Enabling Design Technology Co-Optimization of SRAMs through Open-Source Software

Matthew Guthaus, Hunter Nichols, Jesse Cirimelli-Low, Joseph Kunzler, Bin Wu
Computer Science and Engineering, University of California Santa Cruz, Santa Cruz, CA, USA, email: mrg@ucsc.edu

***Abstract*—OpenRAM is an open-source memory compiler infrastructure that can enable Design Technology Co-Optimization of SRAMs. SRAM DTCO is often plagued by limited access to robust, featured memory compilers. In particular, each technology often “reinvents the wheel” whereas using open-source memory compilation can leverage recurring needs of verification, circuit implementation, and tool compatibility to give a true insight from technology all the way to design.**

I. Introduction

In advanced technology nodes, the scaling impact on Static Random Access Memories (SRAM) is one of the most fundamental and challenging issues. SRAM leakage, performance, and density are all of utmost importance and often have conflicting requirements.

Memories, however, are very regular structures and can benefit from automation in a variety of ways. Memory compilers have become an integral part of design flows for fabless chip designers, foundries, and Integrated Device Manufacturers (IDMs). Fabless chip designers typically buy silicon-proven Intellectual Property (IP) memory designs, often packaged as a memory compiler. Foundries and IDMs, on the other hand, design or use compilers tightly coupled to the process to develop customized, high-performance memory macros.

Memory compilers are often a set of technology-specific, proprietary scripts that perform netlist and physical layout generation. Some may also perform timing characterization and create file formats for use in design flows. Commercial memory IP compilers are typically non-mutable and closed source while proprietary scripts are usually not distributed, unsupported, limited to a particular technology, and often lack advanced features or flexibility.

OpenRAM is an open-source memory compiler that is freely available under the BSD 3-clause license [1], [2]. System designers are using OpenRAM in open-source designs [3] and for new architectures to measure the impact on design power, performance and area (PPA) [4], [5]. Device and technology researchers, on the other hand, can utilize OpenRAM to quickly prototype and evaluate the effects of technology. Specifically, they can see system-level results and quickly iterate with designers for even more insight into design technology co-optimization (DTCO).

While memory compilers are not new, open-source memory compilers that are feature rich can enable a new era of tightly coupled SRAM DTCO. We present OpenRAM which creates SRAM designs in a variety of technologies, can be ported to new technologies, and has an array of design, characterization and compatibility features. The rest of this paper discusses the OpenRAM framework and how it can be used for SRAM DTCO.

II. Overview

OpenRAM currently supports three open-source process technologies: FreePDK 45nm [6], Skywater/Google Open-Source PDK 130nm [7], and MOSIS Scalable CMOS (SCMOS) $0.35\mu mo$ [8] and has been used in a range of other proprietary technologies. OpenRAM technology input includes a simplified set of back end of line (BEOL) design rules, a parameterized transistor and associated front end of line (FEOL) design rules, and creation of some custom cells as shown in Fig. 4.

Any cell or module in OpenRAM can be overridden with a custom one. Some custom cells (e.g., bit cells) necessitate the use of other custom cells (e.g. custom decoders, sense amps) due to dimensional requirements or pin alignment.

Some other cells are created through parameterization and generated on-the-fly in OpenRAM. These use technology design rules and are constructed from a parameterized transistor. In general, these cells use restricted, simplified design rules and may not be optimal for area. But since they are infrequently used, the overall impact is insignificant. For example, the control logic is constructed using the parameterized gates. On the other hand, parameterized cells allow dynamic sizing of control drivers, decoders, word line drivers, precharge circuitry, and other components for improved PPA.

Using software engineering best practices, OpenRAM has unit tests to verify each module so the composability of custom cells can be quickly verified and corrected if necessary. Testing is a key component of the OpenRAM framework and aids portability. The custom and parameterized cells are combined hierarchically into arrays, decoders, address ports, data ports, and control logic as shown in Fig. 1. Together, these implement banks and multi-bank structures. The global bit cell array implements an (optional) hierarchical word line using global and local word line drivers as shown in the schematic in Fig. 3.

OpenRAM supports numerous SRAM features including multiple ports, types of ports (read, write, read/write), column muxing, hierarchical decoders, write masking, and replica bit line timing with configurable guard band. Row and column redundancy is also available for software-aided testing and self repair. As with custom cells, any module can be overridden with custom implementations. OpenRAM output, shown in Fig. 4, includes logical models (Spice, LVS, Verilog), physical models (LEF, GDS2), power and timing (Liberty) as well as data sheets (HTML).

OpenRAM integrates with both commercial and open-source design tools. This includes a variety of physical design tools for design rule checking, extraction, verification and logical/circuit simulation. OpenRAM supports spice simulators through standard spice syntax and measure statements. Using estimations of technology parameters, OpenRAM provides analytical models for fast system-level design space exploration without simulation as shown in Fig. 4.

III. SRAM DTCO

Technology affects SRAM design considerations such as manufacturability, reliability, power, performance, and area. Open-source memory compilation can be utilized for DTCO in each of these aspects. In particular, we discuss physical design and circuit design with issues of lithography, noise, timing, and characterization.

Since bit cells are the most significant component of SRAM area, size and manufacturability which are the most important design consideration. SRAM arrays typically have periodic strap cells (substrate ties), local word line buffers, and row and column end caps. These cells and the SRAM bit cells often include OPC and design rule waivers. OpenRAM has a default array or a customized array can be implemented. OpenRAM supports a parameterized bit cell that can be generated using user design rules, but this obviously is inferior to a custom cell and is only suggested for quick prototyping.

Reliability due to noise margins and bit line leakage is a significant technology dependent circuit issue. The performance ultimately depends on entire arrays and their implementation. In particular, the (dynamic) read noise margin is of critical concern and depends on device models including leakage and bit line parasitics. The bit line leakage also impacts read reliability and performance. Each of these can be evaluated with both focused timing characterization and pseudo-random functional testing that is available in OpenRAM. Focused timing characterization can examine worst case corners whereas functional testing examines data-dependent cases.

Performance is also highly dependent on technology. In particular, word line and bit line parasitics, bit line leakage, and transistor drive current affect read delays. In addition, word line driver sizing, topology, and placement are dependent on BEOL parasitics and can greatly improve performance [9], [10]. Transistor off current directly affects leakage. All of these can be quickly evaluated with characterization in OpenRAM.

Debug assistance including sense amplifier enable (SAE) timing margin, process, voltage, temperature (PVT) corner failure analysis and so on can greatly improve designer productivity and aid DTCO. Specifically, OpenRAM instruments debugging checks for possible errors. For example, the sense amplifier enable (SAE) timing can affect performance if too slow or cause read failures if too early. SAE timing is difficult to predict across multiple corners and memory configurations. OpenRAM can use simulation to guide design iterations for improved robustness or performance considering the respective technology.

IV. Results

Fig. 2 shows an 8 kilobyte, 2 port (1 RW, 1 R), 32-bit data word, byte-writable SRAM in FreePDK 45nm for a RISC-V microprocessor [11]. The memory uses an 8-way column mux and includes two replica bit lines for 258 columns and 258 cells per bit line (66,564 bits total). The hierarchical word line uses eight 32-bit local arrays with local word line drivers on both sides for each port. The memory is $242,605\mu m^2$ for an effective bit area of $3.64\mu m^2/bit$. For reference, a D Flip-Flop in the same technology is $7.08\mu m^2$.

A similar memory was created for Skywater process which is a hybrid 150nm/130nm node with 5 metal layers and 1.8V nominal supply voltage. That memory is $739,263\mu m^2$ for an effective bit area of $11.1\mu m^2/bit$. For reference, a D-flip flop in the same technology is $41.3\mu m^2/bit$. A smaller 1 kilobyte version is currently being fabricated as part of of the Google/Skywater OpenPDK project [7].

Table I shows the single- and dual-port bit cell areas in FreePDK45 and SCMOS as compared to a custom D Flip-Flop (DFF). The automatically generated parameterized bit cells have roughly 2x area over the custom bit cells but are still roughly 3.5x smaller than a DFF.

Fig. 5 shows a read simulation including the precharge and timing. The timing uses a replica bit line structure for PVT tracking of the bit line delay while the decoder and word line delay is matched with a delay line. The replica bit line uses two pull down cells tied internally to ground to enable the sense amplifier using s_en, but can be altered to utilize either multiple replica lines or more pull-down cells to adjust the replica timing. Overall, the SRAM interface is positive-edge synchronous with a setup/hold times comparable to a flip-flop.

Fig. 6 shows the delays of the word line path and delay line over multiple PVT corners in a 45nm SRAM. The supply voltage was varied by $1V \pm 10\%$, temperature was either 25°C or 125°C, and slow, typical, and fast process corners were used. Since technology affects both the process corners and the voltage dependence, the robustness of technology changes can be quickly assessed. In this case, the delay line has an average 5.2% greater delay than the word line delay, but the delay line clearly tracks the delay of the word line over different corners.

Fig. 7 shows a single bank delay trend in 45nm when we utilize DTCO to insert local word line buffers at optimal locations with optimal sizes as in the circuit from Fig. 3. The algorithm uses dynamic programming [10] in a bottom-up manner while pruning suboptimal sub-solutions. The proposed DTCO approach reduces the bank delay when the word line size is large greater than about 512 bits. It also shows that even distribution of buffers is not optimal for delay, but the exact distribution is technology dependent due to transistor drive and BEOL parasitics.

V. Conclusion

We have introduced OpenRAM and discussed its uses in DTCO along with several examples critical to SRAM scaling.

References

[1] M. R. Guthaus *et al.*, “OpenRAM: An open-source memory compiler,” in *ICCAD*, 2016.
[2] ——, “OpenRAM,” https://github.com/VLSIDA/OpenRAM, 2020.
[3] H. Nichols *et al.*, “Automated synthesis of multi-port memories and control,” in *VLSI-SOC*, 2019.
[4] K. Al-Hawaj *et al.*, “Towards a reconfigurable bit-serial/bit-parallel vector accelerator using in-situ processing-in-sram,” in *ISCAS*, 2020.
[5] A. de Gennaro *et al.*, “Design and implementation of reconfigurable asynchronous pipelines,” *TVLSI*, vol. 28, pp. 1527–1539, March 2020.
[6] J. E. Stine *et al.*, “FreePDK: An open-source variation-aware design kit,” in *MSE*, June 2007, pp. 173–174.
[7] Google/Skywater, “SkyWater SKY130 PDK,” https://skywater-pdk.readthedocs.io.
[8] MOSIS, “MOSIS Scalable CMOS (SCMOS),” https://www.mosis.org.
[9] B. Wu *et al.*, “Fast and area-efficient sram word-line optimization,” in *ISCAS*, 2019.
[10] ——, “Bottom-up approach for high speed SRAM word-line buffer insertion optimization,” in *VLSISOC*, 2019.
[11] C. Wolf, “PicoRV32 - A Size-Optimized RISC-V CPU,” https://github.com/cliffordwolf/picorv32, 2020.

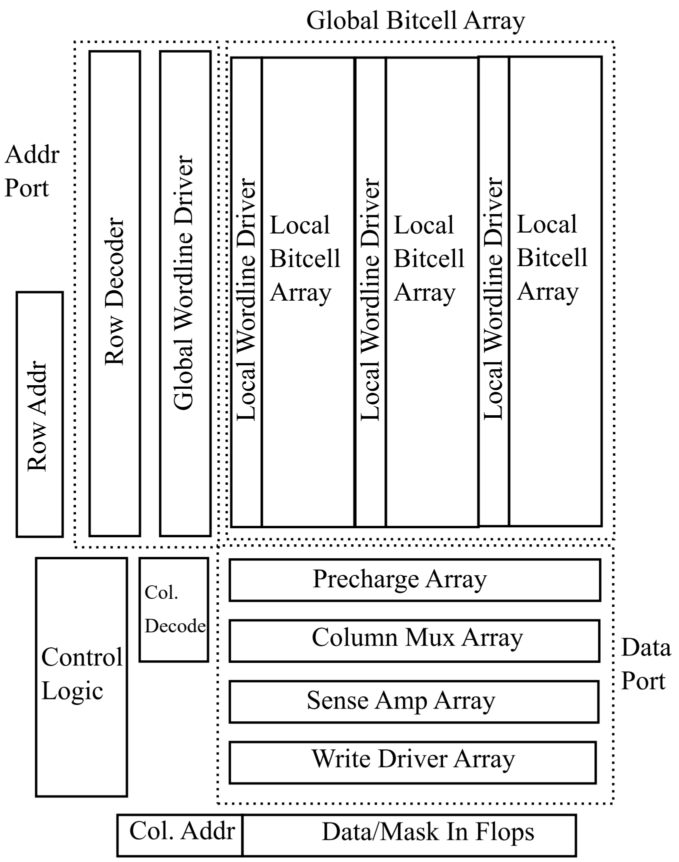

Fig. 1: Block diagram for a single-ported SRAM showing a hierarchical wordline bit cell array along with peripheral circuitry.

	FreePDK45			SCMOS		
	Area	A/Port	A/Port	Area	A/Port	A/Port
	μm^2	μm^2		μm^2	μm^2	
Cust. 6T	0.95	0.95	1.00X	63	63	1.00X
Cust. 1RW/1R	1.97	0.99	1.04X	122	61	0.95X
Cust. DFF	7.08	7.08	7.45X	436	436	6.92X
Auto. 1RW	2.05	2.05	2.16X	134	134	2.12X
Auto. 1RW/1R	3.92	1.96	2.06X	272	136	2.16X

TABLE I: The area of custom bit cells and DFF compared to automatically generated bit cells.

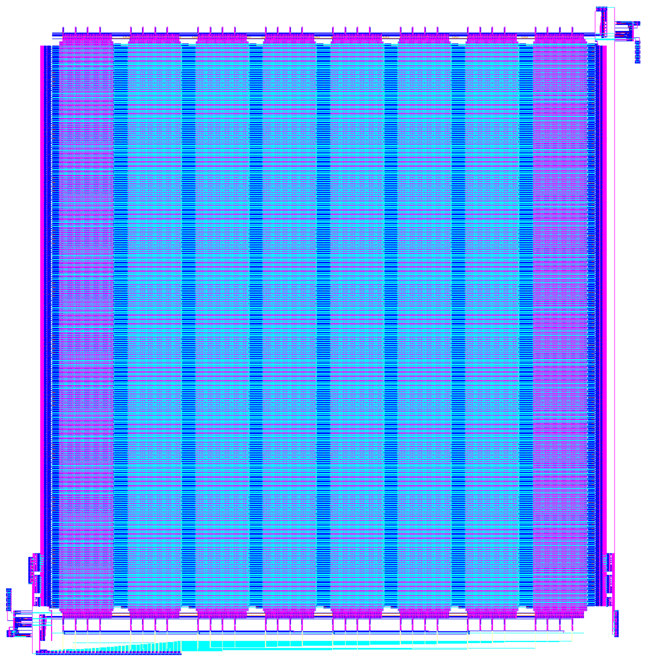

Fig. 2: An 8 kilobyte, 2 port (1 RW, 1R), 32-bit data word, byte-writable SRAM in FreePDK 45nm.

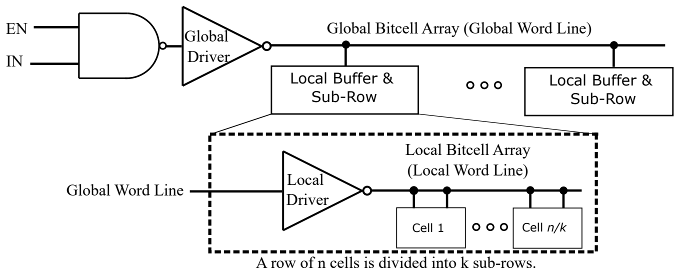
EN
IN
Global Driver
Global Bitcell Array (Global Word Line)
Local Buffer & Sub-Row
Local Buffer & Sub-Row
Global Word Line
Local Driver
Local Bitcell Array (Local Word Line)
Cell 1
Cell n/k
A row of n cells is divided into k sub-rows.

Fig. 3: Hierarchical word lines are technology dependent due to driver sizing and wordline parasitics.

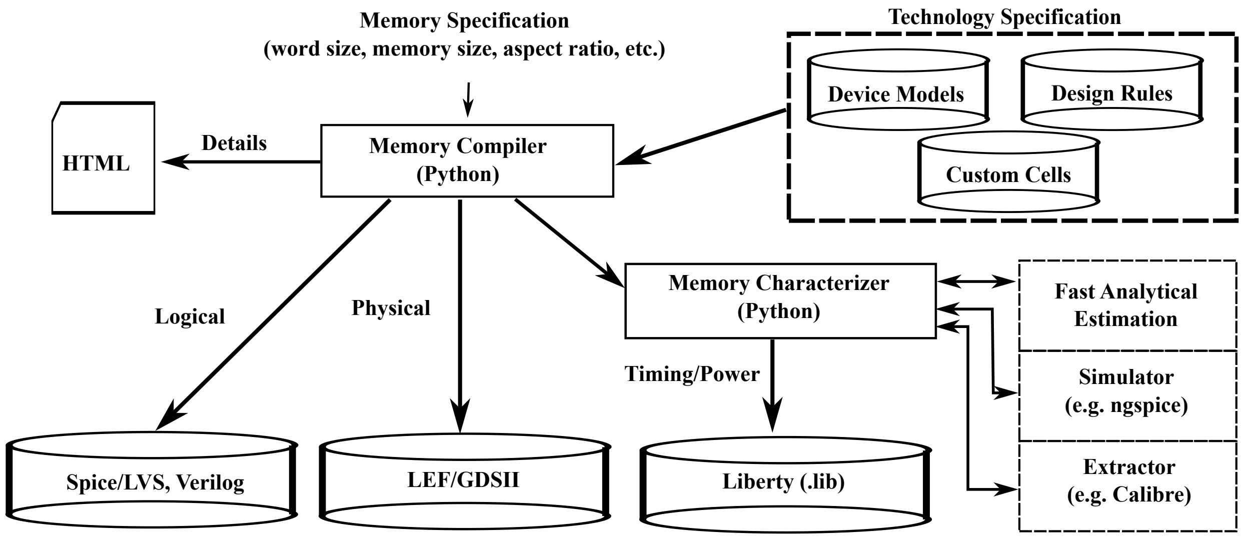

Fig. 4: OpenRAM is an open-source framework that utilizes technology and device data to produce SRAM designs.

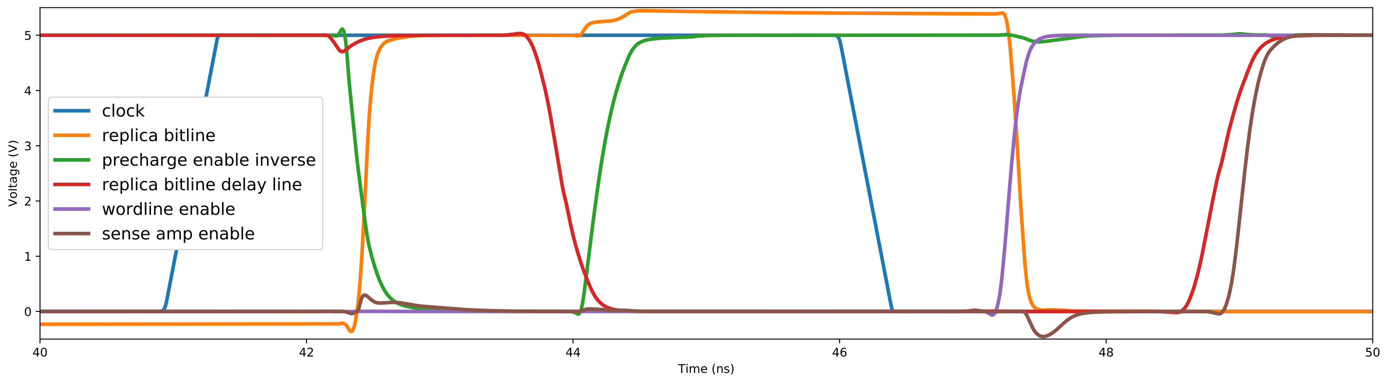

Fig. 5: Simulation excerpt showing replica bit line and sense amplifier timing during a read operation. The bit line precharges on the positive clock half and the sense amplifier is enabled once the inverting replica bit line delay goes high.

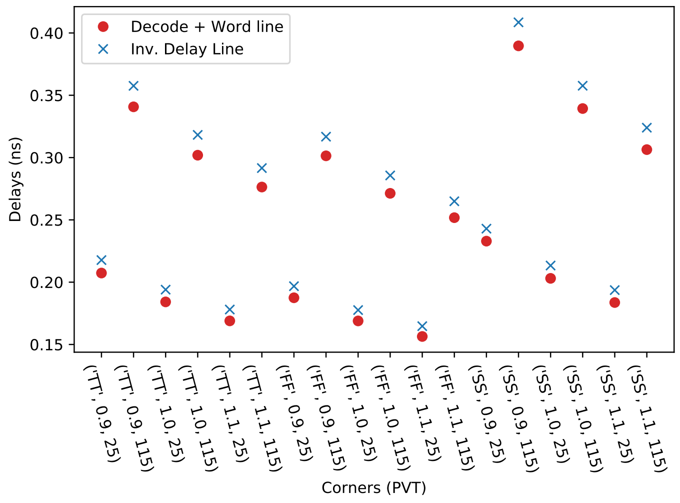

Fig. 6: Word line path and delay line timing are nearly matched with a small offset over different PVT corners.

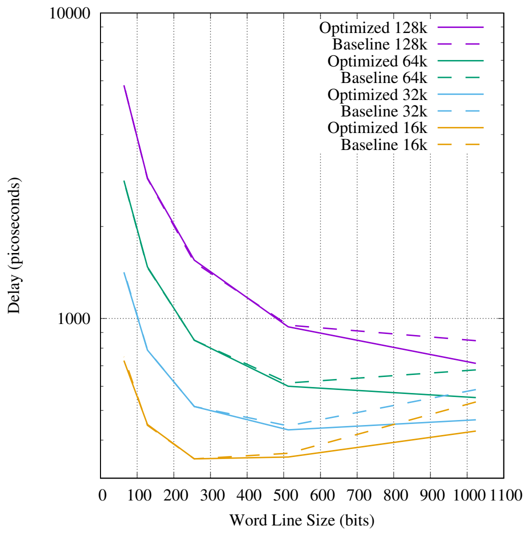

Fig. 7: Hierarchical word line DTCO can reduce bank delay when array width is around 512 bits.