary from the beginning to ensure the optimal operation and sizing. From this idea, we set a standard sizing for transistors, 600nm for pMOS and 400nm for nMOS, for the passive components such as the multiplexers and buffers. This resulted in a standard component size, power, and ground rail separation of 1.82µm. With this determined standard sizing, the rest of the passive components were designed to fit within the rail gap, which necessitated the use of multi-finger transistors for the logic gates. Figure 32 displays the tight packing of logic gates and buffers to create a 4:4 input-output switch box with minimal area use due to the standard sizing of transistors to fit within the same rails.

The sizing of the passive components provided a basis for the transistors in the LUTs and CLBs. Through testing and simulation results, we further altered the sizes of the pull-up and pull-down transistors in the LUTs and MTJs to increase performance. We found that slightly increasing the transistor sizes resulted in a noticeable improvement in the performance of the LUTs and MTJs. We also further adjusted the resistance values of the MTJ states to achieve the desired performance through testing results. The layout of the sense amplifier, while complex, adheres to the rail gap determined by the components presented earlier, which was achieved through the continued use of multi-finger transistors and the use of higher metal layers to achieve connections around existing transistors.

In the final 9-CLB matrix layout design, the layout was optimized through the use of adding a small gap between sets of switch boxes. This gap allowed for input traces which would have otherwise been placed in multiple layers around the perimeter of the component, to pass through the middle and collect on the left side. This allowed for increased separation between magnetic components and inputs, reducing the possibility for the strong magnetic fields of the MTJs to interfere with the input signals of the CLBs.

ENGINEERING STANDARDS

Simulation and Verification - IEEE 1800.2 (UVM)

Although originally intended for digital verification, we referenced the structure and principles of IEEE 1800.2 (Universal Verification Methodology) to organize and scale our analog testbenches. By adopting a modular, hierarchical approach, we were able to test each CLB and switch box component independently before integrating them into larger matrices. Our MTJ devices were modeled in Verilog-A, and simulations were carried out using Cadence Virtuoso.

Transistor Modeling - NCSU Standard Library and BSIM Models

To maintain device-level accuracy, we employed the NCSU standard cell library along with BSIM (Berkeley Short-channel IGFET Model) transistor models. These models are widely used in academic and industrial contexts for SPICE-level simulations. We validated both individual and integrated circuit blocks using these models, confirming logic behavior, timing characteristics, and signal integrity under varied loads and switching activity.

Layout Design - GDSII Format and IPC-2221 Compliance

All physical layouts were completed in GDSII format, the industry standard for mask generation and fabrication workflows. Layout rule checks (DRC) and layout-versus-schematic (LVS) tests were performed to ensure structural accuracy and design compliance. In addition, we applied guidelines from IPC-2221, a widely adopted PCB design standard, to verify our pin routing compatibility with printed circuit board designs and conventional packaging styles.

Placement Optimization - Stockmeyer's Algorithm

For hierarchical placement of layout components, which include CLBs, switch boxes, and peripheral logic, we implemented a variation of Stockmeyer's algorithm. Our approach treated each I/O pin as a discrete block that needed to be positioned along the perimeter of the design area, ensuring routability and minimizing congestion. This strategy enabled us to maintain a balanced layout, reduce critical path lengths, and improve wire accessibility without relying on brute-force or manually constrained methods.

Parasitic Extraction - PEX Analysis

Following the physical layout, we performed PE to analyze resistance and capacitance across key nets. These extracted values were then used to evaluate timing and signal degradation effects, with particular attention paid to high-capacitance nodes. Where possible, we identified contributing factors such as dense routing intersections or overextended nets and documented these as optimization targets for future iterations.

In applying these engineering standards, we ensured that our design is both functionally correct and aligned with physical design practices that are implemented in modern FABs.

TESTING AND SIMULATION

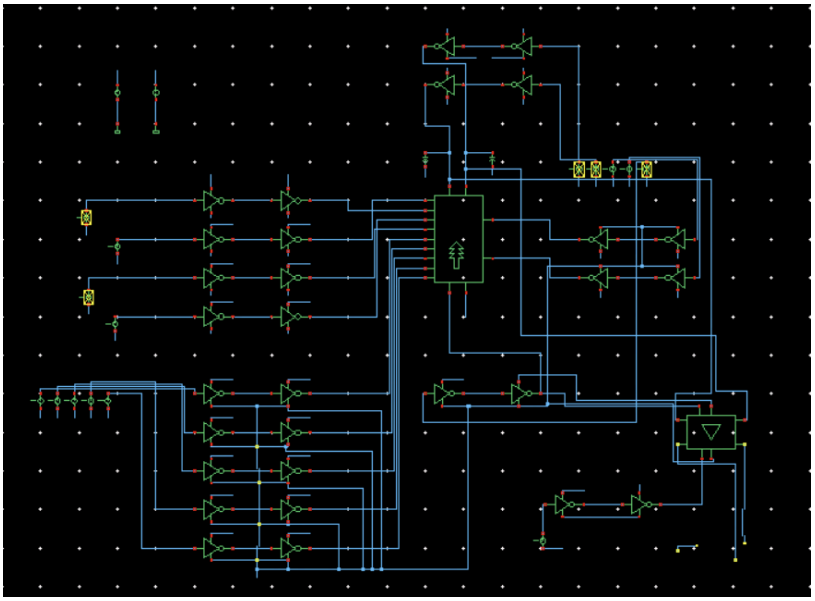


Figure 17: *2:1 LUT testbench design*

In Figure 17, each input of the 2:1 LUT is loaded by 8 inverters (4 in series with 4 to not alter vbit test bench values), and each output by the same quantity. A vbit stream is generated and the corresponding output compared in order to demonstrate correct functionality of the programmed read and write values. The first two MTJs are written as 0 and 1, respectively. Bitlines are manually driven high, and a sense amplifier is included.

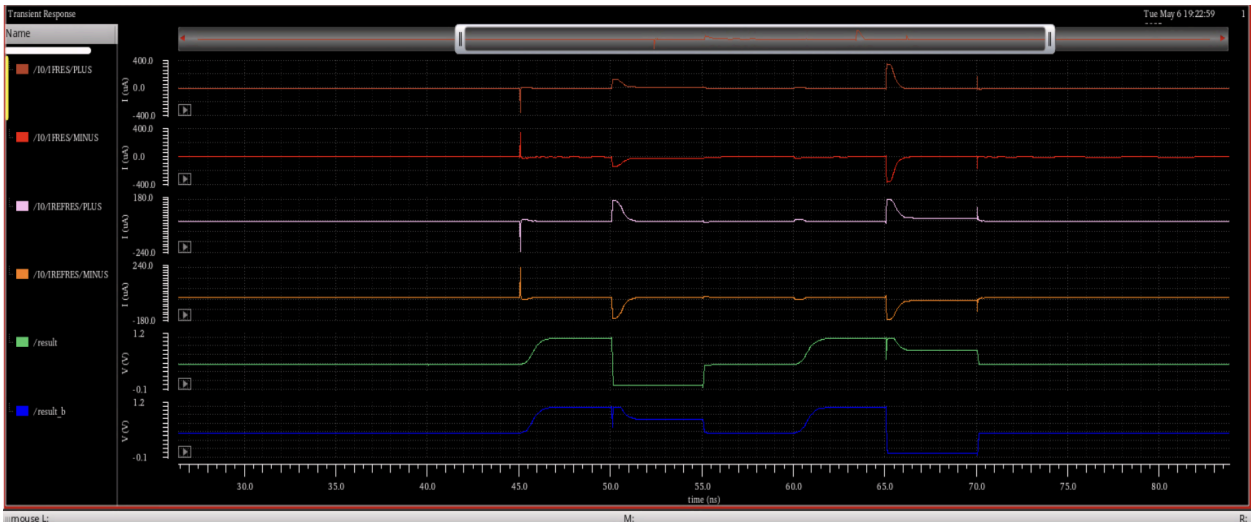


Figure 18: *2:1 LUT testbench waveforms*

From Figure 18, we see that when values are read, a current difference exists between the reference resistance (pink waveform) and the resistance of the MTJ (brown waveform). Currents

are shown in µA in brown, red, pink, and orange waveforms. Green (result) and blue (inverted result) waveforms (V) show the output of the sense amplifier.

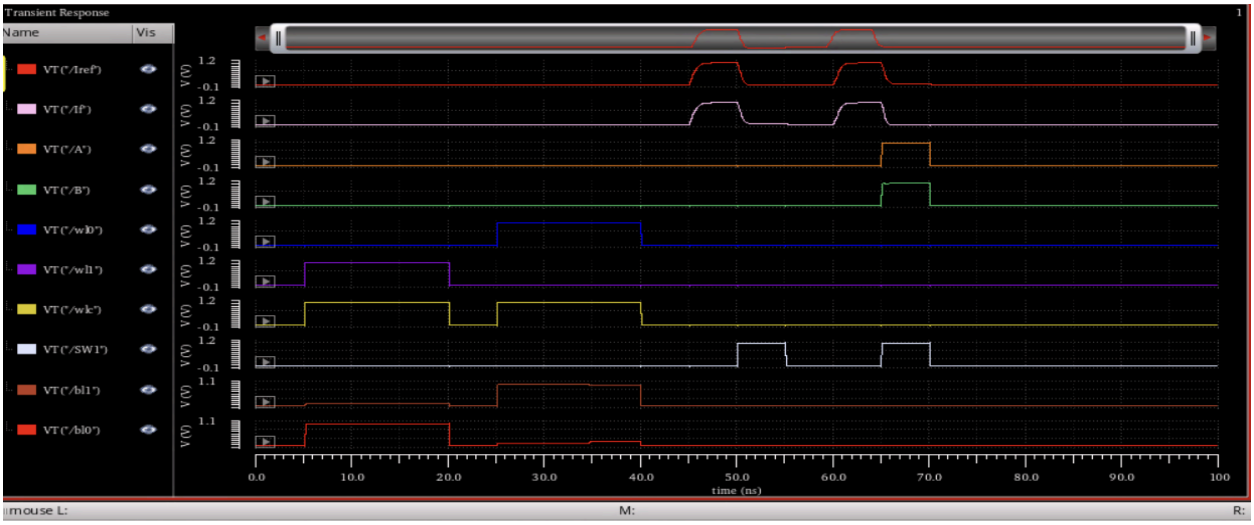


Figure 19: *2:1 LUT testbench waveforms*

Figure 19 shows what external commands are needed for read/write operations in an LUT. We see that the direction of the flow of the current between bitlines correctly encodes the states. Based on the bitlines, necessary wordlines, depending on which MTJ is going to be written, are turned on with a high gate signal (as LUTs use NMOS logic) alongside the common worldline. These allow the current to flow in one direction and thus set the spin of the free layer when current passes for a long enough time, in this case, above 9.5 nanoseconds of current flow changes the spin of the MTJ.

For instance, we see that at the second write instance, bl0 is low and bl1 is driven high, thus the current must flow from bl1 to bl0. For the current to pass from the desired MTJ, wl0 (corresponding to the second MTJ) is turned on. And during the read operation, when both A and B are equal to 1, the first MTJ is along the path, and its resistance is compared to the reference resistor. In our case, making the current flow from bl1 to bl0 puts our MTJ defined in Verilog-A into parallel state, thus 1 is read as a result from our sense amplifiers, as shown in the green waveform in Figure 19.

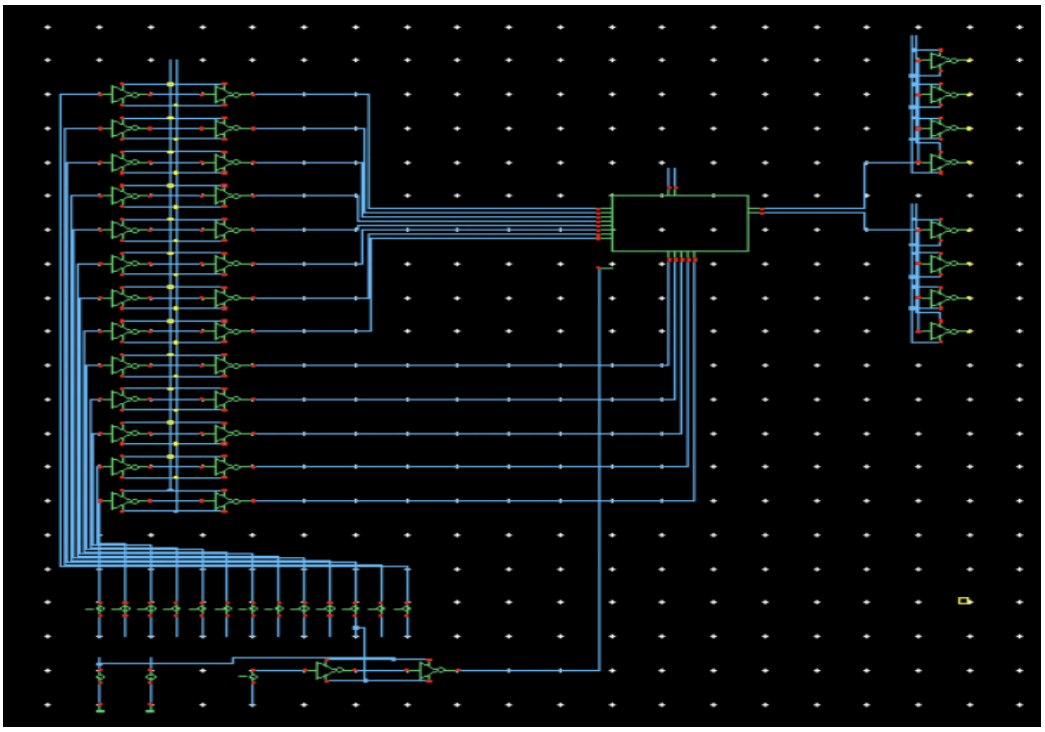


Figure 20: *2:1 CLB test bench schematic*

As shown in Figure 20, each input on the 2:1 CLB was loaded with 8 inverters (4 inverters in series with another 4), with the 2 outputs (the output and its complement) each loaded by 4 inverters to simulate a hypothetical usage scenario. The 2:1 CLB was preprogrammed with the XOR function, and each of its inputs had a vbit test stream to check whether a dynamic range of inputs matched their respective output.

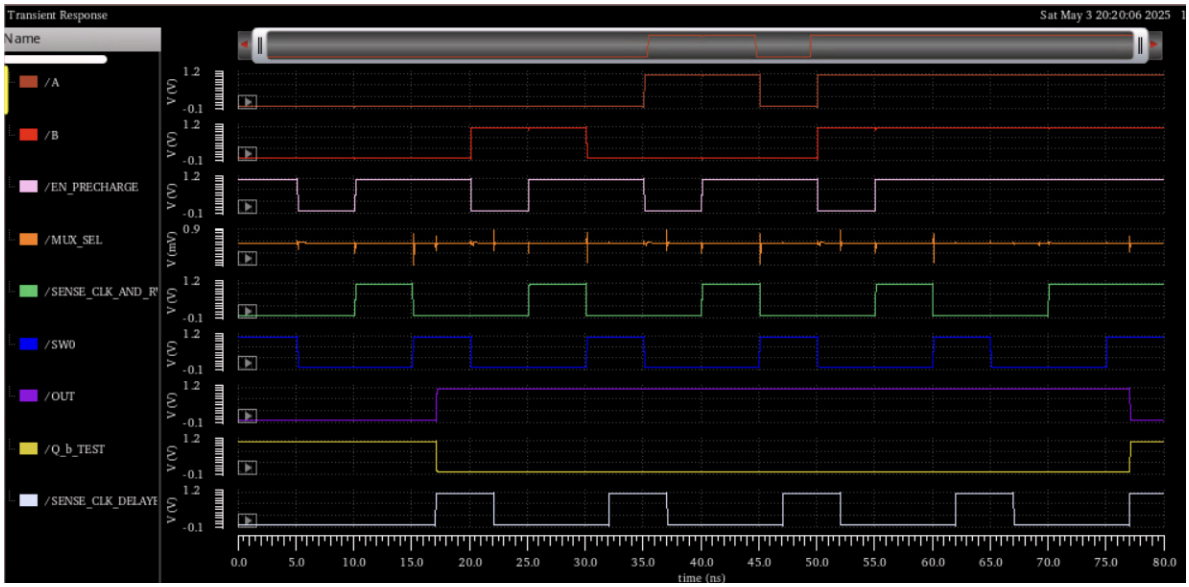


Figure 21: *2:1 CLB XOR testbench waveforms*

In Figure 21, as can be seen in the waveform chart, when “A” and “B” are zero, “OUT” correctly has a zero. When exclusively either of them has a one and the other a zero, the “OUT” is set to one, and finally, when both inputs are set to one, “OUT” is zero. Note that due to the presence of the DFF in the CLB, the output has a slight delay. This can be seen upon the last transition at 50ns, which is reflected in the output after DFF has two rising edge triggers due to the temporal placement of the inputs with respect to the SENSE_CLK.

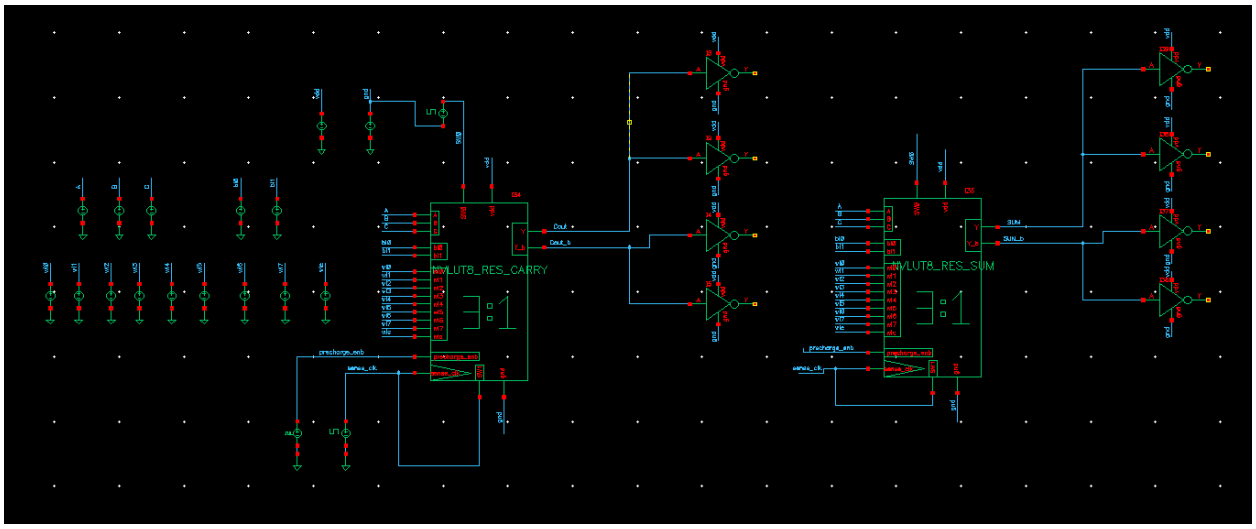


Figure 22: *Two 3:1 NVLUTs (full adder) testbench schematic*

Figure 22 shows two 3:1 NVLUTs in parallel, with their outputs loaded with inverters to perform a pre-programmed full adder test bench scenario

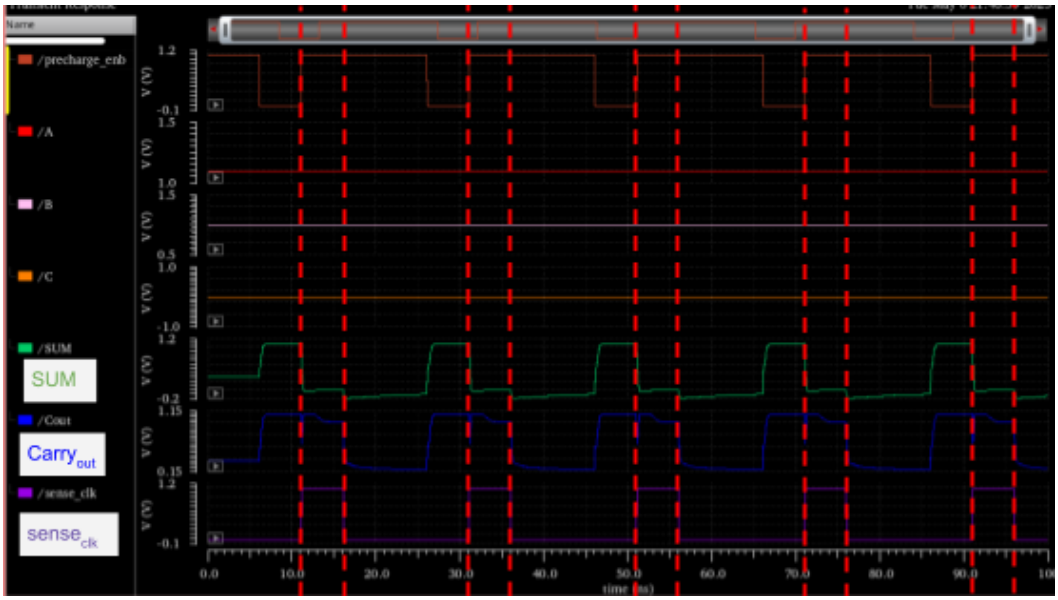
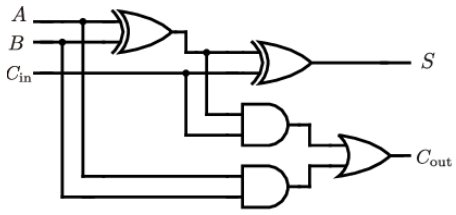


Figure 23: *Two 3:1 NVLUTs (full adder) testbench schematic*



Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 24: *Reference truth table for full adder functionality*

Figures 22, 23, and 24 above illustrate the testing scenario of the 3:1 pre-programmed NVLUT schematics. In order to demonstrate the increased abilities brought on by designing a LUT with more inputs, a full adder was created with one NVLUT programmed with the carry operation, and another with the sum operation. The waveform in Figure 16 above illustrates the case of inputting "110" for "A,B,C_{in}", which can be seen to generate a 0 for the sum and 1 for the carry, just as prescribed by the above truth table.

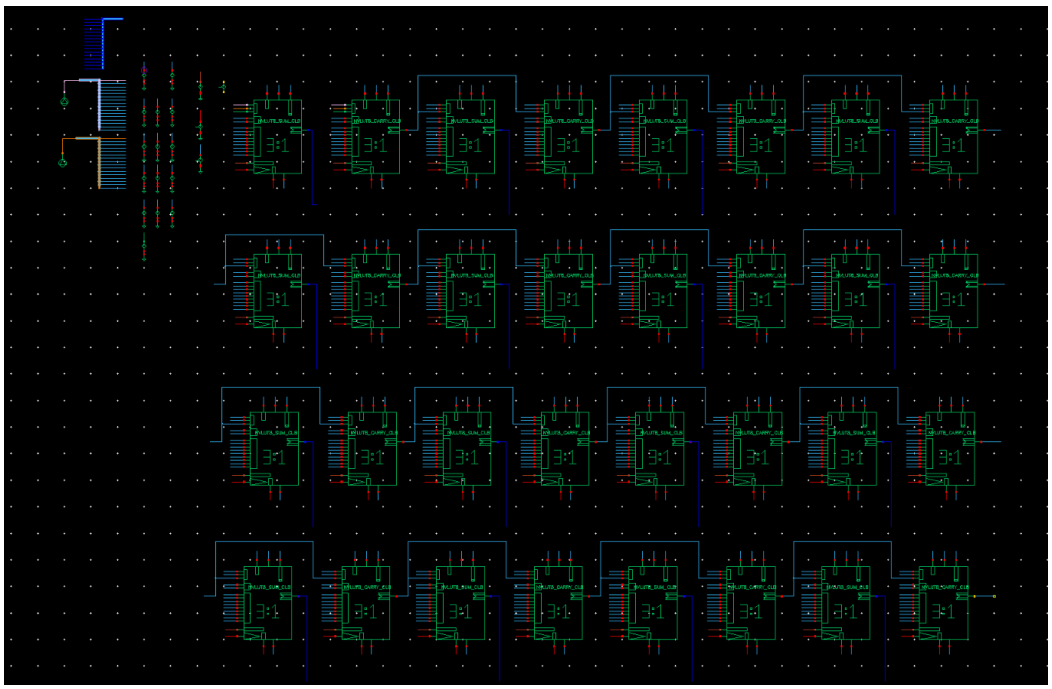
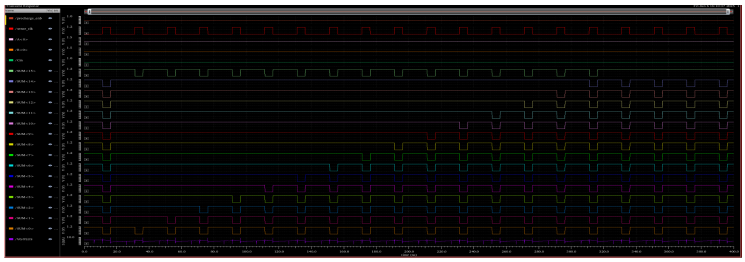


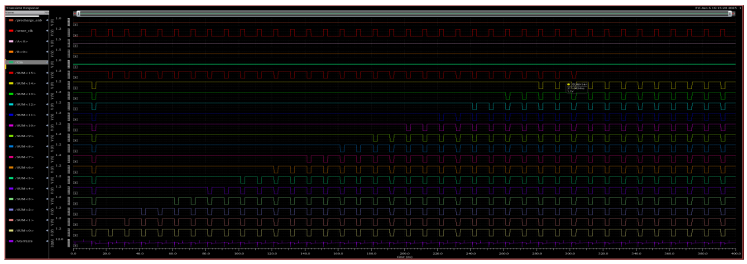
Figure 25: *16-bit adder test bench (programmed CLBs for SUM/CARRY)*

As shown in Figure 25, 32 3:1 CLBs were preprogrammed (16 as SUM, 16 as CARRY) and connected to formulate a 16-bit ripple carry adder. The reason such an inefficient design was chosen for the adder was that testing the timing, delay, and power results in a relatively

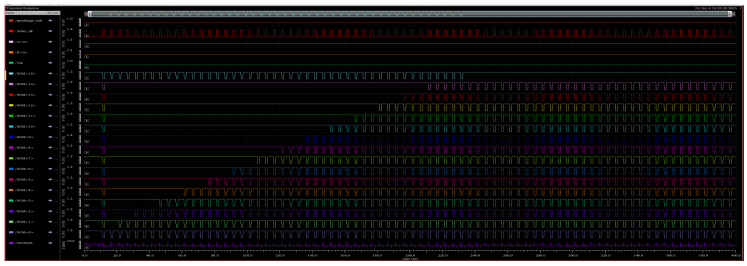
easy-to-comprehend scaling factor (e.g., the hardware itself linearly scales, and this linear scaling can be compared to the delay when the clock speed is changed). The bench consists of two binary bus generators (shown in upper left), which will generate 16 bit-lines corresponding to a decimal value as programmed via Verilog-A, allowing ease of demonstration of proper functioning. The current programmed value in the test bench is 0x7FFF and 0x0001, which corresponds to the worst case scenario for addition in a ripple carry adder (the result should be 0x8000, a one in the 15th bit, and zeros until the 0th bit). This was chosen instead of 0xFFFF and 0x0001 since all zeros is a common value for even an improperly functioning device.



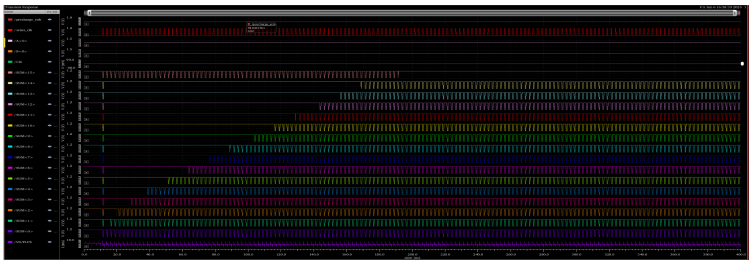
(a) 20ns Clock Cycle



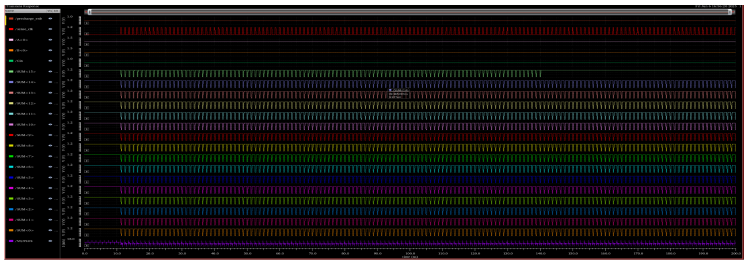
(b) 10ns Clock Cycle



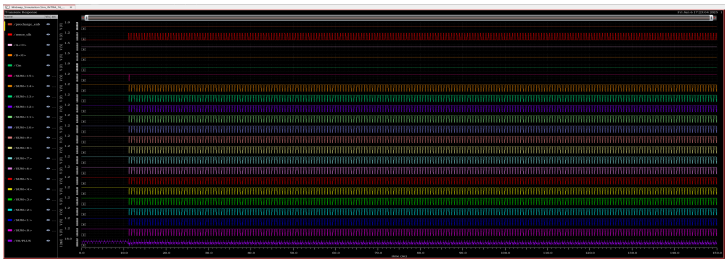
(c) 5ns Clock Cycle



(d) 2.5ns Clock Cycle



(e) 1.25ns Clock Cycle



(f) 0.625ns Clock Cycle

Figure 26: *Clock Speed and 16-bit addition graphs (0x7FFF + 0x0001=0x8000)*

Clock Cycle/Clock Active	Delay Until Correct Value	Energy Consumed During Operation
20ns/5ns	331ns	1.8009nJ
10ns/2.5ns	311ns	1.6962nJ
5ns/1.25ns	236ns	1.2856nJ
2.5ns/0.65ns	193.5ns	1.0539nJ
1.25ns/0.3125ns	141.1ns	0.76621nJ
0.625ns/0.15625ns	[breaks]	[breaks]

Table 1: *Clock Speed and 16-bit addition table (0x7FFF + 0x0001=0x8000) for delays and energy consumed to perform calculation derived from Figure 26*

Table 1 describes the relationship between clock speed and calculation delay, and energy consumed for the 16-bit 3:1 CLB-based ripple carry adder. The simulations were formed in ADE L, then exported into CSV files, and then imported into MATLAB in order to integrate the current over time and multiplied by the voltage to equal energy. When the clock cycle shrank to around 0.625ns, the ripple-carry adder ceased to function. This implies that our CLB-based design is only about to function until reaching a frequency of around 800MHz.

On the other hand, a single MTJ model has been programmed to change its values if a current passes through it in the same direction for more than 9.5 nanoseconds (Appendices Section 1). So the planned write clock cycles are approximately 10 nanoseconds. However, the read cycles are determined by the waveform delays between the sense clock signal and the sense amplifiers' response time. This delay is found to be 0.109 nanoseconds in 2:1 CLBs (Appendices Section 3). This finding proves that the maximum possible frequency is limited by the 16-CLB matrix and thus the theoretical maximum is 800 MHz.

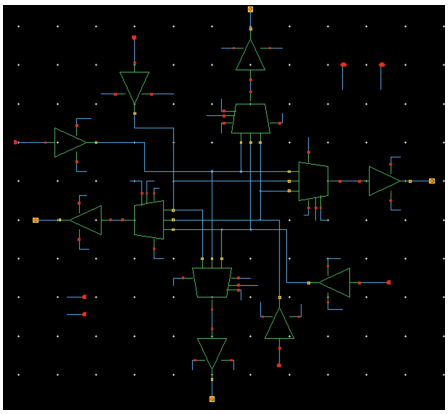


Figure 27*: 4in:4out switch box schematic, composed of 3:1 MUXs and buffers*

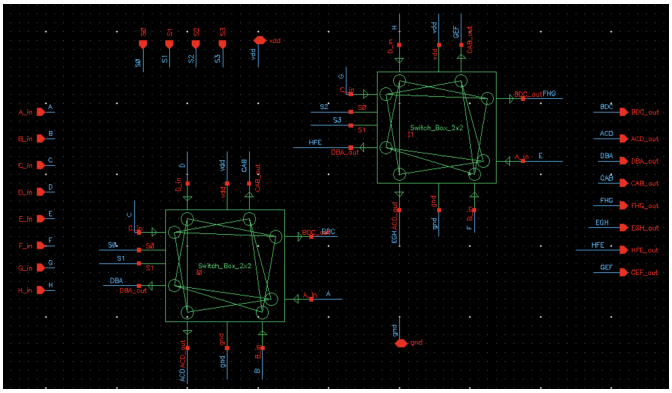


Figure 28: *8in:8out switch box schematic, composed of 4in:4out switch boxes*

Figures 27 and 28 above reveal the schematic design of our 4in:4out and 8in:8out switch boxes. As described earlier, the switch boxes are composed of 3:1 MUXs and buffers, which provide standard loads on the inputs and outputs of the boxes. The goal of testing these components is to determine that they exhibit the proper functionality, meaning that for any given input combination on the selection wires that each output line is conducting a different input line. For the 8in:8out switch box specifically, it is also important to test the independent functionality of the two internal sections to ensure that signals will not be mixed between the various inputs and outputs passing through.

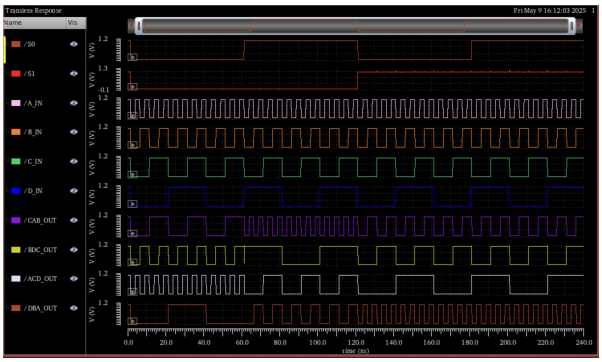


Figure 29: *4in:4out switch box schematic simulation results*

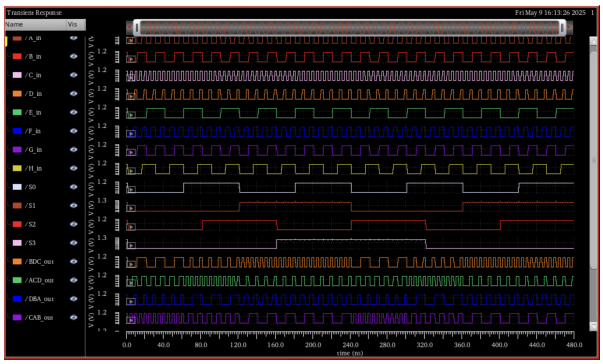


Figure 30*: 8in:8out switch box schematic simulation results*

Figure 29 (left) reveals the simulation results of the 4in:4out switch box schematic, while Figure 30 (right) shows the simulation results of the 8in:8out schematic. In Figure 29, it is clear that for each of the given selection inputs, S0 and S1, each output line conducts a different input, thus demonstrating proper operation of the switch box. Figure 30 similarly shows that the 4 outputs tied to the first 4 inputs only conduct those inputs, while the other 4, EFGH, are not considered and instead are passed through the other 4 inputs in the box.

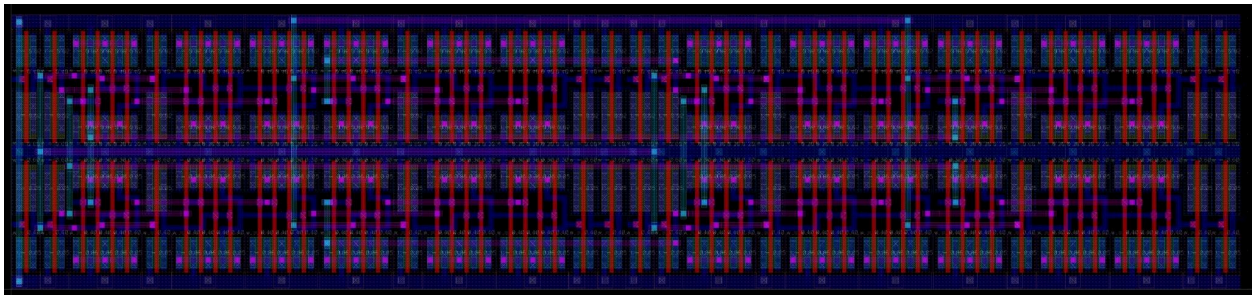


Figure 31*: 4in:4out switch box layout design*

With the working simulations for the switch boxes, the next step is to create the layout design for the component to be incorporated into our final deliverable layout model. Figure 31 above shows the layout design of the 4in:4out switch box complete with full functionality as demonstrated in the figures below, along with validated DRC and LVS test results. The design of this component in the layout optimizes for size by maintaining a uniform height between components. Power and

timing will be studied further along with the other components, but it is important to note that upon initial testing, it was found that the component possesses a delay of roughly 20ps, found through measuring the time between input and output responses.

In practice with standard design ideas, Metal 1 is able to travel freely between horizontal and vertical placements, while Metal 2 is reserved for horizontal routing, and Metal 3 is reserved for vertical routing in the design. The simulation test results in Figure 32 below demonstrate that the layout design is operational and can be implemented into the final layout design and perform as expected.

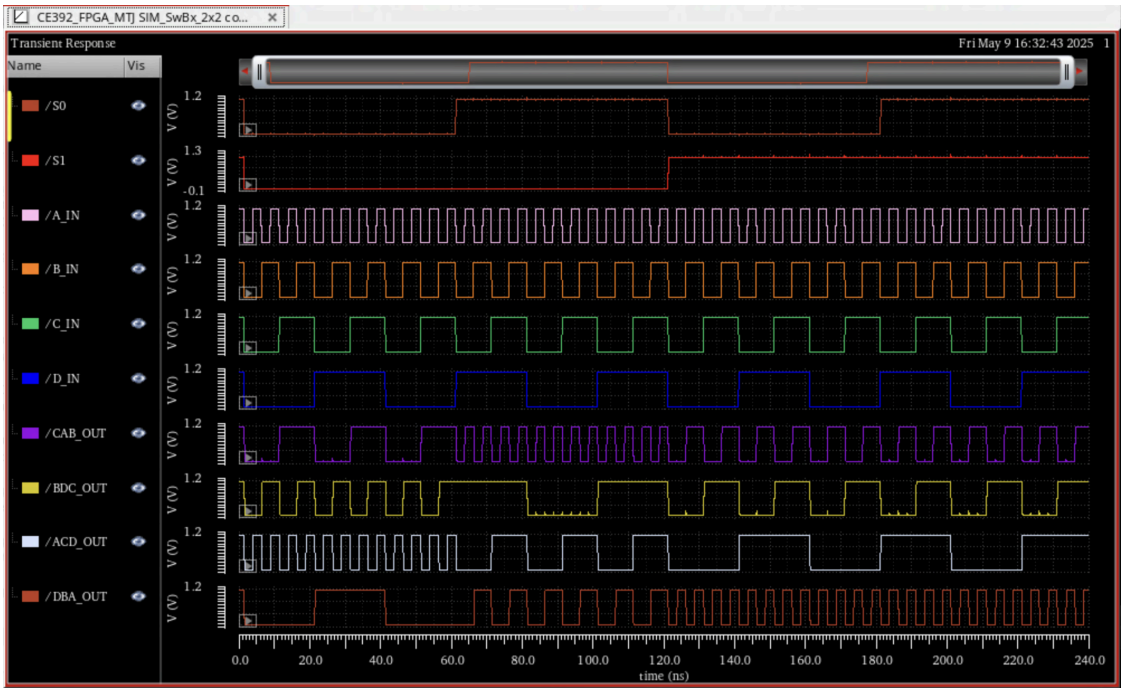


Figure 32: *4in:4out Final Switch Box Layout (Figure 31) Simulation*

The circuit regarding the final demonstration is shown in Figure 33, and details on that circuit are shown in Figures 34 and 35 below. Inputs and outputs are driven with inverters for impedance matching.

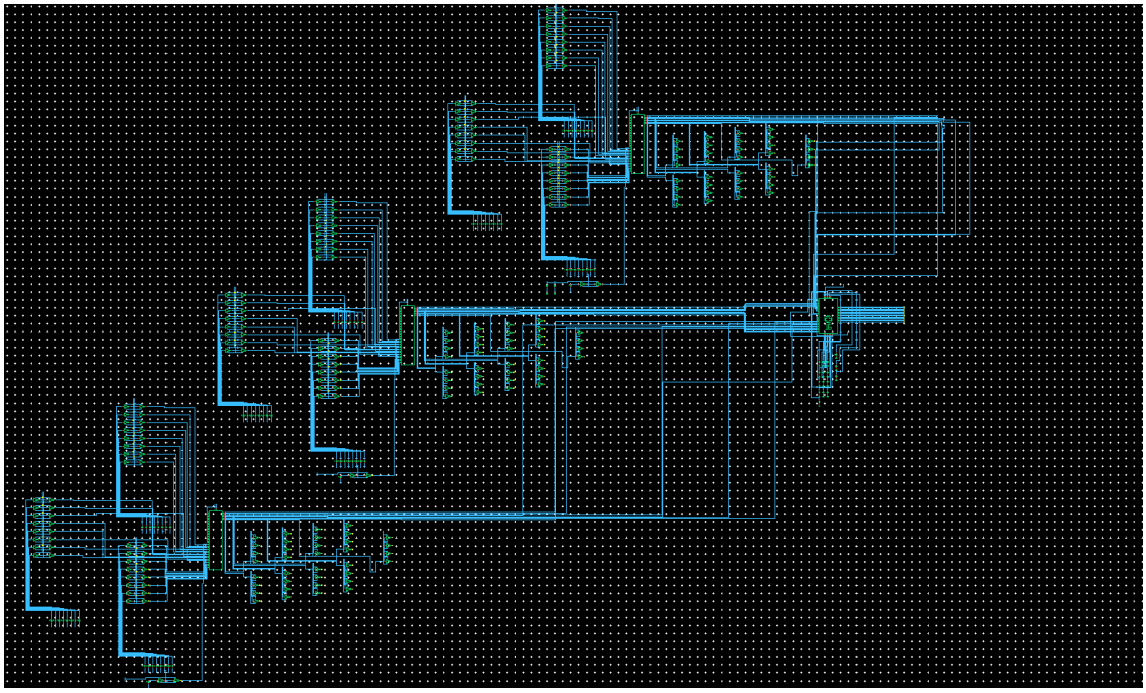


Figure 33: *Final Simulation Schematic*

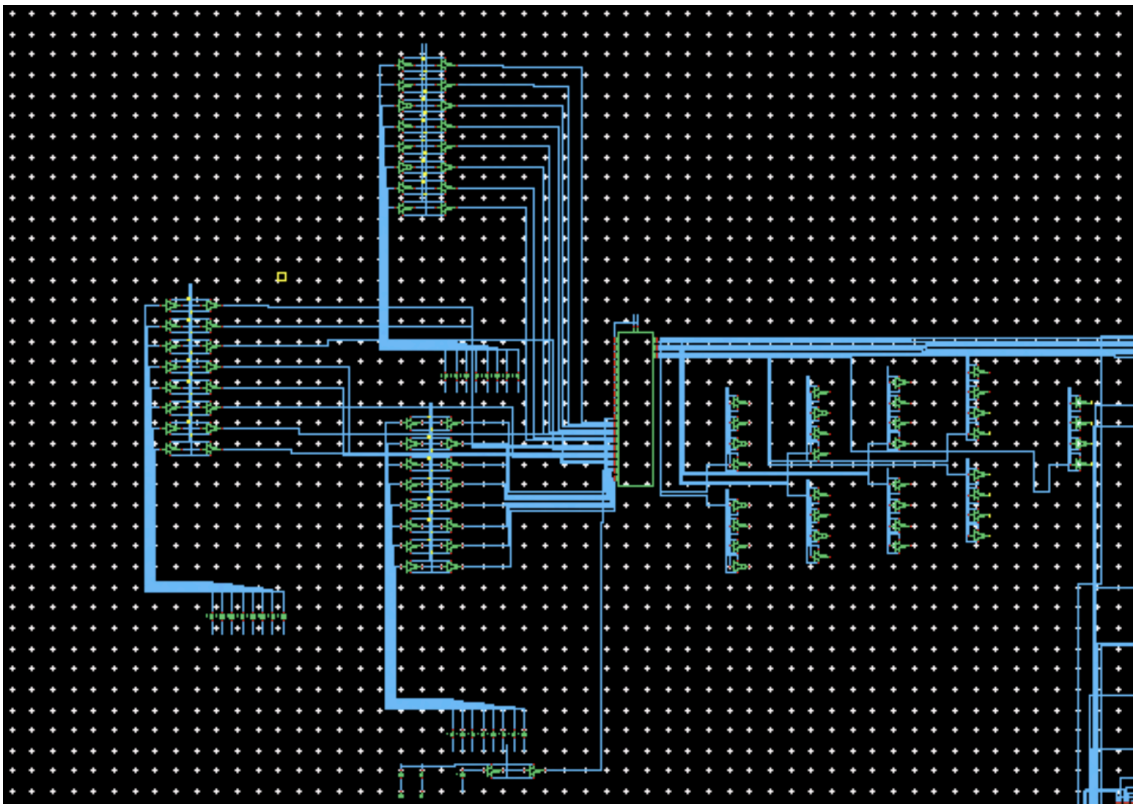


Figure 34: *Final Simulation - Integration of 9-CLB Matrix*

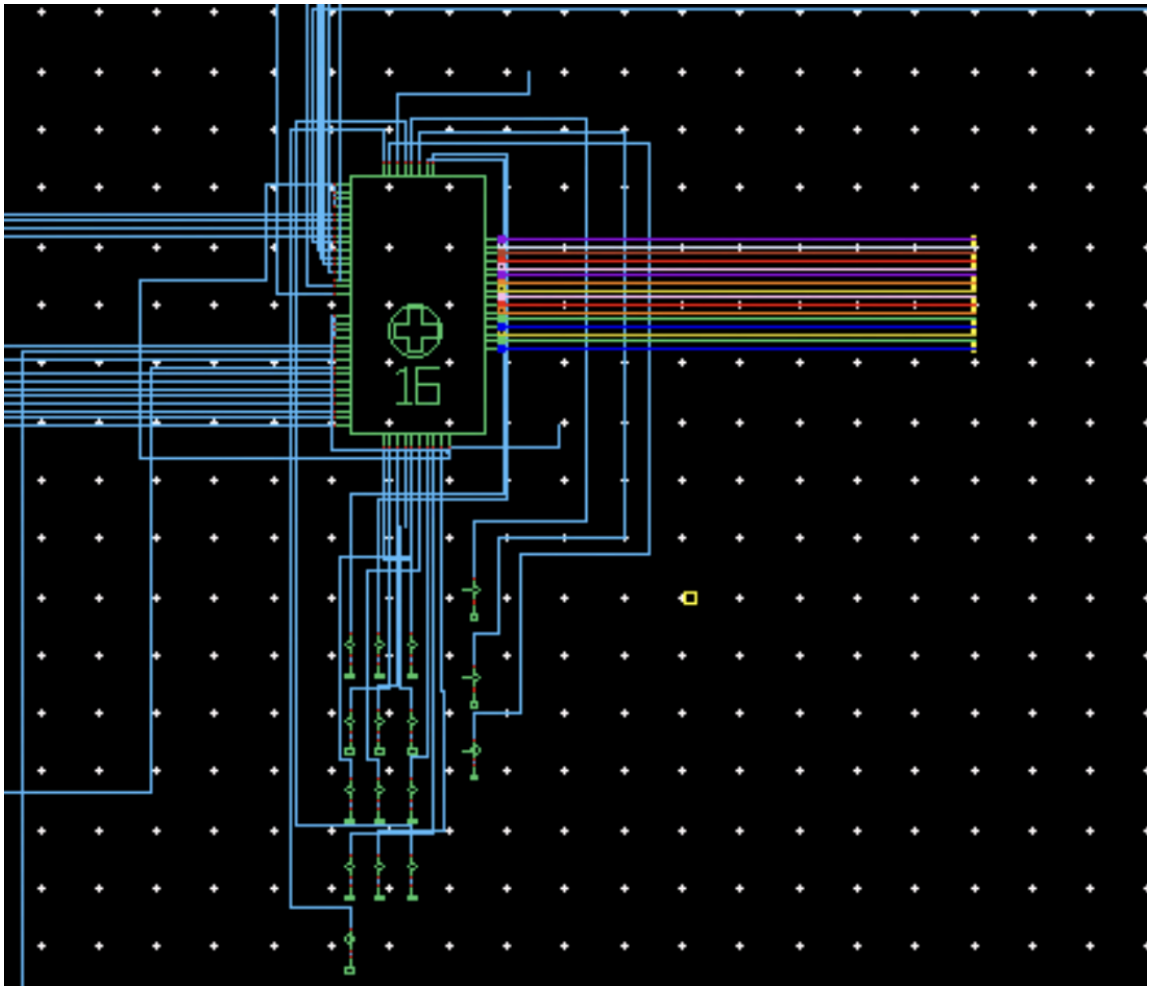


Figure 35: *Final Simulation - Integration of the 16-CLB Matrix and Probed Outputs*

This design demonstrates an effective mechanism for detecting and interpreting patterns through a combination of logic and arithmetic operations. Two 12-bit binary input values are taken and XORed in pairs to expose bit-level differences (using all 3 of the 9-CLB matrices), with the two resulting values then added using a ripple-carry adder, which is programmed from the 16-CLB matrix. All switch box select inputs across 3 9-CLB matrices are programmed to 0 to allow for a single logic level. The system is observed beginning at 22 nanoseconds, the point at which the adder clock starts. When the inputs are exactly the same, the XOR results become predictable, and the sum output from the adder remains at zero (Figure 36). This steady zero output indicates that the input patterns have not changed, which can be interpreted as a logical '1' in the context of stable behavior.

Conversely, when the input binary bitstreams differ, the XOR operations produce non-zero outputs, which are then processed by the ripple-carry adder to yield varying results. In each bitstream, the XOR operation compares corresponding bits from the most significant to the least significant positions, effectively measuring the bitwise differences between the paired inputs. This process reveals the Hamming distance between the input pairs.

Figure 37 illustrates that when the inputs are dissimilar, the output values continue to fluctuate and do not stabilize even up to 100 nanoseconds, indicating that the system is still processing and resolving the differences. By 200 nanoseconds, the outputs begin to settle, and the resulting waveforms show distinct bits that remain high, corresponding to positions where the input digits differed. While the XOR outputs alone reveal the Hamming distance by marking each differing bit with a '1', the ripple-carry adder does not simply count these differences; instead, it sums them as binary values, meaning each differing bit contributes to the total based on its positional weight; producing a result that reflects both how many bits differ and where those differences occur.

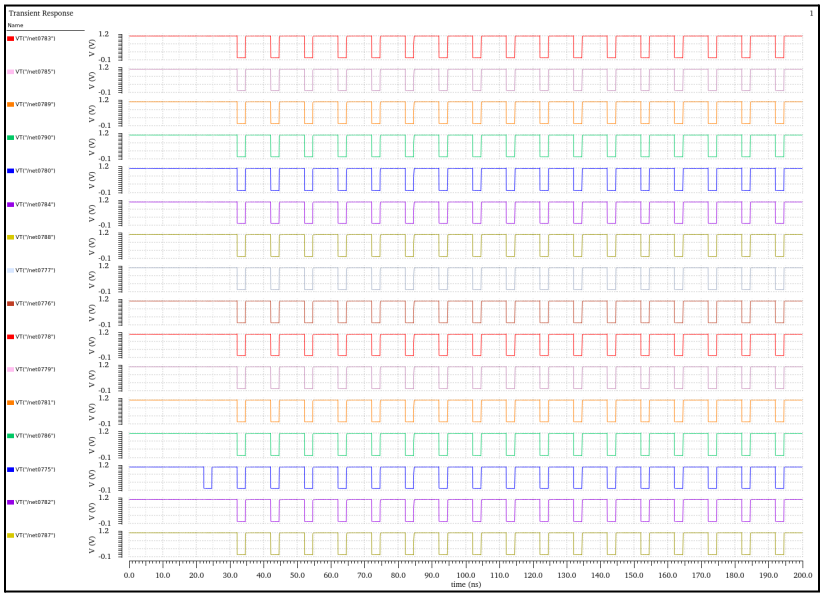


Figure 36: *Final Matrix Testbench - Matching Outputs*

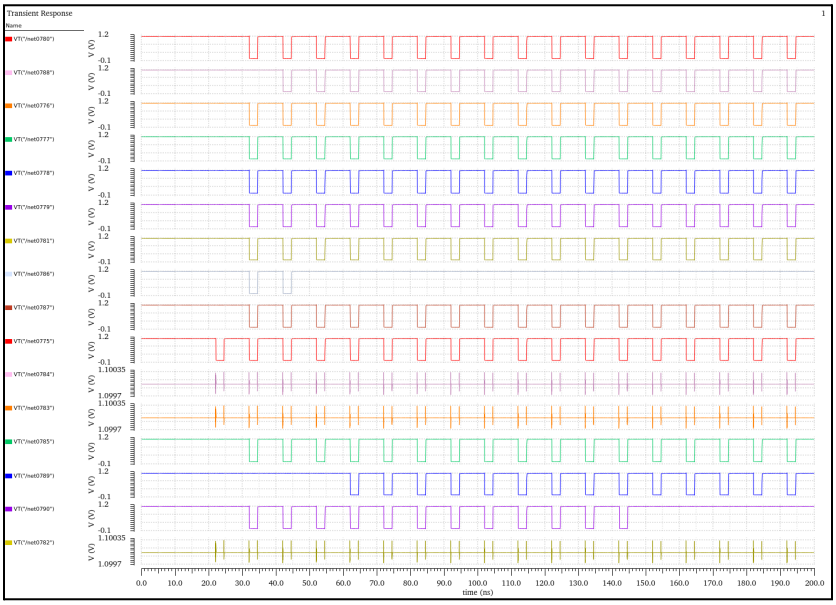


Figure 37: *Final Matrix Testbench - Random Outputs*

IMPLEMENTATION AND SYNTHESIS

The central technology used in the layout process was the NCSU FreePDK45 library for all digital aspects of the design. We were not able to utilize a manufacturable layout for the analog components (resistors, MTJs, capacitors), though we left space for an underlying abstraction as prescribed by the Verilog-A models we conceived, as well as components from the analogLib library. For our design, the most vital inputs included switchbox routing programming, CLB worldline and bitline enable values, and the functional inputs into the CLBs (for binary tree/truth table traversal). The result outputs were the differential bitlines from the CLBs, that is, a single bit output which was routed through the network of matrices and viewed via a voltage rise or drop via ADE L simulation. For our design, our initial target was a clock cycle of 5ns, or a frequency of 200MHz. After iteration in our design, we found that this cycle could be reduced all the way to 1.25ns, corresponding to 800MHz. After that point, the outputs of our design ceased to meet verification.

Layout proceeded according to reasonable utilization of area in design and was primarily downstream from clock speed requirements. For instance, the width of NMOS transistors within the tree-like structure of the LUTs was intentionally oversized in comparison to a unit inverter, and transistors such as the bitline access transistors of the bitline conditioner were multiple times larger than required to allow for minimal resistance and thus enable even faster clocking. A notable example of this is the PMOS switch transistor on the sense amplifier, which we specified as 4.5 micrometers and thus required 12 fingers in order to match the height of the cells it was integrated within. Though we could have achieved similar performance with a much smaller width switch transistor, the very slight edge in switching speed was suitable for our purposes.

Despite individual components not optimizing for area, the integration of components within the larger matrix, which we synthesized, followed Stockmeyer's Algorithm. An example of how we calculated the area we should expect by using arbitrary routing channel areas is shown below with the example of the 9-CLB Matrix:

Step 1 - Module dimensions

We start with two building blocks. The first is the switch box, which is 15.9025 micrometers wide and 3.545 micrometers tall. The second is the logic block (CLB), which is 14.8085 micrometers wide and 4.405 micrometers tall. Multiplying width by height gives each module's area:

- Switch box area = 15.9025 × 3.545 = 56.3743 square micrometers
- Logic block area = 14.8085 × 4.405 = 65.2314 square micrometers

Step 2 - Bus wiring block

We set aside one square region of area 350 square micrometers for the main routing channel. To find its side length, take the square root of 350, which is about 18.7083 micrometers.

Step 3 - Clustering and justification

- We have 16 switch boxes that talk to each other heavily. Splitting them into two groups of eight cuts the average connection length roughly in half.
- We place the 350-square-micrometer bus block between those two groups so that long wires run only through that central channel.
- The nine logic blocks go off to one side in a single stack, since each logic block mainly needs short links to the nearby switch boxes.

Step 4 - Floor plan structure

At the highest level, imagine a straight horizontal row of four items:

- The first cluster of eight switch boxes
- The central bus block
- The second cluster of eight switch boxes
- The stack of nine logic blocks

Step 5 - Cluster heights

• Eight switch boxes stacked vertically give a total height of $8 \times 3.545 = 28.36$ micrometers.

• Nine logic blocks stacked vertically give a total height of $9 \times 4.405 = 39.645$ micrometers.

• The overall floorplan height is the taller of the two: 39.645 micrometers.

Step 6 - Cluster widths and overall bounding box

Add up the widths in the horizontal row:

15.9025 (switch cluster) + 18.7083 (bus) + 15.9025 (second switch cluster) + 14.8085 (logic stack) = 65.3218 micrometers.

So the rectangle enclosing everything is 65.3218 micrometers wide by 39.645 micrometers tall.

Step 7 - Total area

Multiplying width by height gives $65.3218 \times 39.645 \approx 2591.8$ square micrometers.

.

While creating a truly manufacturable layout of some of our analog and magnetic components was unfeasible, we sized our layout involving these components according to the expectation that features such as MTJs would be manufactured beneath (n-well/p-well) and between the contact points we allocated for them in the layout. As such, the digital components of our design were fully realized in our layout and could be matched against their ideal schematic form via LVS and PEX. Analog components (CLBs, matrices) were also matched against their schematic and

validated to pass LVS and PEX, sans their analog aspects (digital-only versions of these components were created to ensure proper connections and validity of the layout). The results of these tests can be reached through the path described in the Appendices Section 2. Without manufacturable and usable layouts of these analog components, it is difficult to optimize for other parameters such as energy consumption or (true) area utilization. This degree of layout optimization to achieve a desired effect will have to be a consideration for future research by creating a physical layout representation with internal electrical properties adequately detailed for simulation and placement in Cadence Virtuoso.

Wiring between digital components was done for minimal area consumption, as long as the wire width and length seemed appropriate for our requirements. For instance, within component modules, wire distance was specified to minimum distance allowable by DRC, unless the wire's integrity was of great importance (e.g. a bitline and its complement), in which case some distance was required to limit resulting parasitic capacitance.

The following table is a summary of the area utilized by the notable components in our design.

Component of Note	Area Taken in Layout
4:4 Switch Box	$15.9025\mu m \times 3.545\mu m = 56.3743\mu m^2$
8:8 Switch Box	$15.9025\mu m \times 6.9125\mu m = 109.9260\mu m^2$
2:1 MTJ-LUT	$7.18\mu m \times 2.115\mu m = 15.1857\mu m^2$
3:1 MTJ-LUT	$15.985\mu m \times 4.6\mu m = 73.5310\mu m^2$
2:1 CLB	$14.8085\mu m \times 4.405\mu m = 65.2314\mu m^2$
3:1 CLB	$27.848\mu m \times 5.905\mu m = 164.4424\mu m^2$
2:1 CLB Matrix	$57.796\mu m \times 45.87\mu m = 2{,}651.1025\mu m^2$
3:1 CLB Adder Matrix	$30.4975\mu m \times 191.25\mu m = 5832\mu m^2$
Fully Featured Design	$89.0855\mu m \times 191.755\mu m = 17082\mu m^2$

Table 2: *Notable components and layout area utilization*

All subcomponents used in design were tested and validated by their respective truth tables in the case of digital, and approximate voltage levels in the case of analog components (a "Sim_" version of each designed schematic/layout was created in order to validate expected behavior).
We followed the standard digital design method of creating foundational designs, testing them, and then incorporating them into larger and larger designs to reduce the ambiguity of errors as much as possible. In the beginning, validation of our programmable design was done via write

and read cycles. For our programmable components (CLBs), we tested multiple different truth table writes via modifying wordlines according to what MTJ was being written to (wl0,1,2,... and wlc are raised to 1 for a specific MTJ) and what was being written (bl0=0, bl1=1 for lowering MTJ resistance corresponding to “1”, and the opposite for raising its resistance). Various truth tables were inscribed in the LUTs to ascertain flexibility of the design (e.g. XOR, NAND, NOT, for the 2:1, and CARRY, SUM, for the 3:1), where the inputs of the CLB were then generated and fed through their network to discover correctness at each corresponding inputs and its respective output (e.g. testing 00 through 11, or 000 through 111 and viewing the output).

Because of the sheer scale of different possible outputs of the 16-bit LUT table adder, a Verilog-A module was written to convert decimal values to 16-bit lines (mentioned earlier), and multiple inputs were tested on the design. The difference between a “0” or “1” in a CLB output is determined by whether the output voltage level settles on remaining at VDD, or whether it drops down to close to zero even after waiting for the maximum delay of the adder. Decimal value outputs were determined via exporting the charts as CSV and importing them into MATLAB, where the bitlines were then sorted and recombined into a decimal value. For relatively quick validation, checking the 16-bit version of the sum of two decimal numbers could also be done via inspection of the bitlines after maximum delay. Due to the large number of components involved in this process, the simulation for each addition typically took several minutes, but after enough random numbers were added together, and every bitline was observed to be able to change, we ascertained the correctness of the programmed truth tables.

ETHICAL CONSIDERATIONS

Our FPGA design introduces non-volatility into the logic layer by using MTJ-based look-up tables, which, while beneficial for performance and power efficiency, comes with a set of ethical risks we must consider seriously.

One of the main concerns is data persistence. Because the logic states in our design remain intact even after power is removed, there's a real risk that sensitive information could be recovered from the hardware without permission. This could be a problem in applications that handle proprietary algorithms, personal data, or security-critical operations. To reduce this risk, we would need to include safeguards like secure erase features, hardware encryption of logic states, and possibly tamper-detection circuits that wipe data automatically if someone tries to access the chip physically.

There's also a material sourcing issue. MTJs require elements like cobalt and tantalum, both of which often come from conflict zones where mining practices can involve child labor, unsafe conditions, and environmental damage. If this technology were scaled up for commercial use, the demand for these materials could increase, which means we have a responsibility to push for ethically sourced materials. In the future, we would aim to use verified conflict-free suppliers and explore alternative materials like ruthenium where possible.

Lastly, while the design is intended for general-purpose logic and computation, it's important to acknowledge that technologies like this can be misused. Because it can retain information and run efficiently with little power, it might be used in surveillance systems, or worse, embedded into systems without users' knowledge. Although we can't control every downstream use case, we believe it's important to think ahead and design with constraints or usage policies in mind where possible.

In short, our design is promising from a technical point of view, but it requires careful consideration when it comes to data privacy, security, material sourcing, and potential misuse. Addressing these concerns early helps ensure this technology is developed and used responsibly.

CONCLUSION

This project demonstrated the design and simulation of a non-volatile logic-in-memory FPGA architecture using STT-MTJ-based LUTs integrated into custom-configured logic blocks. Our system met the key requirements defined at the outset: maintaining state without power, supporting configurable logic operations, enabling scalable routing with custom switch boxes, and verifying functionality across complete logic matrices.

The final integration consists of four CLB matrices and was implemented as a unified layout-level design in our final simulation. Each digital component was independently placed and verified, and the integration accurately reflects the schematic structure validated in earlier simulation stages. All layout-driven verification steps that include Design Rule Check (DRC), Layout Versus Schematic (LVS), and Parasitic Extraction (PEX) were completed. Parasitic values remained within expected ranges, and no anomalies were observed that would compromise timing or logic behavior.

While full analog integration could not be realized at the layout level, this was due to limitations in the design kit provided. Specifically, our custom-defined analog models (STT-MTJ Model1 and Model2), along with standard analogLib components (resistors and capacitors), lack layout layer definitions within the NCSU FreePDK45 process. As a result, those analog elements were excluded from the final layout. In future work, analog simulation at the layout level will require access to full-process PDKs that support analog device layout definitions. Commonly used PDKs that offer such support include TSMC 16nm, GlobalFoundries 22FDX, and SkyWater 130nm, all of which provide more realistic layer mapping and analog component modeling suitable for industry-aligned verification.

Despite this limitation, our layout design was pursued to validate parasitic impacts and assess readiness for fabrication. By comparing layout-based simulations against previously validated schematic models, we confirmed the correct functionality of the digital design. The floorplanning included careful separation of magnetic and logic components, precise pin placement for routing, and adherence to a scalable layout strategy.

All major objectives outlined in our mid-year progress report were completed, including the modeling of MTJ devices in Verilog-A, simulation of arithmetic and pattern recognition functions, layout of CLB and switch box modules, and full integration of matrices. The demonstration of a Hamming distance calculator successfully exercised all programmable features in the final design.

Looking ahead, future improvements should focus on analog layout support through industry-grade PDKs, magnetic shielding strategies for security and signal integrity, and expanding the matrix size to support more complex applications. The design infrastructure, floorplanning, pin placements, and verification framework developed here provide a strong foundation for scaling this architecture into a more advanced, manufacturable prototype.

REFERENCES

[1] B. Dieny, R. B. Goldfarb, and K.-J. Lee, Introduction to Magnetic Random-Access Memory. Hoboken, NJ, USA: Wiley-IEEE Press, 2017. doi: 10.1002/9781119079415

[2] U. Farooq, Z. Marrakchi, and H. Mehrez, Tree-based Heterogeneous FPGA Architectures: Application Specific Exploration and Optimization, 1st ed. New York, NY: Springer, 2012. [Online]. Available: https://doi.org/10.1007/978-1-4614-3594-5

[3] Peng, Shouzhong & Kang, Wang & Wang, Mengxing & Cao, Kaihua & Zhao, Xiaoxuan & Wang, Lezhi & Zhang, Yue & Zhang, Youguang & Zhou, Yan & Wang, Kang & ZHAO, Weisheng. (2017). Interfacial Perpendicular Magnetic Anisotropy of 1x nm Tunnel Junctions for Large-Capacity STT-MRAMs. IEEE Magnetics Letters. PP. 1-1. 10.1109/LMAG.2017.2693961.

[4] N. H. E. Weste and D. M. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th ed. Boston, MA, USA: Pearson, 2010. ISBN: 978-0321547743.

[5] Glochip, “SOT-MRAM Manufacturing Challenges,” Glochip Semiconductor, [Online]. Available: https://www.glochip.com/h-nd-1039.html [Accessed: Jun. 6, 2025].

APPENDICES

1. Verilog-A Model For Final STT-MTJ Model

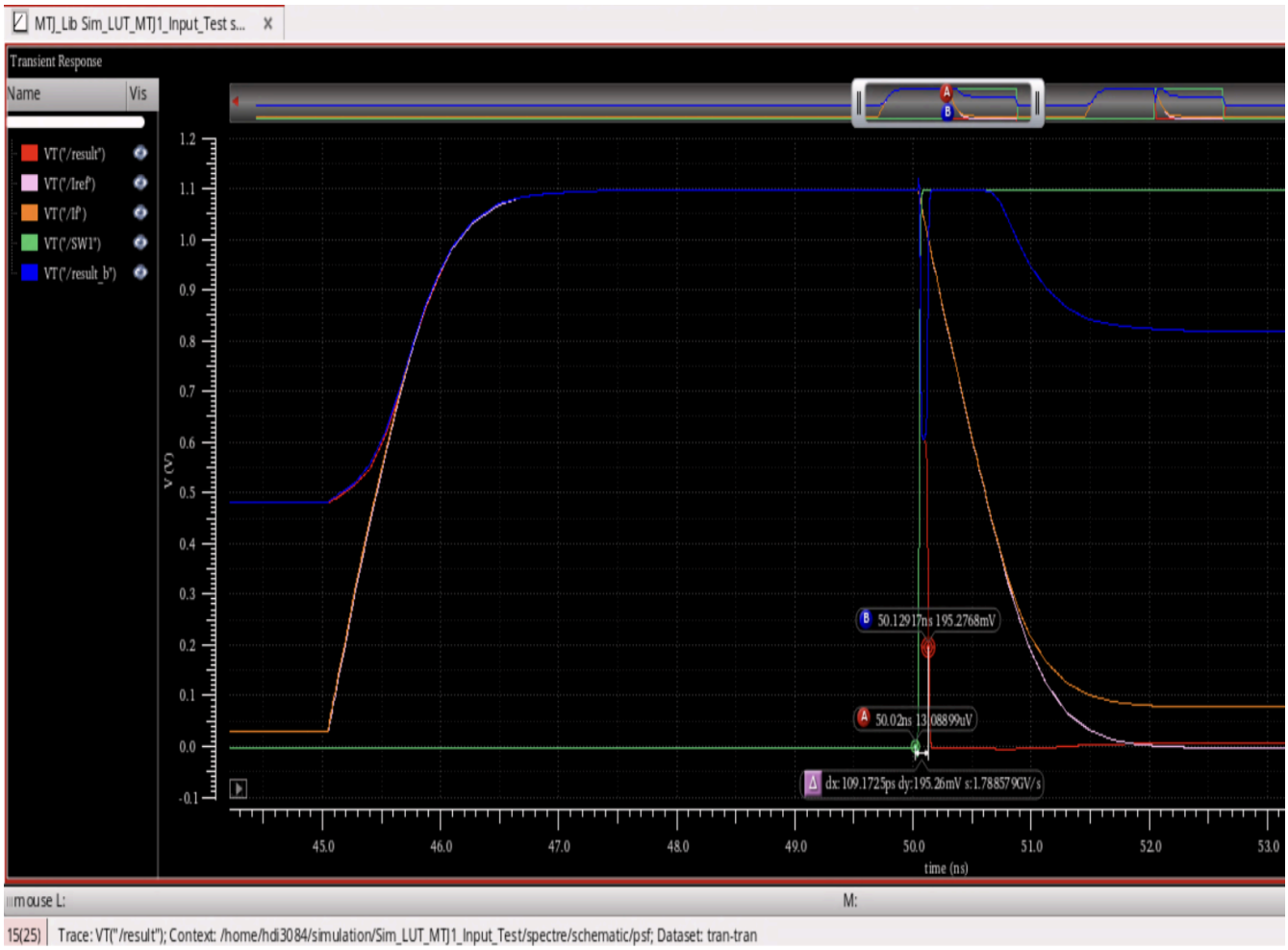
```
// VerilogA for MTJ_Lib, STT_MTJ_Model1, veriloga
`include "constants.vams"
`include "disciplines.vams"

module mtj_basic(top, bottom);
  inout top, bottom;
  electrical top, bottom;
  // Parameters
  parameter real R_P = 1000.0;            // Parallel resistance (Ohms)
  //Assuming R_ref is 2k Ohms
  parameter real R_AP = 3000.0;           // Antiparallel resistance (Ohms)
  parameter real I_THRESH = 5u;          // Threshold current (A)
  parameter real T_SWITCH_MIN = 9.5n;    // Switching time threshold (s)
  parameter real T_TRANSITION = 100p;    // transition time constant
  real state;             // 0 = AP, 1 = P
  real target_state;      // Target state to transition toward
  real R_eff;             // Effective resistance
  real dstate_dt;
  real Imeas;             // Measured current
  real delta;             // Rate control
  real switch_timer;      // Accumulated time for switching
  real last_time;         // For computing delta_t
  real delta_t;
  analog begin
    // First time step initialization
    @(initial_step) begin
      state = 0.0;
      target_state =  0.0;
      switch_timer = 0.0;
      last_time = 0.0;
    end
    Imeas = I(top, bottom);
    // Time delta calculation
    delta_t = $abstime - last_time;
    last_time = $abstime;
    // Current-based switching logic with time tracking
    if (Imeas > I_THRESH) begin
      switch_timer = switch_timer + delta_t;
      if (switch_timer >= T_SWITCH_MIN)
        target_state = 1.0; // Switch to P
    end
       else if (Imeas < -I_THRESH) begin
      switch_timer = switch_timer + delta_t;
      if (switch_timer >= T_SWITCH_MIN)
        target_state = 0.0; // Switch to AP
    end else begin
      switch_timer = 0.0;        // Reset timer when under threshold
      target_state = state;      // Hold state
    end
    // Smooth state transition
    delta = (target_state - state) / T_TRANSITION;
    dstate_dt = delta;
    state = idt(dstate_dt, state);// idt => integrate w/ respect to time
    // Clamp state between 0 and 1
    if (state > 1.0) state = 1.0;
    if (state < 0.0) state = 0.0;
    // Linear resistance interpolation
    R_eff = R_AP + (R_P - R_AP) * state;
    // Ohms Law
    V(top, bottom) <+ R_eff * Imeas;
  end
endmodule
```

2. Location of Final Files Used in Demonstration and Presentation

During our time working on this project in Cadence Virtuoso, we learned that post-file creation organization has the potential to lead to design files being locked from edits and unopenable. With three people working on this project and their own components for much of the early design phase, our files got spread out and copied between libraries. All libraries are within the "cadence" folder, which is located at: `/vol/eecs392/projects/2025/Group1/cadence`.
A note on naming conventions is that the 4:4 Switch Boxes are named 2x2, and the 8:8 Switch Boxes are named 4x4 due to originally denoting the number of connections on each side before transitioning to the input-output naming convention. The .zip file containing the contents of the "cadence" folder has been submitted to the Final Design assignment on Canvas.

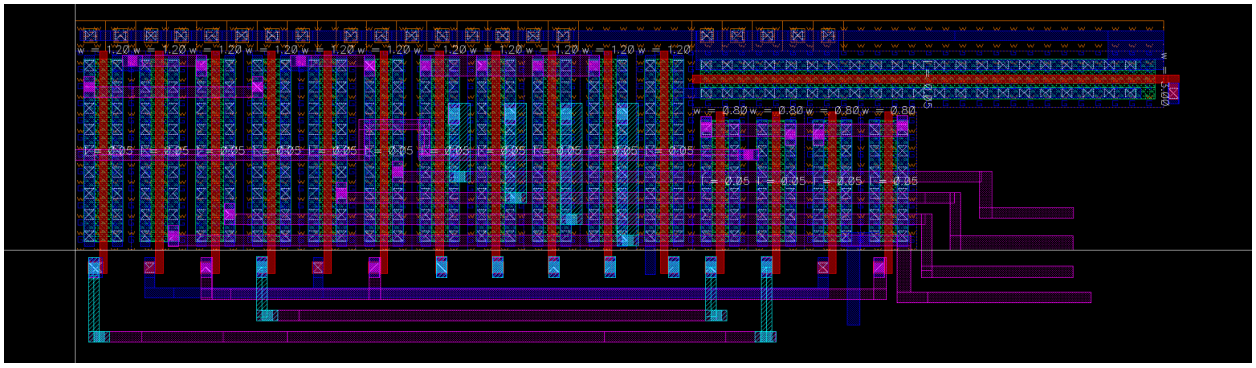
3. STT-MTJ Read Speed Test



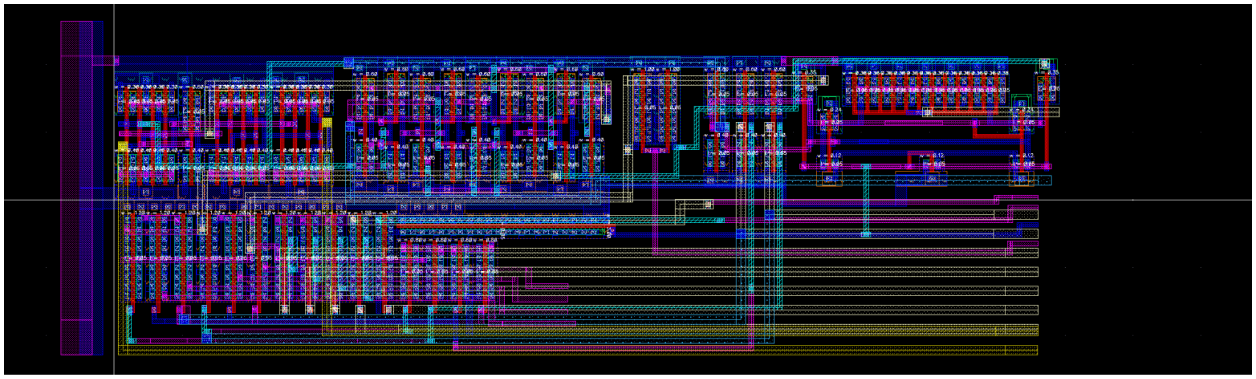
In this waveform a 2:1 CLB's read test is shown. The non-inverted output of the sense amplifier is shown in red, and the clock signal is shown in green. The inputs of the sense amplifiers that come from the logic and reference branches of the LUT are If and Iref, respectively. We see that the delay between the clock signal and the output from the sense amplifier is 0.109 nanoseconds.

4. Layout of Various Other Components

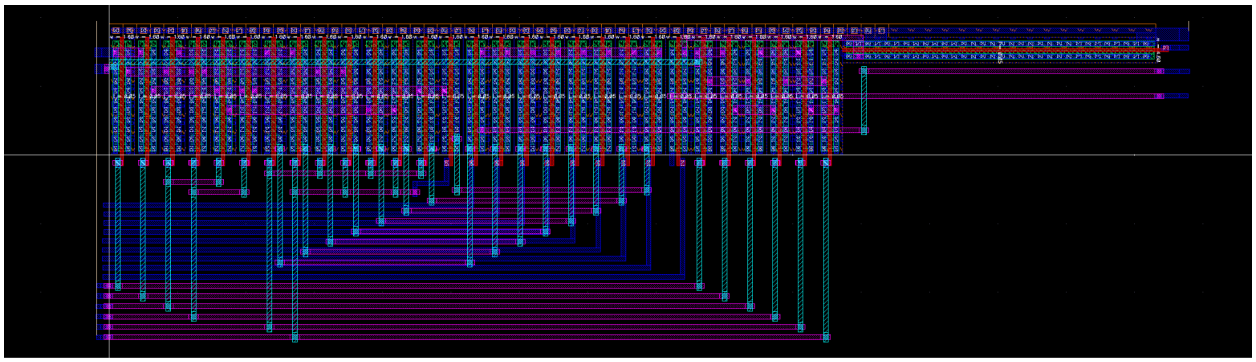
2:1 MTJ-LUT



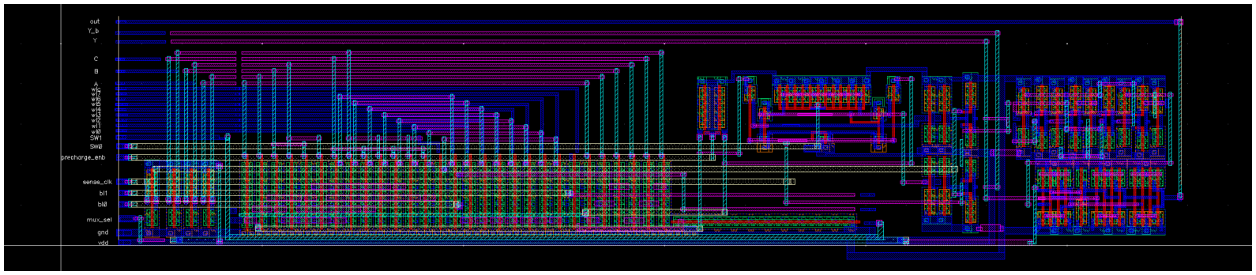
2:1 CLB



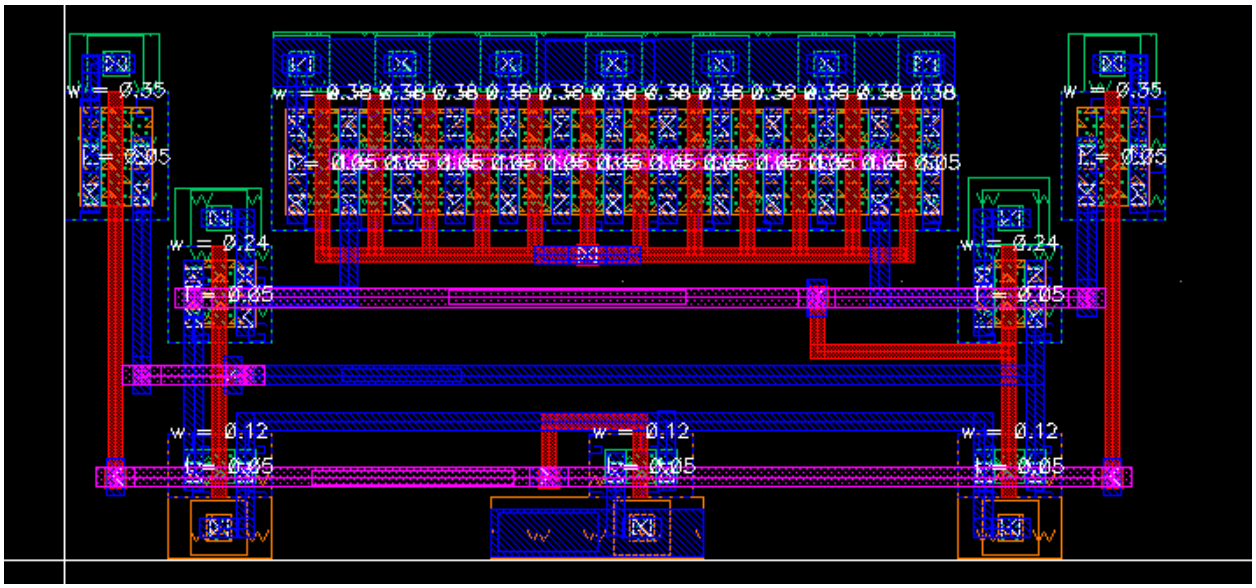
3:1 MTJ-LUT



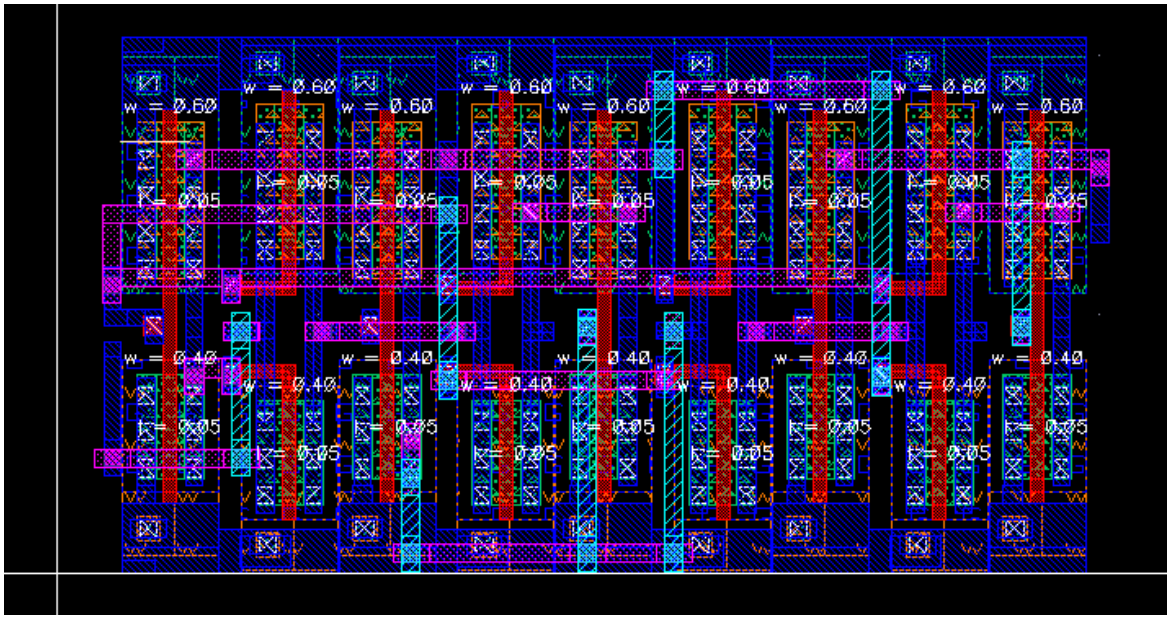
3:1 CLB



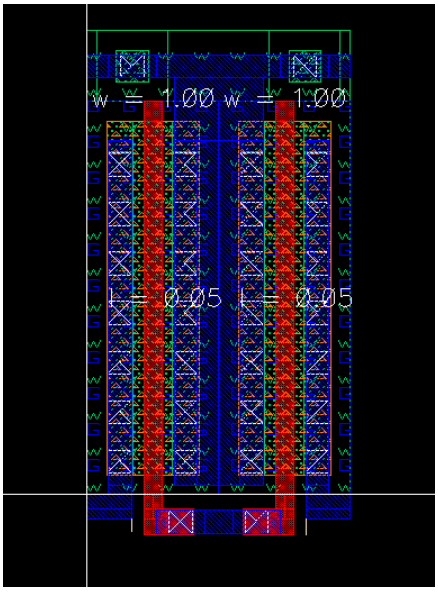
Sense-Amplifier



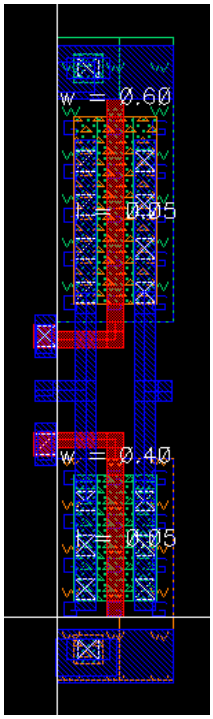
D-Flip Flop



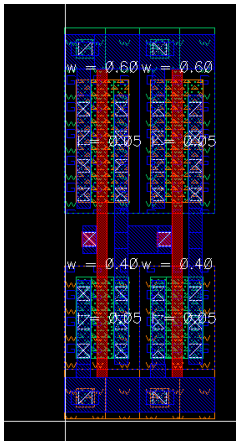
Bitline Conditioner



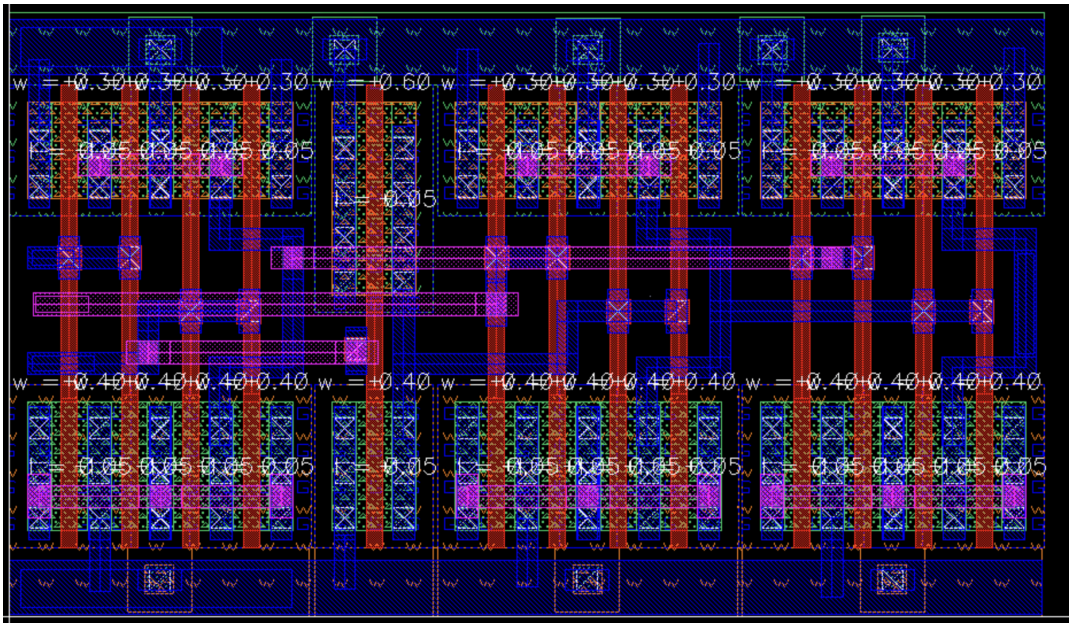
Transmission Gate



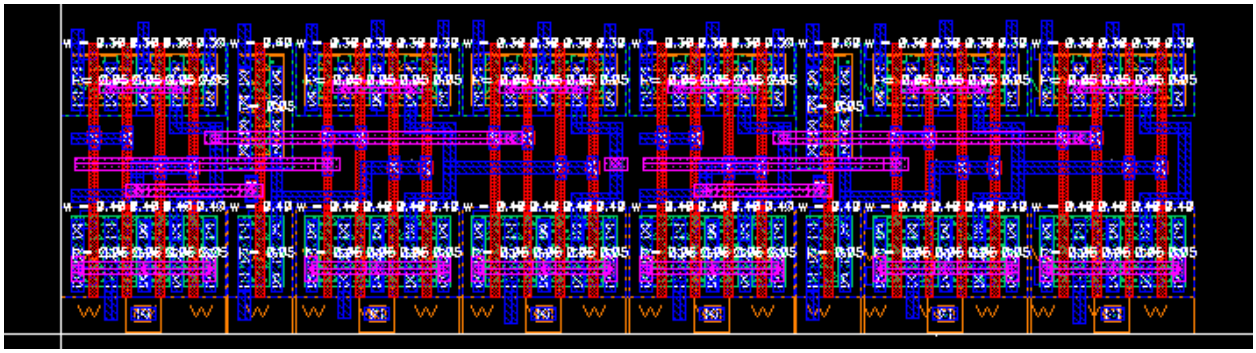
Buffer



2x1 MUX



3x1 MUX



Note: TAPs and power rail are not present due to altering of components to result in clean integration into switch box design. The 3x1 MUX was never used by itself in any other design.