Automated Synthesis of Multi-Port Memories and Control

Hunter Nichols, Michael Grimes, Jennifer Sowash, Jesse Cirimelli-Low, Matthew R. Guthaus
Computer Science and Engineering
University of California Santa Cruz
Santa Cruz, CA 95064
{hznichol,mtgrimes,jsowash,jcirimel,mrg}@ucsc.edu

***Abstract*—High performance systems often employ multi-ported memories to enhance the throughput and flexibility of the memory. Existing SRAM compilers offer limited control over the SRAM design and port configurations while SRAMs are commonly dual-ported. Experimental designs could benefit from design exploration of multi-port configurations. We propose an open-source, multi-port solution that extends the OpenRAM memory compiler. A parameterized bitcell is presented which can support any combination of read, write, and read-write ports. The bitcell layout is generated for these port combinations, and the SRAM layout can support any combination of two ports. In addition, support for multi-port characterization and functional testing ensures correctness and incorporation into design methodologies.**

I. Introduction

The performance of modern multi-core System-on-Chips (SoCs) and processors is not only dictated by the speed of the memory but also the bandwidth. Memories such as register files and Static Random Access Memories (SRAM) are commonly multi-ported to allow for concurrent operations to increase the throughput of the system at the expense of memory area overhead. In addition, memories typically occupy substantial portions of these same designs. Therefore, fast and dense multi-ported memories are necessary in modern systems but have not received significant attention in SRAM compiler implementations.

Multiported memories are distinctly different than register files. Many-ported memories like the 12 read, 8 write register file in the Itanium [1] often do not use sense amplifiers. The complexity of the multi-port cells used in these register files makes them impractical for large memories. Multi-ported SRAMs on the other hand have additional complexities which can affect the performance and the layout requires many specific considerations [2].

While high performance microprocessors often have custom designed SRAMs, ASIC designs often use pre-made memory IP. Since manually designing an SRAM has a high engineering cost, system designers and architects are often prevented from doing broad design space exploration. Memory compilers partially alleviate this problem by automating layout, netlist generation, and characterization. Contemporary SRAM compilers, however, are often limited to a small range of predefined

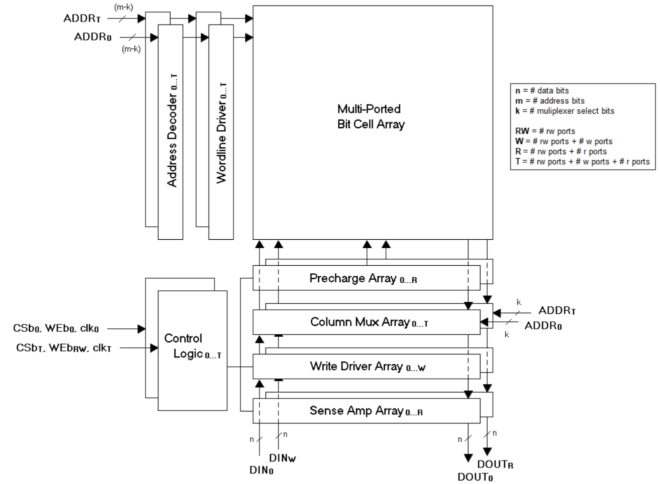

Fig. 1: Block diagram for a multi-ported SRAM showing a shared bit cell array with duplicated instances of peripheral circuitry.

configurations and are usually specific to a given process technology.

The purpose of this work is to augment the OpenRAM [3] memory compiler with multi-port extensions. The contributions presented in this paper are:

- The first open-source multi-port memory compiler.
- A framework for multiport functional testing and characterization.
- Support for custom and synthesized multi-port bitcells.
- A sensitivity matching approach for delay line sizing.

The rest of this paper is organized as follows. We present an overview of multi-port SRAMs in Section II. We discuss design methods including parameterized multi-port bitcell design and the timing model in Section III. We present some results in Section IV and conclusions in Section V.

II. Overview

A. Multi-port Peripheral Circuitry

A multi-ported SRAM allows multiple simultaneous accesses to a shared bitcell array as shown in Figure 1. Single-port peripheral circuitry is duplicated for each read, write, or read/write port. Precharge arrays and sense amplifier arrays are

978-1-7281-3915-9/19/$31.00 ©2019 IEEE

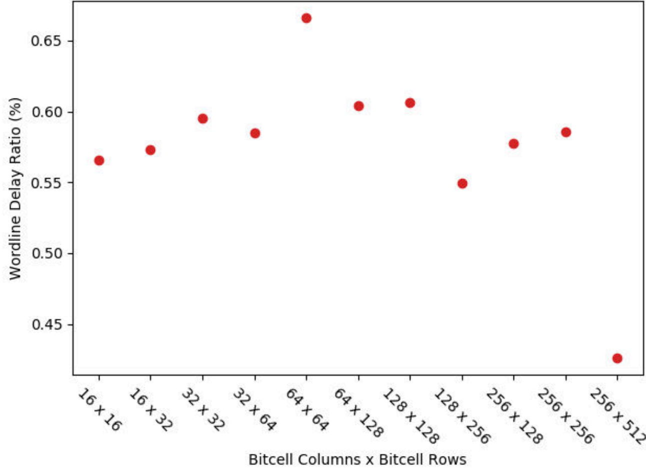

Fig. 2: The wordline path delay as a percentage of the total SRAM delay represents a significant portion for various configurations.

created for read ports while write driver arrays are created for write ports. Column mux arrays, wordline drivers, and address decoders are created for all ports since they are used in both read and write operations.

The control logic for each port type is similar but can be slightly reduced depending on the port operations. For example, a write-only port does not need precharge circuitry or sense amplifiers and therefore doesn't need the associated timing and control circuits. Similarly, a read-only port does not need write drivers.

The control logic also determines the timing of the peripheral circuitry. The sense amplifiers (SA) are used in read operations to amplify the difference between the bitlines in order to reduce the delay of a single bitcell discharging the entire bitline. However, improper timing of the SAs can lead to read failures. This issue becomes worse when considering process, voltage, and temperature (PVT) corners which requires careful sizing of delay lines and the replica bitline (RBL) [4]. The RBL makes the read timing more robust against variation and has had ubiquitous adoption into SRAMs with differential bitlines.

The SRAM read timing is made up of logic and bitline delays which is mirrored by a delay line and RBL to generate the SA enable (SAE) signal. The RBL is sized to match the bitline to output path delay and is determined by the bitline discharge, bitcell array height, and other peripheral circuitry such as the column mux and SA. The wordline path delay is comprised of the control logic delays and the wordline delay which is replicated using a delay line. The wordline path delay can represent a significant percentage of the delay depending on the SRAM configuration and size as shown in Figure 2. The delay is typically replicated with a chain of inverters while matching the rise and fall delays in the wordline path.

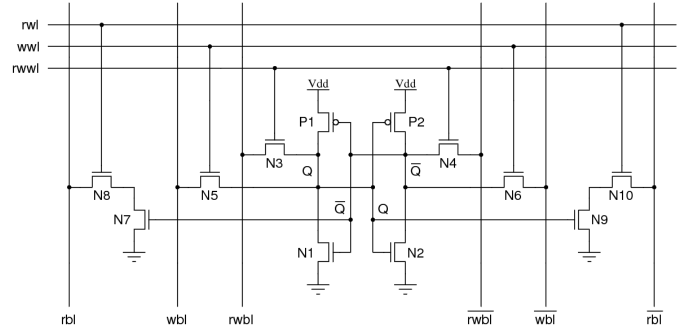

Fig. 3: A multi-ported SRAM cell has a single inverter pair (N1, P1, N2, P2) for storing a bit with a read/write port (N3, N4), write-only port (N5, N6), and read port (N7, N8, N9, N10) to share access to the differential storage nodes.

B. Multi-port Bitcells

A multi-port bitcell is shown in Figure 3 with one read/write, one read-only, and one write-only port. The SRAM cell has a single inverter pair (N1, P1, N2, P2) for implementing the storage node inverter pair. The read/write and write ports are implemented with the N3,N4 and N5,N6 access transistors, respectively. A decoupled read-only port is implemented with an access transistor along with one additional transistor per bitline N7, N8, N9, N10. All ports share access to the differential storage nodes Q and $\overline{Q}$ as shown. Each port also has its own wordline: $rwwl$, rwl, or wwl and pair of differential bitlines: $rwbl$ ($\overline{rwbl}$), wbl ($\overline{wbl}$), or rbl ($\overline{rbl}$).

Read/Write ports are capable of performing both read and write operations. Read/write ports are synonymous with the access transistors in a 6T bitcell (N3 and N4 in Figure 3), i.e. an SRAM with a single read/write port simply uses the 6T bitcell. Adding read/write ports requires additional connections to the cross coupled inverter storage. This storage node can be accessed simultaneously from multiple read ports. The bitlines of these ports are connected to write drivers, precharges, column mux, and sense amplifiers.

Only one port will be writing a bitcell on any given cycle. The write Static Noise Margin (SNM) ensures that the access transistor is able to over-write the bitcell contents sufficiently to store a new value [5], [6]. This is often formulated as the PR ratio where

$$PR = \frac{W_{P1}/L_{P1}}{W_{N3}/L_{N3}} = \frac{W_{P2}/L_{P2}}{W_{N4}/L_{N4}} \tag{1}$$

according to Figure 3. Since a single access transistor can perform a write, this means that the size relationship between the access transistor and the inverter PMOS remains the same with additional read/write ports.

Multiple read operations, on the other hand, can done simultaneously on a single bitcell. The read SNM is affected by the ratio of the NMOS pull-down transistor to the PMOS access transistor [5], [6] which is often formalized as the CR ratio (often called β) where

$$CR = \frac{W_{N1}/L_{N1}}{W_{N3}/L_{N3}} = \frac{W_{N2}/L_{N2}}{W_{N4}/L_{N4}} \tag{2}$$

according to Figure 3. For a stable read with multiple read/write ports, the NMOS pull-down must sink more charge than the combined currents of multiple simultaneous access transistors.

Transistor sizes for the multi-port design can then be derived from the baseline transistor sizes of a single-port 6T-cell assuming it already has satisfactory read and write SNMs. The required size of the NMOS is therefore directly proportional to the number of read/write ports. Since the access transistor and the pull-up transistor do not change with the number of ports, their sizes are set to the same values as the 6T-cell.

Write ports are only capable of performing write operations and are nearly identical to the read/write ports as shown in Figure 3. Two access transistors (N5 and N6) are controlled by a wordline (wwl), with bitlines (wbl and $\overline{wbl}$) and an output from the pair of cross couple inverters connected to each source and drain respectively. Write ports and read/write ports also differ in their associated peripheral circuitry. Since these ports are only performing write operations, the bitlines only need connections to the column mux and a set of write drivers.

The write access transistors are sized the same as the read/write access transistors in order to satisfy the write SNM, but they do not need to perform reads and therefore need not satisfy the read SNM. Since the size of the NMOS of the inverters is contingent only on read operations, they are not affected by the number of write ports.

Read ports are only capable of performing read operations with a different circuit from read/write and write ports. Instead of a single pass transistor connecting the bitline and the storage component of the cell, read ports use a second access transistor to isolate the port from the storage node. The isolation transistors (N7,N9 in Figure 3) have their gate terminal driven by the output from the cross coupled inverter while the other transistors (N8,N10) are the access transistors [7]. Since these ports are only performing read operations, the bitlines only need connections to precharges, column mux, and sense amplifiers, and can forgo connections to write drivers.

The process of performing a read operation is generally the same as that of a read/write port except that the read transistors (N7,N9) do not have a significant effect on the read SNM and therefore does not affect the transistor sizing of N1,N2. Since the read is isolated, the read transistors (N7,N9) and read-access transistors (N8,10) have a size that is independent of the number of ports while the inverters' NMOS is also sized sufficiently to drive the read transistor gates [8].

III. Multi-Port Implementation

The implementation of multi-port SRAMs in OpenRAM required several challenges in the layout, netlist, and timing. To make an SRAM with configurable ports, we created a bitcell with configurable ports considering the sizing in Section II-B. The layout allows any number of ports on a single bitcell using user-design rules. We designed the cell with the intent that it can easily be replaced with a handmade or foundry cell. The peripheral circuitry from Section II-A is also generated to support the bitcell port configurations. The top-level layout is presently limited to any two port combination since additional ports would require more than four metal layers. Additionally, we created automated functional tests and updated the characterization to support multi-port implementation.

A. Parameterized Bit Cell Layout

Bitcell layouts are typically manually designed with area and manufacturability as the primary concerns. The lack of modularity in the layout, however, hampers port scalability by requiring a unique optimized cell for each configuration. Foundry bitcells achieve even greater density by waving some DRC rules but makes layout particular to that technology node. These layout techniques cannot be extended for multi-port configurations across multiple process technologies. Instead, we have taken a parameterized cell approach that performs regular transistor placement utilizing each technology's design rule set to enable fast exploration and prototyping.

In our approach, we have created a very simple regular structure using restrictive design rules. Possibly the most important rule, in this regard, is the orientation of poly-silicon (poly). It is permissible in older CMOS technologies to route poly either horizontally or vertically, but double-patterning lithography used by newer technologies has restricted the orientation of poly to the vertical direction [9]. To accommodate newer nodes, all transistors in our approach are oriented in a single direction. While this may increase area, it allows the design to be easily applied to many technology nodes.

To accommodate shifting design rules, our parameterized bitcell dynamically places components based on relative constraints similar to layout compaction. Two adjacent write ports, as in Figure 4, for example, have three design rule constraints: active to active spacing between the two access transistors, metal1 to metal1 spacing between two contacts, and metal2 to metal2 spacing between the wordline routing and a bitline. The required spacing is the minimum distance that satisfies each of the present constraints, and is dynamically assigned based on process design rules.

To achieve the modularity for any combination of ports, two metrics are relevant for the regular placement of the ports: the width and spacing of access transistors. The bitlines for each port are placed at the outermost edge of the access transistors. All ports require routing to the storage component of the cell while read ports require additional routing to *gnd*. This resolves the tiling into three types: read ports, write ports (read/write and write ports have identical layouts on the bitcell level), and any port with port to inverter spacing. Figure 5 displays a multi-port cell which follows these rules.

B. Multi-Port Peripheral Circuitry

OpenRAM limits itself to three layers of metal for all signal routing and four levels for power distribution. Four metal layers restricts how the peripheral circuitry can be placed and routed which creates a two port limitation for physical layout. However, netlists without layout can be generated with any number of ports and can be used for simulation.

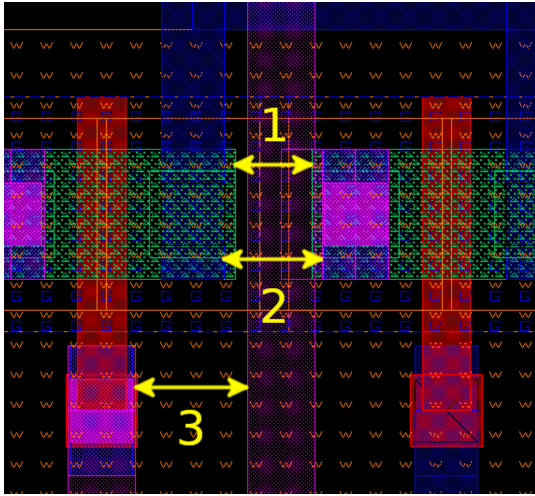

Fig. 4: (1) active-active spacing, (2) metal1-metal1 spacing, (3) metal2-metal2 spacing.

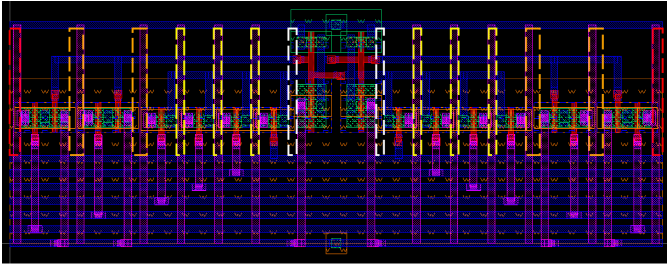

Fig. 5: Regular spacing allows any combination of ports to be placed in layout - (White) inverter to port spacing, (Yellow) write port spacing, (Orange) read port spacing, (Red) bitcell to bitcell spacing

The SRAM netlist duplicates peripheral circuitry based on the multi-port configuration. The current design supports single 1RW configuration along with all two port configurations: 1RW/1R, 1RW/1W, and 1W/1R. In a single-port SRAM, the row address decoder is placed to the left of the bitcell array while the rest of the peripheral circuitry is placed underneath it. The dual-port arrays place the second decoder to the right of the bitcell array with its peripheral circuitry on the top of the bitcell array. Because of the peripheral placement, 1R/1RW, 1W/1RW, and 1R/1W are symmetric but different configurations.

C. Control Logic and Sense Enable Timing

OpenRAM uses standard external control signals including chip select (csb), write enable (web) and a clock (clk) for each port. In addition, data, address, and control values are stored on the positive clock edge. The external control signals are used to generate internal control signals for the decoding, writing, precharging, and sensing, depending on the operation. The control logic dynamically sizes drivers for all control signals based on the SRAM size.

The most critical signal among these internal signals is the sense amplifier enable (SAE) signal which must be late enough to allow the bitcell to discharge the bitlines for reading but waiting too long will sacrifice performance. The control logic generates the SAE signal by delaying the clock through a delay line and a replica bitline. The delay line represents the decode and wordline delay while the replica bitline is sized based on the desired bitline swing needed for correct sensing. For our design, we utilize an analytical model to adjust the delay line based on the critical read path of the SRAM.

The critical read path of the SRAM is the wordline path delay and the bitline delay. The wordline path delay is comprised of the control logic delay to the wordline drivers and the wordline delay on the selected row. The bitline delay is the delay it takes to discharge the bitlines through the selected bitcell. The delay line in the control logic is sized to match the wordline delay and the RBL is sized to match to match the desired bitline swing. Matching the falling and rising delay is also important to better match the variations of the two paths.

The delays of the decoder paths are estimated analytically [10] and compared. The delay line is resized to match the relative delay of the wordline. To further improve the delay estimation, rise and fall delays are separately calculated and used in the delay line sizing to better match process variations. The delay line can have different delays based on the fanout generated per stage.

The bitline delay is entirely timed by the RBL structure. The RBL is made of the the same bitcells used to make the array and a replica bitcell (RBC). The bitcells mimic the bitline load in the array while the RBC is forced to store a logic ‘0’ and discharges the bitline [4]. The output is connected to an inverter which enables the SAs when the bitline was drained past its switching point. The height of the RBL determines the desired bitline swing in the bitcell array which is set large enough to overcome the input offset of the SAs. The bitline delay is not only due to the combined capacitances of the access transistor discharges but also from the column mux which is incorporated into the model by increasing the matching bitline capacitance.

D. Delay Line Design

One method of creating a delay line is to cascade minimum-sized inverters until a desired delay is reached. This is often done iteratively to find the best match as the delays of the wordline can be difficult to predict. The delay line is sized for the worst case delay of the wordline over different variations to prevent read failures which increases power and access time. Cascading inverters is a simple method to add delay but poorly tracks the wordline delay over process, voltage, and temperature.

The delay of the wordline path contains two stages with a large fanout: the wordline enable (WLEN) which enables the row decoder output and the wordline which activates a bitcell row. The wordline must output a logic ‘1’ and has the same polarity as the WLEN. Therefore, both are dependent on the pull-up network and variations that effect the PMOS transistors can have a large effect on these stages.

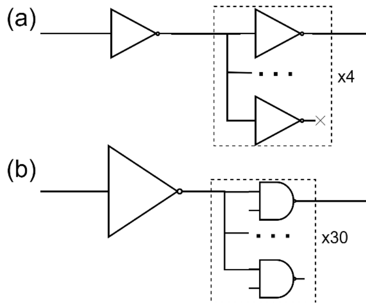

Fig. 6: Stage delays matched using similar size-fanout ratios with (a) using minimum sized inverters and dummy loads to match (b) the example wordline enable NAND fanout.

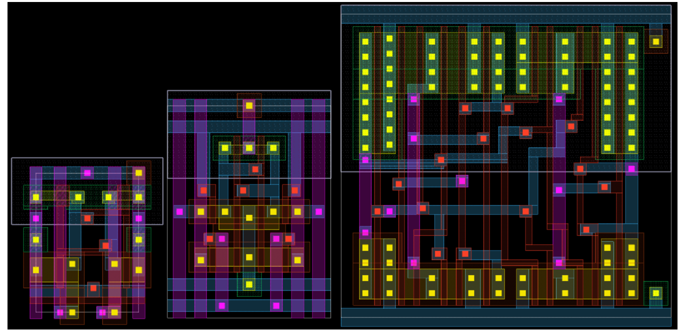

Fig. 7: In SCMOS, the area of the 1RW (6T) and 1RW/1R (10T) is significantly smaller than a DFF [13].

The wordline and WLEN drivers are automatically sized (transistor widths and stages) based on the loads being driven. Such an implementation can easily be replicated by an inverter based delay line, but replicating the high fanout cannot be done by simply adding more stages to the delay line. The delay line must be increased by two stages at a time to maintain a constant output polarity; adding two stages will add a rising delay from a pull-up network in one inverter and a falling delay from a pull-down network in another inverter. Delays added in the delay line will be dependent on both NMOS and PMOS process variations while the high fanout stages are only dependent on PMOS.

There are several ways to change the delay of an inverter delay line without increasing the number of stages such as adding fanout to stages using dummy gates and changing the size of individual stages. Fanout can increase the delay of a single stage by adding more capacitance to be charged/discharged to selectively match the fanout of the wordline path. Increasing fanout can better discretize the delay over adding stages but the delay can be non-linear with the fanout. Increasing the size of a single stage will decrease the delay of a single stage but increase the output load of the previous stage.

The best approach to applying these variable techniques is to make the delay as similar as possible to the wordline path by matching the stages, fanouts, and sizes. This method will provide the closest matching delay but will increase area and power of the SRAM. Rather than total replication, the delays per stage can be replicated with similar a fanout-size ratio to preserve sensitivity to process, voltage, and temperature and therefore improve overall robustness.

For example as shown in Figure 6, the WLEN may have a fanout of thirty 2-input NAND gates driven by a 10x larger than a minimally sized inverter. The ratio between size and fanout can be matched rather than replicating the exact fanout and size. The delay line could use 1x inverter and add a fanout of four inverters to match the relative fanout and size. Logical effort would estimate both have the same delay assuming the difference in parasitics is negligible.

This is an application of a simple linear model to match the delays in the wordline without needing to match the exact fanout and sizes in the wordline path. This method will maintain better rise/fall matching that inverter chains while maintaining comparable delays to a replica structure. Non-linear delays will cause the model to have some inaccuracy when estimating the delay line parameters, so equivalently, the delay matching can be done with characterized data and use similar delays between different sized cells which will be more accurate.

IV. Results

A. Experimental Setup

OpenRAM currently supports two process technologies: the FreePDK generic 45nm process [11] and the MOSIS Scalable CMOS (SCMOS) [12] 350nm process. All SRAMs are automatically designed using user-specified values for the data width, number of data words, and the number of ports and port types. The SRAMs are DRC clean and pass LVS.

All delay measurements were performed with HSPICE on a single SRAM with a bitcell array of 128 rows and 256 columns. SRAMs were generated by OpenRAM using a single port 6T bitcell. The simulations were done using FreePDK45 models.

B. Bitcell Area

The custom bitcells for OpenRAM are 1RW ($63\mu m^2$), 1RW/1R ($151\mu m^2$) in the MOSIS SCMOS process. Compared to the OSU Standard Cell DFF ($436\mu m^2$), these are substantially smaller as can be seen in Figure 7.

The automated layout and modularity of the parametrized bitcells results in a trade-off of higher area overhead than handmade bitcells but allows fast prototyping of many-port options. We generated bitcells of up to 5 ports and measured the area per port. In addition, we compare 1RW and 1RW/1R custom bitcells with the generated bitcells in Table I. The generated bitcell has an average area approximately 2 times greater than that of the handmade 6T cell, but it is still substantially smaller than a DFF.

C. Sense Amplifier Enable Timing

This section shows the delay tracking of the stage-delay matched inverter chain against the wordline path delay. The stage-delay matching chain is an inverter delay line which uses the same number of stages as the wordline path and

	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 bitcells and DFF compared to automatically generated bitcells.

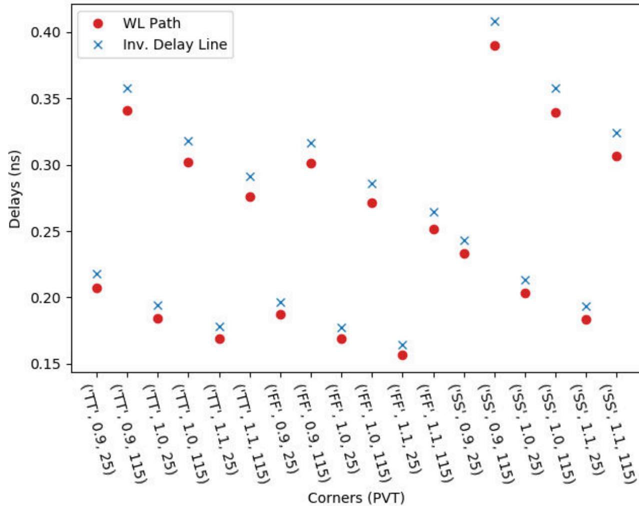

Fig. 8: Wordline path and delay line timing are nearly matched with a small offset over different PVT corners.

partially replicates the high fanout stages. The delay-per-fanout is determined then a number of dummy fanouts are added per stage to match the delay. Initial required fanouts were estimated with an analytical delay model and then tuned to more closely match the stage delays. The wordline path WLEN stage has a fanout of of 128 2-inputs NANDs driven by a 10x inverter. This is matched in the delay line with 20 dummy fanouts driven by a 1x inverter.

Figure 8 shows the delays of the wordline path and delay line over multiple PVT corners. 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. The delay line has an average 5.2% greater delay than the wordline delay, but the delay line clearly tracks the delay of the wordline over different corners. The percentage difference is due to errors in delay granularity that can be added to the delay line. Scaling by their minimum and maximum removes constant differences from the delay line and results in only a 0.9% difference. The stage-delay matched inverter chain represents a robust and automated method to match the wordline delay within OpenRAM.

V. Conclusion

This paper described the extension of a single-port memory compiler into multi-port by using an automated layout and netlist generation. By remaining conservative restrictive design

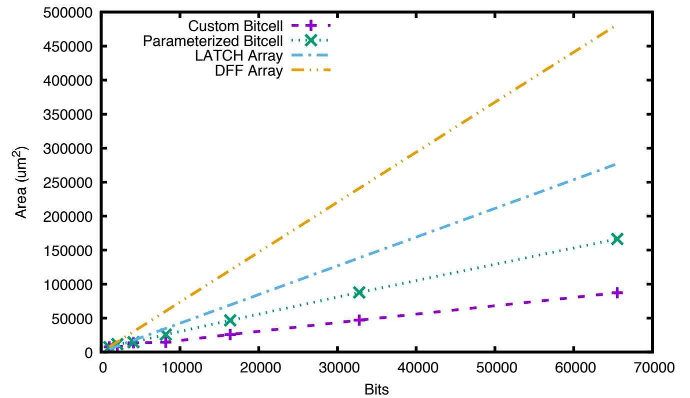

Fig. 9: FreePDK 45nm area comparison of 32-bit word memories of varying sizes shows both the custom and parameterized bitcell are more efficient than latch or DFF arrays.

rules, we maintained the principles set by OpenRAM to maintain technology independence. The layout is automated in FreePDK45 and SCMOS, with the ability to adapt to any process technology. Regular placement of ports and resizing transistors to satisfy read and write stability allow for any configuration of read/write, write, and read ports. We achieved layout for this design on the bitcell and bitcell array level, and achieved netlist generation at the top level. This design achieves a high level of modularity, but at the cost of area. A test designed to simulate random memory usage verified the functionality of numerous SRAM configurations. This work also implemented variable automatic read timing based general SRAM configurations.

References

[1] E. S. Fetzer *et al.*, "A fully bypassed six-issue integer datapath and register file on the Itanium-2 microprocessor," *JSSC*, vol. 37, no. 11, pp. 1433–1440, Nov 2002.
[2] S. Ataei, M. Gaalswyk, and J. E. Stine, "A high performance multi-port SRAM for low voltage shared memory systems in 32 nm CMOS," in *MWSCAS*, Aug 2017, pp. 1236–1239.
[3] M. R. Guthaus *et al.*, "OpenRAM: An open-source memory compiler," in *ICCAD*. New York, NY, USA: ACM, 2016, pp. 93:1–93:6.
[4] B. S. Amrutur and M. A. Horowitz, "A replica technique for wordline and sense control in low-power SRAM's," *JSSC*, vol. 33, no. 8, pp. 1208–1219, Aug 1998.
[5] S. Kim and M. Guthaus, "SNM-aware power reduction and reliability improvement in 45nm SRAMs," in *VLSISOC*, 2011.
[6] J. Rabaey, A. Chandrakasan, and B. Nikolic, "Digital integrated circuits: A Design Perspective. Second Edition," *Prentice Hall*, Jan 2003.
[7] H. Noguchi *et al.*, "Which is the best dual-port SRAM in 45-nm process technology? – 8T, 10T single end, and 10T differential –," in *ICICDT*, June 2008, pp. 55–58.
[8] T. Song *et al.*, "Fully-gated ground 10T-SRAM bitcell in 45 nm SOI technology," *Electronics Letters*, vol. 46, pp. 515–516(1), April 2010.
[9] R. S. Ghaida, G. Torres, and P. Gupta, "Single-mask double-patterning lithography for reduced cost and improved overlay control," *IEEE Transactions on Semiconductor Manufacturing*, vol. 24, no. 1, pp. 93–103, Feb 2011.
[10] I. E. Sutherland, R. F. Sproull, and D. F. Harris, *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann, 1999.
[11] J. E. Stine *et al.*, "FreePDK: An open-source variation-aware design kit," in *MSE*, June 2007, pp. 173–174.
[12] MOSIS, "MOSIS Scalable CMOS (SCMOS)," https://www.mosis.org.
[13] J. Stine, "OSU Standard Cell Library," https://vlsiarch.ecen.okstate.edu/flows/, 2017.