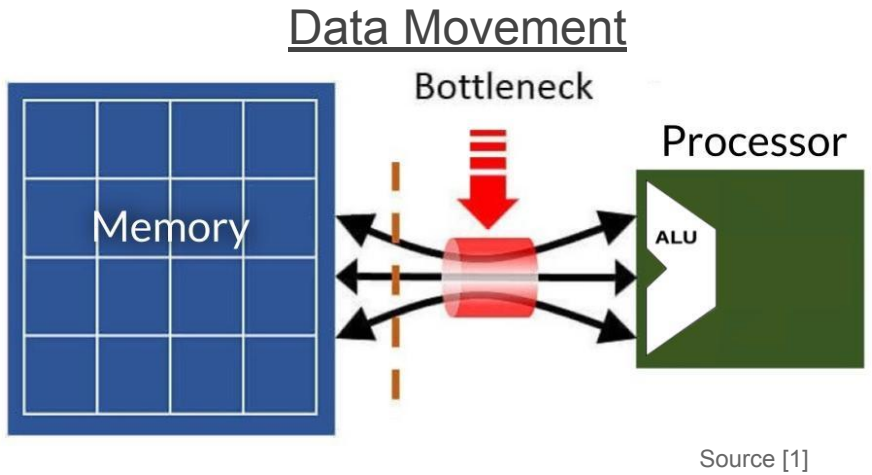
SDMay25-19: ASIC design for Compute-In-Memory Research Vehicle

By: Noah Mack, Olivia Price, Travis Jakl, Sam Burns

Clients/Advisors: Dr. Henry Duwe and Dr. Cheng Wang

Problem Statement

- Matrix-vector multiplication (MVM) is crucial in machine learning computations
- Data transfer between memory and CPU creates bottlenecking
- **What can be done about this?**

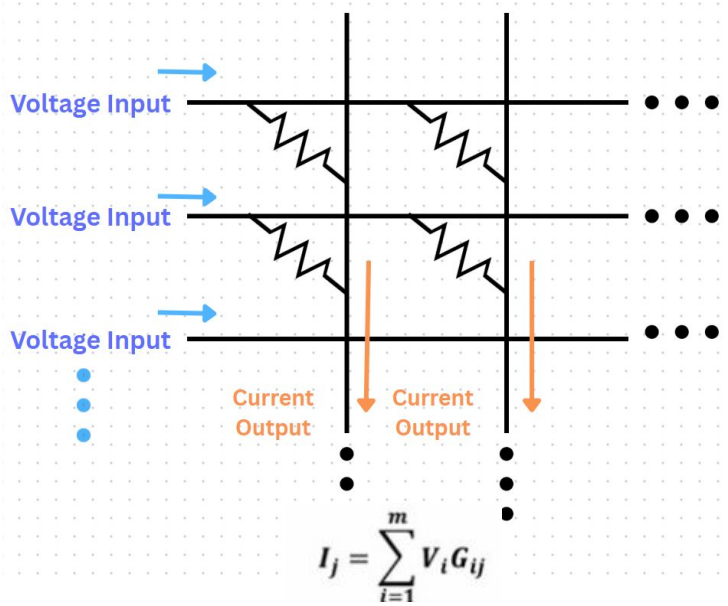


Source [1]

For more information, see section 1.1 of the design document

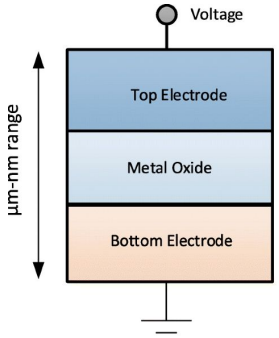
Compute-In-Memory

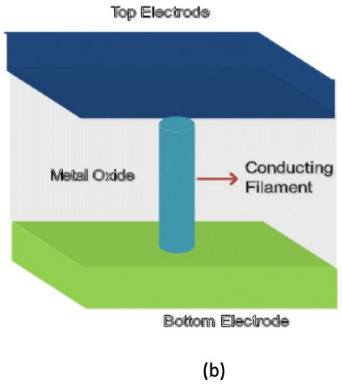
- MVM is performed within memory system
  - Works via multiply and accumulate process
- Alleviates bottleneck between memory and CPU
- Many technologies for this system
  - Resistive Memory (ReRAM) is the one our clients are interested in because of density and efficiency

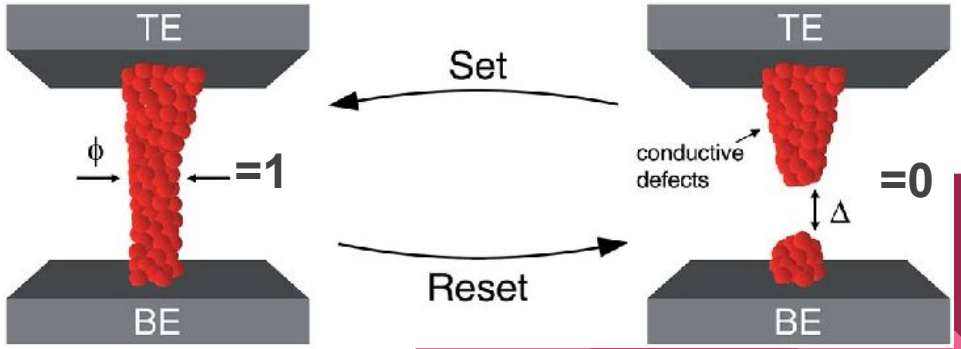


What is ReRAM?

- Non-volatile memory technology
- High and low conductance states controlled by filament formation
- Can perform MAC operations (Ohm's Law) by taking advantage of the two states



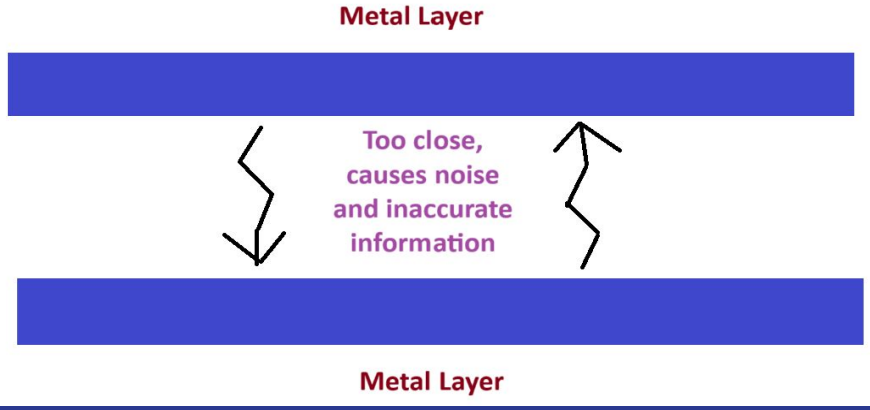


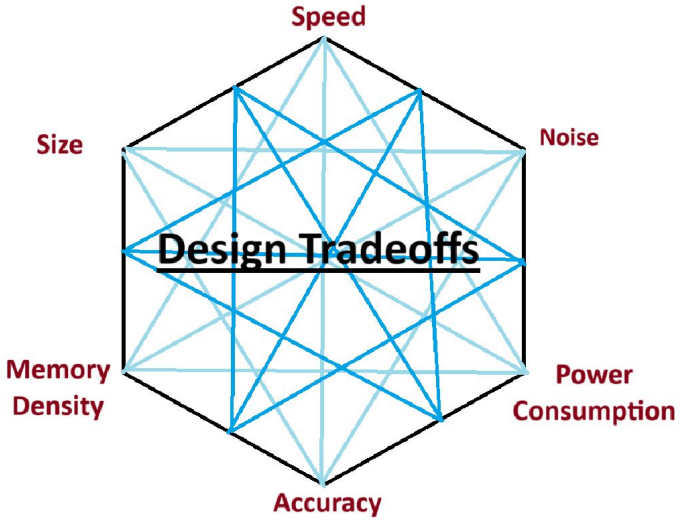


For more information about ReRAM see section 1.3 on the design document

Our Project

- Design and fabricate a test chip implementing four ReRAM-based compute-in-memory architectures (Research Test Vehicle)
- Characterize system performance across architectures and components
- Use Skywater 130nm process and Efabless tools for fabrication and tape-out





Users

Primary Users:

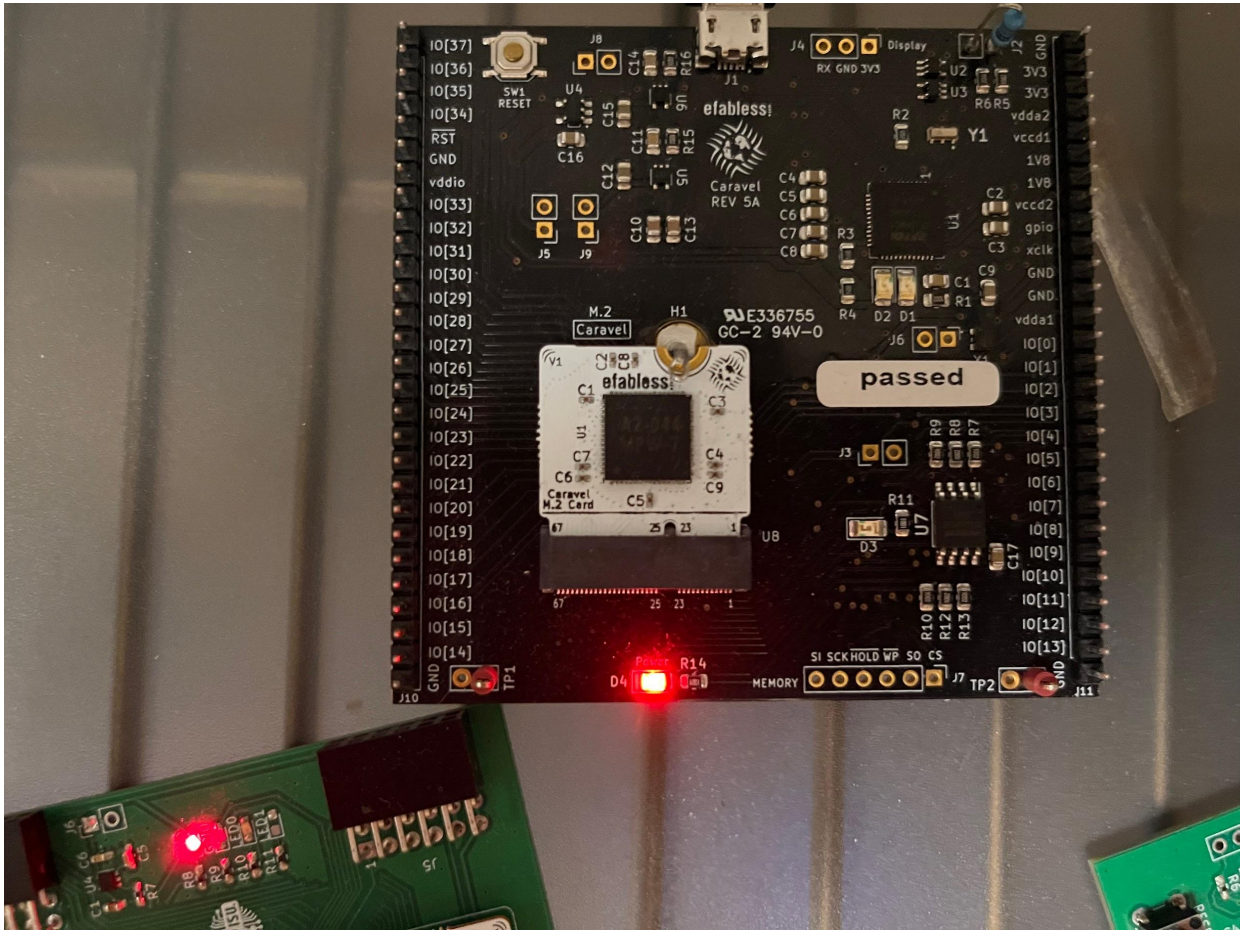
- ISU ECpE faculty and research teams (graduate and undergraduate).

Secondary Users:

- ChipForge: An Iowa State University club
- Students in CPRE 4870/5870, EE 5260

Tertiary Users (Public):

- Public (which will further research in the field and help the field evolve)



A prior efabless project being used by Chip Forge

Requirements

- Component circuits are individually characterizable and accessible through external analog I/O
- Four different ReRAM compute crossbar architectures
  - Different configurations of source, bit, word lines
  - Different peripheral circuitry
  - True crossbar vs. 1T1R grid
- Uncertainty evaluation for implemented architectures
  - Crossbar noise
  - ADC noise
  - Design for worst case crossbar noise within one ADC step

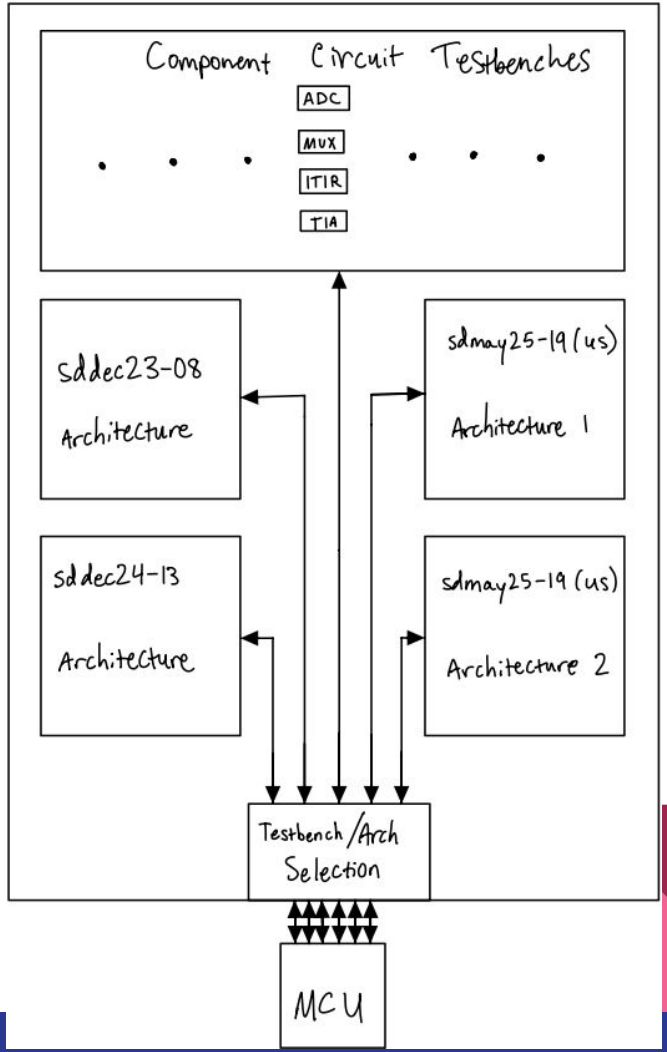
Requirements Cont.

- Bring-up Documentation and code for validation and debug
  - Characterizing the component circuitry via individual test benches.
  - FORMing the ReRAM cells
  - 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.
  - Refine and update Chip Forge documentation

For more information, see section 2.1 of the design document

Design Strategy

1. Gain familiarity with open source tools
2. Test components from previous teams and fix where needed
3. Implement our own unique architectures
4. Create top level schematic with ability to select between different architectures and component circuitry
5. Write C code and documentation



Project Timeline

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									

For more information, see section 3.4 of the design document

Fall: Goals and Challenges

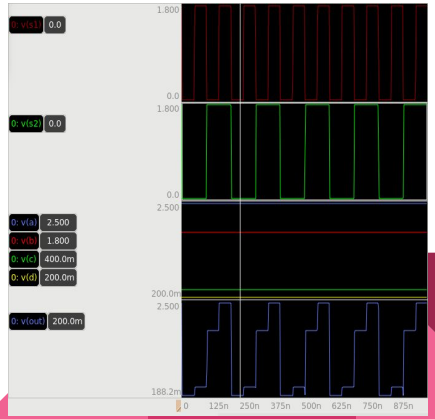
- Goals:
  - Install toolflow
  - Learn basics about each aspect of mixed-signal toolflow
  - Understand function and simulation of ReRAM
  - Familiarize ourselves with past teams' work to inform our own design decisions

XSCHEM

NGSPICE

Magic VLSI Layout Tool

Current distribution version 8.3



Gtk analog waveform viewer

Fall: Testing and Results

- Learned toolflow by taking an inverter all the way through precheck
- Started testing previous teams designs

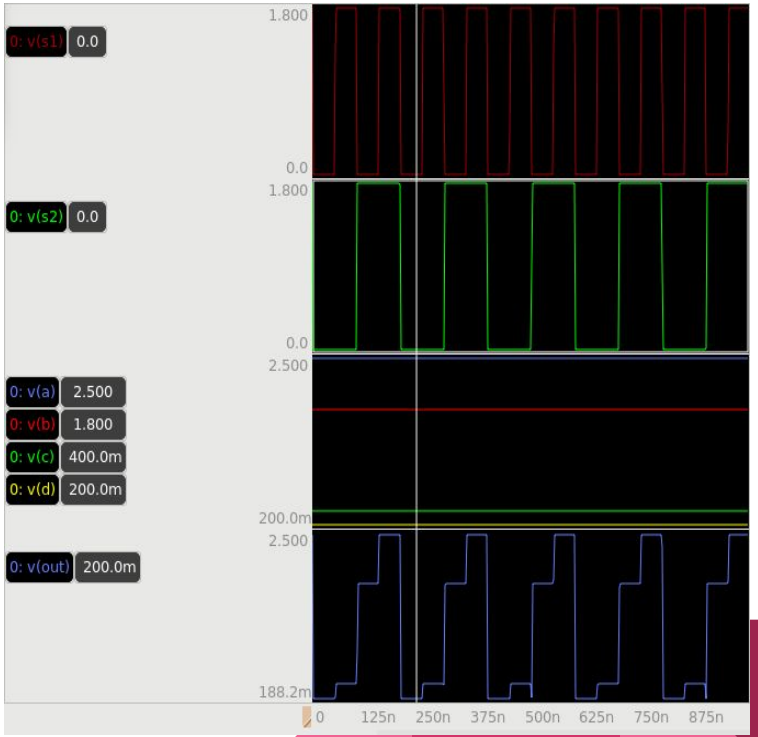
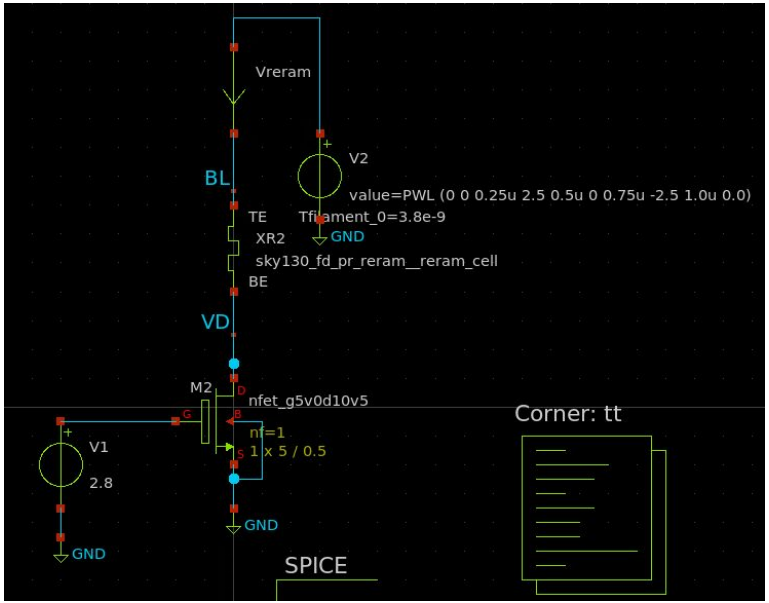


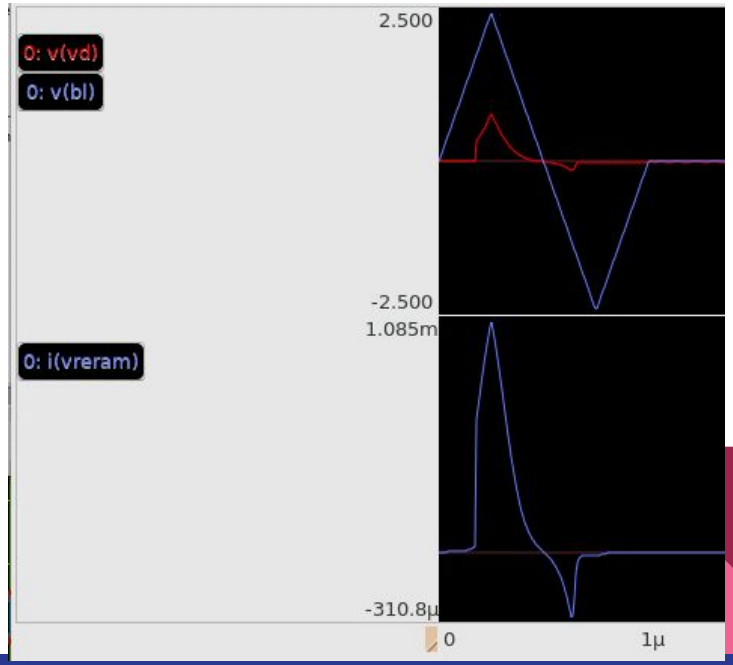
Figure 5.7: Resulting waveforms from 4-to-1 multiplexer testbench from team sddec23-08

For more information, see section 5.8 of the design document

Fall: Testing and Results Cont.

- Able to characterize ReRAM through simulation and reading skywater documentation





For more information, see section 5.8 of the design document

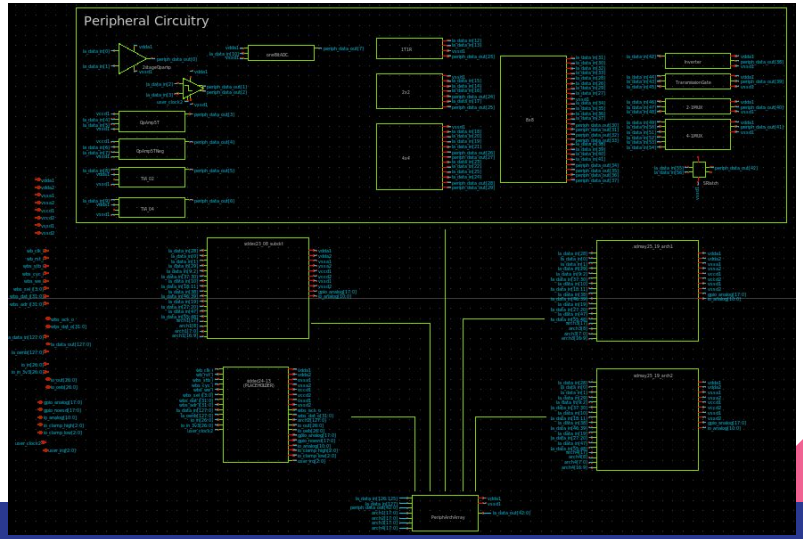
Project Timeline

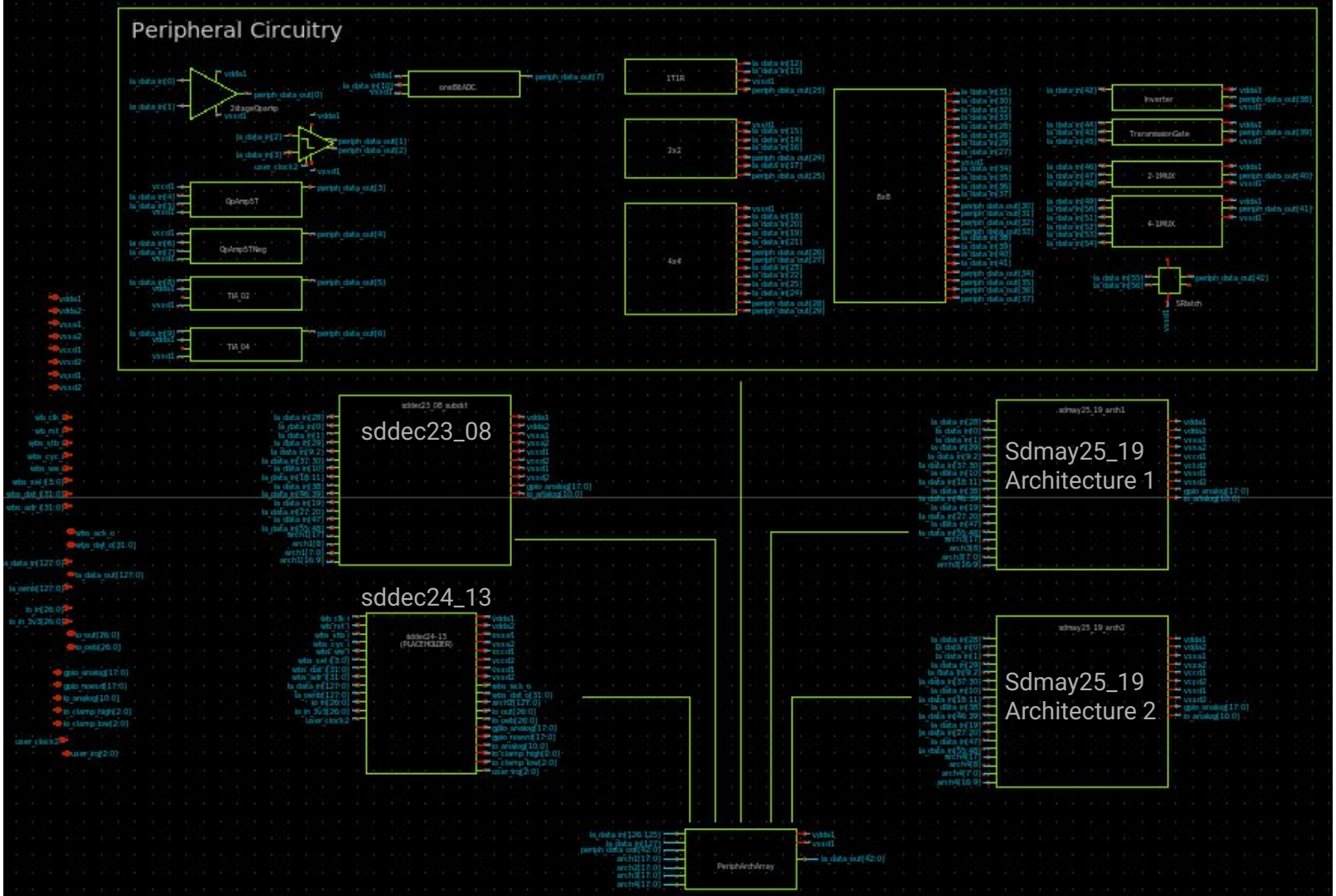
Task Decomposition		September	October	November	December	January	February	March	April	May
Task 1: Figure out the tools and research ReRAM functionality	Complete									
Installing toolchain	Complete									
Demonstrate competency with tools and refine ChipForge tutorials	Complete									
Get an analog 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									

For more information, see section 3.4 of the design document

Spring: Goals and Challenges

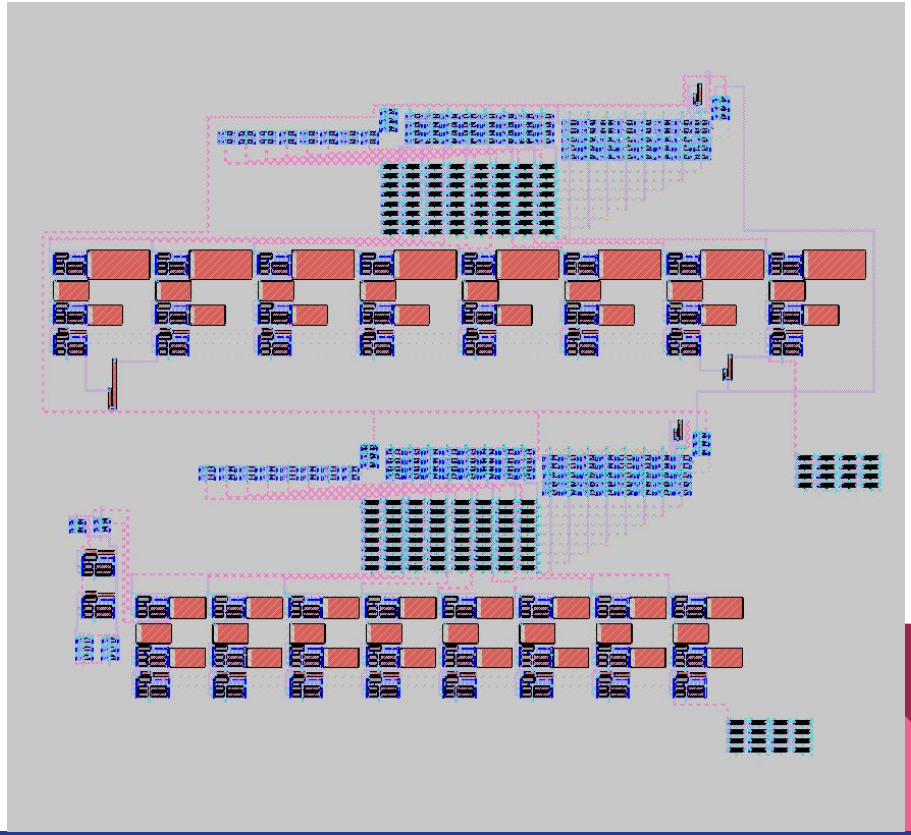
- Create a top level design with 4 total architectures and component testbench
- Designs pass local precheck and hosted checks before April 21st to meet fabrication deadline
- Write bring up code and documentation for end users to begin testing our chip




Peripheral Circuitry
sddec23_08
Sdmay25_19
Architecture 1
sddec24_13
Sdmay25_19
Architecture 2

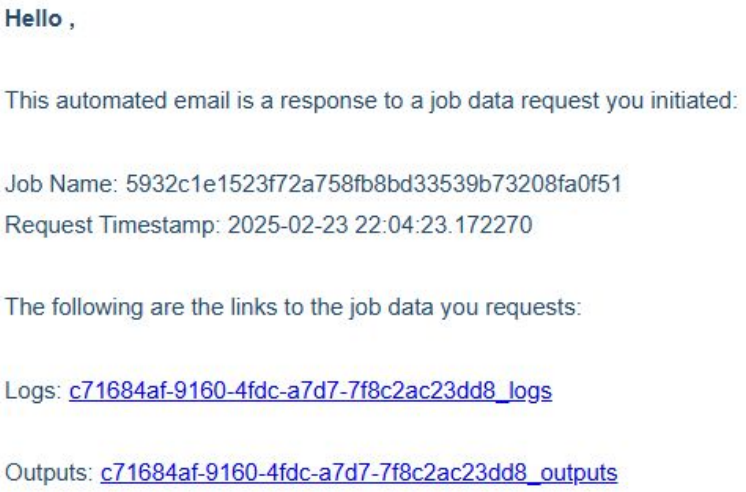
Spring: Work and Results Cont.

- Finished layout of one of our architectures, and layouts of all relevant subcomponents



Spring: Work and Results Cont.

- Successfully got components through precheck and tapeout-check on efabless servers.



Hello ,

This automated email is a response to a job data request you initiated:

Job Name: 5932c1e1523f72a758fb8bd33539b73208fa0f51
Request Timestamp: 2025-02-23 22:04:23.172270

The following are the links to the job data you requests:

Logs: c71684af-9160-4fdc-a7d7-7f8c2ac23dd8_logs

Outputs: c71684af-9160-4fdc-a7d7-7f8c2ac23dd8_outputs

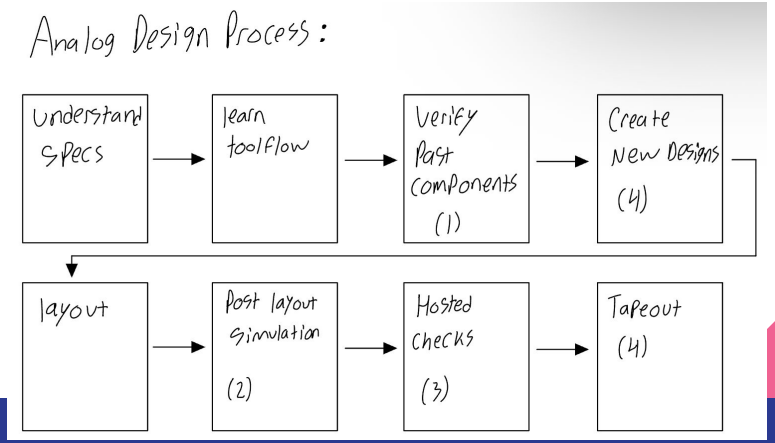
Spring: Work and Results Cont.

- Bring up code for testing architectures and characterizing components
- Provide API with read/write functionality for logic analyzer pins and pre defined pin names

```
// 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);
}
```

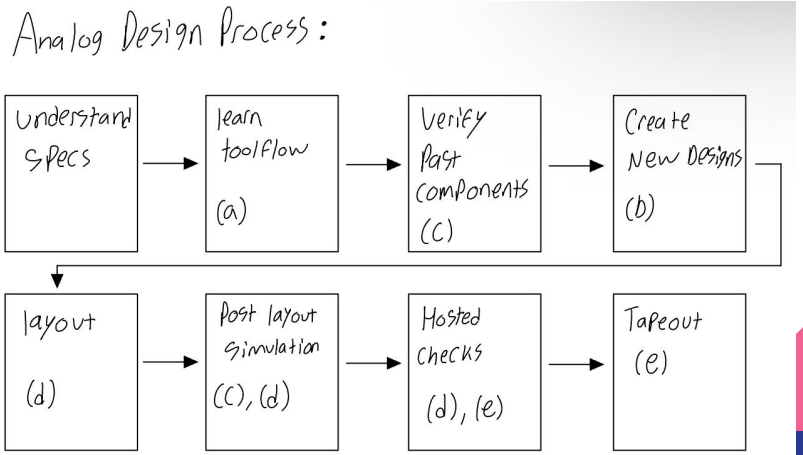
Risks

- Past team's design does not pass functional tests or pre-check (1)
    - Risk 40%
- At least one of the new designs does not satisfy the requirements (2)
    - Risk 15%
- Integrated top-level project wrapper design fails functional tests or pre-check (3)
    - Risk 55%
- Flicker noise is more impactful on the Skywater process than expected (4)
    - Risk 15%



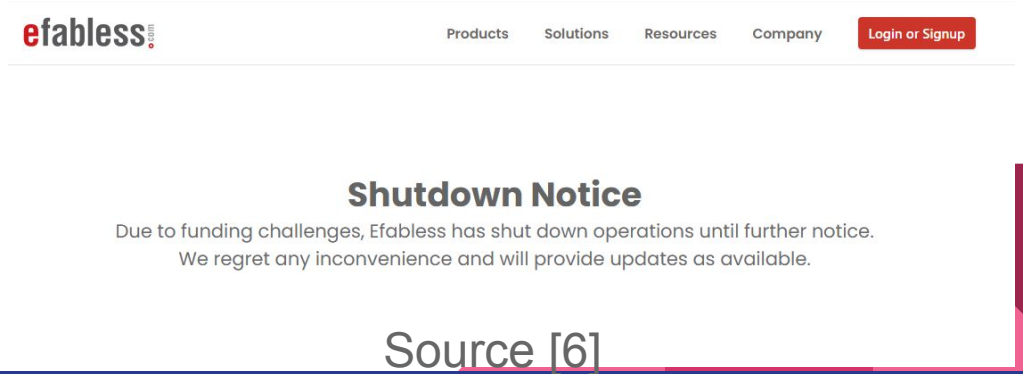
Mitigation

- Utilize efabless Slack community for tool related issues (a)
- Use distinct designs and create component testbench (b)
- Characterize all components from past teams before using in design (c)
- Resolve any issues with simulation or layout with individual components (d)
- Make use of paid tech-support from Efabless for Tapeout related issues (e)



Efabless Shutdown

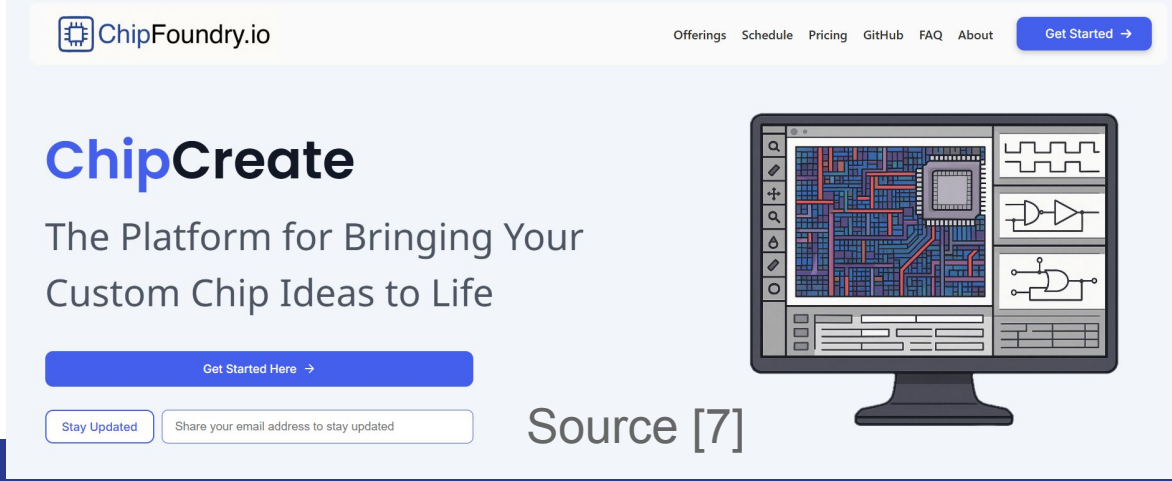
- Shut down in early March, 2025
- Lost access to paid support from Efabless team
- Open-source slack community shut down as well
- Issues with ReRAM SPICE model would go unfixed
- Continue project as planned



Source [6]

Next Steps

- Pursue the design through another similar company:
  - ChipFoundry.io
  - Tiny Tapeout
  - Etc.
- Use the designs as an educational tool for Chip Forge Members



Source [7]

Learning and Value Provided

- Gained in-depth understanding of tapeout process
- Developed an understanding of hierarchical analog design
- Laid out and tested our unique crossbar architecture
- Documentation can provide an example of the design process for Chip Forge members to use in the future
    - Analog documentation
    - Mixed signal integration process

Conclusions

- Able to deliver:
  - Two unique architectures
  - Layout of our first unique architecture
  - A component testbench for cell characterization
  - A top level design that incorporates all 4 compute-in-memory architectures
  - Bringup code that would allow users to test and characterize our design
- Gained valuable insight into design of a hierarchical analog system with mixed signal components
- Our documentation will be a useful reference for future Chip Forge Members pursuing an analog design

Image Sources

[1] https://www.linkedin.com/pulse/memory-bottleneck-ali-farahany/

[2] https://skywater-pdk.readthedocs.io/en/main/

[3] https://efabless.com/

[4] https://pubs.rsc.org/en/content/articlehtml/2019/fd/c8fd00106e

[5] https://link.springer.com/article/10.1186/s11671-020-03299-9

[6] https://efabless.com/notice

[7] https://chipfoundry.io/