# OpenRAM

## Multiport Memory IP

H. Nichols, M. Grimes, J. Cirimelli-Low, J. Sowash, M. Guthaus*
*Speaker: mrg@ucsc.edu


https://github.com/VLSIDA/OpenRAM

# Why an Open Source Memory Compiler?

- Many IP cores are available for “free”...
  - but no SRAMs or even models are included.
- Memories are a bottleneck for performance and power…
  - so they are really important.
- Memories have a regular structure which leads to automation…
  - so tool support should be helpful.
- Cell libraries and PDKs exist, but not memory IP…
  - so this could help research new solutions.

$2(2\pi r)$ is twice as round? Let’s stop reinventing the wheel.

# OpenRAM Guiding Principles

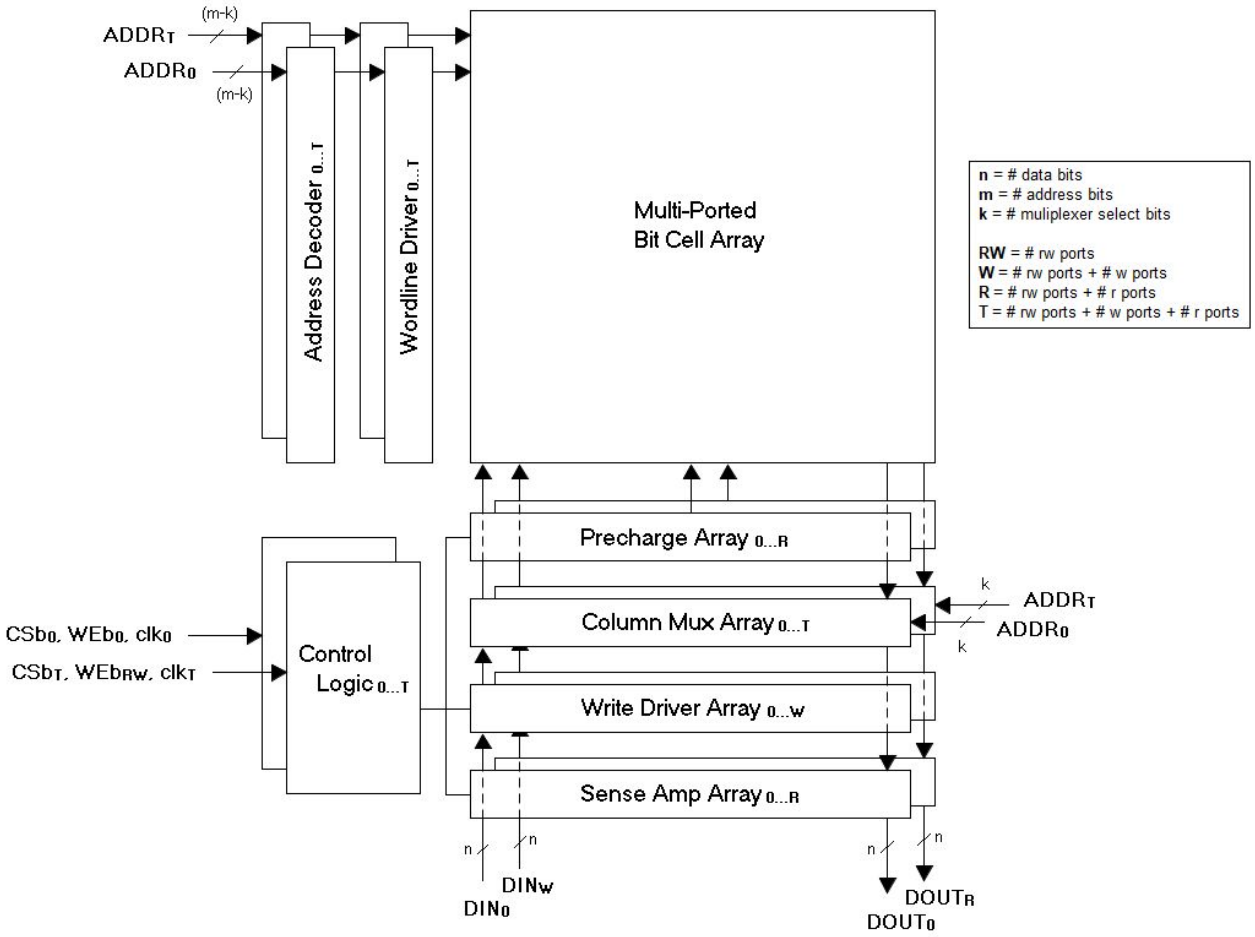
1. Be “extensible” by hardware engineers
2. Be “independent” of technology
3. Be “independent” of specific tools or methodologies
4. Support multiple memory types and configurations

# OpenRAM Features

- Implemented in Python 3.5+
- Licensed with 3-clause BSD
- Provides reference circuit and physical implementations
  - FreePDK45 (generic, non-fabricable 45nm)
  - MOSIS Scalable CMOS (scn4m_subm, TSMC 0.35um)
- Provides a timing/power characterization methodology and functional verification
- Generates GDSII layout data, SPICE netlist, Verilog model, DRC/LVS verification reports and P&R macro view.
- Wrappers for both open-source and commercial tools

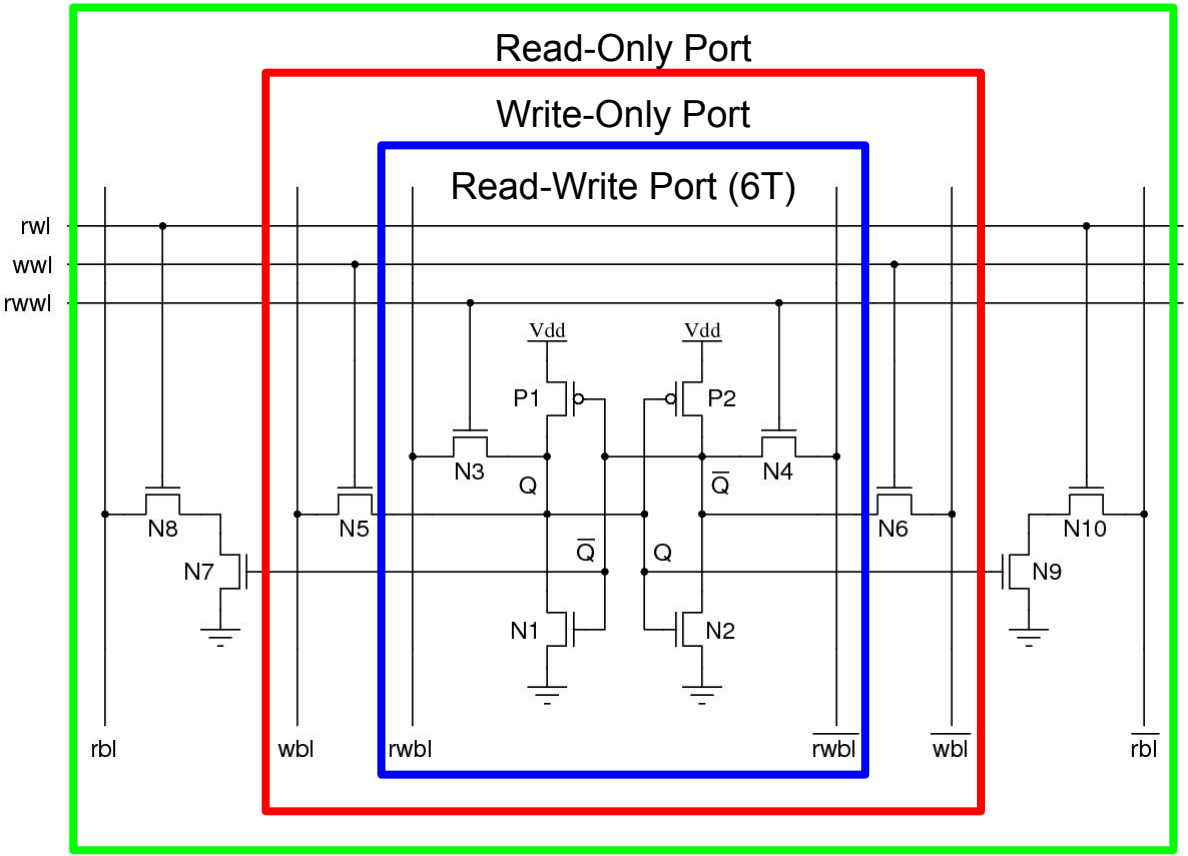
# OpenRAM SRAM Architecture

- Bit cell Array
  - Multi-port bitcells
  - Foundry bitcells
- Each port:
  - Address Decoder(s)
  - Wordline Driver(s)
  - Column Multiplexer(s)
  - Bitline Precharge(s)
  - Sense Amplifier(s)
  - Write Driver(s)
  - **Control Logic with Timing Circuitry**



# Multiport Bitcells

- Based on 6T SRAM cell
  - Standard read-write
  - Isolated read-only
  - Write-only (not sized for reads)
- Can accommodate foundry bitcells



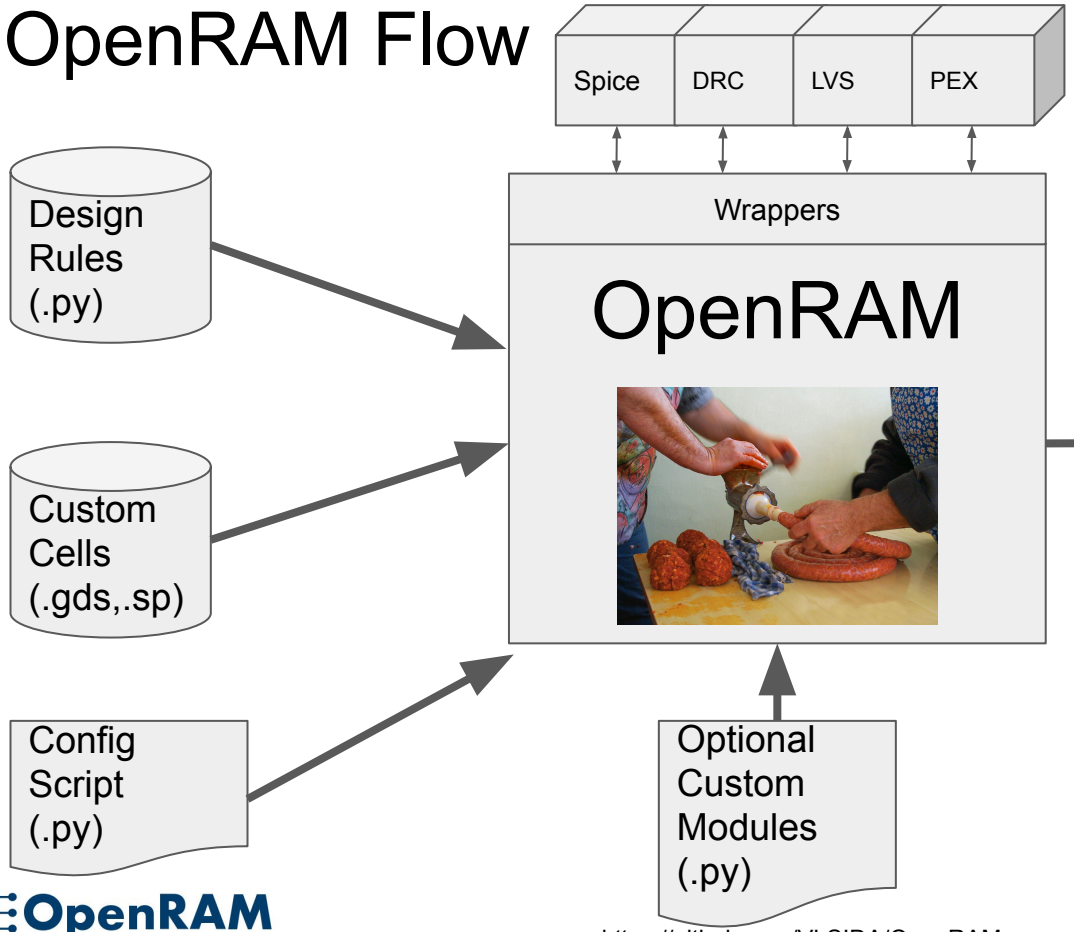
# OpenRAM Implementation

- Front-end mode
  - Generates spice, macro views, timing models
    - Netlist only mode can skip the physical design too
    - Doesn't finish layout (GDS) but provides LEF
  - Doesn't perform DRC/LVS
  - Estimates power/delay analytically
- Back-end mode
  - Generates spice, layout views, timing models
  - Performs DRC/LVS
    - Can perform at each level of hierarchy or at the end
  - Simulates power/delay
    - Can be back-annotated or not

Architects
System Designers

VLSI Designers
Circuit Designers
Memory Designers

# OpenRAM Flow




IP Deliverables

| Type | Description |
|---|---|
| .gds | GDSII layout views |
| .lef | LEF files |
| .log | OpenRAM compile log |
| .v | Verilog simulation models |
| .html | This datasheet |
| .lib | Synthesis models |
| .py | OpenRAM configuration file |
| .sp | SPICE netlists |

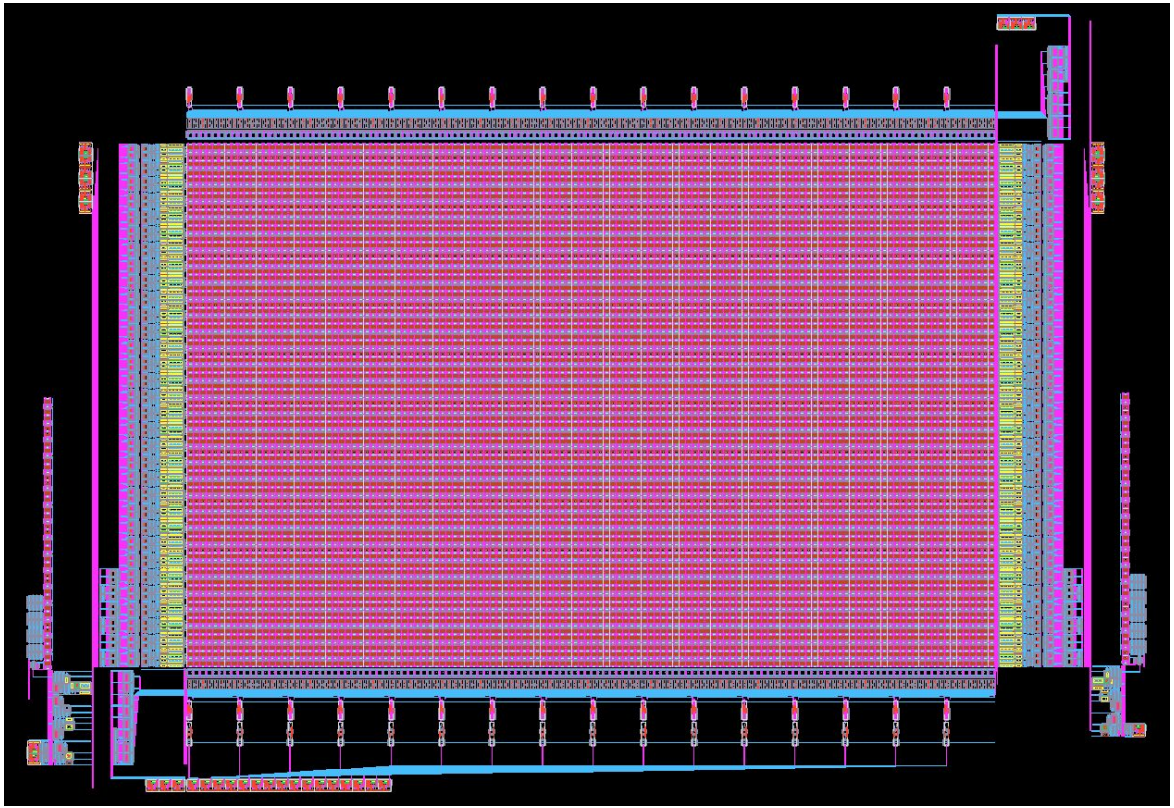
OpenRAM

https://github.com/VLSIDA/OpenRAM

# Regression and Continuous Integration (CI)

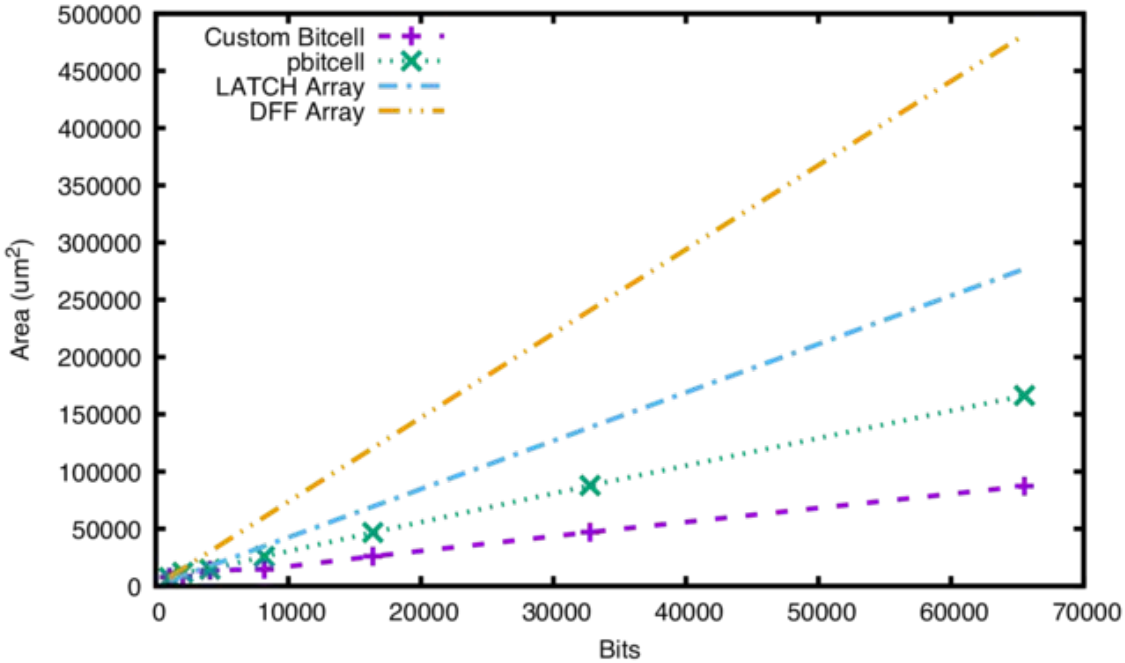
- Uses Python unit test framework.
- Uses coverage.py to measure code coverage.
- Guides users when porting to new technologies.
- Allow users to add features without worrying about breaking functionality.
- Roughly 80% code coverage in about 50 minutes and considers:
  - SRAM, cell and module DRC/LVS
  - Timing simulation
  - Functional simulation
  - Technology portability (FreePDK45 and SCMOS run in parallel)

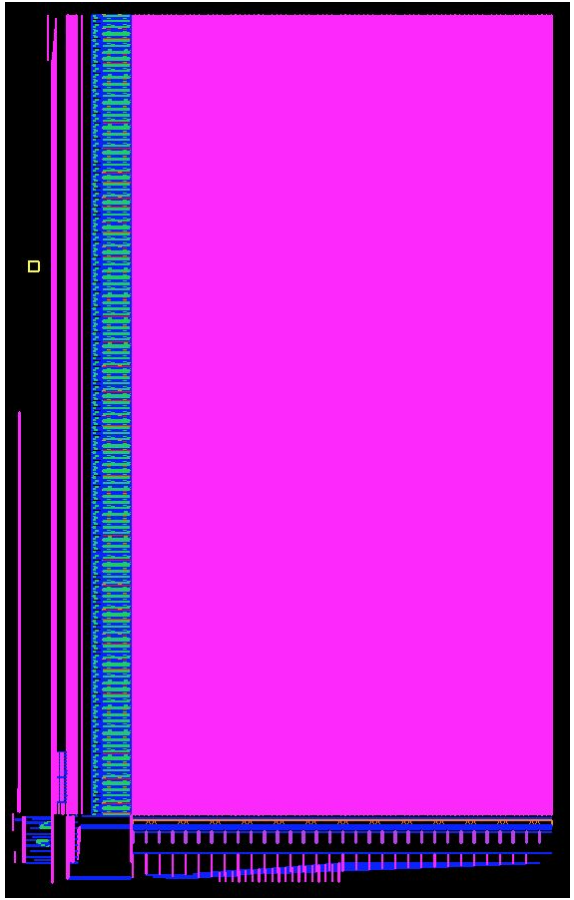
# Small Multi-port Memory (16 x 512, 1rw/1r)



OpenRAM

# FreePDK45 Layout Results*



2048 x 32b x 1rw

*Not compared with other IP because I have not signed NDAs.

# Conclusions

- OpenRAM generates the circuit, functional model, and layout of variable-sized SRAMs.
- OpenRAM is open-sourced, flexible, and portable.
- We are actively porting to different technologies and fabricating designs.
- We are actively developing new features and improvements.
- We are actively seeking feedback and collaborations.
- Are you a company willing to fund open source development?