Title
Fast and Area-Efficient SRAM Word-Line Optimization

Permalink
https://escholarship.org/uc/item/98s4c1hp

ISBN
978-1-7281-0397-6

Authors
Wu, Bin
Stine, James E
Guthaus, Matthew R

Publication Date
2019-05-29

DOI
10.1109/iscas.2019.8702518

Peer reviewed

Fast and Area-Efficient SRAM Word-Line Optimization

Bin Wu
Computer Science and Engineering
University of California Santa Cruz
Santa Cruz, CA 95064
bwu8@ucsc.edu

James E. Stine
Electrical and Computer Engineering
Oklahoma State University
Stillwater, OK 74078
james.stine@okstate.edu

Matthew R. Guthaus
Computer Science and Engineering
University of California Santa Cruz
Santa Cruz, CA 95064
mrg@ucsc.edu

Abstract—**A word line driver controls the access of cells in a row in Static Random Access Memories (SRAMs) and has a significant impact on SRAM speed and power consumption. When gate delay is the dominant factor, simple models are a good guideline for fast word lines. However, routing wire delay is significant when the row size is large, which causes these designs to be suboptimal. This paper presents an analytical optimization technique using a delay model that includes gate delay, wire resistance, and wire capacitance to optimize high-performance word line driver topologies for SRAMs. The proposed methodology has a maximum 45% delay improvement and 42% buffer cost reduction.**

I. Introduction

In modern SRAMs, a word line controls the access of cells in a row and has a significant impact on SRAM speed and power. This paper focuses on detailed optimization of word line driver sizes and topologies in order to investigate the optimal word line driver configuration for a given memory size and technology. Such techniques can be included in memory compilers for high-performance memory variants [1].

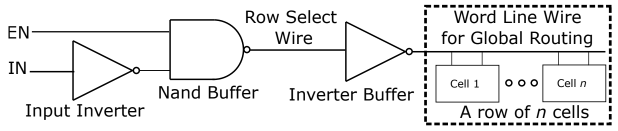


Fig. 1. Conventional memories use a single buffer topology in which the word line wire delay increases quickly as the word line size increases.

A conventional word line driver using a single buffer topology is shown in Figure 1. The driver has a decode input (IN) and an enable (EN) to access a single row after decoding is complete. The NAND gate is typically used with a timed enable signal to ensure that the word line is enabled after the bit lines are precharged and the address is decoded. The delay of the word line has five parts: the input inverter delay, the NAND buffer delay, the row select wire delay, the inverter buffer gate delay, and the word line wire delay.

The power of the single buffer topology increases dramatically for large row sizes. To address this, hierarchical divided word lines (DWL) modify the word line topology so that portions of the word line that are not accessed can be disabled using a select to save power [2]. This is shown in Figure 2 where an additional sub-row select signal is used to logically divide the word line.

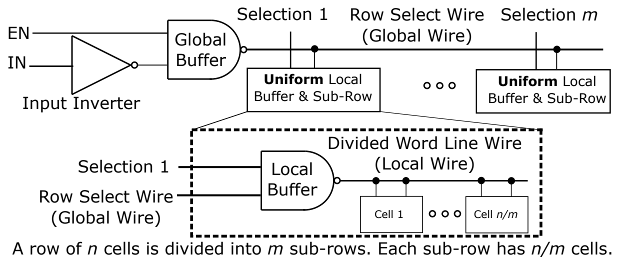

Selection 1
Row Select Wire (Global Wire)
Selection m
EN
IN
Global Buffer
Input Inverter
Uniform Local Buffer & Sub-Row
Uniform Local Buffer & Sub-Row
Divided Word Line Wire (Local Wire)
Selection 1
Row Select Wire (Global Wire)
Local Buffer
Cell 1
Cell n/m
A row of n cells is divided into m sub-rows. Each sub-row has n/m cells.

Fig. 2. DWL saves power by disabling unused portions of a row.

It is well known that wire delays grow quadratically as the length increases. A DWL approach, though created to save power, can be used to improve the speed of long word lines much like the repeater insertion in global interconnects [3], [4]. In addition, the grouping of cells into sub-rows and the size of the local buffers can be further exploited to improve the performance of all memory paths simultaneously.

While bit line delay is significant, word line delay must also be optimized in a well-balanced SRAM design. Large memory array designs are often split into multiple sub-arrays in both the bit line direction [5]–[8] and the word line direction [2], [9]–[13] for overall delay reduction. Interconnect delay has a large impact on the delay of the word-line while the bit-line delay is predominantly due to the weak bit cell drive and large capacitance. The impact of word line interconnect delay becomes more significant as technology scales down, since word lines are often longer than bit lines. Our simulations show that word line delay can be a significant contributor to delay in high-speed SRAMs.

This paper proposes both uniform and non-uniform divided word lines for SRAM performance improvement. Specifically, our proposed optimization uses analytical models that include both gate and wire delays to improve the speed while considering power. Results are verified with SPICE simulation.

In summary, this paper presents:

- The first use of uniform divided word lines for speed improvement as opposed to power savings.
- The first non-uniform divided word lines with higher accuracy gate and interconnect delay models.

- Comparisons of the previous approaches with conventional single driver word line SRAMs.

The rest of this paper is organized as follows: Section II explains the overview of the new idea, Section III explains a simple uniform DWL optimization, Section IV explains our non-uniform DWL optimization methodology details, Section V summarizes the experimental setup, and Section VI analyzes the results and makes conclusions.

II. Overview

The single buffer word line topology speed is sensitive to the row size due to long word line wires. A long word line has resistance that, when combined with the large fanout capacitance of the row, causes delay comparable to gate delays. The word line wire fanout has high capacitance compared to a simple wire since it connects all the 6T cells and thus creates a critical path sensitive to the array width.

The capacitance attached to the word line per 6T cell consists of two parts: the input capacitance of two access transistors and the capacitance of a wire that is the width of the cell. The input gate capacitance is much higher than the wire capacitance in this scenario. Hence, the word line capacitance per 6T cell is much higher than that of a simple wire. Due to the adjacent connection of the cells, the signal has to go through this slow word line wire, which is as wide as the array, to reach the farthest bit cell. This causes the total row delay to grow quadratically as word line size increases.

Traditional repeater insertion [3], [4] is a bad option for reducing the single buffer topology delay. While repeater buffers can reduce the quadratic growth of a wire delay, it is not intended for high-fanout connections like the 6T cells in an SRAM row. In particular, the connection of all cells would still be sequential and the critical path is still exposed to the same total amount of capacitance.

The DWL approach, as shown in Figure 2, can be used to improve speed of an SRAM in addition to, or instead of, saving power. First, the DWL topology solves the sequential connection issue by changing the critical path from a single word line to the slowest word line of several smaller divided sub-rows. The divided word lines, denoted as the local wires, are shorter and faster and thus reduce the overall delay. Second, the DWL topology solves the high capacitance issue by using the row select wire, denoted as the global wire, for global routing. The global wire, which connects the global buffer and local buffers, only sees the input capacitance of local buffers and thus shields much of the capacitance and is much faster than the local wires and the word line wire in the single buffer topology.

In the DWL topology, delay imbalance between sub-rows ultimately limits the speed as the size of a row increases. Hence, not only the number of sub-rows, but also the size of the sub-rows and the size of local buffers need to be optimized. Specifically, using non-uniform, tapered sizes for both sub-row and local buffer sizes can compensate for the delay difference between the left-most and right-most sub-rows.

III. Fast Delay Optimization for Uniform Divided Word Lines

Logical effort sizing of buffers is a simple and fast approach to optimize a uniform DWL topology. To find the best solution in our uniform DWL approach, we enumerate the number of sub rows from a single sub row until the word line delay begins to increase. In each iteration, we use logical effort to simultaneously size the buffers in each branch of the uniform DWL. Increasing the number of sub rows reduces the overall delay, because it reduces the local word line delay. On the other hand, it simultaneously increases the row select wire delay which is included in each sub-row delay. Eventually, the select wire delay becomes the critical path and enumeration is stopped.

The main drawback of such a fast method is that the logical effort model only includes the interconnect capacitance and ignores resistive shielding of the word line interconnect. Such an effect exists and is included in our SPICE simulation for final results and verification of the uniform DWL. Hence, a simple sizing approach tends to use more buffers than necessary. Our non-uniform DWL optimization (presented next) uses more accurate interconnect models by using a π-model [14] and computing the effective capacitance of the reduced π-model uses a fast lookup table using SPICE pre-characterization [15].

IV. Non-Uniform Divided Word Lines

The proposed non-uniform DWL, shown in Figure 3, has global and local wires that can span a row layout similar to the uniform DWL topology. The global wire uses a high metal layer, thus can route over local buffers and 6T cells without design rule violations. The local wire is the existing 6T cell word line wires which are connected by bit cell adjacency.

Both the local array size and the local buffer placement affect the global wire delay. The local buffer size has a minor effect compared to the bit cells and is not included in our model. In addition, the width of a local buffer is much smaller than the sub-row size when the word line size is large so we do not account for this effect on wire length.

Two approaches are used to improve the speed of the DWL topology. First and simplest, the local buffers are inverters rather than NAND gates. Second, sub-rows and local buffers are non-uniform and can compensate for the delay imbalance from the monotonically increasing global wire delay in sub-rows that are farther from the global NAND gate driver.

A. Row and Sub-Row Nomenclature

An SRAM row has a number n of cells in the row and a number m of sub-rows. Thus, there is one input inverter, one global NAND gate, m local buffers in each row. In Figure 3, the left-most sub-row is the first one and the right-most sub-row is the m-th. The input inverter is minimum size to lower the load on the decoder while the global NAND size is s_{gbuf}. The sizes of the sub-rows and local buffers are denoted respectively as a $1 \times m$ matrix A and $1 \times m$ matrix S, where

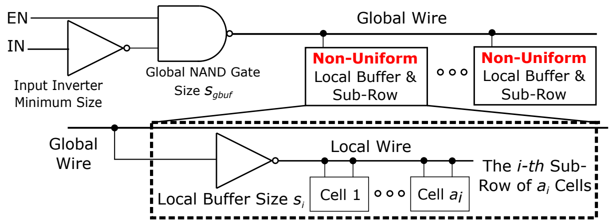

EN
IN
Input Inverter Minimum Size
Global NAND Gate Size S_{gbuf}
Global Wire
Non-Uniform Local Buffer & Sub-Row
Non-Uniform Local Buffer & Sub-Row
Global Wire
Local Wire
The i-th Sub-Row of a_i Cells
Local Buffer Size s_i
Cell 1
Cell a_i

Fig. 3. The proposed topology optimizes global buffer size s_{gbuf}, the number of local buffers m, sizes s_i of each local buffer and the number of cells a_i in each sub-row.

the i-th sub-row size and the i-th local buffer size are denoted as a_i and s_i, respectively.

The delay of all the sub-rows is a $1 \times m$ matrix D where the i-th element d_i is the delay from the input buffer to the farthest bit cell in that sub-row. The row buffer critical path depends on the slowest sub-row, so the delay of the row as a whole is the largest element in matrix D,

$$f_{delay}(s_{gbuf}, S, A) = \max(d_i(s_{gbuf}, S, A)), 1 \leq i \leq m. \quad (1)$$

where each d_i and hence f_{delay} are a function of the row parameters, s_{gbuf}, S, and A.

B. Delay Optimization

This paper optimizes delay while ignoring area overhead since the area difference between different buffer topologies is insignificant compared to the 6T array itself. However, the power-delay product or other metrics could easily be applied if users desire. The proposed design uses a nonlinear programming formulation due to multiple inputs and the nonlinear delay models. We use the Python SciPy library "optimize" function as our nonlinear optimizer.

1) Overall Optimization Structure: Figure 4 is the overall optimization flow. The delay optimization of a row cannot be solved by the nonlinear programming library function directly because number of sub rows must be an integer number. Hence, the delay optimization is split into several sub-problems where the number of sub-rows m is fixed to solve a larger integer nonlinear programming optimization. The tool iterates over the number of sub-rows m, examines each sub-problem, and finds the optimized result by comparing sub-problem results.

2) Non-Linear Sub-Problem: Each sub-problem is a nonlinear programming problem with an initial input and an objective function to find the optimized solution. The SciPy "optimize" function uses a sequential least squares programming (SLSQP) algorithm, estimates the gradient numerically with the default finite-difference setup. Several initial row solutions are generated with different strategies to prevent local minima in $f_{delay}(s_{gbuf}, S, A)$.

With a fixed number of sub-rows, the optimization sub-problem defines the constraints and boundaries of the inputs

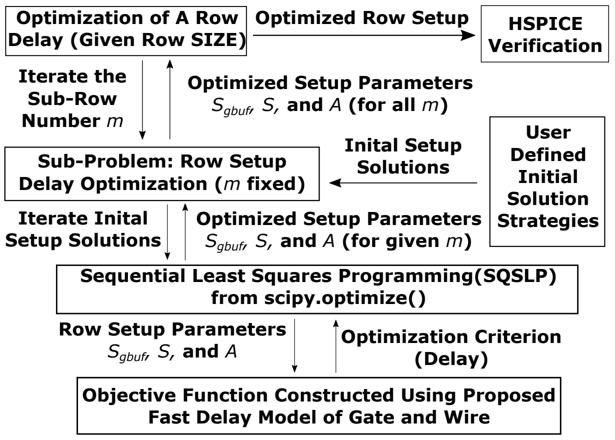

Optimization of A Row Delay (Given Row SIZE)
Optimized Row Setup
HSPICE Verification
Iterate the Sub-Row Number m
Optimized Setup Parameters S_{gbuf}, S, and A (for all m)
Sub-Problem: Row Setup Delay Optimization (m fixed)
Inital Setup Solutions
User Defined Initial Solution Strategies
Iterate Inital Setup Solutions
Optimized Setup Parameters S_{gbuf}, S, and A (for given m)
Sequential Least Squares Programming(SQSLP) from scipy.optimize()
Row Setup Parameters S_{gbuf}, S, and A
Optimization Criterion (Delay)
Objective Function Constructed Using Proposed Fast Delay Model of Gate and Wire

Fig. 4. The problem is break into several optimization sub-problems that can be formalized as Equation (2) and solved by a SciPy library function. A fast objective function that models delay of gates and wires is crucial for optimization.

and is based on Equation (1) and Section IV-A as

$$\begin{cases} \min f_{delay}(s_{gbuf}, S, A) \\ \sum_{i=1}^{m} a_i = n \\ 1 \leq a_i \leq n - m + 1, 1 \leq i \leq m \\ 1 \leq s_{gbuf} \\ 1 \leq s_i, 1 \leq i \leq m. \end{cases} \quad (2)$$

Each sub-row size a_i has an upper limit of $n - m + 1$, where the other $m - 1$ sub-rows have one cell to keep the total cell number as n. The global buffer size s_{gbuf} and local buffer size s_i have to be positive numbers bigger than one due to the minimum size of buffers.

3) Objective Function: The nonlinear programming solver calls the objective function to calculate the row delay, so it needs to be fast and relatively accurate. While SPICE simulation would be accurate, it is too slow. Hence, SPICE is only used for final verification and fast/accurate analytical delay models are used for the delays instead.

All interconnect delays use the Elmore delay model. Both global and local wires are modeled as a distributed RC model with each segment represented as a π-model. The 6T cell access transistor gates are a added as a single capacitor.

The gate delay models use lookup tables to calculate the output voltage delay/slew of a gate based on its size, gate type, input voltage slew, and output node effective capacitance. The input voltage slew is from the driving gate delay/slew calculation. The gate's output net is reduced to a single π-model [14], so that the effective capacitance of the reduced π-model can used by the gate lookup table [15].

V. Experimental Setup

We use the FreePDK45 design rules [16] and local/intermediate interconnect model parameters from the Predictive Technology Model [17]. A 6T cell has a width of $0.7um$ and so the word line wire resistance across the cell is 1.0269Ω and the wire capacitance is $0.0987fF$, assuming a $0.075\mu m$ wire width and $0.01\mu m$ wire spacing. The P to N ratio is 2 and unit transistor gate capacitance is $0.119fF$.

The optimized baseline is a single buffer topology sized using logical effort. Two optimized DWL topologies are analyzed: a uniform DWL topology sized using logical effort [18] and a non-uniform DWL topology sized with our nonlinear programming approach. Both DWL topologies use the lowest delay solution by iterating over the number of sub-rows m from 1 to 10. We considered word lines from 256 to 2048 bits with a 256 bit step size.

VI. Simulation Results with a 45nm Technology

Figure 5(a) shows the final delays of each design strategy from SPICE simulation. Our non-uniform DWL design has the smallest delay among all design strategies. Creating multiple sub-rows is not needed when the row size is small, but is essential for high-speed performance when the row size is bigger than 512 bits. After 512 bits, the single buffer topology delay increases at much faster rate than the DWL topologies. The delays of all topologies can be a few hundred picoseconds, and are thus significant for fast SRAM designs.

Figure 5(b) shows the delay improvement of all design strategies compared to the optimized baseline. Both DWL topologies show a steady increasing delay improvement as the row size increases. The non-uniform is slightly lower delay overall and at 2048 bits they are 45.35% lower delay and 42.79% lower delay than the baseline.

The non-uniform DWL shows slightly more delay (6ps) when the word line size is 512 bits. This is due to mismatch between SPICE simulations and our analytical models. The mismatch leads to slightly sub-optimal row setups and caused a delay increase only within a narrow intermediate row size range when all delay components are very similar.

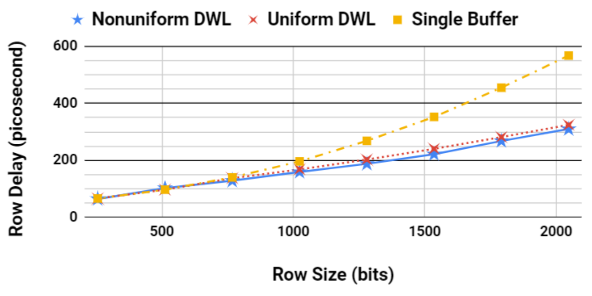
Figure 6 shows the area consumed by the row buffers for the different design strategies. The buffer area is approximated by the total transistor width of the buffers normalized to a minimum width transistor. The uniform DWL topology uses the same buffer size for the local buffer of each branch. The non-uniform DWL topology uses the optimized row setup from our methodology.

As mentioned in Section III, the baseline design and the uniform DWL topology tend over-size buffers since they exclude the shielding effect. The non-uniform DWL topology has fewer buffers except at the row size of 512 bits which is the same minor model mismatch as previously discussed. Overall, Figure 6 shows an increasing area savings compared to the baseline. The maximum is 42.84% when the row size is 2048 bits.

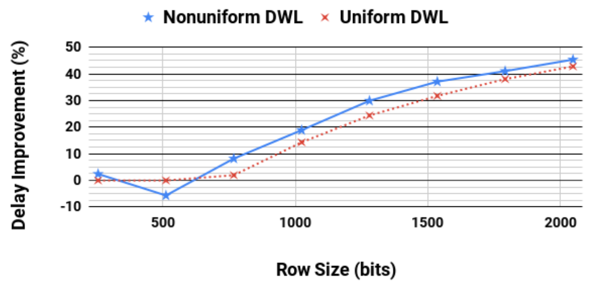
VII. Conclusion

In summary, the non-uniform DWL has the fastest speed of all approaches. An optimized uniform DWL topology can significantly reduce delay compared to a single buffer topology, but it does so with extra area compared to the non-uniform DWL approach. An optimized non-uniform DWL topology further reduces this delay and also saves buffer area. For large row sizes, the difference in delay between the uniform DWL and non-uniform DWL approaches becomes less significant, but the area savings is significant in these situations.

The word line driver delay is significant for high-speed SRAMs. To address the slow single buffer delay, this paper proposed using both a fast uniform DWL topology and a fast, area-efficient non-uniform DWL topology. The non-uniform DWL topology uses an accurate analytical word line model that considers both gate and interconnect delays and can be used in a non-linear programming approach to enhance the performance of SRAMs word lines in memory compilers [1].



(a) Delay Comparison



(b) Delay Reduction Compared to The Baseline

Fig. 5. The non-uniform DWL achieves the fastest speed compared to the rest by using a non-uniform DWL topology.

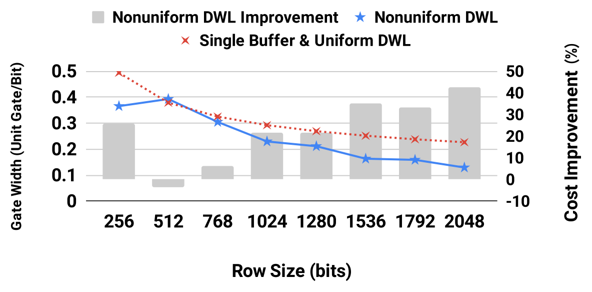


Fig. 6. When the row size is large enough, non-uniform DWL can save significant buffer area.

References

[1] M. R. Guthaus, J. E. Stine, S. Ataei, B. Chen, B. Wu, and M. Sarwar, “OpenRAM: An open-source memory compiler,” *ICCAD*, pp. 1–6, 2016.

[2] M. Yoshimoto, K. Anami, H. Shinohara, T. Yoshihara, H. Takagi, S. Nagao, S. Kayano, and T. Nakano, “A divided word-line structure in the static RAM and its application to a 64K full CMOS RAM,” *JSSC*, vol. SC-18, no. 5, pp. 479–485, 1983.

[3] S. Dhar and M. A. Franklin, “Optimum buffer circuits for driving long uniform lines,” *JSSC*, pp. 32–40, 1991.

[4] V. Adler and E. G. Friedman, “Repeater design to reduce delay and power in resistive interconnect,” *TCAS II*, vol. 45, no. 5, pp. 607–616, May 1998.

[5] R. Taylor and M. Johnson, “A 1-Mbit CMOS dynamic RAM with a divided bitline matrix architecture,” *JSSC*, vol. SC-20, pp. 894–902, 1985.

[6] H.-B. Kang and H.-W. K. et.al, “A hierarchy bitline boost scheme for sub-1.5 V operation and short precharge time on high density FeRAM,” *ISSCC*, vol. 1, pp. 158–159, 2002.

[7] X. Zheng and M. Liu, “A 512 kb SRAM in 65nm CMOS with divided bitline and novel two-stage sensing technique,” *DDECS*, pp. 191–192, 2012.

[8] N. K. Saini and A. Gupta, “Low power circuit techniques for optimizing power in high speed SRAMs,” *ICACCI*, pp. 2399–2404, 2016.

[9] T. Hirose, “A 20 ns 4-Mb CMOS SRAM with hierarchical word decoding architecture,” *JSSC*, vol. 25, no. 5, pp. 1068–1073, 1990.

[10] T. Chen, D. Lauderback, and G. Sunada, “Optimization of the number of levels of hierarchy in large scale hierarchy memory systems,” *ISCAS*, vol. 5, pp. 2104–2107, 1992.

[11] A. J. Bhavnagarwala, S. Kosonocky, and J. D. Meindlt, “Interconnect-centric array architectures for minimum SRAM access time,” *ICCD*, pp. 400–405, 2001.

[12] B. Rooseleer, S. Cosemans, and W. Dehaene, “A 65 nm, 850 MHz, 256 kbit, 4.3 pJ/access, ultra low leakage power memory using dynamic cell stability and a dual swing data link,” *JSSC*, vol. 47, pp. 1784–1796, 2012.

[13] B. Rooseleer and W. Dehaene, “A 40 nm, 454 MHz 114 fJ/bit area-efficient SRAM memory with integrated charge pump,” *ESSCIRC*, pp. 201–204, 2013.

[14] P. O’Brien and T. Savarino, “Modeling the driving-point characteristic of resistive interconnect for accurate delay estimation,” *ICCAD*, pp. 512–515, 1989.

[15] R. Macys and S. McCormick., “A new algorithm for computing the effective capacitance in deep sub-micron circuits,” *CICC*, pp. 313–316, 1998.

[16] “Freepdk45,” https://www.eda.ncsu.edu/wiki/FreePDK45:Contents.

[17] “Predictive technology model,” http://ptm.asu.edu/.

[18] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits.*, 1999.