ASIC Design of ReRAM-based AI Accelerators

Design Document

sdmay25-19
Client: Henry Duwe
Advisors: Henry Duwe & Cheng Wang

Team Members/Roles:
Sam Burns (Mixed Signal designer),
Travis Jakl (Mixed Signal Designer),
Noah Mack (Digital Signal Designer), &
Olivia Price (Analog Signal Designer)

sdmay25-19@iastate.edu
http://sdmay25-19.sd.ece.iastate.edu/

Executive Summary

Matrix-vector multiplication (MVM) is a fundamental operation in many computing tasks, yet performing it in traditional systems is inefficient due to frequent data transfers between memory and the arithmetic logic unit (ALU). One promising solution is to perform computation in memory (e.g., ReRAM) using the memory array itself. Such computational memories are an active area of research but have challenges associated with the analog nature of their computation. Our project focuses on implementing a test vehicle for computational ReRAM and constituent analog components

The key requirement for this project is to have four unique ReRAM architectures that can be tested individually on a test chip. Two designs were inherited from past student groups, and our team developed two additional architectures. The first design reorients the ReRAM cell structure by aligning the word line and the source line, which may improve routing efficiency and layout density. The second design eliminates the transistors creating a true crossbar that consists of just memristor which could help scalability, but introduce challenges such as sneak path current.

All designs utilize the Skywater 130nm process and the components are implemented using open-source tools including Xschem, Magic, and Ngpice. In addition to circuit and layout design, we are developing C-code for a microcontroller to interface with each ReRAM cell, allowing analysis testing and performance evaluation of the different architectures.

The progress made in the first semester included learning how to utilize the tools to our given criteria, researching the ReRAM technology and how to implement it, and creating two new unique designs to go into our research chip. By the end of the first semester, we had four distinct ReRAM designs planned and partially implemented. We also learned how to use the tools so our designs can be tested, fabricated, and placed on the project wrapper when they are prepared.

The second semester focused on implementing our designs using open-source tools and thoroughly testing each to ensure proper functionality. We also developed C code to interface with a microcontroller, enabling performance comparisons between the different ReRAM architectures. Extensive testing was conducted, layouts were completed, and three out of the four designs reached final layout status.

However, during the second semester an unexpected shutdown of Efabless fabrication removed our ability to perform pre-check, tapeout check, and to submit the chip for fabrication. In response, we shifted our focus to completing the simulation, validation, and the documentation process. Despite this change, we met the project's primary objectives: to create, test, and compare four unique ReRAM architectures using an open-source workflow.

Next steps include finding a new company to fabricate our design, completing functional verification of designs, finalizing the MCU codebase for automated MVM testing, and publishing detailed documentation. This project demonstrates the viability of ReRAM for in-memory computation and lays the groundwork for future research into low-power, high efficiency artificial intelligence.

Learning Summary

Development Standards & Practices Used

Below is a bulleted list of our circuit and hardware design practices. Also listed is the software that we are using to implement our designs and the engineering standards we have been following.

Circuit Design Practices:

- Analog and digital circuit integration
- Noise management
- Timing analysis
- Simulation and validation
- Device sizing and parameter optimization

Hardware Design Practices:

- Design for fabrication
- ReRAM-specific design considerations

Software Practices:

- Xschem (circuit design software)
- Magic (layout design software)
- Ngspice (simulation software)

Engineering standards:

- *IEEE 1481-2019- IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA)*: This is applicable to our project since it specifies how our integrated circuit should be examined using a variety of design automation tools for timing and power consumption.
- *IEEE 1076.4-2000- IEEE Standard VITAL ASIC Modeling Specification*: This standard is relevant to our project because it calls for the testing of an ASIC chip using extremely precise and effective simulation models.
- *IEEE 1149.4-2010- IEEE Standard for a Mixed-Signal Test Bus*: This is relevant to our project since it will have both digital and analog components, and we will need to properly test each one separately and in tandem.

- *IEEE 1364-2005- IEEE Standard for Verilog Hardware Description Language*: Since our project requires us to create Verilog code to facilitate communication between the wrapper and the analog portion, this standard is appropriate for us.

Summary of Requirements

- Four different ReRAM compute crossbar architectures must be present in the final tape-out.
- Component circuits are individually characterizable and accessible through analog pins.
- Uncertainty evaluation on architectures being implemented; the difference between simulated ideal crossbar current and actual within one ADC step.
- C Code for the MCU to interface with the ReRAM that enables testing and demonstrates that the ReRAM can compute an MVM within an epsilon tolerance.
- Bring up Documentation for FORMing the ReRAM Cells and characterizing the component circuitry via individual test benches.

Updated Summary of Requirements

We provided an updated summary of requirements due to efabless shutting down. This shows what is not attainable after the shutdown.

- Component circuits are individually characterizable and accessible through analog pins.
- Four different ReRAM compute crossbar architectures must be present in the final top level design and layout.
- C Code for the MCU to interface with the ReRAM that enables testing and demonstrates that the ReRAM can compute an MVM within an epsilon tolerance.
- Bring up Documentation for FORMing the ReRAM Cells and characterizing the component circuitry via individual test benches.

Applicable Courses from Iowa State University Curriculum

- EE 330 - Integrated Electronics
- EE 465 - Digital VLSI Design
- EE 435 - Analog VLSI Circuit Design
- EE 501 - Analog and Mixed-Signal VLSI Circuit Design Techniques
- CPR E 281 - Digital Logic
- CPR E 288 - Embedded Systems I: Introduction
- CPR E 381 - Computer Organization and Assembly Level Programming

New Skills/Knowledge acquired that was not taught in courses

- ReRAM technology
- Hierarchical analog design
- Compute in memory design and application
- Open-source software: Xschem, Magic, Ngspice
- Skywater 130nm process
- Tape-out process of silicon chips

Table of Contents

List of Figures/Tables/Definitions

Figures

Tables

Definitions

ReRAM:	Resistive Random Access Memory
1T1R:	ReRAM cell consisting of 1 transistor and 1 memristor
ADC:	Analog to Digital Converter
XSchem:	Open-source schematic creation software
Magic:	Open-source layout creation software
Ngspice:	Open-source analog simulation software
Gaw:	Open-source analog waveform viewer

1. Introduction

1.1 Problem Statement

Matrix-vector multiplication (MVM) is one of the most common operations in machine learning applications. Performing MVM requires many multiply and accumulate operations, which takes a lot of time and energy in a typical CPU. Moving the required data back and forth in between the CPU and memory consumes a lot of energy in a traditional system. Compute-in-memory (CIM) technologies pose a potential solution to speeding up these processes by eliminating the memory bottleneck and allowing for parallel computation. ReRAM is an emerging, low-power, and non-volatile memory technology which may be used for CIM. Using ReRAM for CIM may require a rethink of ReRAM architectures. First, there is significant potential for impact from noise, both from the internal architecture of the ReRAM matrix as well as device noise from other chip components. Second, there are limited opportunities for ReRAM chip fabrication. For these reasons, we will design a fabricated test chip for ReRAM architecture exploration and characterization. The test chip will include multiple distinct ReRAM implementations in one design, resulting in a final chip that can test multiple implementations for research purposes. Ultimately, the final design will be laid out with four distinct architectures and a testbench for component characterization with simulation data demonstrating functionality.

1.2 Users and User Needs

The ISU ECpE faculty and their research teams (grad and undergrad) serve as the main client(s)/user(s) for this project. Secondary users include ChipForge, an Iowa State University club, while tertiary users encompass researchers and enthusiasts outside of our department. The objective of our project is to create a research test chip with multiple ReRAM architectures from past teams, in addition to our own, on the chip. In conjunction with the design of these architectures, comprehensive documentation will be produced to guide users on tool utilization, troubleshooting, and the design process. ISU research faculty will then use the chip to evaluate and characterize the different ReRAM architectures. Providing a finished top-level design, testbenches, and layout will meet the user needs. Additionally, in-depth documentation for the theoretical bring-up and testing methodology will be necessary for the users teams to evaluate the top level design and proposed architectures. To perform sufficient testing on each architecture, the research teams must be able to test and characterize each component individually to verify the functionality of system subcomponents. Along with testbench results, C code must be provided to allow the MCU to interface with the chip, performing testing and measurements.

1.3 What is ReRAM?

ReRAM is a part of emerging nonvolatile technologies that is aiming to address the limitations of conventional memory systems. ReRAM operates as a resistive switching, where a voltage induces a filament to grow between two electrodes. When the resistive material creates a filament it provides low resistance which entails a one. If the filament is broken then the resistance is high which creates a zero. There are three distinct modes for ReRAM; forming, writing and reading. Forming is when voltage is first applied to the cell to create the initial oxygen filament. Writing is where the filament is changed by the application of the electric field. Lastly, reading is where the resistance of the cell is transformed into binary data.

2. Requirements, Constraints, and Standards

2.1 Requirements

- Functional Requirements
    - Four different ReRAM compute crossbar architectures must be present in the final tapeout
        - Two architectures will come from the previous two team's final designs.
            - Both read from the bit line, and the source and bit lines are parallel in both designs.
            - One utilizes a 4-bit ADC, while the other utilizes a 1-bit ADC
        - Two additional architectures will be newly designed by our team
            - One will be a true crossbar design, with no transistors; a matrix of memristors
            - One will parallelize the source and word lines
    - Component circuits are individually characterizable and accessible through analog pins, and include the following:
        - One, three, and four bit ADCs
        - Transimpedance amplifier
        - ReRAM cell(s)
        - One, three, and four bit DACs
- Resource Requirements
    - Uncertainty evaluation on architectures being implemented; difference between simulated ideal crossbar current and actual within one ADC step
    - C Code for the MCU to interface with the ReRAM that enables testing and demonstrates that the ReRAM can compute a MVM within an epsilon tolerance
    - Bring-up Documentation for FORMing the ReRAM Cells and characterizing the component circuitry via individual testbenches.

2.2 Constraints

- Must use Efabless open source tools for design process
- Must use Skywater 130nm process
- Must use previous teams' architectures in our final design
- Final layout design must fit inside project wrapper

2.3 IEEE Standards

- *IEEE 1481-2019- IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA)*: This is applicable to our project since it specifies how our integrated circuit should be examined using a variety of design automation tools for timing and power consumption.
- *IEEE 1076.4-2000- IEEE Standard VITAL ASIC Modeling Specification*: This standard is relevant to our project because it calls for the testing of an ASIC chip using extremely precise and effective simulation models.

- *IEEE 1149.4-2010- IEEE Standard for a Mixed-Signal Test Bus*: This is relevant to our project since it will have both digital and analog components, and we will need to properly test each one separately and in tandem.
- *IEEE 1364-2005- IEEE Standard for Verilog Hardware Description Language*: Since our project requires us to create Verilog code to facilitate communication between the wrapper and the analog portion, this standard is appropriate for us.

2.4 Applicable Courses From ISU Curriculum

- EE 330
- ENGL 314
- EE 230
- EE 465
- EE 435
- CPR E 381
- CPR E 288

3. Project Plan

3.1 Project Management/Tracking Procedures

We plan on using the agile methodology for managing our project. Since we meet with our advisors every week, we will do one-week sprints where we can give updates on what was accomplished in the past week. We will use GitHub issues to track tasks on our project. We will also be very deliberate when deciding to commit our changes, ensuring that any time we have our project in a state that we want to save or talk about we can look at the corresponding commit.

3.2 Task Decomposition

Task 1: Figure out the tools and research ReRAM functionality

- Installing toolchain
- Demonstrate competency with tools and refine ChipForge tutorials
- Get an analog device through Pre-check

Task 2: Verify and integrate previous architectures and peripheral circuitry

- Looking at other team's components and testing if they work
- Get the other team's components through pre-check

Task 3: Research and implement new architures

- Create schematic for new architecture #1
- Create schematic for new architecture #2
- Integrate new architectures into top-level design

Task 4: Create Final Layout of Design

- Create layout of circuit component testbench
- Add each unique architecture to the layout

- Clear all DRC Errors from the design
- Make sure the design passes LVS

Task 5: Verify Behavior of Final Design

- Perform post extraction simulation on components
- Perform post extraction simulation on unique architectures
- Verify that simulation results match expected behavior

Task 6: Get Final Design Through Efabless Checks

- Get final design through Efabless hosted precheck
- Get final design through Efabless hosted tapeout check

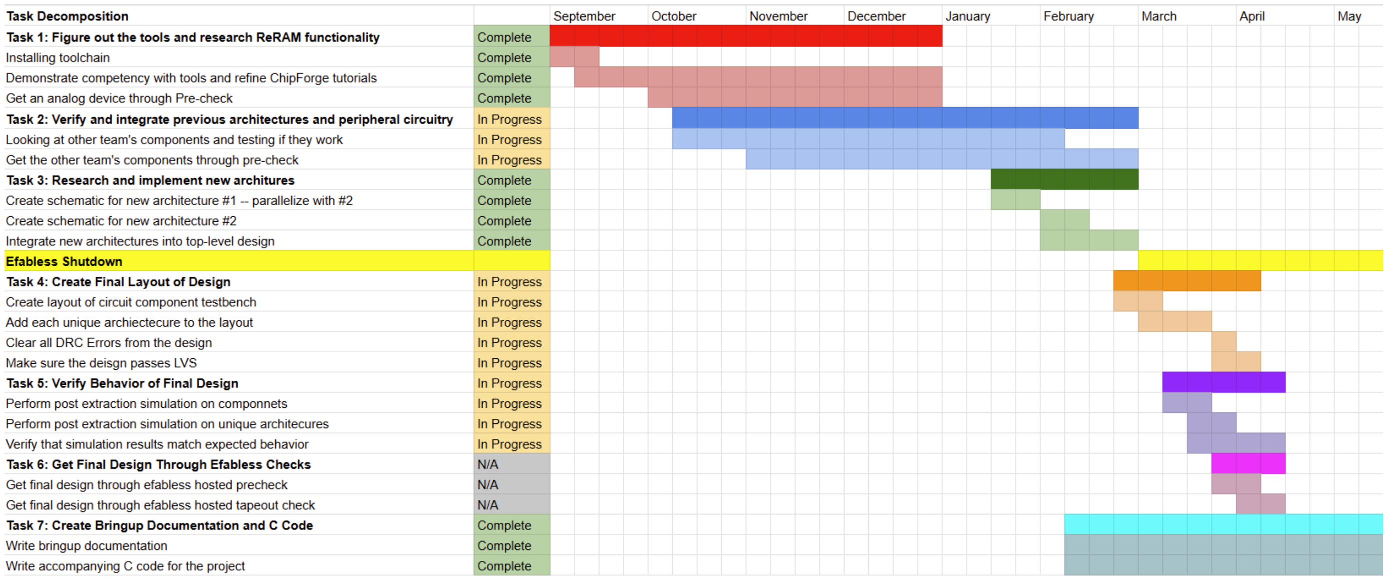
Task 7: Create Bring-up Documentation and C Code

- Write bring-up documentation
- Write accompanying C code for the project

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- All of the previous team's modules function as expected
- Post-integration of the previous team's modules function as expected while passing DRC and LVS checks
- We have created our own architectures that function as expected and pass DRC and LVS checks
- All four architectures are integrated and pass DRC, LVS, and precheck
- The component testbench has been integrated and passes DRC, LVS, and precheck

3.4 Project Timeline/Schedule



Task Decomposition		September	October	November	December	January	February	March	April	May
Task 1: Figure out the tools and research ReRAM functionality	Complete									
Installing toolchain	Complete									
Demonstrate competency with tools and refine ChipForge tutorials	Complete									
Get an analog device through Pre-check	Complete									
Task 2: Verify and integrate previous architectures and peripheral circuitry	In Progress									
Looking at other team's components and testing if they work	In Progress									
Get the other team's components through pre-check	In Progress									
Task 3: Research and implement new architures	Complete									
Create schematic for new architecture #1 -- parallelize with #2	Complete									
Create schematic for new architecture #2	Complete									
Integrate new architectures into top-level design	Complete									
Efabless Shutdown										
Task 4: Create Final Layout of Design	In Progress									
Create layout of circuit component testbench	In Progress									
Add each unique archiectecure to the layout	In Progress									
Clear all DRC Errors from the design	In Progress									
Make sure the deisgn passes LVS	In Progress									
Task 5: Verify Behavior of Final Design	In Progress									
Perform post extraction simulation on componnets	In Progress									
Perform post extraction simulation on unique architecures	In Progress									
Verify that simulation results match expected behavior	In Progress									
Task 6: Get Final Design Through Efabless Checks	N/A									
Get final design through efabless hosted precheck	N/A									
Get final design through efabless hosted tapeout check	N/A									
Task 7: Create Bringup Documentation and C Code	Complete									
Write bringup documentation	Complete									
Write accompanying C code for the project	Complete									

Figure 3.1: Gantt chart

3.5 Risks and Risk Management/Mitigation

- **Past team's design does not pass functional tests or pre-check**
    - **Risk 40% :** Past team's design does not pass functional tests or pre-check. Efabless tools have evolved over time, so past designs may no longer pass verification. Models may have been updated or files may be missing. To prevent a roadblock, we will begin by testing each component individually before integrating any of the designs to catch any problems early on. We may need to re-layout these components if they fail.
- **At least one of the new designs does not satisfy the requirements**
    - **Risk 15% :** There is a lot of potential for noise to interfere with the functionality of our circuits, as well as other various issues that could arise from our component circuitry. To mitigate this, we will make sure to do our research on different design architectures to produce a different process that our clients and end users will benefit from.
- **Integrated top-level project wrapper design fails functional tests or pre-check**
    - **Risk 55% :** Combining all of our components into one cohesive design may provide some issues while running the final checks. To mitigate this issue, we will make sure to run constant LVS and prechecks as we go along the integration process.
- **Flicker noise is more impactful on the Skywater process than expected**
    - **Risk 15%:** Finding the necessary values in the Skywater PDK to calculate the corner frequency between flicker noise and thermal noise has not been successful. In terms of simulation, there is not one agreed upon method of simulating flicker noise. Currently, we are unsure of what method our software is using, making simulation results more difficult to interpret. However, for most modern processes, the effects of thermal noise in the megahertz region are much more impactful than that of flicker noise [1]. Additionally, the impedance of the memristor will be much greater than that of any MOS device used in an architecture, so the dominant source of noise in these designs will be the ReRAM cells. To mitigate this potential risk, we can increase the clock speed of our circuit to around 20MHz where thermal noise will certainly be dominant.

3.5.1 Updated Risks and Risk Management/Mitigation Post Efabless-Shutdown

Due to Efabless shutdown we had to update a few of our risks.

- **Final design is unable to be fabricated**
    - **Risk 100%:** Due to the efabless shutdown in early march, there is no possibility of our design being fabricated at this time. Despite losing the resources from efabless and their open-source Slack community, we are working through the project to meet our deadlines as if this event did not occur. Working towards our original goal is the best mitigation strategy to reach a successful outcome.
- **Past team's design does not pass functional tests or pre-check**
    - **Risk 100% :** Past team's design does not pass functional tests or pre-check. Efabless tools have evolved over time, so past designs no longer pass verification.

Two of the main reasons for this stem from tool versioning issues, and the ReRAM SPICE file not being able to pass precheck. Models may have been updated or files may be missing. To try and work through roadblocks, we will test each component individually before integrating any of the designs to catch sources of error. We will need to re-layout these components if they fail or did not have a layout to begin with.

- **Integrated top-level project wrapper design fails functional tests or pre-check**
    - **Risk 100% :** As previously mentioned, the ReRAM cell SPICE model is able to be simulated, but is not able to pass pre-check. This was an issue we planned to work through with the efabless employees, but after the shutdown this was no longer an option. To mitigate this issue, we will make sure that our final design passes both DRC and LVS checks to get as close to achieving our original goal as possible.

3.6 Personnel Effort Requirements

	Time
Task 1: Figure out the tools and research ReRAM functionality	**70 hrs**
Installing toolchain	10 hrs
Demonstrate competency with tools and refine ChipForge tutorials	40 hrs
Get an analog device through Pre-check	20 hrs
Task 2: Verify and integrate previous architectures and peripheral circuitry	**120 hrs**
Looking at other team's components and testing if they work	60 hrs
Get the other team's components through pre-check	60 hrs
Task 3: Research and implement new architures	**90 hrs**
Create schematic for new architecture #1	30 hrs
Create schematic for new architecture #2	30 hrs
Integrate new architectures into top-level design	30 hrs
Task 4: Create Final Layout of Design	**50 hrs**

Create layout of circuit component testbench	20 hrs
Add each unique architecture to the layout	20 hrs
Clear all DRC Errors from the design	5 hrs
Make sure the design passes LVS	5 hrs
Task 5: Verify Behavior of Final Design	**80 hrs**
Perform post extraction simulation on components	20 hrs
Perform post extraction simulation on unique architectures	30 hrs
Verify that simulation results match expected behavior	30 hrs
Task 6: Get Final Design Through Efabless Checks	**60 hrs**
Get final design through Efabless hosted precheck	30 hrs
Get final design through efabless hosted tapeout check	30 hrs
Task 7: Create Bring-up Documentation and C Code	**40 hrs**
Write bring-up documentation	20 hrs
Write accompanying C code for the project	20 hrs

Table 3.1: Personnel effort requirements

3.7 Other Resource Requirements

All of the CAD tools are open-source software. However, the software that will be used is not well documented so we have to rely heavily on previous teams documentation and tutorials. The other resources are used to create schematics and layouts and test the functionality of the components and circuits. These resources include:

- Xschem: Schematic capture
- Magic: Layout
- Netgen: LVS
- Slack: online community for help with open source toolset

We also have access to the lab in Durham 310, which has FPGAs that we can use through their comparch computer. We may need to design a PCB breakout board once our design is fabricated, for which we can use ChipForge documentation. Finally, we have $9,750 for the

ReRAM Efabless tapeout for ChipIgnite 2504, which is provided to us by our client, Dr. Duwe, through an NSF award.

4. Design

4.1 Broader Context

4.1.1 Broader context

The test chip is designed for the research and engineering community. This chip will explore the next-generation of in compute memory. The communities that are being affected by this design are data-intensive industries such as healthcare, finance, autonomous systems, and machine learning will benefit from the reduced energy consumption associated with ReRAM compute in memory systems. Lastly, the societal needs that our project addresses are energy efficiencies and technological advancement which create competitiveness in the future for technology.

Area	Description	Examples
Public Health, Safety, and Welfare	The ReRAM test chip will demonstrate computational efficiency potentially reducing energy consumption of artificial intelligence machine learning. This can benefit communities by lowering energy demand and associated emissions	Reducing energy consumption would decrease greenhouse gas emissions which will improve the quality of the air in areas with significant computational infrastructure.
Global, Cultural, and Social	By advancing energy-efficient technology, this project aligns with the global efforts towards sustainable and energy reductions, valued by diverse and professional communities.	The project supports globe sustainability initiatives by reducing the carbon footprint of computational operations.
Environmental	ReRAM aims to reduce energy consumption and create low powered technologies compared to traditional memory. However, the fabrication process might use hazardous material, so that effect is unknown.	While the energy usage during the operation of the ReRAM compute-in-memory systems is reduced. The manufacturing process of the ReRAM cells, such as transition metal oxide, must be evaluated to ensure sustainability.
Economic	The cost of this device may be expensive at first but it creates a more energy efficient outcome. In the long run it will make	Successful implementations of ReRAM compute-in-memory systems will create job opportunities in low-power hardware design and manufacturing.

Table 4.1: Broader context

4.1.2 Prior Work/Solutions

Since ReRAM is an emerging technology, especially for compute purposes, there are no commercially available solutions on the market. We have looked closely, of course, at the previous two senior design team's projects for reference. In addition, the ISAAC design is an example of an academic implementation of a ReRAM-based true machine learning accelerator. The table below lists pros and cons of each of these designs, as well as our design.

Product	**Pros**	**Cons**
ISAAC ReRAM-based CNN Accelerator [3]	● Acts as a true AI accelerator ● Uses 1T1R grid for more precise memristor writes ● Shared ADC allows for better precision	● Shared ADC requires sample and hold circuits ● Very specialized to one application, not as general for research
Design made by sddec23-08	● Uses 1T1R grid for more precise memristor writes ● Multiple ADCs; one for each read line	● Multiple ADCs = less precision (1 bit) ● Only includes one ReRAM architecture
Design made by sddec24-13	● Uses 1T1R grid for more precise memristor writes ● Shared ADC allows for better precision	● Shared ADC requires sample and hold circuits ● Only includes one ReRAM architecture
Our design	● Includes four individually accessible unique ReRAM architectures ● Includes individually characterizable component circuitry ● Generalized for research purposes	● Research-focused, unlikely to be useful in a true machine learning context ● Could suffer from over-complication since it includes many different designs

Table 4.2: Pros/Cons of prior work/solutions

4.1.3 Technical Complexity

Our design presents a variety of technical challenges:

1. Since we are using open source software, none of our team members are familiar with the tools we are using. This provides a barrier of entry to getting started with our actual design.
2. ReRAM is an emerging technology, and there are very few opportunities available for fabrication of ReRAM chips. Also, there is little information about ReRAM usage for compute-in-memory applications, so we are exploring a new frontier.
3. We are including four different ReRAM architectures in our design, which of course increases the complexity of our design. However, this will likely help us down the line: if we find that one of our architectures doesn't work as intended, there are still three others to test.
4. We are also integrating a number of components acting as the peripheral circuitry of the design. This includes S&H circuits, TIAs, DACs, and ADCs. While our team doesn't have direct experience with each of these components, they are very well studied in academia and in industry, so finding documentation about them shouldn't be a struggle.

4.2 Design Exploration

4.2.1 Design Decisions

1. **New ReRAM Architectures**

The first design decision we had to make was what kinds of crossbars we would use in our unique ReRAM architectures. For our first crossbar, we decided to do a similar 1T1R matrix as the two senior design teams that came before us, but ours would be different by parallelizing the bitlines and sourcelines in our layout. The previous two teams had their sourcelines parallel to their wordlines. We decided to change things up with our second crossbar design and explore a true ReRAM crossbar, which uses no transistors and only uses a matrix on ReRAM cells. We felt that this would offer a unique solution that maximizes density within our crossbar. The operations on this crossbar are more difficult and noise-prone, but the potential for high-density ReRAM computation could be worth the risk.

2. **ADC Resolution**

We had a few options for ADC resolution in our project. We could use the 1-bit ADC from team sddec23-08, the 4-bit ADC from team sddec24-13, or design our own new ADC at a different resolution. We were partly swayed in our decision due to verification difficulties with the 4-bit ADC (seemingly due to differences in our tool installations), but we ended up using the 1-bit ADC in our design. The advantage to this decision is that each bitline has its own ADC, so there is no need for additional circuitry for sharing a single ADC among each line. This simplistic nature should work well for a research chip that is more for proving the concept than getting precise results.

3. **Multiplexing Outputs**

The next major design decision came when integrating all of our designs and our component testbench into our top level schematic. The user needs a way to choose which architecture to get output from, since there are not enough pins in the Caravel wrapper to connect to all of our architectures and component circuit testbenches. So, we designed a multiplexing circuit that takes in two select lines for choosing between our four architectures, as well as an additional select line for choosing whether the output should come from the architectures or the component testbench. The user can then change the values of these select lines depending on their desired output. The logic analyzer output will always be from one of our crossbar architectures or the component testbench.

4.2.2 Ideation

One of the key uncertainties in our design process is whether we will be able to achieve precise and accurate data retrieval from the output line. Currently, we are still in the process of determining the length of the interconnects and assessing the level of noise that must be accounted for when acquiring data from the output. This is why the selection of an appropriately designed Analog-to-Digital Converter (ADC) is critical to the success of the project. The ADC choice will not only determine whether the project is feasible but will also play a crucial role in ensuring the functionality and scalability of the system in the broader context of the project. This includes the goal of enhancing the performance of AI accelerators by reducing power consumption and improving processing speed.

Additionally, we are reviewing the designs developed by our previous team, with the possibility of incorporating elements of their architecture into our own designs. Their work includes the use of a 4-bit Flash ADC and a Sample-and-Hold (S&H) ADC, both of which provide useful insights and potential integration points for our architectures. Below are five potential ADC design approaches that we are considering in our two architectures.

- Current_Mode ADC
- 4-bit Flash ADC
- 1-bit S&H ADC
- Delta Sigma ADC
- Pipeline ADC

4.2.3 Decision-Making and Trade-off

Below is a table that weighs the benefits and drawbacks of each ADC design. With the use of this table and previous designs, we will choose whether we want to implement a new ADC within our designs that differentiates from the previous teams or utilize one of theirs.

Architecture Type	Pros	Cons
Current-Mode ADC	It can handle small variations in current levels and use low power consumption.	Susceptible to small amounts of noise and low resolution compared to other ADC designs.
4-bit Flash ADC	Fast conversions as well as simplicity in design with ideal parallel processing.	High power consumption as well as large area and susceptible to noise environments.
1-bit sample and hold ADC	Extremely fast conversions, extremely low power consumption, and simplicity in design.	Only limited to binary decision-making, not suitable for high precision applications, and sensitive to noise.
Delta Sigma ADC	Very high resolution with exceptional precision and accuracy. Great noise reduction.	Very slow and has higher power consumption compared to other ADC designs.
Pipeline ADC	High speed with fast conversion rates, making it useful for high-speed applications.	More complex than other ADC designs and has high power consumption. It may lack high resolution based on the application.

Table 4.3: Comparing ADC architectures

We also needed to explore pros and cons for different ReRAM crossbar architectures. Below is a table to visualize the positives and negatives of each design.

Crossbar Type	Pros	Cons
1T1R grid, parallel word and source lines	Transistors make it easier to select individual cells for operations	Less dense
1T1R grid, parallel bit and source lines	Transistors make it easier to select individual cells for operations, also able to test an additional layout for noise	Less dense, need to create and verify an entirely new layout
True crossbar	Higher density	More overhead for operations

Table 4.4: Comparing ReRAM crossbar architectures

4.3 Proposed Design

4.3.1 Overview

Our design aims to characterize four unique matrix-vector multiplication modules that utilize ReRAM for performing compute-in-memory operations. These modules will consist of component circuitry, the first of which is the ReRAM crossbar itself. The ReRAM crossbar is a matrix of ReRAM cells where the actual computation will take place. Another component circuit is the trans-impedance amplifier (TIA), which will convert currents into voltages which will represent the result of the operation. The final major component circuit is the ADC, which will convert our analog result into a digital value readable by the CPU. Each component circuit will be individually characterizable as part of our design as well.

4.3.2 Detailed Design and Visuals

The goal of this design is to combine four different ReRAM crossbar architectures into one top-level block design. These will all be connected to the microcontroller via Logic Analyzer Pins, which are responsible for communication between the microcontroller and the ReRAM crossbar systems, such as reading values from the current crossbar or sending values to it. Two of the ReRAM crossbars will be sourced from the previous teams' designs, while the other two will be original designs. The original designs should closely resemble the previous designs, with minor architectural changes, such as swapping the directions of the source and bit lines. The top-level design should also include separate, individually characterizable testbenches for the different types of peripheral circuitry, including the ADC, DAC, and TIA. The top-level diagram outlines the design, while a lower diagram presents a potential internal design for a ReRAM crossbar. The ReRAM crossbar consists of a matrix of ReRAM cells connected by lines called bit lines, source lines, and word lines. The matrix is preloaded with values by setting the conductances of each memristor in the matrix, and input values are sent into the matrix via either the bit line or the source line, depending on the specific architecture. The previous teams' designs should be used in depth as references when designing the new architectures.

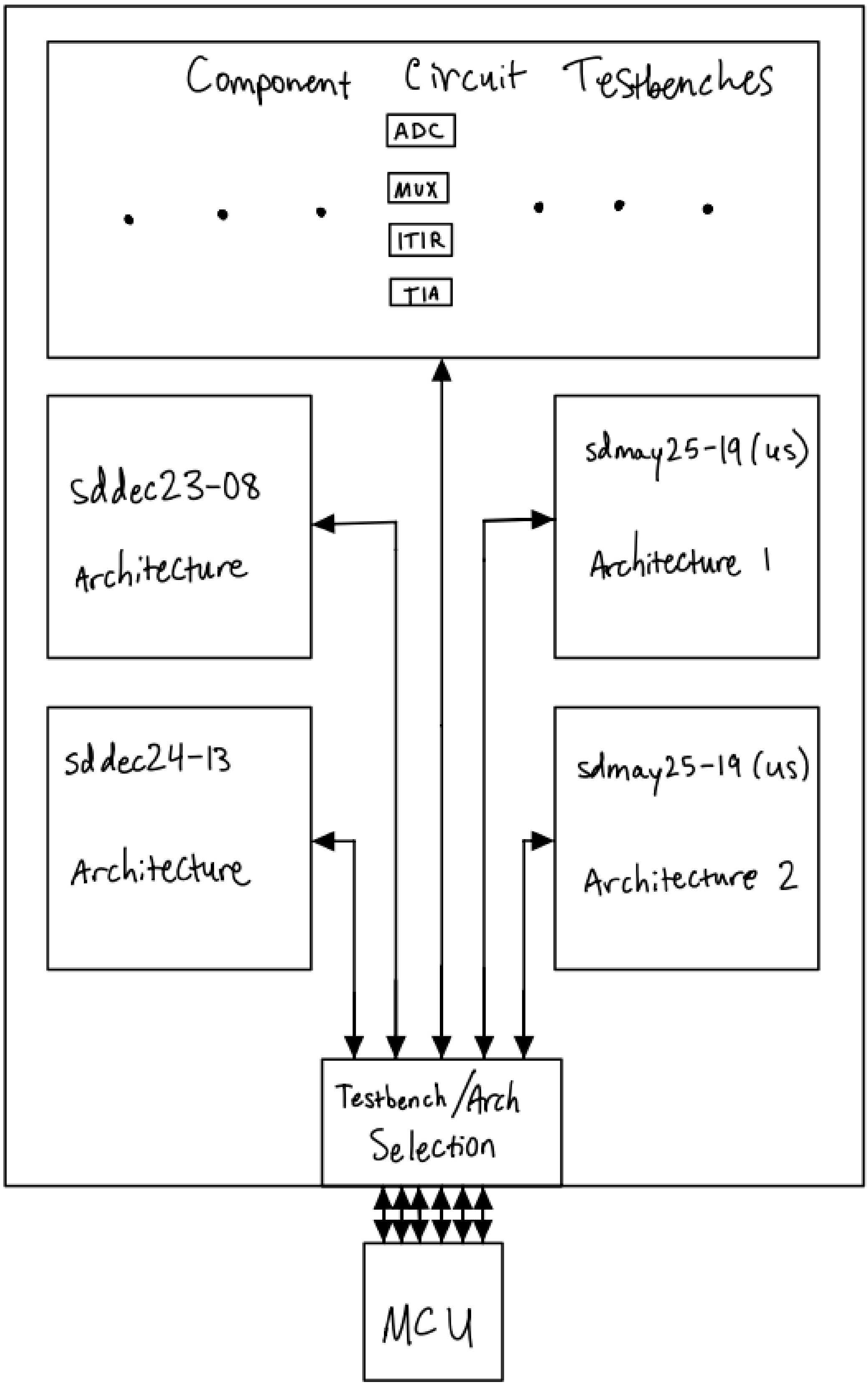


Figure 4.1: Top level block design

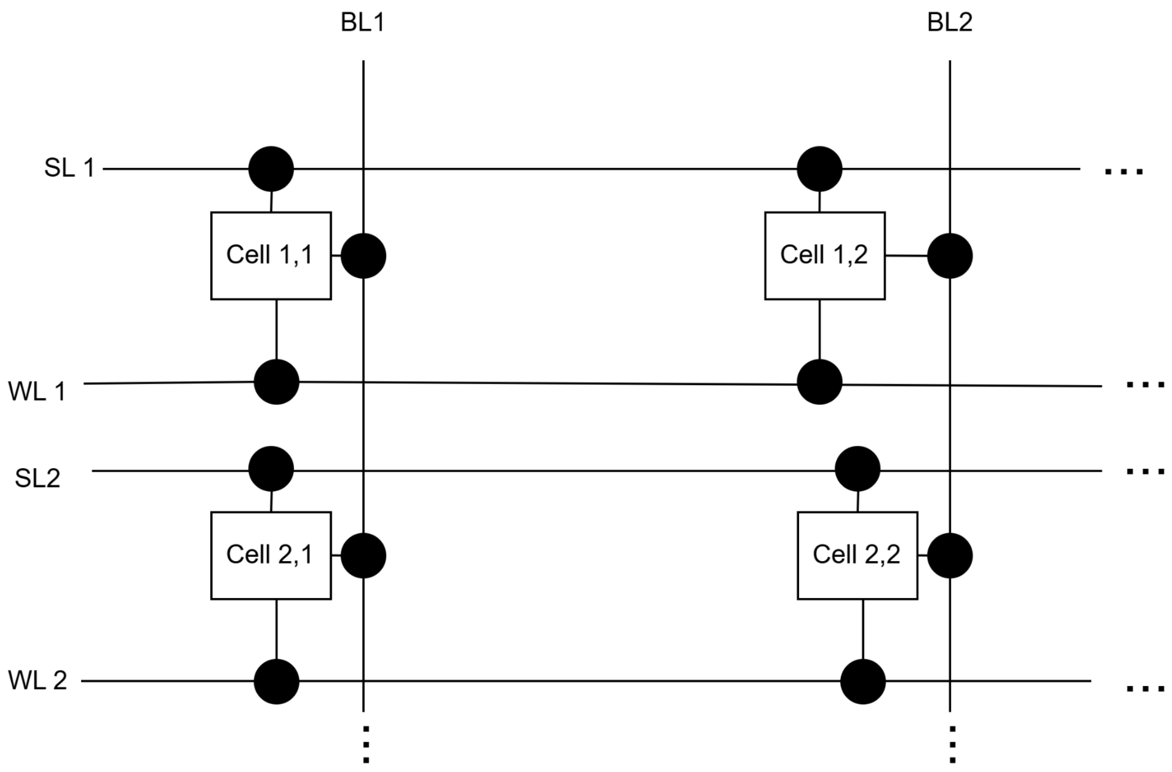


Figure 4.2: Schematic for one ReRAM crossbar architecture

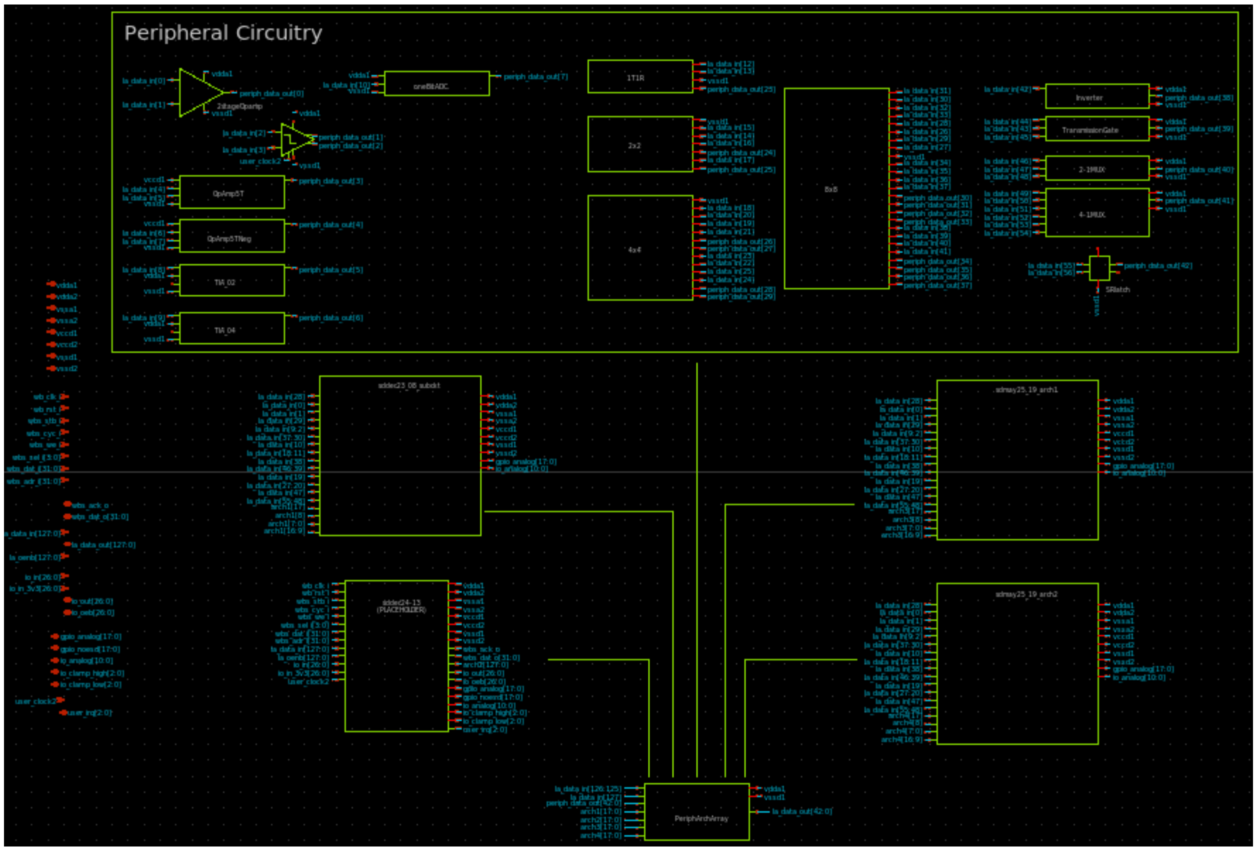


Figure 4.3: Schematic for our final top level design

Our top level schematic matches up well with our original vision that we created in our block diagram. It consists of our four architectures and our component/peripheral circuitry testbench area, all connected to a multiplexer that determines our final output.

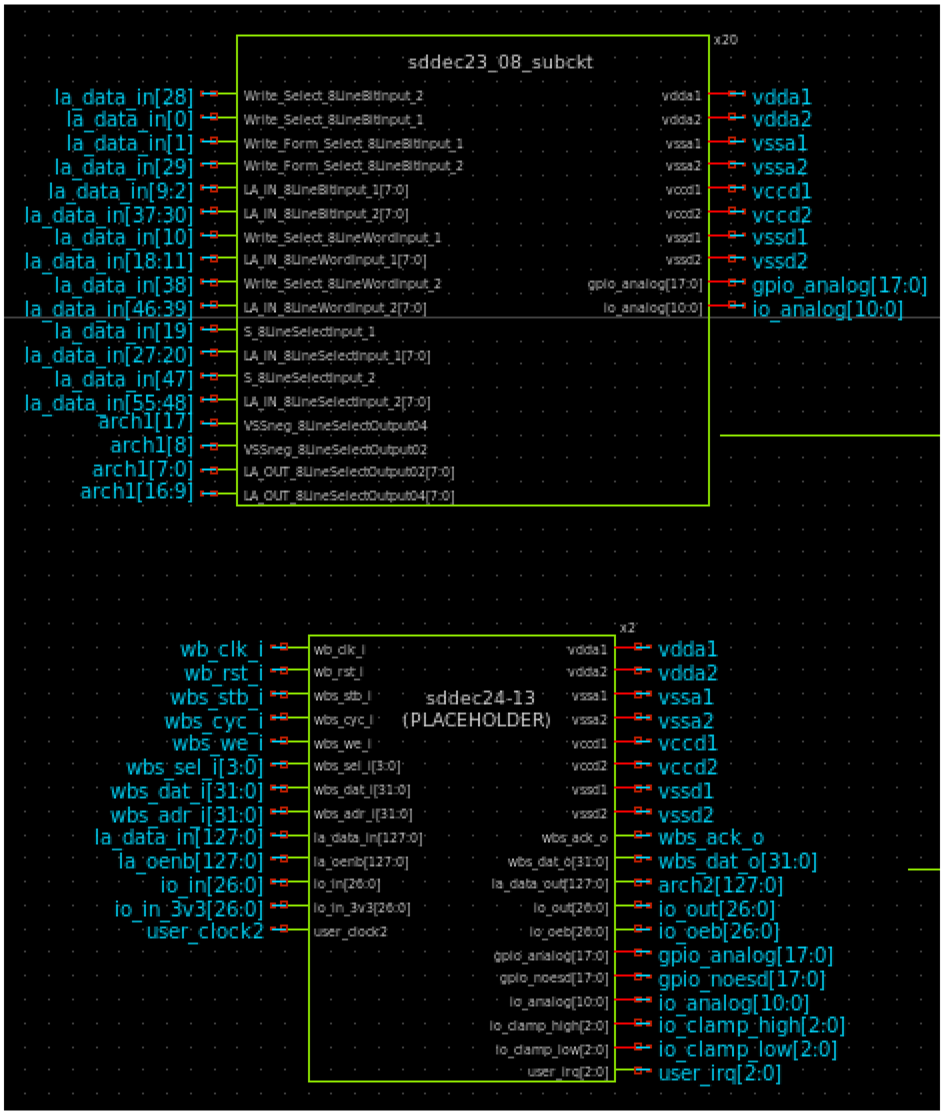


Figure 4.4: Architectures from SDDec38-08 and SDDec24-13 in our top level design

Two of our four unique architectures were designed by the previous two teams. They both used the same crossbar architecture, but their designs differed in peripheral circuitry. Team sddec23-08 used a 1-bit ADC, while one of sddec24-13's major contributions was a 4-bit ADC. Due to this difference, the design with the 4-bit ADC shared a single ADC among each bitline, while the other design had individual ADCs for each bitline.

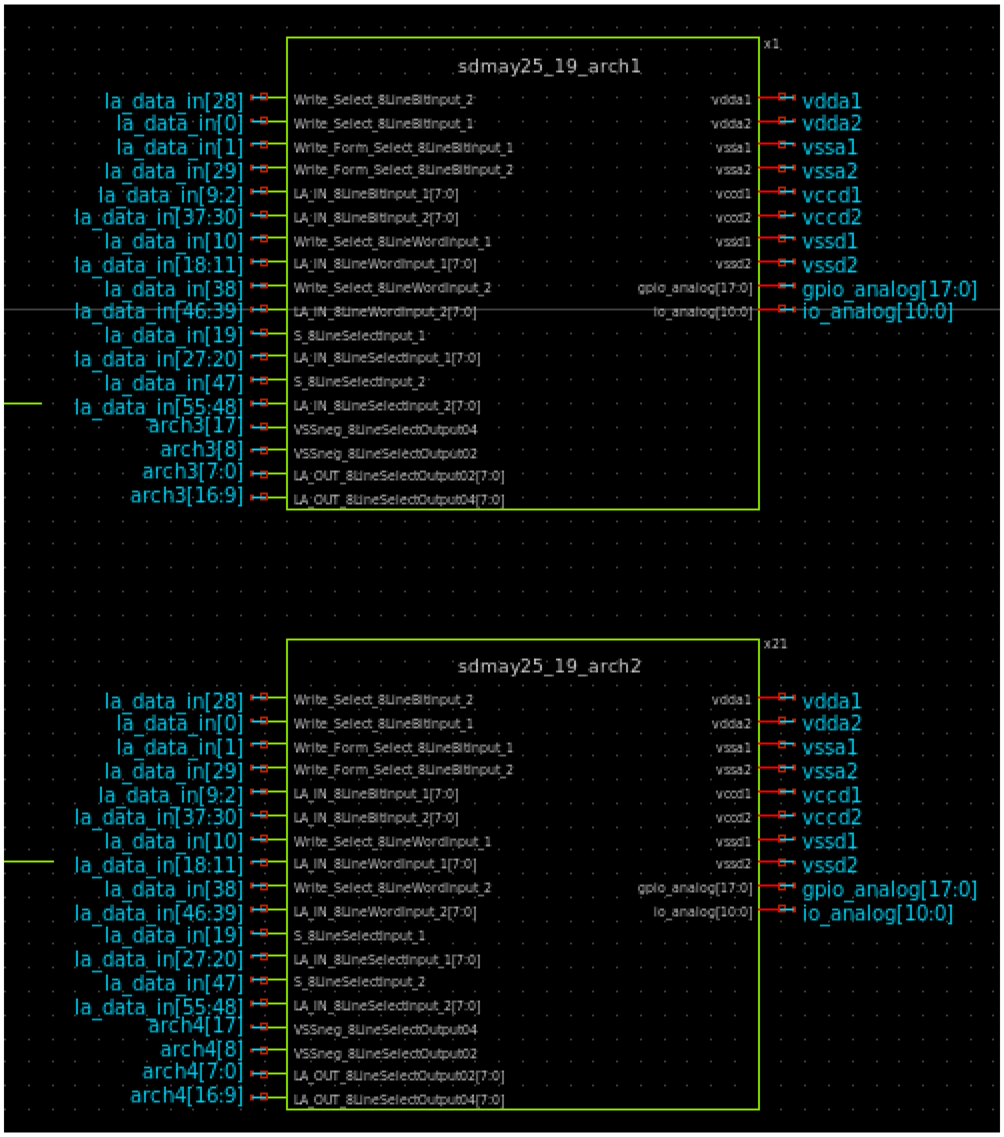

sdmay25_19_arch1
x1
la_data_in[28]
la_data_in[0]
la_data_in[1]
la_data_in[29]
la_data_in[9:2]
la_data_in[37:30]
la_data_in[10]
la_data_in[18:11]
la_data_in[38]
la_data_in[46:39]
la_data_in[19]
la_data_in[27:20]
la_data_in[47]
la_data_in[55:48]
arch3[17]
arch3[8]
arch3[7:0]
arch3[16:9]
Write_Select_8LineBitInput_2
Write_Select_8LineBitInput_1
Write_Form_Select_8LineBitInput_1
Write_Form_Select_8LineBitInput_2
LA_IN_8LineBitInput_1[7:0]
LA_IN_8LineBitInput_2[7:0]
Write_Select_8LineWordInput_1
LA_IN_8LineWordInput_1[7:0]
Write_Select_8LineWordInput_2
LA_IN_8LineWordInput_2[7:0]
S_8LineSelectInput_1
LA_IN_8LineSelectInput_1[7:0]
S_8LineSelectInput_2
LA_IN_8LineSelectInput_2[7:0]
VSSneg_8LineSelectOutput04
VSSneg_8LineSelectOutput02
LA_OUT_8LineSelectOutput02[7:0]
LA_OUT_8LineSelectOutput04[7:0]
vdda1
vdda2
vssa1
vssa2
vccd1
vccd2
vssd1
vssd2
gpio_analog[17:0]
io_analog[10:0]
sdmay25_19_arch2
x21
arch4[17]
arch4[8]
arch4[7:0]
arch4[16:9]

Figure 4.5: Both architectures from this team in top level design

Our other two architectures are the ones we contributed, which use the same peripheral circuitry as sddec23-08. Our first architecture has a similar 1T1R grid crossbar, but we chose to layout the crossbar such that the sourcelines were parallel to the bitlines rather than the wordlines. We hoped that this would offer the opportunity to analyze and compare the effects of noise between the two layouts. Our second architecture uses a true crossbar design, which is just a grid of memristors. Since testing and verification took the bulk of the time spent on this project, we went with the same peripheral circuitry for consistency – this made the selection between different architectures much less complicated.

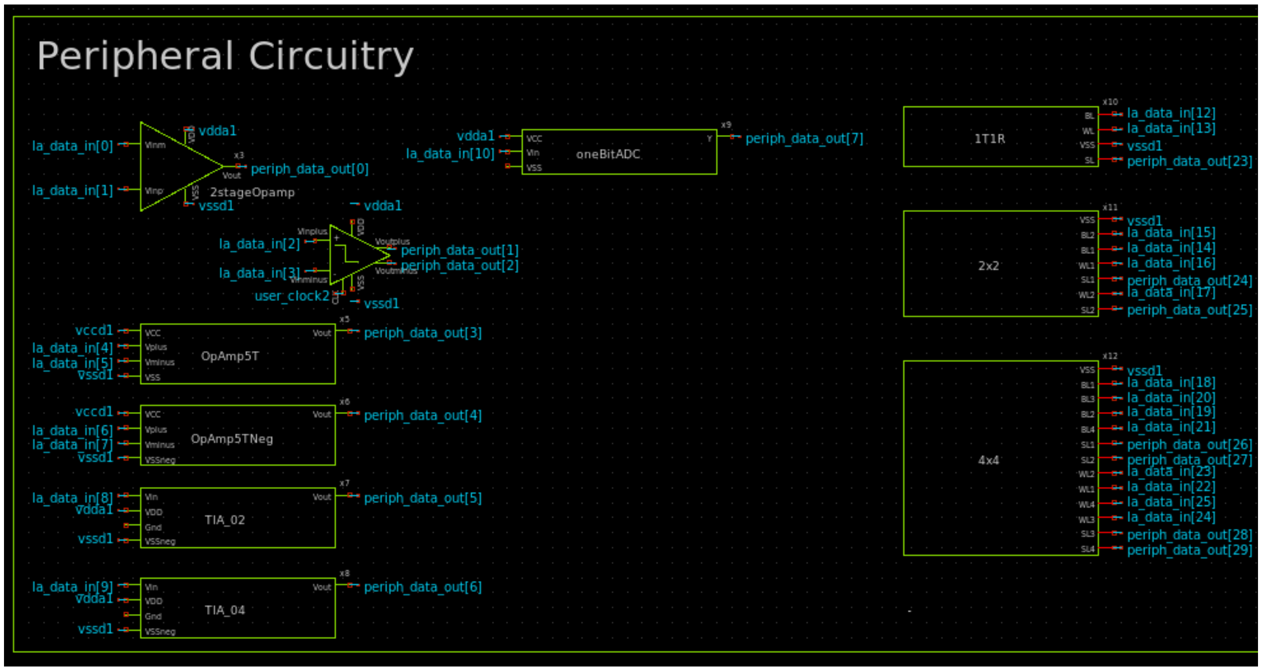


Figure 4.6: Peripheral circuitry in our top level design

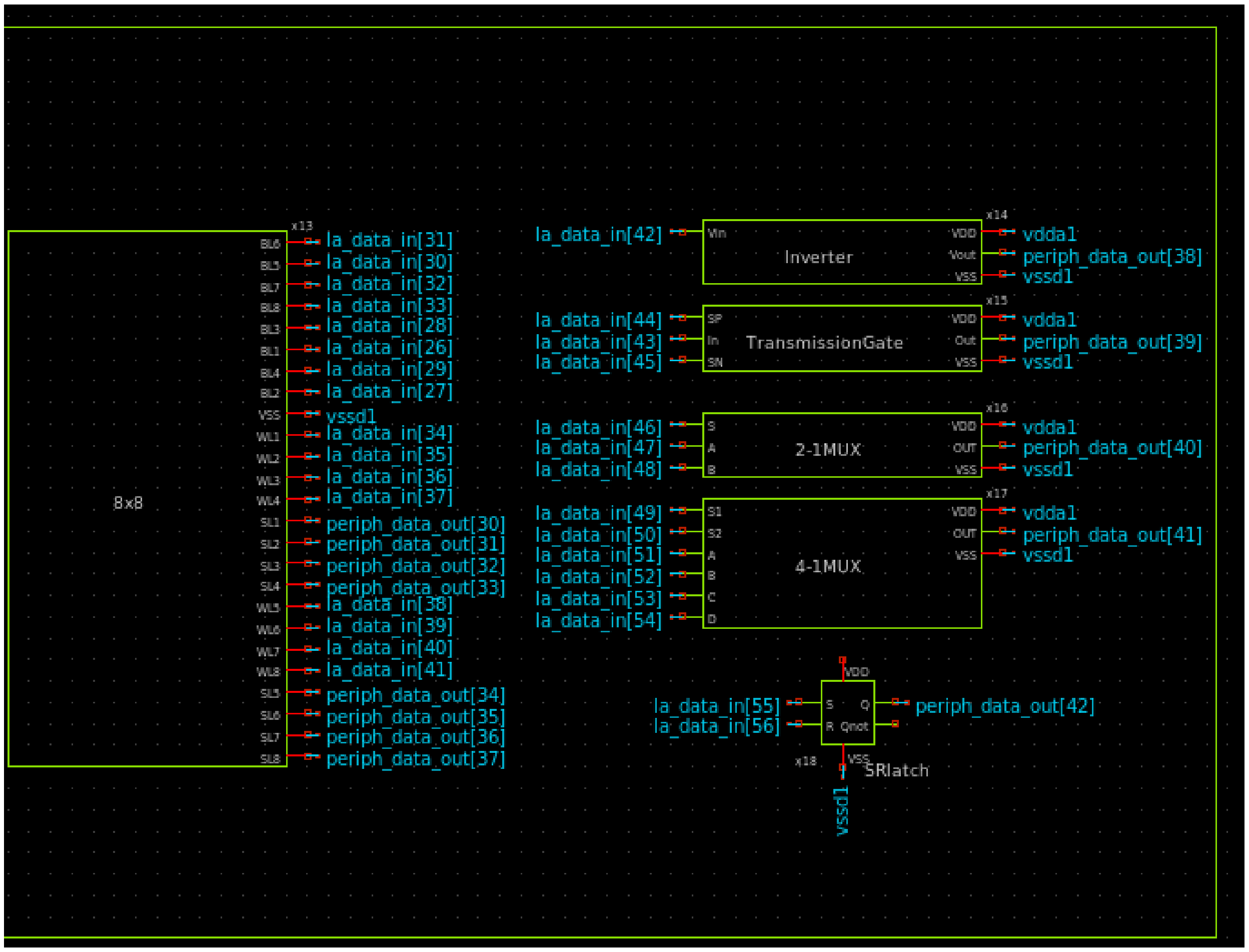


Figure 4.7: Peripheral circuitry in our top level design (cont.)

Our peripheral circuitry area consists of individually accessible versions of each component circuit in our architectures. This includes the various sizes of crossbars, simple circuits like multiplexers and inverters, and peripheral circuits such as ADCs and TIAs.

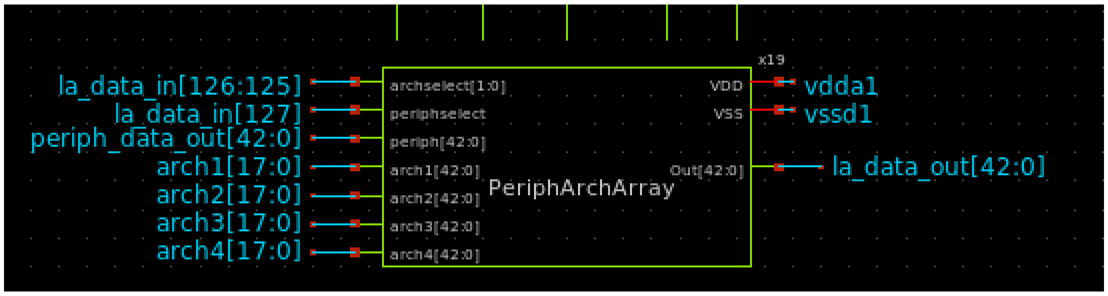


Figure 4.8: Architecture select MUX outputting to MCU

Our selection circuit took in data from each of our architectures as well as our component circuits, and used three select bits to give the user freedom to choose where they want to take output from.

This circuit will allow us to interface with the MCU (designed by efabless), enabling us to test and characterize the components and architectures we have spent the year designing. Notes on how to use this circuit will be included in our bring-up documentation for future users. The documentation will also help minimize the risk that any chips/ReRAM cells get damaged in the bring-up and testing process.

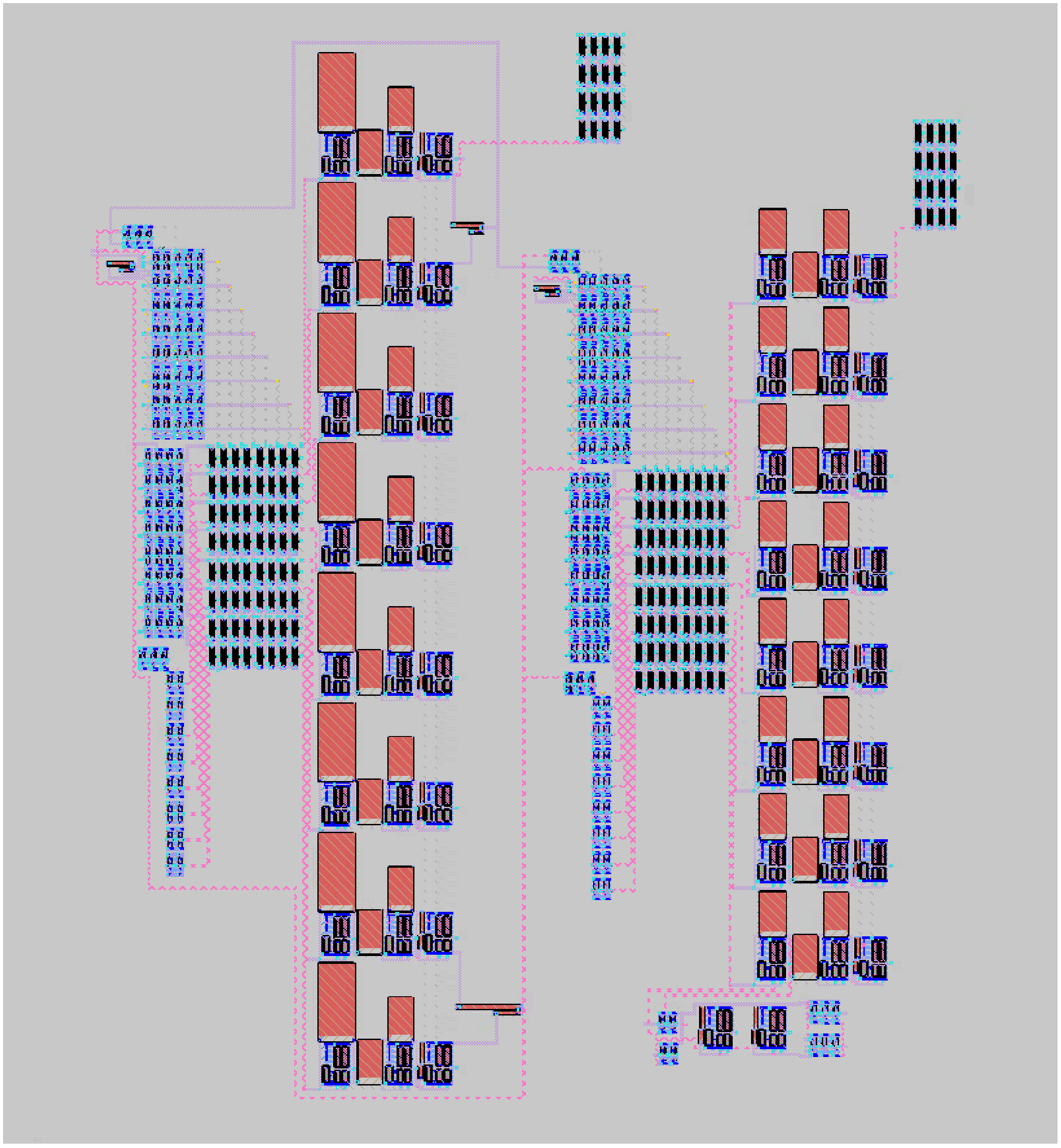


Figure 4.9: Our architecture one final layout

4.3.3 Functionality

Once fabricated, our chip will act as a daughter board that can be connected to a microcontroller. The microcontroller will perform FORMing operations to set up the ReRAM cells for computation. The microcontroller will also be able to send and receive data from each architecture on the chip, in addition to the peripheral circuitry testbench. During read/write operations to the chip, the microcontroller will be able to perform multiply and accumulate operations on each architecture. This data, in combination with characterizations performed in the peripheral circuitry testbench will allow the users to characterize the components and also get a better understanding of which ReRAM architectures are better suited for compute in memory workloads.

4.3.4 Areas of Concern and Development

Our design plan currently meets our user requirements by containing four unique ReRAM crossbar architectures and individually characterizable component circuitry. Our biggest concern moving forward with our design will be about the feasibility of our ReRAM crossbar architecture designs. As we complete simulations, we may find that certain configurations of ReRAM cells don't effectively carry out the operations necessary for our design to meet requirements. If this is the case, we will need to either change the configuration of the faulty architecture or come to a conclusion backed by concrete evidence that four unique adequate architectures do not exist.

4.4 Technology Considerations

Our design presents a variety of technical challenges:

1. Since we are using open source software, none of our team members are familiar with the tools we are using. This provides a barrier of entry to getting started with our actual design.
2. ReRAM is an emerging technology, and there are very few opportunities available for fabrication of ReRAM chips. Also, there is little information about ReRAM usage for compute-in-memory applications, so we are exploring a new frontier.
3. We are including four different ReRAM architectures in our design, which of course increases the complexity of our design. However, this will likely help us down the line: if we find that one of our architectures doesn't work as intended, there are still three others to test.
4. We are also integrating a number of components acting as the peripheral circuitry of the design. This includes S&H circuits, TIAs, DACs, and ADCs. While our team doesn't have direct experience with each of these components, they are very well studied in academia and in industry, so finding documentation about them shouldn't be a struggle.

5. Testing

The nature of our project requires rigorous testing, as there are many different components that all must work separately and also must be integrated together in order to reach our final product. Each component circuit, whether inherited from previous group's designs or newly designed by our group, must have an individual testbench that works as expected. In addition, these components must remain functional when integrated together. To ensure that there are no unexpected issues, our team's testing philosophy will be to continuously test and integrate new components and run our design through pre-check each time we make changes. This ensures that we never add too many features at one time, so that if issues arise we can solve them swiftly by identifying the most recent change.

5.1 Unit Testing

The individual components of every circuit must be tested. For each component, we created a schematic testbench using XSchem which was used to simulate the behavior of our circuit and compare it to the theoretical behavior. The testbenches can also be used for post-layout simulations which were carried out for each component in Magic. In the layout tool Magic, parasitics such as unintended capacitance can be introduced during simulation due to layers being placed too closely together in the layout. These parasitic effects are especially relevant in post-layout analysis. Magic also includes tools for extracting SPICE netlists from our layouts. Whether simulating pre-layout or post-layout, we used Ngspice to carry out our simulations and Gaw and built in graphs in XSchem for viewing out resulting waveforms. Additionally, for the ReRAM computer architectures, each individual component required thorough testing. Furthermore, the components that we have created also have testbenches. All of the individual testbenches were included in our final top-level design to enable post-tapeout simulation of our component circuitry.

5.2 Interface Testing

Our design includes two primary interfaces: one that connects the individual components within our four ReRAM architectures, and another that links all of our individual analog components to the on-chip RISC-V processor. Within our ReRAM architectures, we connected our components to the appropriate lines in our crossbar, which is dependent on our unique architectures. This was tested iteratively to ensure accuracy. Using XSchem for testbenches, Magic for layouts, Ngspice for simulations and Gaw/built in graphs for viewing waveforms. Component circuits were gradually integrated, one at a time, and tested at each step to ensure proper functionality as the design progressed. Using this strategy, we will be able to determine what component is acting out of accordance with our expectations and make adjustments to our design as needed. For the interface between our circuity and the processor, we integrated our full design into the Efabless-provided analog wrapper. This crucial step establishes the necessary connections to the logic analyzer pins, which creates an efficient communication with the processor for system analysis and verification

5.3 Integration Testing

The most critical integration path in our design was the integration of our analog circuitry into the provided analog wrapper that allows communication between the RISC-V processor and our circuitry through logic analyzer pins. As per the project requirements, our four ReRAM architectures must be accessible via logic analyzer pins, as well as individual component testbenches for characterization and testing purposes. To validate this integration, we first tested our high-level analog circuitry as a complete unit to ensure that it is working properly using the same methods mentioned in the above sections. Once these tests simulated properly, we proceeded with the integration of the high-level analog circuitry into the analog wrapper. This step entailed establishing connections between the appropriate logic analyzer pins and the inputs and outputs of our circuit.

5.4 System Testing

Following the completion of our circuit and its integration within the analog wrapper, the next stage involved the establishment of test cases to evaluate the functionality of our design. C code was developed to implement the multiplication function across each of our four ReRAM architectures. This code served to validate the processor's capacity to effectively dispatch these computational tasks to our analog circuitry and to receive the resultant output. Furthermore, C code was also generated for the purpose of testing the discrete component circuit. The utilization of these C programs facilitated a comprehensive assessment of the entire system, encompassing a detailed examination of the interface mediating communication between processor and our analog circuitry.

5.5 Regression Testing

For this project, regression testing is critical for a smooth reliable development process. The strategy for a successful regression testing involves the systematic re-execution of previously successful test cases following any modifications or additions to the design. The primary objective is to verify that new changes do not interfere negatively with the functionality of our design particularly within the analog circuitry and its interface with the RISC-V processor. This will include re-running tests for ReRAM architecture operations, processor-circuitry communication, and individual component behavior. Furthermore, targeted tests were created to address potential negative setbacks on the specific nature of design changes, with the goal of maintaining the integrity of the system's functionality through the development lifecycle.

5.6 Acceptance Testing

Acceptance testing is crucial in the testing process that determines whether our design meets the requirements and has the components to be successful. This includes four ReRAM architectures with accompanying component circuitry, as well as individually characterizable component circuit testbenches. If these were present in the design, then we moved on to the second acceptance criterion, which is that our design must pass the Efabless-provided pre-check and tapeout check on the Efabless servers by the tapeout date, April 21st, 2025. If our design passes all checks and contains all required components, it will be acceptable.

5.7 User Testing

User testing is a vital part of the design process and since this project is intended to be fabricated in the future, we made sure to follow the criteria and unexpected changes the client and the advisor brought forth. We also had weekly meetings to ensure we were on task and if we ran into a problem, that it was addressed in accordance with their desires. Although there was no direct user interaction with our design, users were kept informed and aligned with the design decisions throughout the process.

5.8 Security Testing

Security testing is not applicable to our design.

5.9 Results

During the entirety of this project, we focused on gaining familiarity with the open-source tools, simulating and building components, and laying out the final designs in Xschem and Magic. This process entailed taking a component through all of the steps: building in Xschem, simulating in Xschem, creating the layout in Magic, performing post-layout simulations, running DRC and LVS checks, and completing pre-check and tapeout checks. However, we only managed to get one component through Efabless pre-check and tapeout before Efabless lost its funding. After the criteria were updated for the project, we continued to follow the steps outlined above, but without performing the pre-check or tapeout check, as those processes rely on Efabless servers. Presented below are some of the simulation results, schematics, and layouts of components that were built from prior team's, and our team.

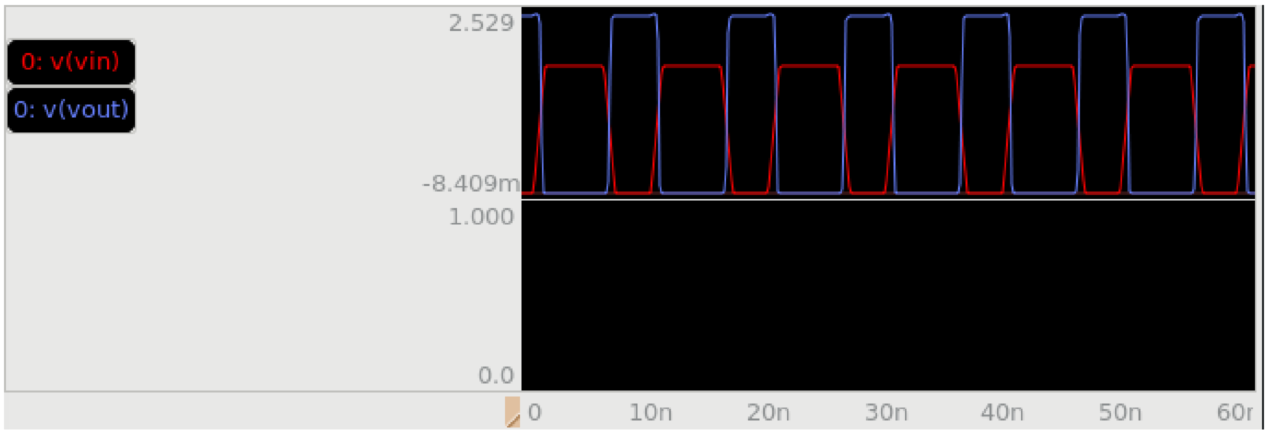


Figure 5.1: Simulation results from inverter, pre-layout

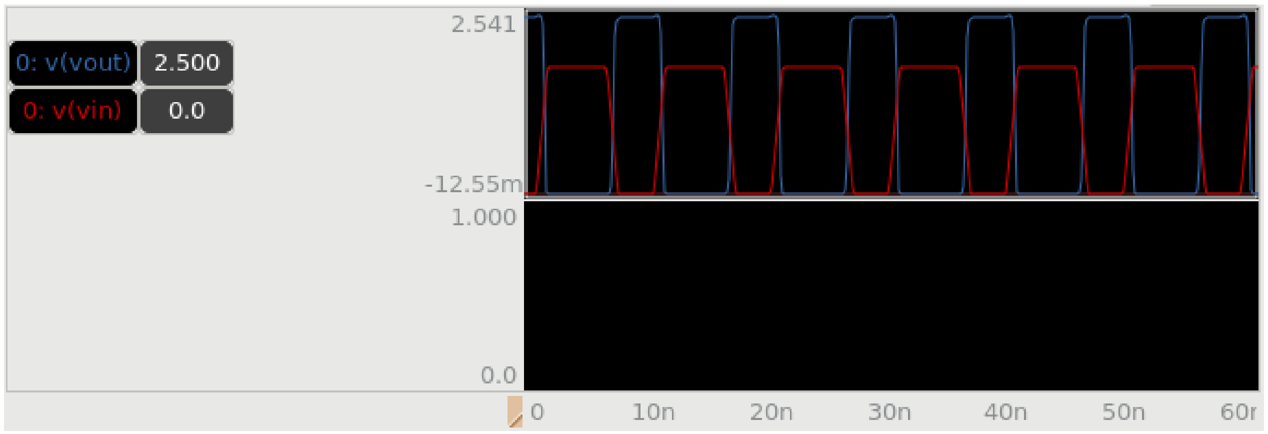
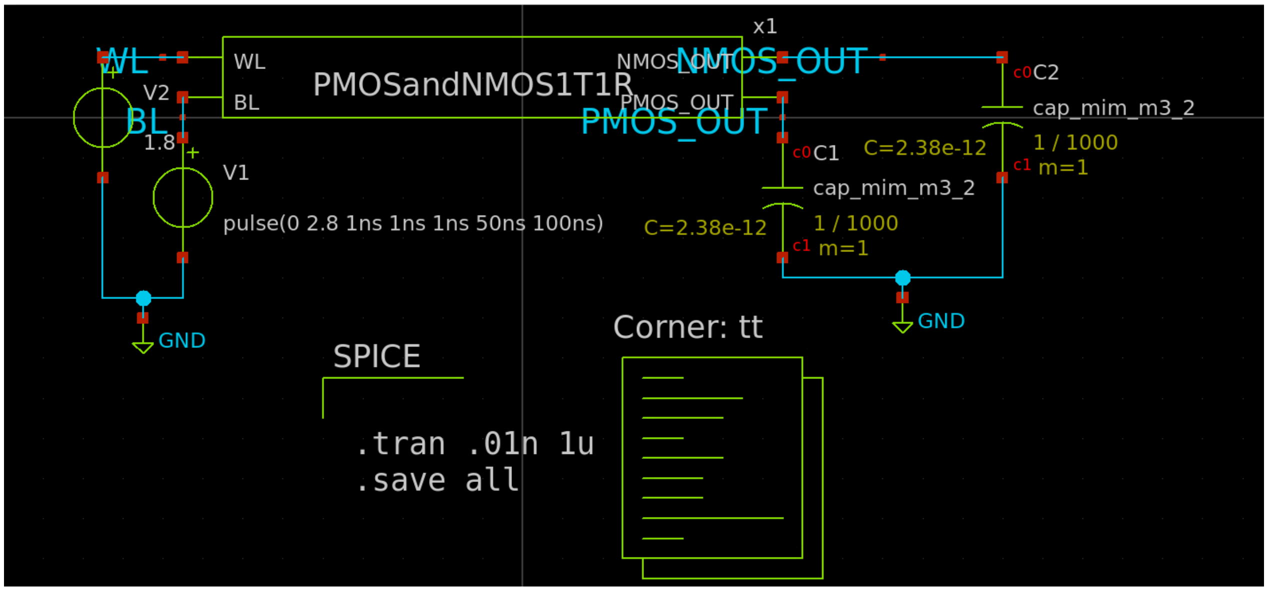


Figure 5.2: Simulation results from inverter, post-layout

When exploring potential design decisions for our unique architectures, we considered the idea of using PMOS transistors instead of NMOS transistors. The previous two design teams both used NMOS transistors. In order to explore this idea further, we designed a testbench which compared two ReRAM cells, one using a PMOS transistor and the other using an NMOS transistor. The poor results of this test, along with our own research and assistance from faculty advisors, caused us to shift our focus to other potential design decisions. Despite the suboptimal results, this was still a design decision worth discussing.



```
.tran .01n 1u
.save all
```

Figure 5.3: Testbench for comparing 1T1R cells using PMOS and NMOS transistors

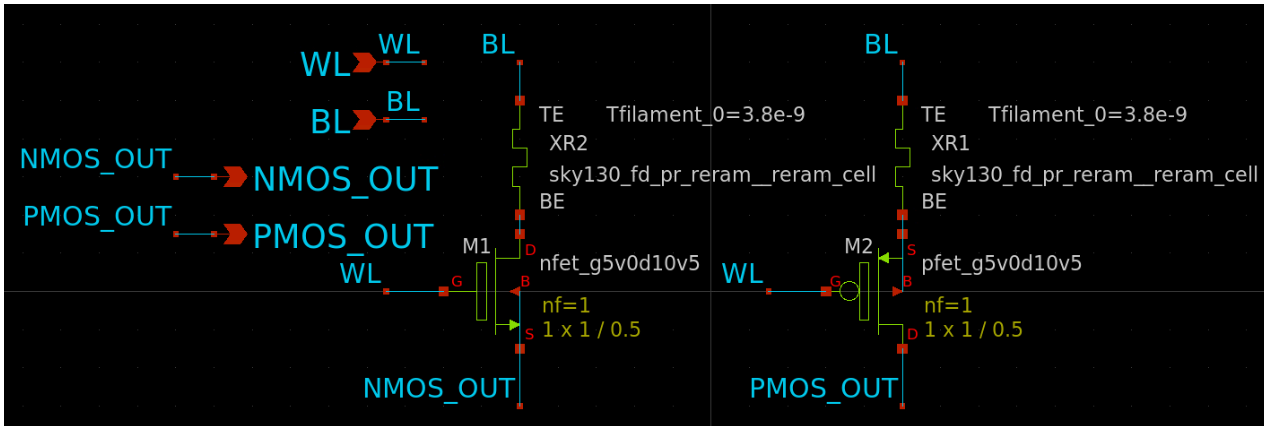


Figure 5.4: Schematic including two 1T1R cells, one using a PMOS transistor and the other using an NMOS transistor

We have many testing results from the sddec23-08 design team's testbenches. Testing these components helped us gain more familiarity with Ngspice and Gaw waveform viewer. It also gave us an opportunity to see what components from the previous team we could utilize in our design, what components we could modify to fit our design, and what components we may need to revamp for our use.

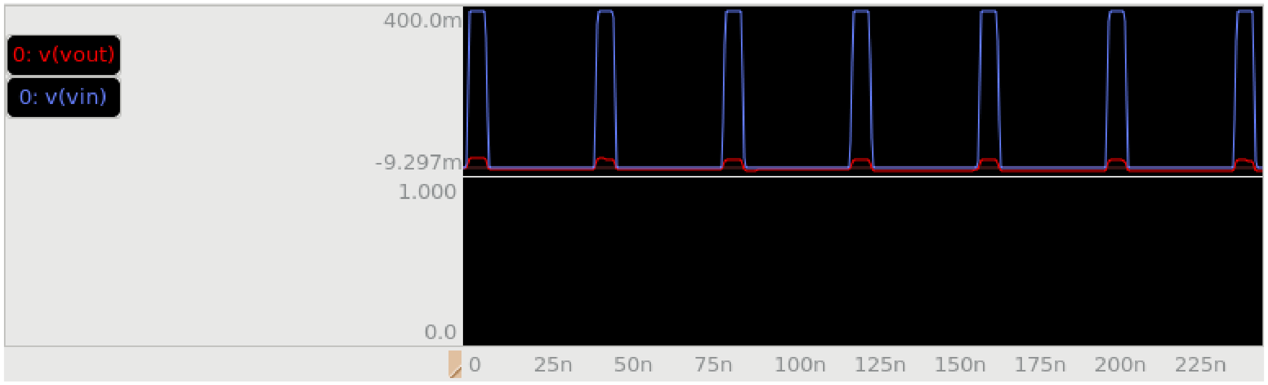


Figure 5.5: Resulting waveforms from buffer testbench from team sddec23-08

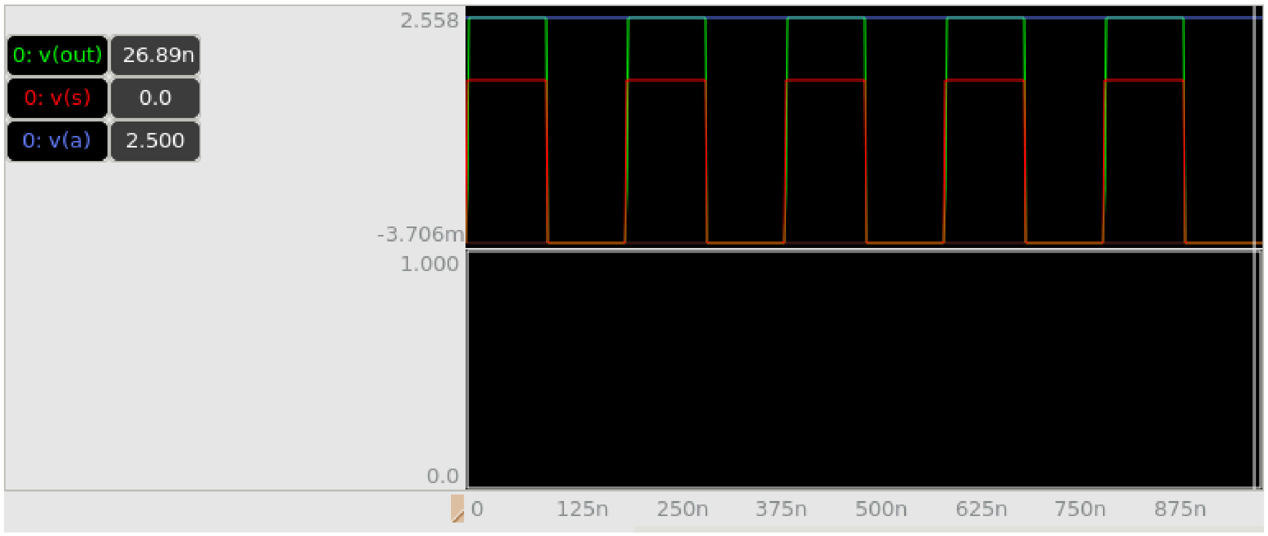


Figure 5.6: Resulting waveforms from 2-to-1 multiplexer testbench from team sddec23-08

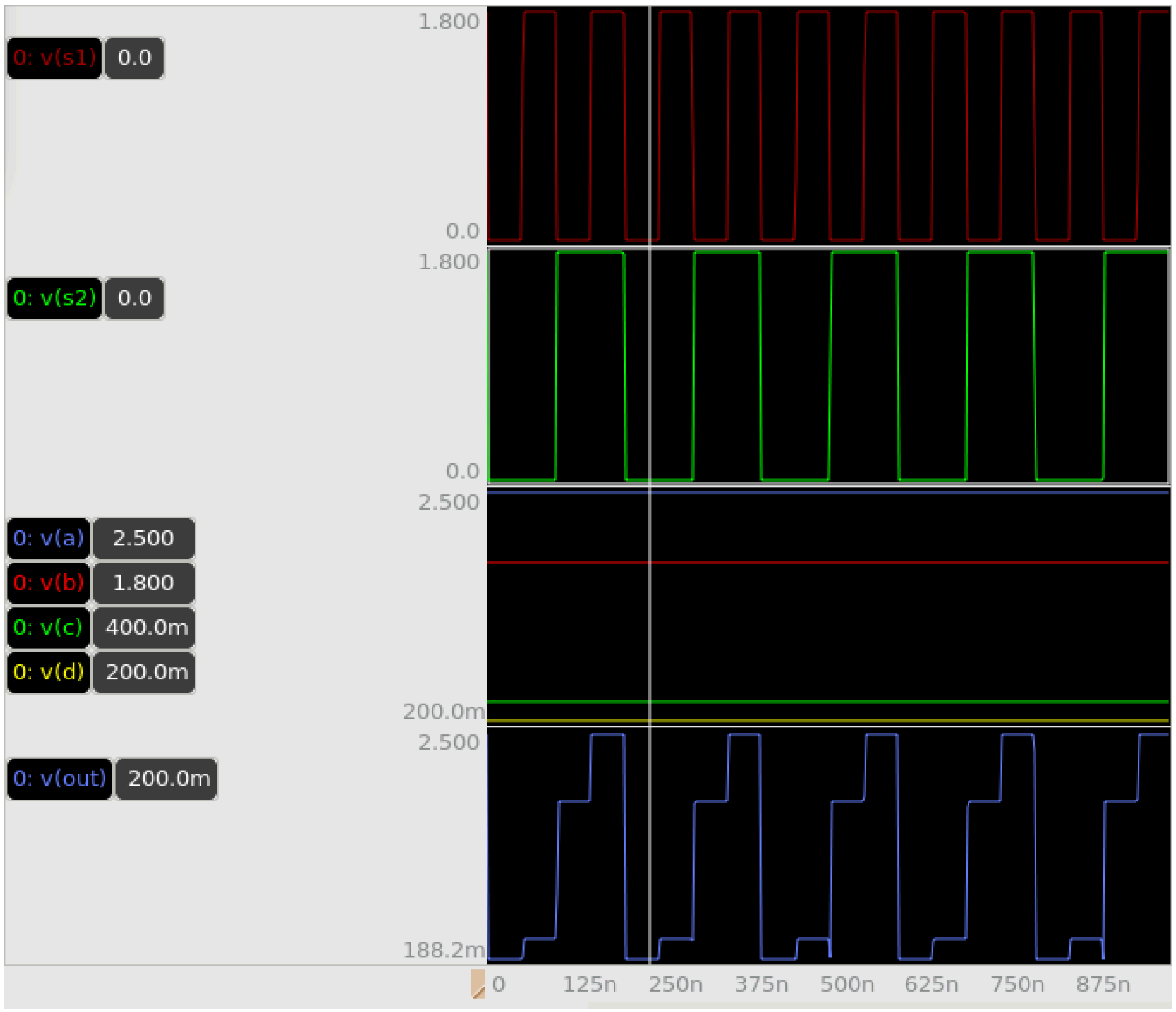


Figure 5.7: Resulting waveforms from 4-to-1 multiplexer testbench from team sddec23-08

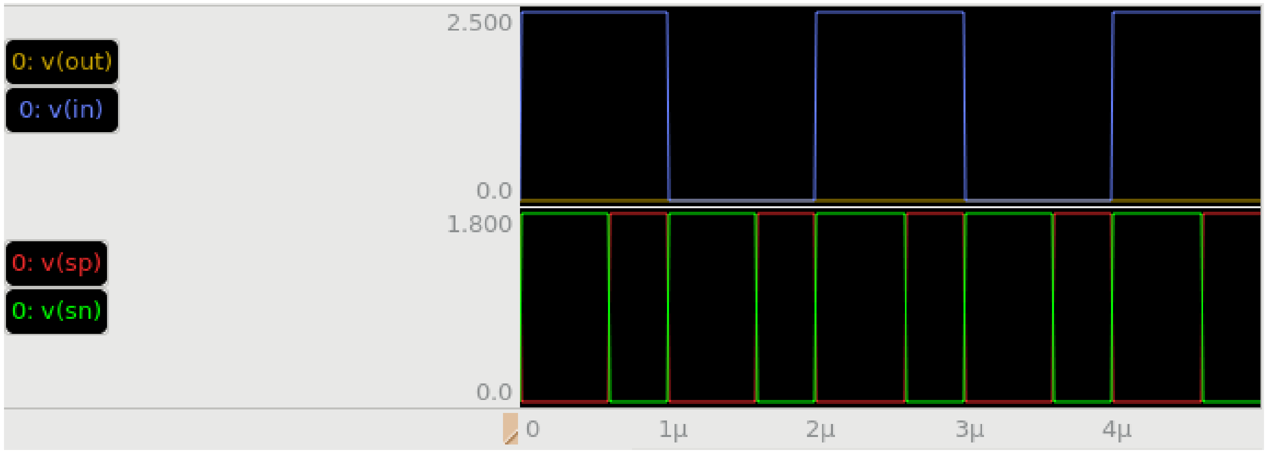
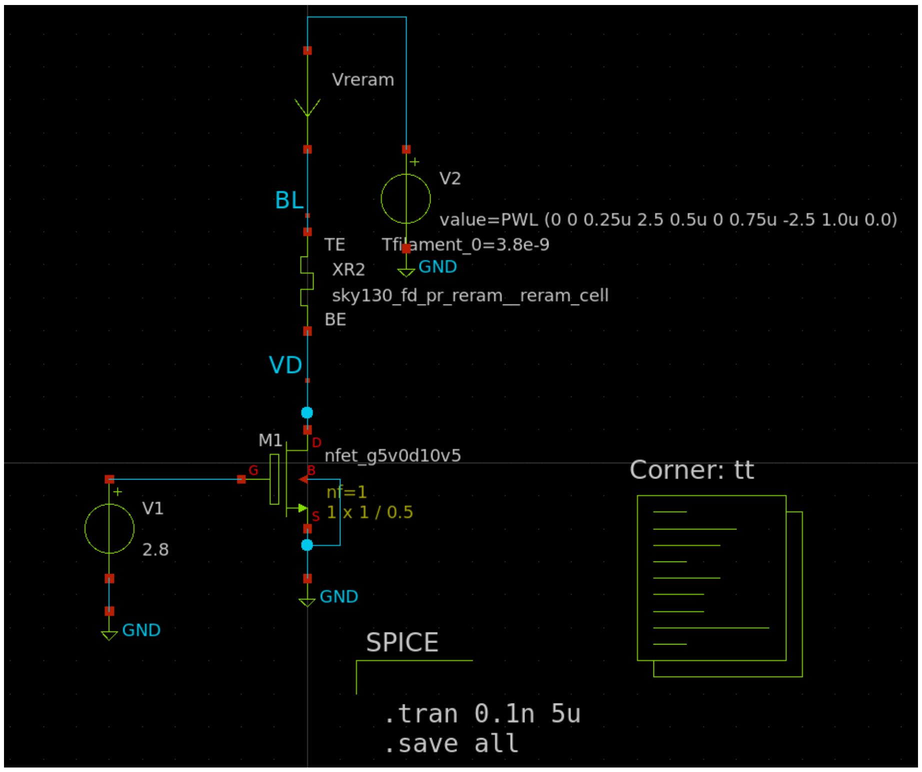


Figure 5.8: Resulting waveforms from transmission gate testbench from team sddec23-08

The schematic below shows our work to test and characterize a pre-layout 1T1R cell in xschem. Since moving away from the idea of using a cell with a PMOS device, the testbench below only includes a NMOS device for testing purposes. In order to read the value of the ReRAM, the MOS device is kept in the triode region. Seen on Vsource component (V2) the input signal is a ramp function that increases from 0V to 2.5V and then decreases from 2.5V to -2.5V before finally settling back at zero. This input signal should demonstrate the SET and RESET states of the device. We measure the state of the device by observing drain-source voltage and drain current of the transistor while comparing them to the input waveform.



```
.tran 0.1n 5u
.save all
```

Figure 5.9: 1T1R testbench

The waveforms below represent the successful SET and RESET operations of the ReRAM cell in the testbench above. The ReRAM cell enters the SET state as i(vreram) begins to peak. When ReRAM enters the SET state, it transitions to a high conductance material, which we can see in increase in current conducted through the transistor. In order to transition the ReRAM cell to the RESET state, a negative potential needs to be applied across the cell. As the negative potential is applied, the ReRAM cell will transition back to the low conductance state. We can see this transition to the RESET state in the waveform below around 600ns where the current drops to near zero when a sufficient negative potential is applied across the cell.

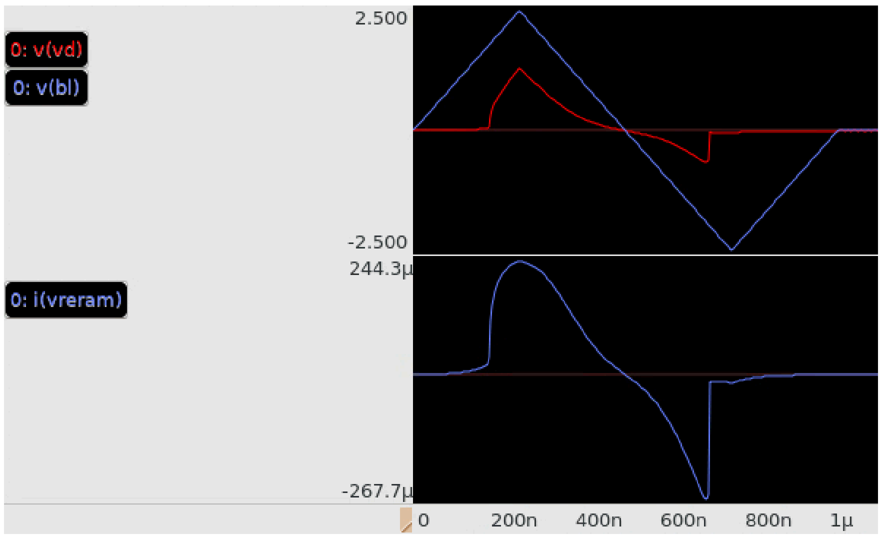
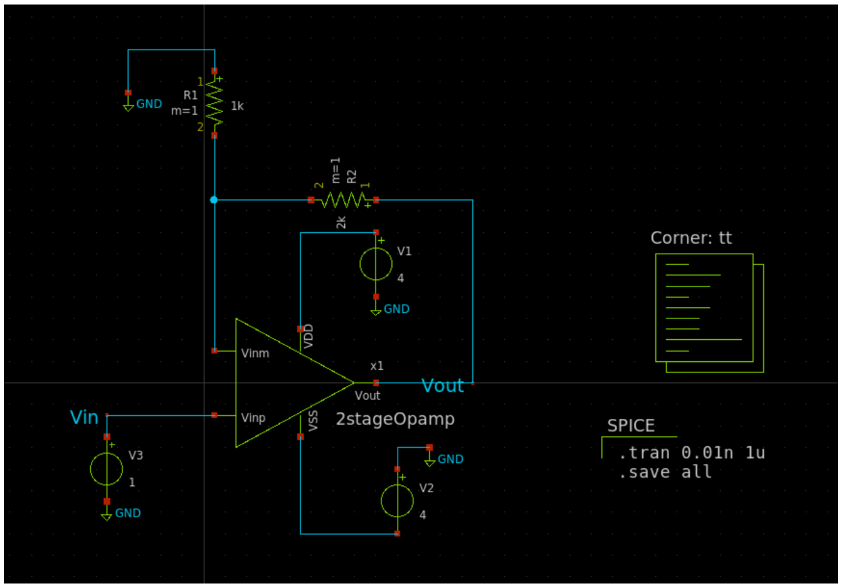


Figure 5.10: Waveforms from our 1T1R testbench



```
.tran 0.01n 1u
.save all
```

Figure 5.11: Transimpedance schematics testbench

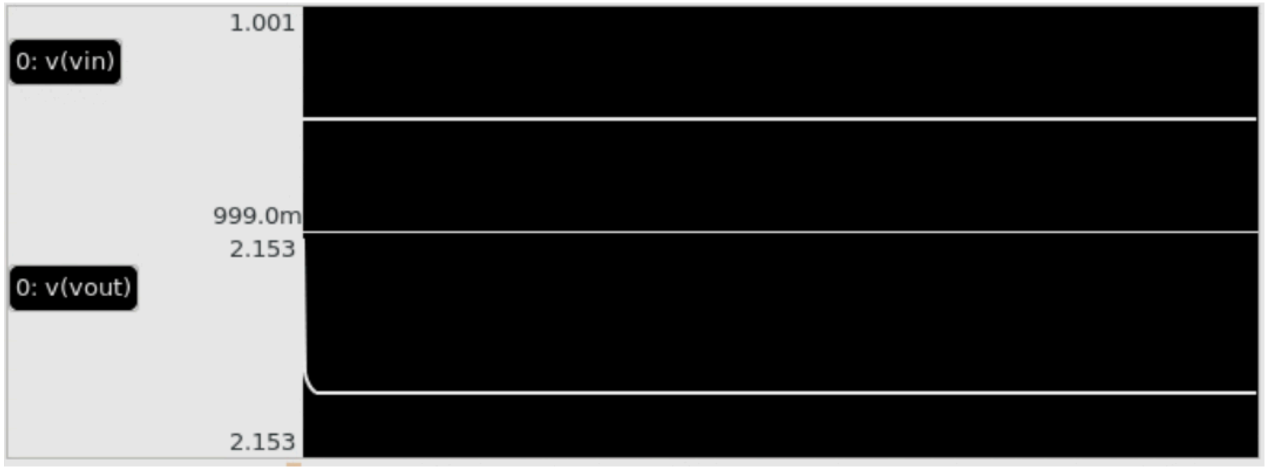
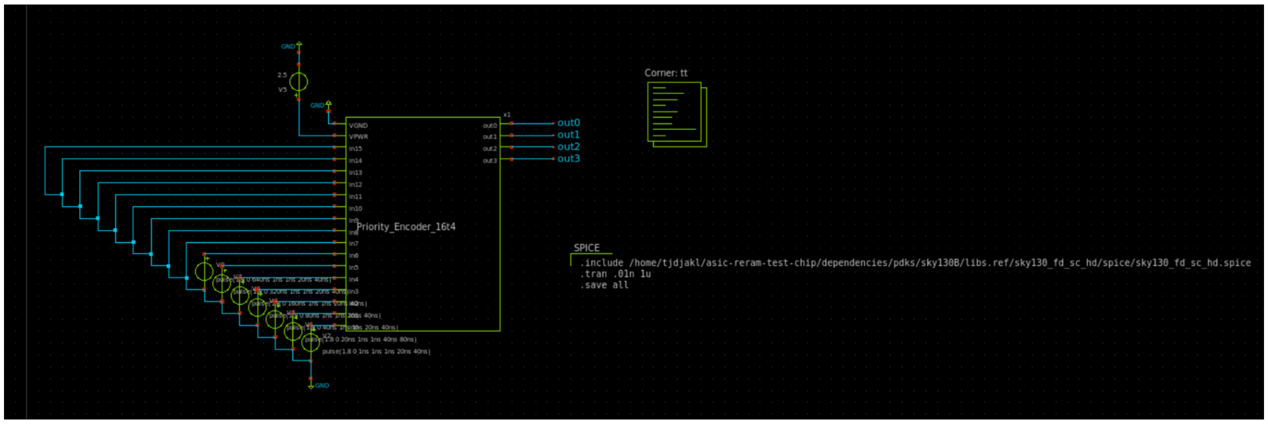


Figure 5.12: Resulting waveform for a transimpedance amplifier



SPICE
```
.include /home/tjdjakl/asic-reram-test-chip/dependencies/pdks/sky130B/libs.ref/sky130_fd_sc_hd/spice/sky130_fd_sc_hd.spice
.tran .01n 1u
.save all
```

Figure 5.13: Priority Encoder testbench

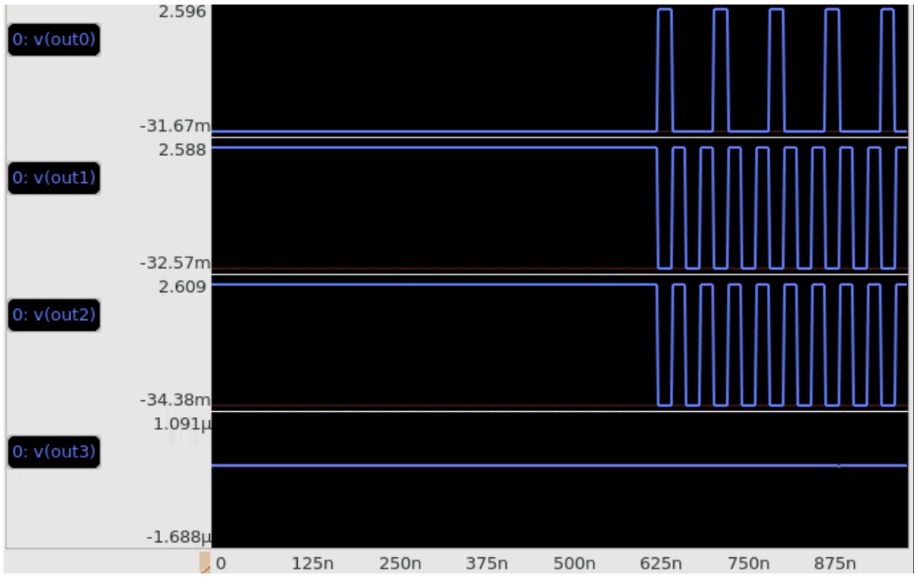


Figure 5.14: Resulting waveform from our priority encoder

6. Implementation

Over the course of the year, our implementation process focused on testing and verifying inherited components from previous teams in parallel with integrating newly designed elements. Key achievements include developing unit testbenches for individual components (e.g., op-amps, muxes, and transmission gates), verifying their behavior using simulation tools such as Ngspice and then laying out components and performing post layout simulation. We have created a top level design where we are able to write and read values to each of the 4 architectures in use in our design. Our design also allows for individual characterization of components through the use of our peripheral circuitry testbench. We also completed the layout of one of our unique architectures, with the designs passing DRC and LVS checks.

However, some of our original goals were not met. Firstly, we were not able to meet the tapeout date with our final design. We were not able to meet this goal for several reasons, first and foremost was the shutdown of efabless in early march. With this shutdown we lost access to the hosted checks that allowed us to run the precheck and tapeout-check jobs that informed us on whether our design was able to be fabricated. We were able to make partial progress on this goal, as we were able to get one of our components through tapeout-check before, as part of learning how their hosted check process worked, before the servers were shut down. Secondly, with the efabless shutdown we lost the support we had paid for to help with ReRAM related issues. One reason we were not able to complete the precheck of our final design was a persistent issue with the ReRAM SPICE model. The model can be simulated and passes DRC & LVS, however fails precheck. Without the support of efabless, the issue could not be resolved. One additional struggle we faced was differences in tooling between our current team and past teams. These issues required us to remake many of their testbenches, as the tools they used for simulation did not work with our version of the toolflow.

Despite these challenges, we continued on with the project to try and meet our original goal as best as possible. In the end we were able to deliver a top level design with all 4 architectures and a component testbench, a layout of one of our unique architectures, got a mux through tapeout check, and also demonstrated simulation results of our crossbar.

6.1 Design Analysis

In the fall we conducted comparative analysis on PMOS- and NMOS-based ReRAM cells using a custom testbench. The results of this analysis informed the selection of transistor types for new architectures shall remain NMOS, as mentioned in section 5.8.

Visual representations, including testbench schematics and simulation results, have been instrumental in guiding the implementation process. Key visuals include schematics comparing PMOS and NMOS testing, pre-layout and post-layout simulation results for constructed circuits, and waveform outputs from inherited designs like buffers and multiplexers. These efforts build upon previous work while ensuring robust and reliable implementation of new designs.

We have implemented two new ReRAM architectures. One will parallelize the source and bit lines. The design for this architecture can be seen in Figure 6.1 [6]. In this design, the source and bit lines are arranged in parallel, which simplifies the routing of signals. This setup can improve signal uniformity and reduce power loss caused by electrical resistance. However, placing the lines so close together increases the risk of coupling noise, where signals from one line interfere with another. The goal of this design is to explore how noise can be managed in such a configuration while seeing if the benefits of simpler routing outweigh the drawbacks.

The other architecture will be a true crossbar design with no transistors, essentially being a matrix of memristors. The design for this architecture can be seen in Figure 6b [6]. This design removes transistors entirely, leaving only a matrix of interconnected ReRAM cells. Without transistors, the cells lack isolation during read and write operations, meaning all cells are connected at once. This can lead to issues like leakage currents, which waste power, and signal interference, where neighboring cells affect accuracy. On the other hand, removing transistors has key advantages. It makes each cell much smaller, maximizing the use of space. This also reduces manufacturing complexity and cost. Additionally, the fully connected structure is excellent for handling many calculations at the same time, which is perfect for compute-in-memory tasks. By testing this design, we hope to find ways to overcome its challenges while taking advantage of its simplicity and efficiency.

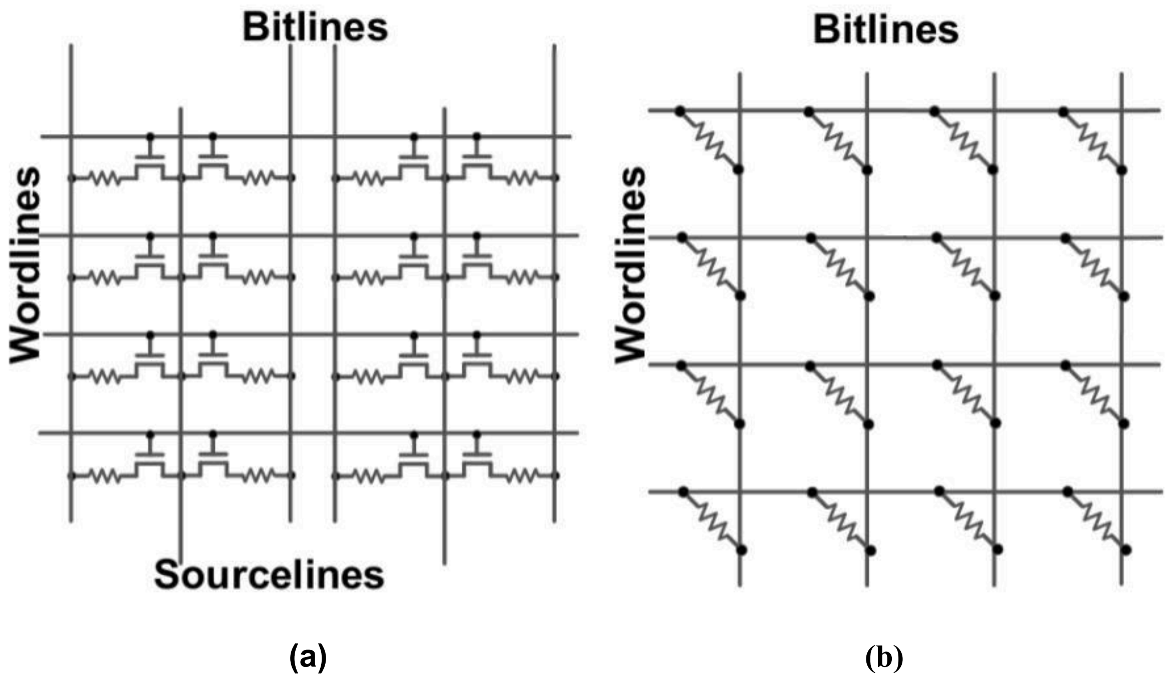


Figure 6.1: (a) Parallel Bitline and Sourcelines. (b) True Crossbar Design (both extracted directly from [6])

In the second semester, we focused heavily on verifying and updating components inherited from previous teams. Each standard cell was fully tested using a combination of simulation and physical verification tools, ensuring compatibility with the updated toolchain and compliance with Efabless fabrication standards. Several components required updates to function correctly, and while we were successful in verifying most of them, the 4-bit ADC from the previous team could not be simulated reliably. As a result, we incorporated the ADC developed by team sddec23-08 into our own designs. In parallel, we developed our two new ReRAM architecture schematics in Xschem and began integrating them into our system. During this period, we also gained experience with the Efabless hosted precheck and tapeout-check processes. These efforts culminated in getting our inverter, transmission gate, and 2-1 multiplexer through hosted tapeout verification—providing a solid foundation for the broader system-level integration.

We also completed the schematic creation of our two custom architectures in Xschem. These were integrated into two corresponding top-level designs, which include the new architectures, two inherited architectures from previous teams, and a comprehensive peripheral testbench. These individual top-level designs were then unified into a full-chip top-level design that routes outputs from all four architectures and peripheral logic into a final output multiplexer. This mux enables user-selectable output routing to the microcontroller unit (MCU) interface, providing flexibility for testing and deployment.

In preparation for a future tapeout, we finalized the layout for one of our new architectures. This layout was verified using DRC and LVS tools. Although the full design could not be submitted due to the Efabless server shutdown in March, we successfully pushed multiple components (e.g., muxes, transmission gate, inverter) through a hosted tapeout check before access was lost. This confirmed that our toolflow and design methodology were compliant with Efabless standards and ready to be fabricated.

Additionally, we completed bring-up firmware for the MCU. This code is designed to initialize the chip and manage data routing once the chip is fabricated and deployed. The bring-up code supports architecture selection and data communication between the MCU and peripheral logic. It also provides intuitive pin definitions and flexible library functions for writing and reading logic analyzer pins. An example main program is also provided for users to see how the code might look in action.

However, several original goals were not met. The team was unable to meet the tapeout deadline due to the Efabless shutdown. This shutdown blocked access to the hosted precheck and tapeout-check tools necessary for final submission. Although one component was successfully pushed through precheck before the shutdown, our full design encountered a persistent issue with the ReRAM SPICE model. While the model simulates and passes DRC/LVS, it fails a specific step of precheck that remains unresolved due to the discontinued support services from Efabless. Tooling inconsistencies between past and present teams also posed difficulties; simulation environments used in prior work were incompatible with our updated toolchain, requiring the team to rebuild many of the inherited testbenches.

Despite these challenges, we delivered a complete top-level design featuring all four ReRAM architectures, a functioning component testbench, and layout verification of one new architecture.

7. Ethics and Professional Responsibilities

Engineering ethics means following guidelines that are in the best interests of the client, the public, and the engineering profession. Adhering to these guidelines, this team seeks to ensure that the ReRAM-based architectures reduce energy consumption and improve computational efficiency for extensive tasks like machine learning. During the process, we are being mindful of being honest and transparent, and being accountable and responsible. Accessing the capabilities, risks, and limitations of ReRAM and providing documentation and testing results will provide the client with accurate data to make an honest decision about their product. Regarding the complexity and uniqueness of ReRAM, we will be proactive in addressing errors that arise during the project, and if we uncover any flaws in the ReRAM architecture, we will work quickly to correct them.

Professional responsibilities mean being held up to specific standards and duties that engineers have to their profession, colleagues, and the public. Thus, our project must amplify the integrity of research, collaboration of knowledge, and advocacy for fair and equitable access. We must ensure that the test chip is capable of producing valid results, which includes rigorous testing, validation, and documentation of our findings, which will help with the future development of the chip. As we explore and test ReRAM architectures, we must consider the long-term implications of this technology and aim to ensure that the benefits of our ReRAM findings will be accessible for further innovations and diverse industries.

7.1 Areas of Professional Responsibility/Codes of Ethics

The chosen code of ethics for the table below is the IEEE.

Area of Responsibility	**Definition**	**Relevant Items from the Code of Ethics**	**Team Interaction with the Code**
Work Competence	The ability of an engineer to perform their job when faced with challenges ensures that they are qualified by enhancing or maintaining their technical skills.	*“To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.”*	Our team has actively sought knowledge through research papers, external consultations, and expert feedback from the open-source company. To stay up to date, we review current literature and continue to ask experts.
Financial Responsibility	Refers to the ethical obligation of engineers to manage financial resources wisely, transparently, and with integrity	*“To avoid unlawful conduct in professional activities, and to reject bribery in all its forms.”*	Most of our tools are open-sourced. However, we have a tape-out date in April to produce one hundred chips for ten thousand dollars. We need to use our time responsibly in order to have a working and cost effective design.
Communication Honesty	Engineers should provide truthful and accurate information to avoid any misleading claims.	*“To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors.”*	We maintain transparency when we encounter both successes and challenges we face. Any design changes are also openly discussed.
Health, Safety, Well-Being	Engineers will ensure the health, safety, and well-being of the public when creating designs and	*“To hold paramount the safety, health, and welfare of the public, to protect the privacy of others, and*	Our team runs regular risk assessments that may pose any safety hazards, especially with concerns

	performing their work.	*to disclose promptly factors that might endanger the public or the environment."*	about overheating and ReRAM material production.
Property Ownership	The ethical responsibility of engineers is to respect the intellectual property rights of others, including patents, copyrights, trademarks, and trade secrets.	*"To be honest and realistic in stating claims or estimates based on available data, and to credit the contributions of others properly."*	We foster a collaborative environment where any ideas are welcome and given credit.
Sustainability	Engineers should strive for environmentally friendly and stable designs that can be used in the long term.	*"To strive to comply with ethical design and sustainable development practices."*	By using ReRAM cells, we are decreasing the power consumption from the memory to the arithmetic logic unit back to the memory.
Social Responsibility	Engineers should create products and services that contribute to the well-being of the community and society, not to their own greed.	*"To improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems."*	With the emerging ReRAM technology, we are also creating tutorials, and documentation that will help future innovators and the public to use for their own ideas and sound minds.

Table 7.1: Codes of ethics

To further contribute to the table above, the professional responsibility that we perform well in is work competence. Knowing little to no knowledge about the emerging ReRAM-based technology, we had to find research papers that were based on the infamous ReRAM cells and how to compute in memory matrix-vector multiplication. We have also consulted experts in various fields for noise minimization, very large circuit integration (VLSI), and tool usage.

One area of professional responsibility that requires improvement is financial responsibility. As previously mentioned in the table, we have a tape-out deadline in April, meaning that all our designs, schematics, layouts, and wrappers must be completed by then. The client is relying on our design to produce one hundred chips, each costing one hundred dollars, so our project must undergo rigorous testing through open-source software to ensure the design functions as intended. We have faced challenges in creating a design that is distinct from previous teams' work. However, given the tight deadline and the significant financial investment involved, it is crucial that we choose a solid design that can be completed successfully within the available time frame. To achieve this, we need to better allocate time in our schedules for teamwork, bringing both efficiency and quality to our efforts to ensure financial responsibility for the project's outcome.

7.2 Four Principles

Below is a table that breaks down the four principles versus the broader context considerations.

	Beneficence	**Nonmaleficence**	**Respect for Autonomy**	**Justice**
Public Health, Safety, and Welfare	Ensuring the ReRAM-based architectures do not cause harm to the public through rigorous testing to ensure reliability.	Make sure the chip is designed to be electrically safe and will not overheat due to energy inefficiencies.	Empowering users to make an informed decision about whether they choose to use the product or not through documentation and tutorials.	Allowing all individuals to access this project and research through the use of the internet and documentation we have provided.
Global, Cultural, and Social	We have designed the tutorials and documentation to ensure that all classes through college can use and learn from our experiences.	The designs and practices through making these chips will not harm the users.	The design does not affect anything related to religion or culture.	Having the documentation presented online ensures that all social, cultural, and global communities will have fair access to this technology
Environmental	Developed to decrease energy consumption and reduce the environmental footprint and waste.	The production of one hundred chips will inevitably have an environmental impact due to the manufacturing process. However, in the long term, ReRAM technology is expected to contribute to smaller chip sizes and a reduction in energy consumption.	Our design, compared to other chips that are manufactured, will create an eco-friendly design due to reduced power consumption.	If the design works it will reduce the environmental footprint due to inefficiencies in chips globally.
Economic	The design will help create job opportunities due to the complexity and manufacturing process of the ReRAM cells.	The design will not disrupt the economy, only help to improve AI implementation.	We provide four potential designs for ReRAM-based architectures that will be allocated to any budget needs.	The implementation of ReRAM in chips causes the creation of new markets, the generation of jobs, and the continuation of more sustainable practices throughout the economy.

Table 7.2: Four principles

One broader context-principle pair that is important to us is environmental beneficence. The primary purpose of this project is to reduce power from data going into the arithmetic logic unit and then back to the memory unit. This causes increased time, which correlates with an increase in power consumption. To ensure that this happens, we will utilize ReRAM's low-power characteristics and the CIM architecture to reduce the energy required for memory-intensive operations like MVM.

A context-principle pair that our final design will be lacking in the end but will not be prolonged is environmental nonmaleficence. When creating something out of theory, you need to test and recreate, and repeat the process until it is successful. At the end of April, this project will be sent to Efabless, and one hundred chips will be manufactured to be a test chip. This will cause a loss of materials and create waste from the manufacturing process. However, if this project shows promising results and grows in the future, it will reduce power consumption. Thus it will end up helping the environment and creating innovation in the other fields as well.

7.3 Virtues

Below are the three virtues that are important to this team.

1. **Clear and thorough documentation:** With the creation of something new, we have decided that one of our most important virtues was clear and thorough documentation. Clear and thorough documentation means that all design decisions, process findings, and project details are well-recorded. To portray that we are following this virtue, we have created tutorials and troubleshooting guides for the open software tools to help people follow in our footsteps. We are also documenting what errors we have run into with the architectures and what solutions we have come up with to overcome these challenges.

2. **Honesty:** Another one of our core virtues is honesty. This virtue correlates with our first virtue, stating that with clear and thorough documentation, we also need to be honest with our clients and the public about our errors and challenges. Honesty means being open and transparent about the issues and challenges we face when completing this project. To demonstrate our commitment to this task, we encourage open communication and try to maintain an environment where team members are comfortable discussing challenges or setbacks without fear of judgment. Any issues regarding timelines and setbacks are openly discussed during our weekly meetings and are addressed with an honest assessment and discussion.

3. **Cooperativeness:** Our last team member's core value is cooperativeness. This project has each member dedicating a certain amount of time that can be more extensive than other projects proposed. It also has a diverse set of technical skills needed to successfully implement the criteria into this project. Cooperativeness means working in a group collaboratively and supporting each other towards a common goal. We actively promote a team-first mentality, where successes and work ethic are recognized within the context of team collaboration. Regular work sessions are held to ensure that all team member's ideas are heard and have a chance to contribute to the designs.

Our individual core virtues are presented below.

- Sam Burns
    - Resourcefulness is one virtue I have demonstrated in my time spent on our senior design project so far. Resourcefulness is a crucial virtue when working through a project because it provides an opportunity to get access to materials and learn things that I wouldn't have known about; this opens the door to making more progress and developing a deeper understanding of the project. A couple of ways I have demonstrated this virtue are by meeting with various faculty with expertise outside that of our own project advisor and client, meeting with the 491 TAs, and reaching out to the Efabless community via their slack channel. Each of these resources helped me learn more about the skills required to succeed on our senior design project.

    - Patience is one of the most important virtues a team can demonstrate when working on a project. There have been times that I have felt frustrated with simulating results or unclear software errors. However, the key for a project to progress is remaining patient and dedicated to working through issues that arise. Every project will face struggles, remaining patient will only help the progress of a team. One way I can work to demonstrate this is taking more unique approaches to persistent issues I face in my circuit design on this project.

- Travis Jakl
    - Perseverance is a vital quality, especially in engineering and research projects, where challenges and setbacks are common. It helps me stay focused on long-term goals and motivates me to push through difficulties. By persevering, I can not only resolve issues but also learn valuable knowledge that improves my troubleshooting skills for this project and future ones. Perseverance helped me to push through the toolset errors I was receiving at the start of this project, utilizing documentation, former and current team members, and Slack to solve these issues. This not only helped myself, but helped my team as well.

    - Time management is essential for balancing multiple responsibilities and ensuring consistent progress on a project. I struggled with managing my time effectively as other coursework and commitments piled up, which led to challenges in prioritizing tasks for the project. To improve, I plan to create a more structured approach by breaking the project into smaller, manageable tasks with clear deadlines. I'll use calendars and task lists to track progress and set aside specific time slots for focused work on the project. By regularly reassessing my workload, I'll ensure I stay on track and reach milestones without compromising the quality of my work.

- Noah Mack

  - Throughout our senior design project so far, I feel that I have demonstrated the virtue of perseverance. As a computer engineering student, I haven't had much experience with schematic creation and I have had no exposure to layout creation before this project. As we worked to get familiar with the open source tools, I kept trying even when things weren't working. However frustrating it was, I just needed to keep trying until I got it figured out. Perseverance is important to me because it is required for learning new things.

  - One important virtue that we haven't had the chance to demonstrate to its fullest extent is writing clear and thorough documentation. We did contribute to the ChipForge analog documentation as we worked through their tutorial, but for our project we will need detailed documentation so that our users have all of the tools they need to successfully utilize our test chip. This is an important virtue to me because I always appreciate when I am using a new technology and it has high quality documentation to streamline the learning process.

- Olivia Price
  - One of the virtues that is important to me is honesty. It is vital to communicate your successes and failures to your team so that they can help you or innovate on your work. I have demonstrated honesty by attending every weekly meeting and communicating the challenges I face or the solutions I have found. I have also asked questions when something is unknown to me or have voiced my concerns when the objective does not align with my knowledge.

  - One virtue that is also important to me is my lack of commitment to quality. Commitment to quality is crucial because it shows the client that they can trust me to maintain their satisfaction. It also improves efficiency, the project will gain a competitive advantage, and it will ensure long-term success. To demonstrate this virtue in the future, I will prioritize dedicating more time to the project and create a schedule that allows me to allocate additional hours for focused work.

7.4 Ethics and Virtue Changes Throughout the Project

From the beginning of our project to the end our team adopted a virtue-based ethical approach, emphasizing integrity, accountability, respect for users, and a commitment to safety. These foundational values guided our decision-making and interactions throughout the design and development process. Despite the evolving scope of our project, our ethical stances have remained consistent. We continued to prioritize communication, especially in documenting design decisions and potential limitations. We also maintained our responsibility to consider environmental implications of our work, even if we are not fabricating. However, there were no major shifts in our ethical approach because the guiding principles that we have established above were broad enough to accommodate the various challenges we encountered. Although, as we progressed through the project our team's ethical awareness has deepened by our project experience.

8. Closing Material

8.1 Conclusions

Over the course of our senior design project, our team developed and constructed a research test vehicle that will compare four different ReRAM architectures. From initial concept development through sketches and final testing, we worked systematically to meet the objectives set at the beginning of the project. Our testing phase consisted of building the circuitry in Xschem, testing it through Ngspice, and post layout testing in Magic, which demonstrated that the design operates reliably and meets key performance criteria set above. One of our major accomplishments was designing two new architectures that could contribute to the research of computation in memory. In addition, we provided documentation and tutorials for future users which allows for them to take up the project after our team is no longer in contact. Our team was also able to figure out how tape-out and pre-check are accomplished through Efabless servers, which has taken a new name; ChipFoundry. However, with Efabless shutting down, some criteria of our project were not attainable during our duration in senior design. In the end, these accomplishments aligned closely with our overall project goals, which were creating a research test vehicle that had four unique ReRAM architectures. Also, if none of the architectures work, we also created a section in our final design where each circuit component can be individually characterized and tested through analog pins.

Our final design not only satisfies the original requirements but also shows potential for future enhancements, learning opportunities, and real-world applications. This project has demonstrated our team’s ability to combine technical knowledge with practical problem-solving and teamwork.

8.2 Value Provided

Our design was developed with a strong focus on addressing the specific needs of our target users, namely our client, ISU ECpE faculty and research teams, ChipForge, students, and the public. By creating a flexible ReRAM-based research test vehicle, we provided a platform for exploring in-memory computing architectures that help reduce energy consumption and data transfer delays associated with traditional processor-memory designs. This benefits our users by allowing them to explore different ReRAM device characteristics and compute-in-memory operations without requiring entirely new fabrication runs.

At the beginning of the project, our primary objective was to reduce the inefficiencies associated with the traditional separation of memory and processing units which causes energy consumption and latency caused by data movement. Our design successfully demonstrates the feasibility of embedding simple compute capabilities directly into the memory array by leveraging the unique switching behavior of ReRAM devices. Additionally, our project addresses the experimental challenge of having a flexible and testable platform for early-stage in-memory computing research, which lacks accessible prototyping solutions.

The focus of our project was to decrease power consumption and data processing time especially for artificial intelligence, neuromorphic computing, and edge computing applications. Our project fits into this context by providing a proof of concept hardware platform that aligns with global research efforts into in-compute memory architectures. An example of the value-provided by this project is our ReRAM research test vehicle allows for multiple architectures which allows for wide exploration without redesigning the entire chip

8.3 Next Steps

As we finished our project, there were some requirements that were not met due to timing and unforeseen challenges. One of the unforeseen challenges was Efabless shutting down which made it impossible to get our chip through pre-check and tapeout check which happens on the Efabless servers. Another unforeseen problem was some of the prior team's components did not work entirely, and we tried to fix them, however, we could not figure out the errors. Lastly, this chip was supposed to be shipped off to fabricate on April 21st, 2025, however with Efabless shutting down, this could not be accomplished. For the next steps, it would be a good idea to test the circuitry components against noise variations to test reliability and resilience of both the memory and logic components. It would also be a good idea to find a company that could potentially fabricate this chip, and do pre-check and tapeout on their servers for all the components. Once the chip is fabricated, there could be a design analysis team that can investigate the four ReRAM architectures and find which one works best and innovate upon its design. These steps will build on our foundation and push ReRAM-based in-memory computing closer to practical, high-impact deployment.

9. References

[1] P. Gray, P. Hurst, S. Lewis, and R. Meyer, "Noise in Integrated Circuits," in *Analysis and Design of Analog Integrated Circuits 4th ed.*, Hoboken, NJ, USA: Wiley, 2001, ch. 11, pp. 748-802.

[2] P. Chi *et al.*, "PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory," *Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA)*, Seoul, Korea, 2016, pp. 27–39, doi: 10.1109/ISCA.2016.13.

[3] A. Shafiee *et al.*, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," *Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA)*, Seoul, Korea, 2016, pp. 14–26, doi: 10.1109/ISCA.2016.12.

[4] IEEE, "IEEE Standards Association," [Online]. Available: https://standards.ieee.org/. [Accessed: 06-Dec-2024]

[5] SkyWater PDK Authors, "Sky130-FD PR ReRAM Documentation," [Online]. Available: https://sky130-fd-pr-reram.readthedocs.io/en/latest/index.html. [Accessed: 06-Dec-2024].

[6] C. Xu et al., "Overcoming the challenges of crossbar resistive memory architectures," 2015 IEEE 21st International Symposium on High Performance Computer Architecture

(HPCA), Burlingame, CA, USA, 2015, pp. 476-488, doi: 10.1109/HPCA.2015.7056056.

[7] J. Thater, “Open-Source Analog Design Flow Using Efabless and the SkyWater 130 nm PDK”, Iowa State University Senior Design Team sddec23-08, Nov. 27, 2023. [Online]. Available: https://sddec23-08.sd.ece.iastate.edu/reports/Senior%20Design%20Enviornment_ReRA%20Setup.pdf. [Accessed: Dec. 6, 2024].

[8] ChipForge, "ISU Chip Fab Documentation: Analog Design," [Online]. Available: https://git-pages.ece.iastate.edu/isu-chip-fab/documentation/#/analog/index. [Accessed: Dec. 6, 2024].

[9] B. Hofer and P. Hyung-Joo, "Sky130 Ngspice ReRAM," [Online]. Available: https://github.com/barakhoffer/sky130_ngspice_reram. [Accessed: Dec. 6, 2024].

[10] E. R. Hsieh et al., "High-Density Multiple Bits-per-Cell 1T4R RRAM Array with Gradual SET/RESET and its Effectiveness for Deep Learning," 2019 IEEE International Electron Devices Meeting (IEDM), San Francisco, CA, USA, 2019, pp. 35.6.1-35.6.4, doi: 10.1109/IEDM19573.2019.8993514.

[11] J. Thater, A. Petersen, M. Ottersen, and R. Dukele,”ReRAM Compute ASIC Fabrication”, Iowa State University Senior Design Team sddec23-08, Nov. 27, 2023. [Online]. Available: https://sddec23-08.sd.ece.iastate.edu/reports/Senior%20Design%20Enviornment_ReRA%20Setup.pdf. [Accessed: Dec. 6, 2024].

[12] K. Kivimagi, J. Xie, G. Moorman, and N. Cook, “ReRAM Compute Crossbar Fabrication”, Iowa State University Senior Design Team sddec24-13, May 2024. [Online]. Available: https://sddec24-13.sd.ece.iastate.edu/SDDEC24_13_DESIGN_DOCUMENT%20(2).pdf. [Accessed: Dec. 6, 2024].

10. Appendices

10.1 Operation Manual

Introduction

This document acts as a guide to using the ASIC ReRAM test chip designed by senior design team SDMay25-19 at Iowa State University. Information enclosed in this document includes C code documentation and pin definitions, as well as the pinout for the chip itself.

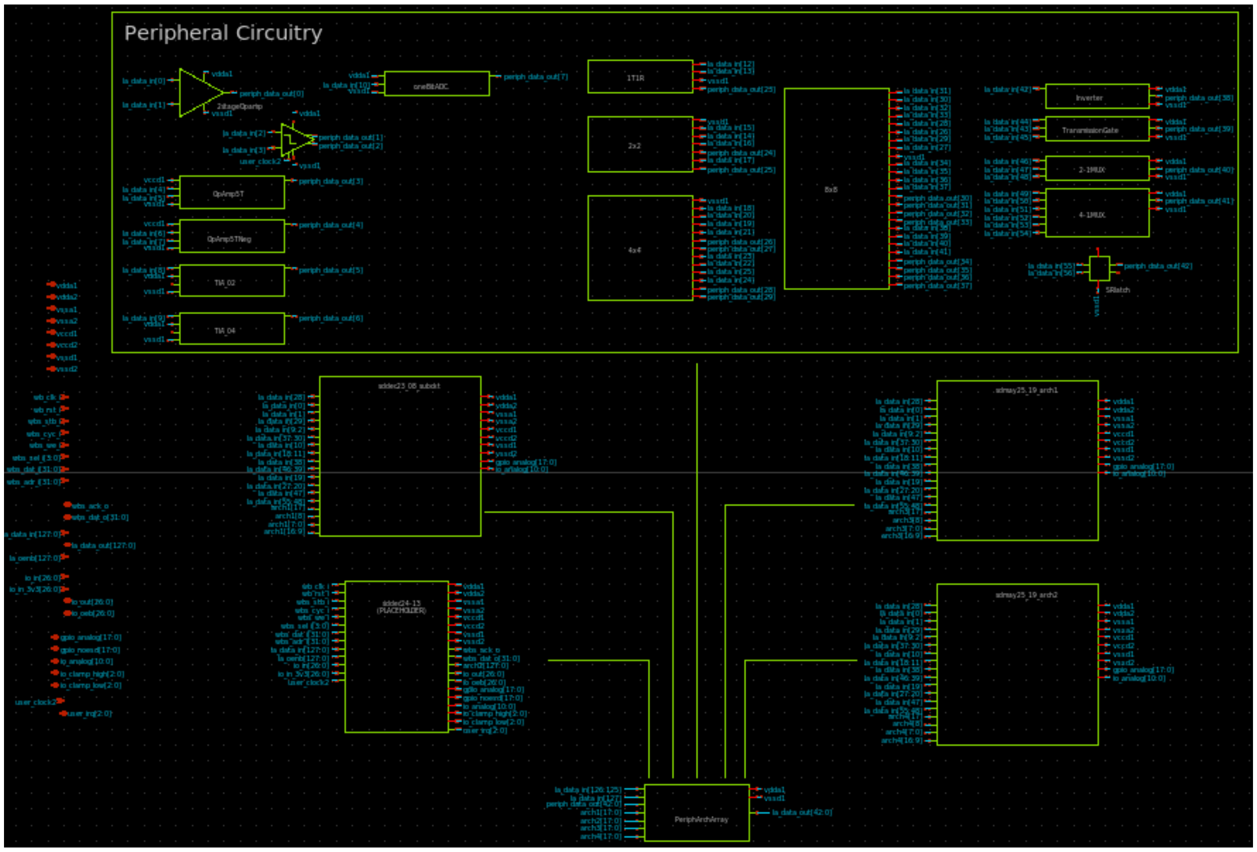


Figure 10.1: Top Level Schematic

The overall design of our chip consists of four ReRAM compute architectures, as well as a peripheral circuitry area with our component circuits individually mapped to pins. All components of this chip can be tested using the code and pin definitions provided in this document.

Code

Our repository contains two C files. The first is called "sdmay25_defs.h", which contains helpful functions for writing and reading logic analyzer pins, as well as pin definitions for ease of use. The second file is an example "main.c", which performs a few example functions on the chip to get the user started.

```
static inline void write_la_pin(uint32_t pin, uint32_t val)
```

*Use this function to write a HIGH or LOW **val**ue to a single **pin**.*

```
static inline void write_la_pins(uint32_t lsb_pin, uint32_t width,
uint32_t val)
```

*Use this function to write a **val**ue to the logic analyzer that spans more than a single bit. The pin definitions provide appropriate values for **lsb_pin** and **width** (see definitions table below).*

```
static inline uint32_t read_la_pin(uint32_t pin)
```

*Use this function to read a single logic analyzer **pin**.*

```
static inline uint32_t read_la_pins(uint32_t lsb_pin, uint32_t width)
```

*Use this function to read a value of a given **width** from the least significant pin (**lsb_pin**). The pin definitions provide appropriate values for **lsb_pin** and **width** (see definitions table below).*

See our repository for example main.c.

Definitions/Pinout

*Note: for signals wider than 1 bit, the full range of bits is noted next to the least significant bit pin. Width

sdmay_defs.h Definition	**Caravel Pin**	**Description/Notes**
PERIPH_SELECT	la_data_in[127]	1: peripheral output, 0: architecture output
PERIPH_DATA_OUT_LSB	la_data_out[42:0]	
PERIPH_DATA_OUT_WIDTH	N/A (value 43)	
TWO_STAGE_OP_AMP_IN_MINUS	la_data_in[0]	2 stage op amp Vin-
TWO_STAGE_OP_AMP_IN_PLUS	la_data_in[1]	2 stage op amp Vin+
TWO_STAGE_OP_AMP_OUT	la_data_out[0]	
COMPARATOR_IN_PLUS	la_data_in[2]	Comparator Vin+
COMPARATOR_IN_MINUS	la_data_in[3]	Comparator Vin-
COMPARATOR_OUT_PLUS	la_data_out[1]	Comparator Vout+
COMPARATOR_OUT_MINUS	la_data_out[2]	Comparator Vout-
OP_AMP_5T_IN_PLUS	la_data_in[4]	5T op amp Vin+
OP_AMP_5T_IN_MINUS	la_data_in[5]	5T op amp Vin-
OP_AMP_5T_NEG_IN_PLUS	la_data_in[6]	5T op amp neg Vin+
OP_AMP_5T_NEG_IN_MINUS	la_data_in[7]	5T op amp neg Vin-
OP_AMP_5T_OUT	la_data_out[3]	
OP_AMP_5T_NEG_OUT	la_data_out[4]	
TIA_02_IN	la_data_in[8]	
TIA_04_IN	la_data_in[9]	
TIA_02_OUT	la_data_out[5]	
TIA_04_OUT	la_data_out[6]	
ADC_IN	la_data_in[10]	

ADC_OUT	la_data_out[7]	
BL_IN_1T1R	la_data_in[12]	Bitline input
WL_IN_1T1R	la_data_in[13]	Worldline input
SL_OUT_1T1R	la_data_out[23]	Sourceline input
BL_IN_2X2_LSB	la_data_in[15:14]	
BL_IN_2X2_WIDTH	N/A (value 2)	
WL_IN_2X2_LSB	la_data_in[17:16]	
WL_IN_2X2_WIDTH	N/A (value 2)	
SL_OUT_2X2_LSB	la_data_out[25:24]	
SL_OUT_2X2_WIDTH	N/A (value 2)	
BL_IN_4X4_LSB	la_data_in[21:18]	
BL_IN_4X4_WIDTH	N/A (value 4)	
WL_IN_4X4_LSB	la_data_in[25:22]	
WL_IN_4X4_WIDTH	N/A (value 4)	
SL_OUT_4X4_LSB	la_data_out[29:26]	
SL_OUT_4X4_WIDTH	N/A (value 4)	
BL_IN_8X8_LSB	la_data_in[33:26]	
BL_IN_8X8_WIDTH	N/A (value 8)	
WL_IN_8X8_LSB	la_data_in[40:34]	
WL_IN_8X8_WIDTH	N/A (value 8)	
SL_OUT_8X8_LSB	la_data_out[37:30]	
SL_OUT_8x8_WIDTH	N/A (value 8)	
INVERTER_IN	la_data_in[42]	
INVERTER_OUT	la_data_out[38]	
TRANSMISSION_GATE_IN	la_data_in[43]	

TRANSMISSION_GATE_SP	la_data_in[44]	
TRANSMISSION_GATE_SN	la_data_in[45]	
TRANSMISSION_GATE_OUT	la_data_out[39]	
MUX_2_1_SEL	la_data_in[46]	0: output A, 1: output B
MUX_2_1_A	la_data_in[47]	
MUX_2_1_B	la_data_in[48]	
MUX_2_1_OUT	la_data_out[40]	
MUX_4_1_SEL1	la_data_in[49]	See next row
MUX_4_1_SEL2	la_data_in[50]	(With SEL1) 00: output A, 01: output B, 10: output C, 11: output D
MUX_4_1_A	la_data_in[51]	
MUX_4_1_B	la_data_in[52]	
MUX_4_1_C	la_data_in[53]	
MUX_4_1_D	la_data_in[54]	
MUX_4_1_OUT	la_data_out[41]	
SR_LATCH_S	la_data_in[55]	
SR_LATCH_R	la_data_in[56]	
SR_LATCH_Q	la_data_out[42]	
ARCH_SELECT_LSB	la_data_in[126:125]	00: sddec23-08. 01: sddec24-13, 10: sdmay25-19 arch1, 11: sdmay25-19 arch2
ARCH_SELECT_WIDTH	N/A (value 2)	
ARCH_DATA_OUT_LSB	la_data_out[17:0]	
ARCH_DATA_OUT_WIDTH	N/A (value 18)	
WRITE_SELECT_8LINEBITINPUT_1	la_data_in[0]	
WRITE_SELECT_8LINEBITINPUT_2	la_data_in[28]	

WRITE_FORM_SELECT_8LINEBITINPUT_1	la_data_in[1]	
WRITE_FORM_SELECT_8LINEBITINPUT_2	la_data_in[29]	
LA_IN_8LINEBITINPUT_1_LSB	la_data_in[9:2]	
LA_IN_8LINEBITINPUT_1_WIDTH	N/A (value 8)	
LA_IN_8LINEBITINPUT_2_LSB	la_data_in[37:30]	
LA_IN_8LINEBITINPUT_2_WIDTH	N/A (value 8)	
WRITE_SELECT_8LINEBITINPUT_1	la_data_in[10]	
WRITE_SELECT_8LINEBITINPUT_2	la_data_in[38]	
LA_IN_8LINEWORDINPUT_1_LSB	la_data_in[18:11]	
LA_IN_8LINEWORDINPUT_1_WIDTH	N/A (value 8)	
LA_IN_8LINEWORDINPUT_2_LSB	la_data_in[46:39]	
LA_IN_8LINEWORDINPUT_2_WIDTH	N/A (value 8)	
S_8LINESELECTINPUT_1	la_data_in[19]	
S_8LINESELECTINPUT_2	la_data_in[47]	
LA_IN_8LINESELECTINPUT_1_LSB	la_data_in[27:20]	
LA_IN_8LINESELECTINPUT_1_WIDTH	N/A (value 8)	
LA_IN_8LINESELECTINPUT_2_LSB	la_data_in[55:48]	
LA_IN_8LINESELECTINPUT_2_WIDTH	N/A (value 8)	
VSSNEG_8LINESELECTOUTPUT_02	la_data_in[8]	
VSSNEG_8LINESELECTOUTPUT_04	la_data_in[17]	
LA_OUT_8LINESELECTOUTPUT02_LSB	la_data_out[7:0]	
LA_OUT_8LINESELECTOUTPUT02_WIDTH	N/A (value 8)	
LA_OUT_8LINESELECTOUTPUT04_LSB	la_data_out[16:9]	
LA_OUT_8LINESELECTOUTPUT04_WIDTH	N/A (value 8)	

Table 10.1: Pinout

10.2 Alternative/Initial Versions of Design

Due to the nature of our project, our design was pretty much set in stone from the beginning. We had two previous team's work to go off of, so we had a good idea of what we needed to do. Our job was integrating all of the designs onto one chip. Rather than revising our design, most of our time was spent getting familiar with tools and verifying/improving work from the previous teams.

10.3 Other Considerations if applicable

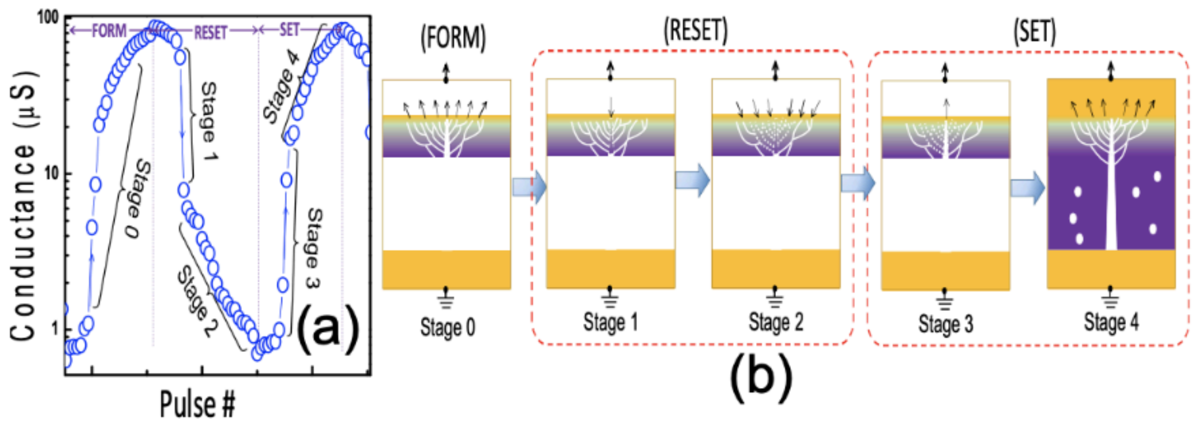


Fig. 10 After FORMing, the transition mechanism during RESET/SET can be divided into 4 stages. During FORMing, in Stage 0, a major filament with a wide radius and small filaments have developed. During RESET, in Stage 1, the major filament recesses and the conductance drops abruptly; in Stage 2, the small filaments slowly recess and the conductance drops gradually. During SET, in Stage 3. the major filament re-connects and the conductance jumps; in Stage 4, the small filaments re-connect and the conductance slowly increases.

Figure 10.2: explains ReRAM structural and operation basics and is extracted directly from [10]

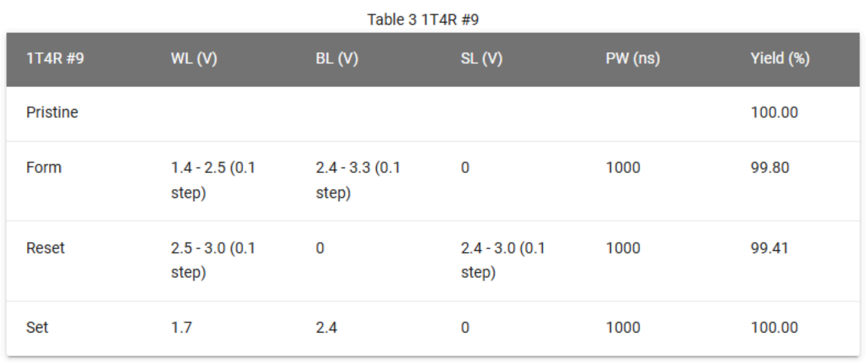


Table 3 1T4R #9

1T4R #9	WL (V)	BL (V)	SL (V)	PW (ns)	Yield (%)
Pristine					100.00
Form	1.4 - 2.5 (0.1 step)	2.4 - 3.3 (0.1 step)	0	1000	99.80
Reset	2.5 - 3.0 (0.1 step)	0	2.4 - 3.0 (0.1 step)	1000	99.41
Set	1.7	2.4	0	1000	100.00

Alternative programming schemes can be used. For example, from our data on the 1T4R test #9 above, one could consider using a fixed reset pulse of WL=3V, SL=3V with a 1000ns pulse to ensure high RESET yields. Additionally, a lower voltage can be tried multiple times (e.g., a SET/READ and verify, or RESET/READ and verify) until the operation is successful. The pulse length is also a free variable, shorter (e.g., 100-200ns SET/RESET) pulses, with a verify scheme can also be used to reduce average write time.

Figure 10.3: explains ReRAM state change requirements and is extracted directly from [5]

ASIC Design Troubleshooting Guide

This guide contains troubleshooting steps for ASIC design workflows, including tools like Magic, Xschem, Netgen, Klayout, and ReRAM integration.

Table of Contents

General Setup Issues

Issue: Incorrect SPICE File Generation in Xschem

Symptoms: SPICE files contain incorrect or extraneous information.
Resolution:

1. Clone the `efabless/caravel_user_project_analog` repository:
   git clone https://github.com/efabless/caravel_user_project_analog
2. Run `make setup` as instructed on the Caravel documentation.
3. If Ngspice fails to install, copy the `pdk` folder from `analog_tutorial` to the new template directory.
4. Re-run simulations to verify SPICE file generation.

Magic Tool Issues

Issue: Incorrect Magic Configuration File

Symptoms: Using the wrong `.magicrc` file results in missing via options (e.g., extraDevices 1&2 menu selections).
Resolution:

1. Open Magic with the correct configuration:
   `magic -rcfile $PDKPATH/libs.tech/magic/sky130A.magicrc`
   a. When using ReRam, make sure to use sky130B
2. Avoid using `magic -rcfile sky130A.magicrc` as suggested in some tutorials.

Supporting Image:

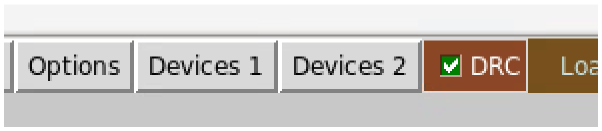


Figure 10.4: Magic interface showing Devices 1 & 2 menu with proper vias.

Issue: Wiring VDD and VSS Pins in Project Wrapper

Symptoms: Incorrect pin connections cause LVS failures.
Resolution:

1. Ensure both pin instances of `VDD` and `VSS` are connected (e.g., wire `vccd1` to `vccd1`).
2. Verify connections in the layout view before running LVS.

Supporting Image:

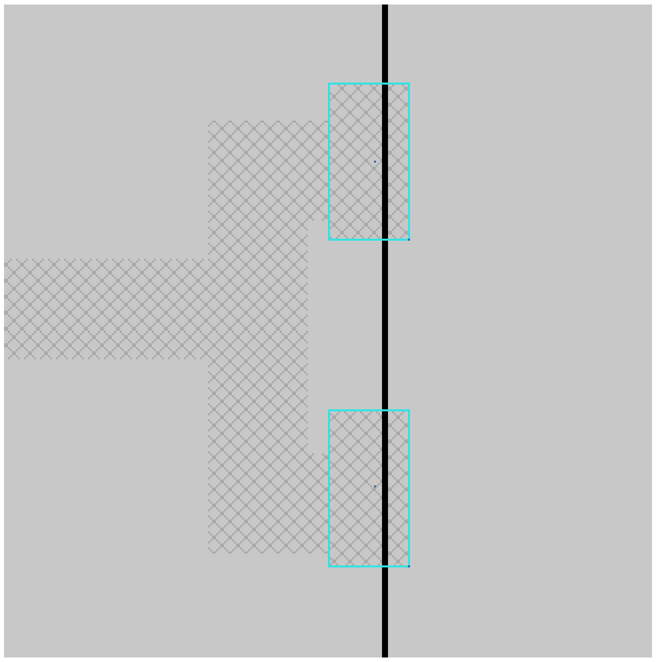


Figure 10.5: Layout showing correct `VDD` pin connections.

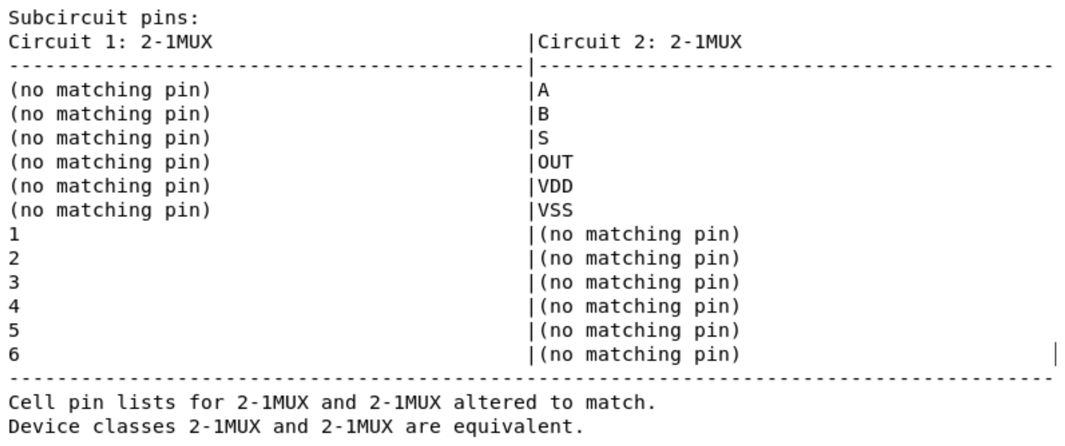
Xschem Issues

Issue: Symbol Type Mismatch (Primitive vs. Subcircuit)

Symptoms: LVS fails due to incorrect symbol type in Xschem after parasitic extraction or wrapper integration.
Resolution:

1. After parasitic extraction, switch the symbol type from “primitive” to “subcircuit” in Xschem.
2. Ensure the symbol is set to “subcircuit” when placing it in the `user_analog_project_wrapper`.
3. Generate the wrapper netlist and verify the symbol type.

Supporting Image:



```
Subcircuit pins:
Circuit 1: 2-1MUX                           |Circuit 2: 2-1MUX
--------------------------------------------|--------------------------------------------
(no matching pin)                           |A
(no matching pin)                           |B
(no matching pin)                           |S
(no matching pin)                           |OUT
(no matching pin)                           |VDD
(no matching pin)                           |VSS
1                                           |(no matching pin)
2                                           |(no matching pin)
3                                           |(no matching pin)
4                                           |(no matching pin)
5                                           |(no matching pin)
6                                           |(no matching pin)                          |
---------------------------------------------------------------------------------------------
Cell pin lists for 2-1MUX and 2-1MUX altered to match.
Device classes 2-1MUX and 2-1MUX are equivalent.
```

Figure 10.6: LVS result indicating symbol issue.

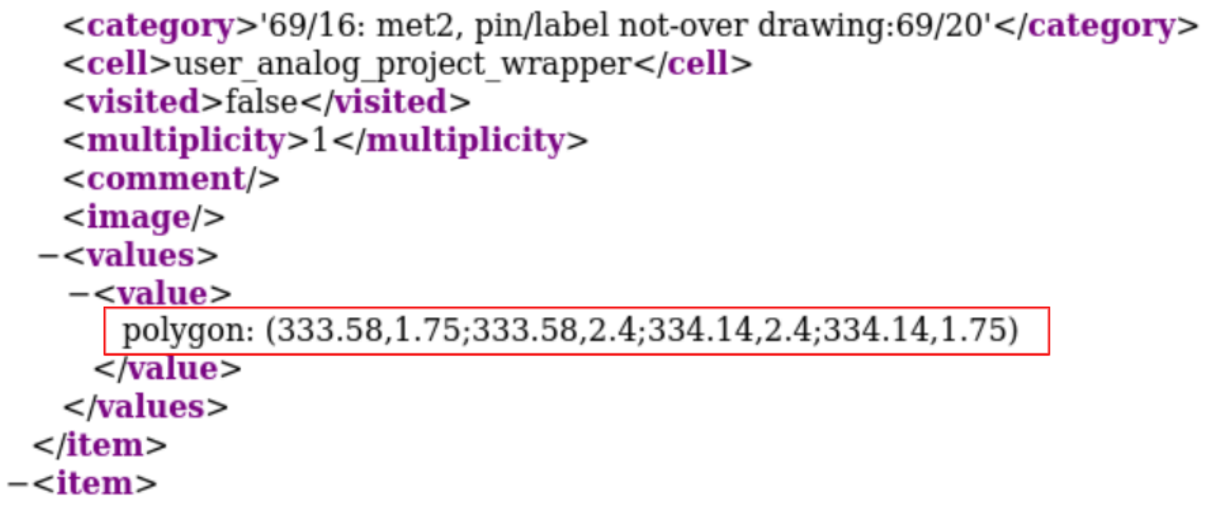
Klayout DRC Issues

Issue: Precheck DRC Failure

Symptoms: DRC errors reported in Klayout, visible in `klayout.xml`.
Resolution:

1. Open `klayout.xml` to locate error coordinates (e.g., polygon at (333.58, 1.75)).
2. Inspect the layout in Magic or Klayout at the specified coordinates.
3. Common cause: Missing or erased pin poly substance, causing errors on pin labels.
4. Correct the layout and re-run DRC.

Supporting Image:



```
    <category>'69/16: met2, pin/label not-over drawing:69/20'</category>
    <cell>user_analog_project_wrapper</cell>
    <visited>false</visited>
    <multiplicity>1</multiplicity>
    <comment/>
    <image/>
  -<values>
    -<value>
       polygon: (333.58,1.75;333.58,2.4;334.14,2.4;334.14,1.75)
     </value>
   </values>
 </item>
-<item>
```

Figure 10.7: Klayout highlighting a DRC error at specified coordinates.

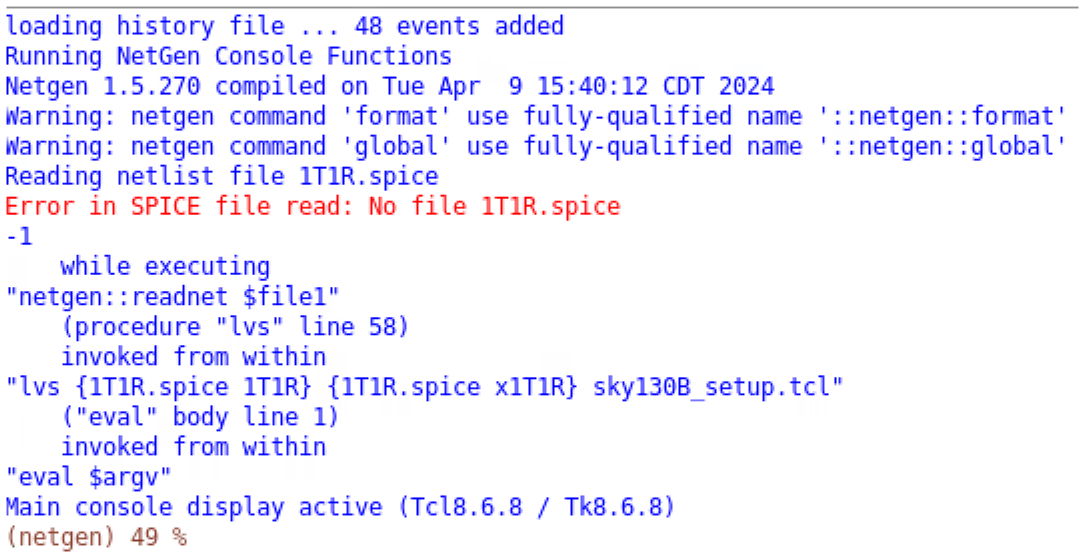
LVS Check Issues

Issue: LVS Execution in Netgen

Symptoms: Errors during LVS check due to incorrect file paths or settings.
Resolution:

1. Include both SPICE files (schematic and layout) in the `/netgen` directory and ensure that it is your current directory.
2. Use the following command to run LVS:
   netgen lvs "/path/to/schematic.spice schematic_subcircuit_name" "/path/to/magic.spice magic_subcircuit_name" sky130A_setup.tcl
   Example:
   ```
   netgen lvs "/home/tjdjakl/analog_tutorials/xschem/inverter.spice
   inverter" "/home/tjdjakl/analog_tutorials/mag/inverterMag.spice
   inverter" sky130A_setup.tcl
   ```
3. Ensure full file paths are provided if files are not found.
4. Do not include the `-batch` flag.
5. When generating the `.spice` netlist, check "set top to .subckt" in the tool.

Supporting Image:



```
loading history file ... 48 events added
Running NetGen Console Functions
Netgen 1.5.270 compiled on Tue Apr  9 15:40:12 CDT 2024
Warning: netgen command 'format' use fully-qualified name '::netgen::format'
Warning: netgen command 'global' use fully-qualified name '::netgen::global'
Reading netlist file 1T1R.spice
Error in SPICE file read: No file 1T1R.spice
-1
    while executing
"netgen::readnet $file1"
    (procedure "lvs" line 58)
    invoked from within
"lvs {1T1R.spice 1T1R} {1T1R.spice x1T1R} sky130B_setup.tcl"
    ("eval" body line 1)
    invoked from within
"eval $argv"
Main console display active (Tcl8.6.8 / Tk8.6.8)
(netgen) 49 %
```

Figure 10.8: LVS output indicating file path error.

Issue: Precheck LVS Failure Despite Local LVS Passing

Symptoms: No `lvs.report` in `precheck_results`, even though local LVS passes.
Resolution:

1. Copy the `lvs_config.json` file from `analog_tutorial/lvs/user_analog_project_wrapper/` to the ReRAM project's LVS directory.
2. Ensure the following lines were added to `lvs_config.json`:
   "STD_CELL_LIBRARY": "...",
   "INCLUDE_CONFIGS": "...",
   "LVS_SPICE_FILES_TO_FIX": "..."

3. Rename the LVS directory folder to `user_analog_project_wrapper` if needed.
4. Ensure the cell file is in the current directory.
5. Re-run the LVS check.

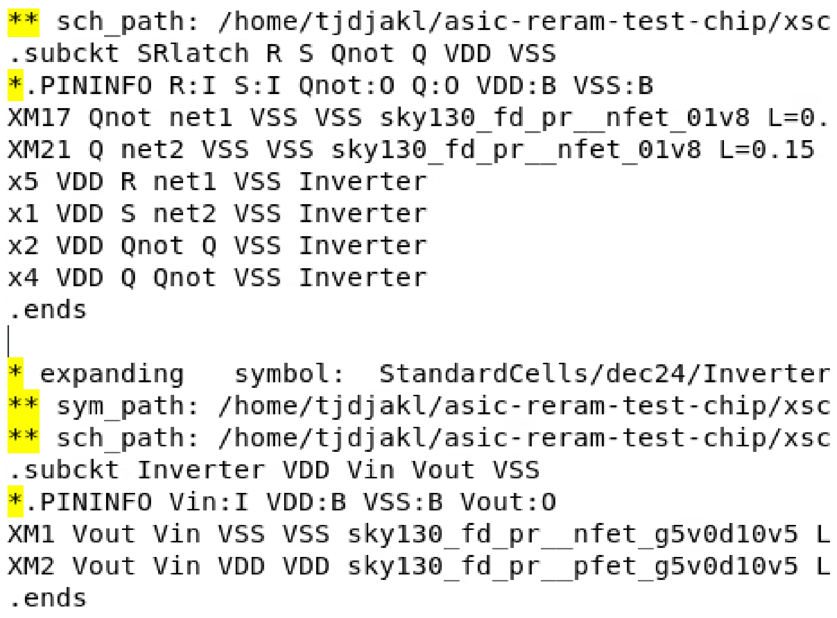
Precheck Failures

Issue: Consistency Error in SPICE File

Symptoms: Precheck fails with a “Consistency” error related to the circuit name.
Resolution:

1. Open the `.spice` file in `/netgen`.
2. Remove any comments within the file (any line beginning with * or **).
3. Re-run precheck.

Supporting Image:



```
** sch_path: /home/tjdjakl/asic-reram-test-chip/xsc
.subckt SRlatch R S Qnot Q VDD VSS
*.PININFO R:I S:I Qnot:O Q:O VDD:B VSS:B
XM17 Qnot net1 VSS VSS sky130_fd_pr__nfet_01v8 L=0.
XM21 Q net2 VSS VSS sky130_fd_pr__nfet_01v8 L=0.15
x5 VDD R net1 VSS Inverter
x1 VDD S net2 VSS Inverter
x2 VDD Qnot Q VSS Inverter
x4 VDD Q Qnot VSS Inverter
.ends

* expanding   symbol:  StandardCells/dec24/Inverter
** sym_path: /home/tjdjakl/asic-reram-test-chip/xsc
** sch_path: /home/tjdjakl/asic-reram-test-chip/xsc
.subckt Inverter VDD Vin Vout VSS
*.PININFO Vin:I VDD:B VSS:B Vout:O
XM1 Vout Vin VSS VSS sky130_fd_pr__nfet_g5v0d10v5 L
XM2 Vout Vin VDD VDD sky130_fd_pr__pfet_g5v0d10v5 L
.ends
```

Figure 10.9: Commented lines within `.spice` file to be removed.

Issue: GPIO Defines Failure

Symptoms: Precheck fails due to unconfigured GPIO pins.
Resolution:

1. Open up `~/verilog/rtl/user_defines.v`
2. Set all GPIO pins to `GPIO_MODE_MGMT_STD_BIDIRECTIONAL`.
3. For OEB pins, configure them as analog to resolve related errors.
4. Verify pin settings in the configuration file.
5. Re-run precheck.

Issue: Missing SPICE File in Xschem

Symptoms: Precheck LVS error: `cp: cannot stat '/path/to/user_analog_project_wrapper.spice'`.
Resolution:

1. Ensure the `user_analog_project_wrapper.spice` file is present in both `/xschem` and `/netgen` directories.
2. Copy the file to the required locations and re-run precheck.

Issue: README and Documentation Errors

Symptoms: Precheck fails due to default issues in documentation files.
Resolution:

1. Replace the contents of `/analog_tutorials/README.md` with `/analog_tutorials/docs/source/index.rst`.
2. Re-run precheck to verify resolution.

ReRAM Integration Issues

Issue: Installing ReRAM Source Files

Symptoms: Errors during ReRAM installation due to `sudo` permissions.
Resolution:

1. Clone the ReRAM repository:
   `git clone https://github.com/barakhoffer/sky130_ngspice_reram`
2. `cd sky130_ngspice_reram`
3. Edit `install.sh` to remove all instances of `sudo` at the bottom of the file.
4. Run the installation: `source install.sh`

Issue: LVS and Precheck with ReRAM

Symptoms: LVS or precheck failures due to SPICE file issues in the ReRAM project.
Resolution:

1. Modify SPICE files by renaming instances and removing unwanted lines, as detailed in referenced guides.
2. Update the project's `Makefile` per external documentation.
3. Run `source setup.sh` and `make setup`.
4. Fix `CARAVEL_ROOT` path errors by updating the export statement:
   export CARAVEL_ROOT=/correct/path
5. Re-run `make setup` and `make precheck`.

Additional Notes

- Always verify file paths and naming conventions to avoid common errors.
- Refer to external documentation (e.g., Jake's and Konnor's guides) for detailed steps on ReRAM and precheck workflows.
- For persistent issues, check the Caravel documentation or community forums(Slack).
- Ensure all tools (Magic, Xschem, Netgen, Klayout) are using the correct PDK version (`sky130A` or `sky130B`).

10.4 Code

We wrote two C files for testing our final chip. These can be found in our Git repository, under the bringup/code directory. The first file, sdmay25_defs.h, consists of utility functions for reading/writing logic analyzer pins as well as pin name definitions for ease of readability. The second file is an example main.c file, which shows how form, write, MAC, and read operations could be tested using the functions and definitions from sdmay25_defs.h.

sdmay25_defs.h:

```c
#include <stdint.h>
#include "caravel/defs.h"

//-----------//
// Utilities //
//-----------//

// gets the relative bit to the la register (% 32)
#define rel_bit(x)   ((x) % 32)

#define HIGH 1U
#define LOW 0U

// gets the la input register for the provided la pin
static inline volatile uint32_t* get_la_in_reg(uint32_t pin) {
    uint32_t index = pin / 32;
    switch(index) {
        case 0:
            return &reg_la0_data_in;
        case 1:
            return &reg_la1_data_in;
        case 2:
            return &reg_la2_data_in;
        case 3:
            return &reg_la3_data_in;
        default:
            return &reg_la0_data_in;    //should never be reached
    }
}

// gets the la output register for the provided la pin
static inline volatile uint32_t* get_la_out_reg(uint32_t pin) {
    uint32_t index = pin / 32;
    switch(index) {
        case 0:
            return &reg_la0_data;
        case 1:
            return &reg_la1_data;
        case 2:
            return &reg_la2_data;
        case 3:
            return &reg_la3_data;
        default:
            return &reg_la0_data;    //should never be reached
    }
}

// sets a single logic analyzer input pin
static inline void write_la_pin(uint32_t pin, uint32_t val) {
    *get_la_in_reg(pin) |= (1U << rel_bit(pin));
}

// sets multiple logic analyzer pins to the given value (use for multi-bit
inputs)
static inline void write_la_pins(uint32_t lsb_pin, uint32_t width,
uint32_t val) {
    uint32_t end_pin = lsb_pin + width - 1;

    // First, handle the first register (starting from lsb_pin)
    uint32_t first_reg = lsb_pin / 32;
    uint32_t first_reg_pin = rel_bit(lsb_pin);  // Bit position in the
first register
    uint32_t first_reg_mask = ((1U << width) - 1) << first_reg_pin;

    // Mask out the value for the first register
    uint32_t first_reg_value = (val << first_reg_pin) & first_reg_mask;
    *get_la_in_reg(lsb_pin) |= first_reg_value;  // Set bits in the first
register

    // If the range spans multiple registers
    if (first_reg != end_pin / 32) {
        // Set full 32-bit registers in between (if any)
        for (uint32_t i = first_reg + 1; i < end_pin / 32; ++i) {
            uint32_t mid_val = (val >> (i * 32)) & 0xFFFFFFFFU;  //
Extract full 32 bits for the middle registers
            *get_la_in_reg(i * 32) |= mid_val;
        }

        // Handle the last register (from the beginning of last pin to the
end of the range)
        uint32_t last_reg_pin = rel_bit(end_pin); // Bit position in the
last register
        uint32_t last_reg_mask = (1U << (last_reg_pin + 1)) - 1; // Mask
to clear up to the last bit

        // Extract the remaining bits for the last register
        uint32_t last_reg_value = val >> (end_pin / 32 * 32);
        *get_la_in_reg(end_pin) |= (last_reg_value & last_reg_mask);  //
Set bits in the last register
    }
}

// gets a single logic analyzer output pin
static inline uint32_t read_la_pin(uint32_t pin) {
    return (*get_la_out_reg(pin) & (1U << rel_bit(pin))) >> rel_bit(pin);
```

```
}

// gets multiple logic analyzer output pins (use for output bus)
static inline uint32_t read_la_pins(uint32_t lsb_pin, uint32_t width) {
    uint32_t end_pin = lsb_pin + width - 1;
    uint32_t result = 0;

    // First, handle the first register (starting from lsb_pin)
    uint32_t first_reg = lsb_pin / 32;
    uint32_t first_reg_pin = rel_bit(lsb_pin);  // Bit position in the
first register
    uint32_t first_reg_mask = ((1U << width) - 1) << first_reg_pin;

    // Get bits from the first register
    result |= (*get_la_out_reg(lsb_pin) & first_reg_mask) >>
first_reg_pin;

    // If the range spans multiple registers
    if (first_reg != end_pin / 32) {
        // Get full 32-bit registers in between (if any)
        for (uint32_t i = first_reg + 1; i < end_pin / 32; ++i) {
            result |= (*get_la_out_reg(i * 32) & 0xFFFFFFFFU) << ((i -
first_reg) * 32);
        }

        // Handle the last register (from the beginning of last pin to the
end of the range)
        uint32_t last_reg_pin = rel_bit(end_pin); // Bit position in the
last register
        uint32_t last_reg_mask = (1U << (last_reg_pin + 1)) - 1; // Mask
to clear up to the last bit

        // Get the remaining bits for the last register and shift into the
result
```

```
        result |= (*get_la_out_reg(end_pin) & last_reg_mask) << ((end_pin
/ 32 - first_reg) * 32);
    }

    return result;
}

//----------------//
// Architectures //
//----------------//

// Architecture Select Bits

#define ARCH_SELECT_LSB 125
#define ARCH_SELECT_WIDTH 2
#define ARCH_SELECT_MASK ((1U << ARCH_SELECT_WIDTH) - 1) <<
(rel_bit(ARCH_SELECT_LSB))

// Architecture Data Out
#define ARCH_DATA_OUT_LSB 0
#define ARCH_DATA_OUT_WIDTH 18
#define ARCH_DATA_OUT_MASK ((1U << ARCH_DATA_OUT_WIDTH) - 1) <<
(rel_bit(ARCH_DATA_OUT_LSB))

//-- Architecture Pin Definitions --//

// bit lines
#define WRITE_SELECT_8LINEBITINPUT_1        0
#define WRITE_SELECT_8LINEBITINPUT_2        28

#define WRITE_FORM_SELECT_8LINEBITINPUT_1   1
#define WRITE_FORM_SELECT_8LINEBITINPUT_2   29

#define LA_IN_8LINEBITINPUT_1_LSB           2
```

```
#define LA_IN_8LINEBITINPUT_1_WIDTH         8

#define LA_IN_8LINEBITINPUT_2_LSB           30
#define LA_IN_8LINEBITINPUT_2_WIDTH         8

// word lines
#define WRITE_SELECT_8LINEBITINPUT_1        10
#define WRITE_SELECT_8LINEBITINPUT_2        38

#define LA_IN_8LINEWORDINPUT_1_LSB          11
#define LA_IN_8LINEWORDINPUT_1_WIDTH        8

#define LA_IN_8LINEWORDINPUT_2_LSB          39
#define LA_IN_8LINEWORDINPUT_2_WIDTH        8

#define LA_IN_8LINEWORDINPUT_1_LSB          11
#define LA_IN_8LINEWORDINPUT_1_WIDTH        8

// select lines
#define S_8LINESELECTINPUT_1                19
#define S_8LINESELECTINPUT_2                47

#define LA_IN_8LINESELECTINPUT_1_LSB        20
#define LA_IN_8LINESELECTINPUT_1_WIDTH      8

#define LA_IN_8LINESELECTINPUT_2_LSB        48
#define LA_IN_8LINESELECTINPUT_2_WIDTH      8

#define VSSNEG_8LINESELECTOUTPUT_02         8
#define VSSNEG_8LINESELECTOUTPUT_04         17

#define LA_OUT_8LINESELECTOUTPUT02_LSB      0
#define LA_OUT_8LINESELECTOUTPUT02_WIDTH    8
```

```
#define LA_OUT_8LINESELECTOUTPUT04_LSB      9
#define LA_OUT_8LINESELECTOUTPUT04_WIDTH    8

//----------------------//
// Peripheral Circuitry //
//----------------------//

// Peripheral Select Bit

#define PERIPH_SELECT 127
#define PERIPH_SELECT_MASK ((1U << (rel_bit(PERIPH_SELECT))))
#define PERIPH_DATA_OUT_LSB 0
#define PERIPH_DATA_OUT_WIDTH 43

//-- Peripheral Circuit Pin Definitions --//

// Two Stage Op Amp
#define TWO_STAGE_OP_AMP_IN_MINUS   0
#define TWO_STAGE_OP_AMP_IN_PLUS    1

#define TWO_STAGE_OP_AMP_OUT        0

// Comparator
#define COMPARATOR_IN_PLUS          2
#define COMPARATOR_IN_MINUS         3

#define COMPARATOR_OUT_PLUS         1
#define COMPARATOR_OUT_MINUS        2

// 5T Op Amp
#define OP_AMP_5T_IN_PLUS           4
#define OP_AMP_5T_IN_MINUS          5
#define OP_AMP_5T_NEG_IN_PLUS       6
#define OP_AMP_5T_NEG_IN_MINUS      7
```

```
#define OP_AMP_5T_OUT                3
#define OP_AMP_5T_NEG_OUT            4

// TIAs
#define TIA_02_IN                    8
#define TIA_04_IN                    9

#define TIA_02_OUT                   5
#define TIA_04_OUT                   6

// ADC
#define ADC_IN                       10
#define ADC_OUT                      7

// 1T1R
#define BL_IN_1T1R                   12
#define WL_IN_1T1R                   13

#define SL_OUT_1T1R                  23

// 2x2 Crossbar
#define BL_IN_2X2_LSB                14
#define BL_IN_2X2_WIDTH              2
#define WL_IN_2X2_LSB                16
#define WL_IN_2X2_WIDTH              2

#define SL_OUT_2X2_LSB               24
#define SL_OUT_2X2_WIDTH             2

// 4x4 Crossbar
#define BL_IN_4X4_LSB                18
#define BL2_IN_4X4_WIDTH             4
#define WL_IN_4X4_LSB                22
```

```
#define WL_IN_4X4_WIDTH             4

#define SL_OUT_4X4_LSB              26
#define SL_OUT_4X4_WIDTH            4

// 8x8 Crossbar
#define BL_IN_8X8_LSB               26
#define BL_IN_8X8_WIDTH             8
#define WL_IN_8X8_LSB               34
#define WL_IN_8X8_WIDTH             8

#define SL_OUT_8X8_LSB              30
#define SL_OUT_8x8_WIDTH            8

// Inverter
#define INVERTER_IN                 42

#define INVERTER_OUT                38

// Transmission Gate
#define TRANSMISSION_GATE_IN        43
#define TRANSMISSION_GATE_SP        44
#define TRANSMISSION_GATE_SN        45

#define TRANSMISSINO_GATE_OUT       39

// 2 to 1 Mux
#define MUX_2_1_SEL                 46
#define MUX_2_1_A                   47
#define MUX_2_1_B                   48

#define MUX_2_1_OUT                 40

// 4 to 1 Mux
```

```
#define MUX_4_1_SEL1                49
#define MUX_4_1_SEL2                50
#define MUX_4_1_A                   51
#define MUX_4_1_B                   52
#define MUX_4_1_C                   53
#define MUX_4_1_D                   54

#define MUX_4_1_OUT                 41

// SR Latch
#define SR_LATCH_S                  55
#define SR_LATCH_R                  56

#define SR_LATCH_Q                  42
```

main.c:

```
#include "sdmay25_defs.h"


int main() {
    // inverter test
    write_la_pin(PERIPH_SELECT, HIGH); // selects for output from
peripheral circuitry testbench
    write_la_pin(INVERTER_IN, HIGH); // sets input to inverter high
    if(read_la_pin(INVERTER_OUT) == LOW) {  // check output of inverter
        printf("Inverter test successful! (Expected LOW, Actual LOW)");
    } else {
        printf("Inverter test failed. (Expected LOW, Actual HIGH)");
    }



    // forming ReRAM cells
    write_la_pin(WRITE_FORM_SELECT_8LINEBITINPUT_1, HIGH);
    write_la_pins(LA_IN_8LINEBITINPUT_1_LSB, LA_IN_8LINEBITINPUT_1_WIDTH,
0xFF);
    write_la_pins(LA_IN_8LINEWORDINPUT_1_LSB,
LA_IN_8LINEWORDINPUT_1_WIDTH, 0xFF);
    write_la_pins(LA_IN_8LINESELECTINPUT_1_LSB,
LA_IN_8LINESELECTINPUT_1_WIDTH, 0x0);
    // INSERT APPROPRIATE DELAY FOR FORMING //
    write_la_pin(WRITE_FORM_SELECT_8LINEBITINPUT_1, LOW);
    write_la_pins(LA_IN_8LINEBITINPUT_1_LSB, LA_IN_8LINEBITINPUT_1_WIDTH,
0x0);
    write_la_pins(LA_IN_8LINEWORDINPUT_1_LSB,
LA_IN_8LINEWORDINPUT_1_WIDTH, 0x0);



    // Writing to ReRAM cells, setting all cells to the same value
    uint8_t value = 0xFF;
    write_la_pins(LA_IN_8LINEBITINPUT_1_LSB, LA_IN_8LINEBITINPUT_1_WIDTH,
value);
    write_la_pins(LA_IN_8LINESELECTINPUT_1_LSB,
LA_IN_8LINESELECTINPUT_1_WIDTH, 0xFF ^ value);
    write_la_pins(LA_IN_8LINEWORDINPUT_1_LSB,
LA_IN_8LINEWORDINPUT_1_WIDTH, 0xFF);
    write_la_pin(WRITE_SELECT_8LINEBITINPUT_1, HIGH);
    // MAY NEED TO ADD DELAY FOR RESULTS TO HOLD //



    // MAC
    write_la_pins(LA_IN_8LINEBITINPUT_1_LSB, LA_IN_8LINEBITINPUT_1_WIDTH,
0x0);
    write_la_pins(LA_IN_8LINESELECTINPUT_1_LSB,
LA_IN_8LINESELECTINPUT_1_WIDTH, 0x0);
    write_la_pins(LA_IN_8LINEWORDINPUT_1_LSB,
LA_IN_8LINEWORDINPUT_1_WIDTH, 0x1);
```

```
    // MAY NEED TO ADD DELAY FOR RESULTS TO PROPAGATE //



    // Read output from architecture 0
    write_la_pins(ARCH_SELECT_LSB, ARCH_SELECT_WIDTH, 0x0);
    write_la_pin(PERIPH_SELECT, LOW);
    uint32_t result = read_la_pins(LA_OUT_8LINESELECTOUTPUT02_LSB,
LA_OUT_8LINESELECTOUTPUT02_WIDTH);
    if(result == 8) {   // each cell has a 1, multiplying by the 1 on our
wordline, 1 * 1 + 1 * 1 + ... 8 times = 8
        printf("MAC test successful! (Expected 8, Actual 8)");
    } else {
        printf("MAC test failed. (Expected 8, Actual %d)", result);
    }
}
```

10.5 Team Contract

10.5.1 Team Members

- ➢ Computer Engineers
    - Noah Mack

- ➢ Electrical Engineers
    - Sam Burns
    - Travis Jakl
    - Olivia Price

10.5.2 Required Skill Set For Your Project

Below is a list of the required technical skill sets that are needed to complete this project successfully.

- Analog Design
    - Fabrication and Layout Design
    - Power Management and Efficiency
    - Circuit Simulation and Analysis
    - Measurement and Debugging

- ReRAM Cell Architecture

- Digital Design
    - Synthesis
    - Logic and Circuit Design
    - Memory and Data Path Design
    - Hardware Description Design (Verilog)
    - Integration and Interface Design (C-code)
    - Testing and Debugging

10.5.3 Skill Sets Covered by the Team

Below is a table of skills that each team has acquired for this project, either during the semester or previously learned from prior semesters.

Skills	Team Member
Fabrication and Layout Design	All members
Power Management and Efficiency	All members
Circuit Simulation and Analysis	Sam Burns, Travis Jakl, & Olivia Price
Measurement and Debugging (Analog)	Sam Burns, Travis Jakl, & Olivia Price
ReRAM Cell Architecture	Sam Burns, Travis Jakl, & Olivia Price
Synthesis	Noah Mack
Logic and Circuit Design	All members
Memory and Data Path Design	All members
Hardware Description Design (Verilog)	Noah Mack
Integration and Interface Design (C-code)	Noah Mack & Sam Burns
Testing and Debugging (Digital)	Noah Mack

Table 10.2: Skill sets covered by team

10.5.4 Project Management Style Adopted By The Team

This team's project management was waterfall. Included in section 3.4 is our Gantt chart, which is a style of water project management. We divided the months across the two semesters into seven different tasks. Each primary task has one or two sub-tasks. We decided not to include hard deadlines because many of our functions depend on figuring out task one: how to use the open-source tools. Not much documentation is provided, so it will take time to acquire the knowledge on how to utilize them to their fullest capabilities.

10.5.5 Initial Project Management Roles

Below is the list of each team member and their acquired roles for this project.

- Sam Burns: Analog Signal Designer & Digital Signal Designer
- Travis Jakl: Analog Signal Designer & Digital Signal Designer
- Noah Mack: Digital Signal Designer
- Olivia Price: Analog Signal Designer

10.5.6 Team Contract

Team Name

_________________SDMay25-19_________________

Team Members:

1) ______________Noah Mack________ 2) ___________Olivia Price__________
3) ____________Sam Burns__________ 4) ______________Travis Jakl________

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
    a. Team meeting with advisors: Thursday, 2 pm, Durham 353
    b. Team meetings: Friday 1 pm, TLA, and Sundays 2 pm at ChipISU Club in Durham
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face): Phone
3. Decision-making policy (e.g., consensus, majority vote): Consensus
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived): Meeting Notes will be kept on a running document in the shared google drive that has the rest of our collaborative resources.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

    Each person attends a set meeting. If attendance is not possible, then communication prior to the meeting is necessary.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

    Each team member maintains and holds their designated responsibilities, communicating progress and expected progress, as well as ensuring important tasks can be completed prior to deadlines

3. Expected level of communication with other team members:

   Communication should be clear and frequent. Communicate about problems, progress, or expected absences of meetings.

4. Expected level of commitment to team decisions and tasks:

   Each team member should be committed to their role and responsibilities, discussing any discrepancies with the rest of the team

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

   The proposed project will include one digital design member, Noah, while the rest will work on analog design. The client interaction will happen every week, face-to-face for an hour. This will include sharing out findings and whether the clients want what we propose. The individual component design will come down to Sam, Olivia, and Noah, while Travis coordinates between the three of us. This will ensure the analog and digital designs do not stray too far from each other.

2. Strategies for supporting and guiding the work of all team members:

   Travis will be the coordinator for digital and analog design. The goal of this project is to get all of the pieces of the FPGA board together. This involves knowing what everyone is doing and someone line streaming between us to keep us together.

3. Strategies for recognizing the contributions of all team members:

Team members will be responsible for making in-depth documentation of their work. Not only will detailed documentation make drafting the design document easier, but it will also allow members to efficiently keep track of their work.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
   a. Noah - Deep knowledge of digital design and embedded C, can handle most of the surrounding digital architecture. Only computer engineering major on the team.
   b. Travis - Knowledge of both digital and analog design. This will be needed to integrate the analog and digital aspects of this project.
   c. Olivia - Electrical engineering with an emphasis in VLSI, which is designing and fabricating elements to be put on a board. This will help with putting the entire design together using analog and digital knowledge.
   d. Sam - Knowledge of analog circuit design and familiarity with digital schematic capture tools.
2. Strategies for encouraging and supporting contributions and ideas from all team members:

   Discuss all ideas from all team members; don't dismiss any ideas without exploring them first.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

   Be upfront in communication. Be willing to bring something up to the group if it is bothering you. If truly uncomfortable with that, reach out to Professors or Advisors for help.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

Be prepared for the tapeout date of April 11, 2025. This involves having a working design that has been thoroughly simulated both in the digital tools and possibly on an FPGA to verify circuit behavior.

2. Strategies for planning and assigning individual and teamwork:

   When tasks/problems arise, discuss as a group how to distribute the workload, with a clear understanding of who is taking on what responsibility

3. Strategies for keeping on task:

   Assisting each other when problems arise, working in group settings with each other on the project, keeping each other accountable and on task

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

   The team discusses with the infractor their wrongdoings, and if no change occurs afterward, meet with advisors to discuss what further actions to take.

2. What will your team do if the infractions continue?

   Meet with the course/team advisors and discuss what further actions to take. If necessary, 4910 instructors will be involved if project advisors are unable to resolve the issue.

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*

b) *I understand that I am obligated to abide by these terms and conditions.*

c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) ______________Travis Jakl__________________________ DATE ______9/13/24______

2) ______________Sam Burns__________________________ DATE ______9/13/24______

3) ______________Olivia Price__________________________DATE ______9/13/24______

4) ______________Noah Mack__________________________ DATE ______9/13/24______