DESIGN AND ANALYSIS OF FAST LOW POWER SRAMs

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF

ELECTRICAL ENGINEERING

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Bharadwaj S. Amrutur

August 1999

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Mark A. Horowitz (Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Bruce A. Wooley

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Krishna Saraswat

Approved for the University Committee on Graduate Studies:

_____

Abstract

This thesis explores the design and analysis of Static Random Access Memories (SRAMs), focusing on optimizing delay and power. The SRAM access path is split into two portions: from address input to word line rise (the row decoder) and from word line rise to data output (the read data path). Techniques to optimize both of these paths are investigated.

We determine the optimal decoder structure for fast low power SRAMs. Optimal decoder implementations result when the decoder, excluding the predecoder, is implemented as a binary tree. We find that skewed circuit techniques with self resetting gates work the best and evaluate some simple sizing heuristics for low delay and power. We find that the heuristic of using equal fanouts of about 4 per stage works well even with interconnect in the decode path, provided the interconnect delay is reduced by wire sizing. For fast lower power solutions, the heuristic of reducing the sizes of the input stage in the higher levels of the decode tree allows for good trade-offs between delay and power.

The key to low power operation in the SRAM data path is to reduce the signal swings on the high capacitance nodes like the bitlines and the data lines. Clocked voltage sense amplifiers are essential for obtaining low sensing power, and accurate generation of their

sense clock is required for high speed operation. We investigate tracking circuits to limit bitline and I/O line swings and aid in the generation of the sense clock to enable clocked sense amplifiers. The tracking circuits essentially use a replica memory cell and a replica bitline to track the delay of the memory cell over a wide range of process and operating conditions. We present experimental results from two different prototypes.

Finally we look at the scaling trends in the speed and power of SRAMs with size and technology and find that the SRAM delay scales as the logarithm of its size as long as the interconnect delay is negligible. Non-scaling of threshold mismatches with process scaling, causes the signal swings in the bitlines and data lines also not to scale, leading to an increase in the relative delay of an SRAM, across technology generations.The wire delay starts becoming important for SRAMs beyond the 1Mb generation. Across process shrinks, the wire delay becomes worse, and wire redesign has to be done to keep the wire delay in the same proportion to the gate delay. Hierarchical SRAM structures have enough space over the array for using fat wires, and these can be used to control the wire delay for 4Mb and smaller designs across process shrinks.

Acknowledgments

I thank my advisor Prof. Mark Horowitz for his invaluable guidance throughout the course of this thesis. His insights and wisdom have been a great source of inspiration for this work. I thank my co-advisor Prof. Bruce Wooley for his quiet encouragement and support during my stay in Stanford. I am grateful to Prof. Teresa Meng and Prof. Leonid Kazovsky for agreeing to be a part of my orals committee. I am indebted to Prof. Krishna Saraswat for reading my dissertation at a very short notice. The dissertation has greatly benefited from the astute comments of the reading committee members and any remaining errors are solely my own.

The many hours I have spent at work have been very stimulating and enriching, thanks to the wonderful students I have been privileged to interact with. I have enjoyed many a conversation with the inhabitants of CIS and would especially like to thank Guyeon Wei, Dan Weinlader, Chih-Kong Ken Yang, Ricardo Gonzalez, Stefanos Sidiropoulos, Ken Mai, Toshihiko Mori, Ron Ho, Vijayaraghavan Soundararajan and Esin Terzioglu.

Thanks to my many friends on campus and outside, I have had many memorable moments outside my work. I would especially like to thank Suhas, Navakant, Bijit, Navin, Anil, Mohan and Shankar.

I am grateful to Sudha, Anand, Raghavan and Meera for being such a wonderful family. I thank my wife, Vidya, for her love and patience during the final stages of my dissertation. Last but not the least, I am eternally indebted to by grandfather and my parents for their love, encouragement and support. This dissertation is dedicated to them.

Table of Contents

.

.

List of Tables

List of Figures

.

.

.

Chapter 1 *Introduction*

Fast low power SRAMs have become a critical component of many VLSI chips. This is especially true for microprocessors, where the on-chip cache sizes are growing with each generation to bridge the increasing divergence in the speeds of the processor and the main memory [1-2]. Simultaneously, power dissipation has become an important consideration both due to the increased integration and operating speeds, as well as due to the explosive growth of battery operated appliances [3]. This thesis explores the design of SRAMs, focusing on optimizing delay and power. While process [4-5] and supply [6-11] scaling remain the biggest drivers of fast low power designs, this thesis investigates some circuit techniques which can be used in conjunction to scaling to achieve fast, low power operation.

Conceptually, an SRAM has the structure shown in Figure 1.1. It consists of a matrix of 2^m rows by 2^n columns of memory cells. Each memory cell in an SRAM contains a pair of cross coupled inverters which form a bi-stable element. These inverters are connected to a pair of bitlines through nMOS pass transistors which provide differential read and write access. An SRAM also contains some column and row circuitry to access these cells. The m+n bits of address input, which identifies the cell which is to be accessed, is split into m row address bits and n column address bits. The row decoder activates one of the 2^m word lines which connects the memory cells of that row to their respective bitlines. The column decoder sets a pair of column switches which connects one of 2^n bitline columns to the peripheral circuits.

In a read operation, the bitlines start precharged to some reference voltage usually close to the positive supply. When word line turns high, the access nfet connected to the

cell node storing a ‘0’ starts discharging the bitline, while the complementary bitline

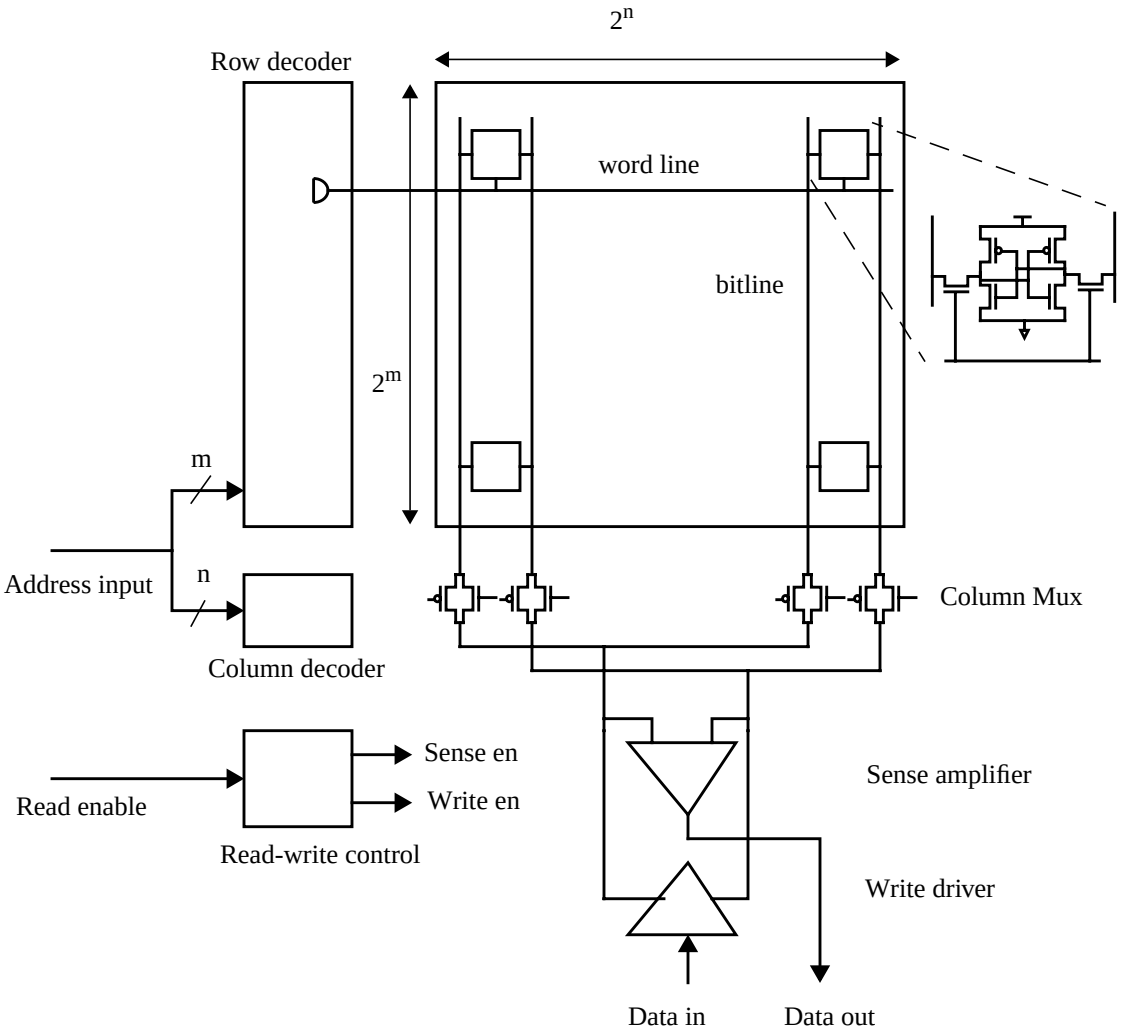


Figure 1.1: Elementary SRAM structure with the cell design in its inset

remains in its precharged state, thus resulting in a differential voltage being developed across the bitline pair. SRAM cells are optimized to minimize the cell area, and hence their cell currents are very small, resulting in a slow bitline discharge rate. To speed up the RAM access, sense amplifiers are used to amplify the small bitline signal and eventually drive it to the external world.

During a write operation, the write data is transferred to the desired columns by driving the data onto the bitline pairs by grounding either the bit line or its complement. If the cell data is different from the write data, then the '1' node is discharged when the access nfet connects it to the discharged bitline, thus causing the cell to be written with the bitline value.

The basic SRAM structure can be significantly optimized to minimize the delay and power at the cost of some area overhead. The optimization starts with the design and layout of the RAM cell, which is undertaken in consultation with the process technologists [4]. For the most part, the thesis assumes that a ram cell has been adequately designed and looks at how to put the cells together efficiently.

The next chapter introduces the various techniques which are used in practical SRAMs and motivates the issues addressed by this thesis. For the purposes of design and optimization, the access path can be divided into two portions: the row decoder - the path from the address inputs to the word line, and the data path - the portion from the memory cell ports to the SRAM I/O ports.

The decoder design problem has two major tasks: determining the optimal circuit style and decode structure and finding the optimal sizes for the circuits and the amount of buffering at each level. The problem of optimally sizing a chain of gates for optimal delay and power is well understood [12-16]. Since the decode path also has intermediate interconnect, we will analyze the optimal sizing problem in this context. The analysis will lead to some formulae for bounding the decoder delay and allow us to evaluate some simple heuristics for doing the sizing in practical situations. We will then look at various circuit techniques that have been proposed to speed up the decode path and analyze their delay and power characteristics. This will eventually enable us to sketch the optimal decode structures to achieve fast and low power operation (Chapter 3).

In the SRAM data path, switching of the bitlines and I/O lines and biasing the sense amplifiers consume a significant fraction of the total power, especially in wide access width memories. Chapter 4 investigates techniques to reduce SRAM power without hurt-

ing performance by using tracking circuits to limit bitline and I/O line swing and aid in the generation of the sense clock to enable clocked sense amplifiers. Experimental results from two different prototypes are also presented in this chapter.

We then take a step back from detailed circuit design issues to look at the scaling trends in the speed and power of SRAMs with size and technology. We use some simple analytical models for delay and power for the decoder and the data path which are a function of the size, organization and technology. The models are then used to obtain the optimum organizations for delay and power for each different SRAM size and technology generation, enabling us to plot the scaling trends (Chapter 5). We finally summarize the main conclusions of this thesis along with future directions for research in Chapter 6.

Chapter 2

Overview of CMOS SRAMs

The delay and power of practical SRAMs have been reduced over the years via innovations in the array organization and circuit design. This chapter discusses both these topics and highlights the issues addressed by this thesis. We will first explore the various partitioning strategies in Section 2.1 and then point out the main circuit techniques which have been presented in the literature to improve speed and power (Section 2.2).

2.1 SRAM Partitioning

For large SRAMs, significant improvements in delay and power can be achieved by partitioning the cell array into smaller sub arrays, rather than having a single monolithic array as shown in Figure 1.1. Typically, a large array is partitioned into a number of identically sized sub arrays (commonly referred to as macros), each of which stores a part of the accessed word, called the sub word, and all of which are activated simultaneously to access the complete word [20-22]. High performance SRAMs can have up to 16 macros, while low power SRAMs typically have only one macro. The macros can be thought of as independent RAMs, except that they might share parts of the decoder.

Each macro conceptually looks like the basic structure shown in Figure 1.1. During an access to some row, the word line activates all the cells in that row and the desired sub word is accessed via the column multiplexers. This arrangement has two drawbacks for macros that have a very large number of columns: the word line RC delay grows as the

square of the number of cells in the row, and bitline power grows linearly with the number of columns. Both these drawbacks can be overcome by further sub dividing the macros into smaller blocks of cells using the Divided Word Line (DWL) technique first proposed by Yoshimoto, et.al. in [17]. In the DWL technique the long word line of a conventional array is broken up into k sections, with each section activated independently thus reducing the word line length by k and hence reducing its RC delay by k^2. Figure 2.1 shows the DWL architecture where a macro of 256 columns is partitioned into 4 blocks each having only 64 columns. The row selection is now done in two stages, first a global word line is

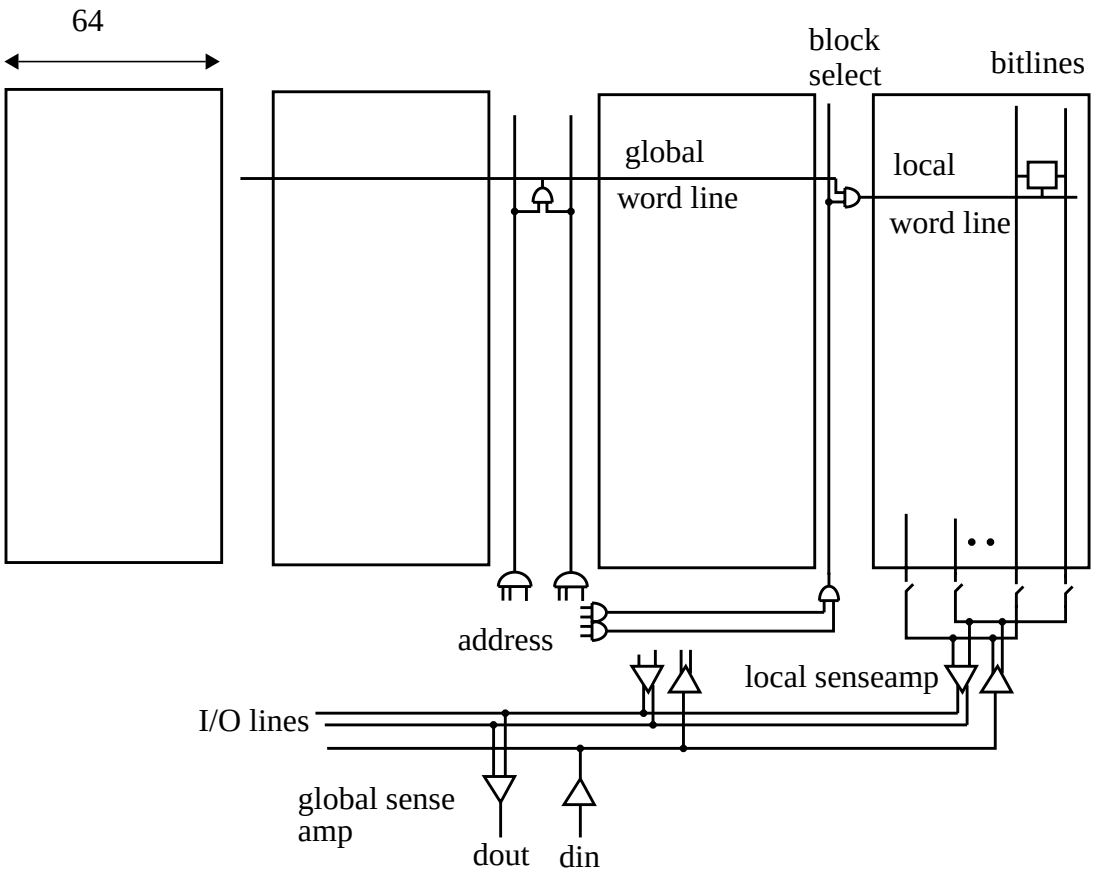


Figure 2.1: Divided Word Line (DWL) Architecture

activated which is then transmitted into the desired block by a block select signal to activate the desired local word line. Since the local word line is shorter (only 64 columns

wide), it has a lower RC delay. Though the global word line still is nearly as long as the width of the macro, it has a lower RC delay than a full length word line since its capacitive loading is smaller. It sees only the input loading of the four word line drivers instead of the loading of all the 256 cells. In addition, its resistance can be lower as it could use wider wires on a higher level metal layer. The word line RC delay is reduced by another factor of four by keeping the word drivers in the center of the word line segments thus halving the length of each segment. Since 64 cells within the block are activated as opposed to all the 256 cells in the undivided array, the column current is also reduced by a factor 4. This concept of dividing the word line can be carried out recursively on the global word line (and the block select line) for large RAMs, and is called the Hierarchical Word Decoding (HWD) technique [45]. Partitioning can also be done to reduce the bitline heights and is discussed further in the next section.

Partitioning of the RAM incurs area overhead at the boundaries of the partitions. For example, a partition which dissects the word lines requires the use of word line drivers at the boundary. Since the RAM area determines the lengths of the global wires in the decoder and the data path, it directly influences their delay and energy. In Chapter 5 we will study the trade offs in delay, energy and area obtained via partitioning.

2.2 Circuit techniques in SRAMs

The SRAM access path can be broken down into two components: the decoder and the data path. The decoder encompasses the circuits from the address input to the word line. The data path encompasses the circuits from the cells to the I/O ports.

The logical function of the decoder is equivalent to 2^n n-input AND gates, where the large fan-in AND operation is implemented in an hierarchical structure. The schematic of

a two-level 8 to 256 decoder is shown in Figure 2.2. The first level is the predecoder where

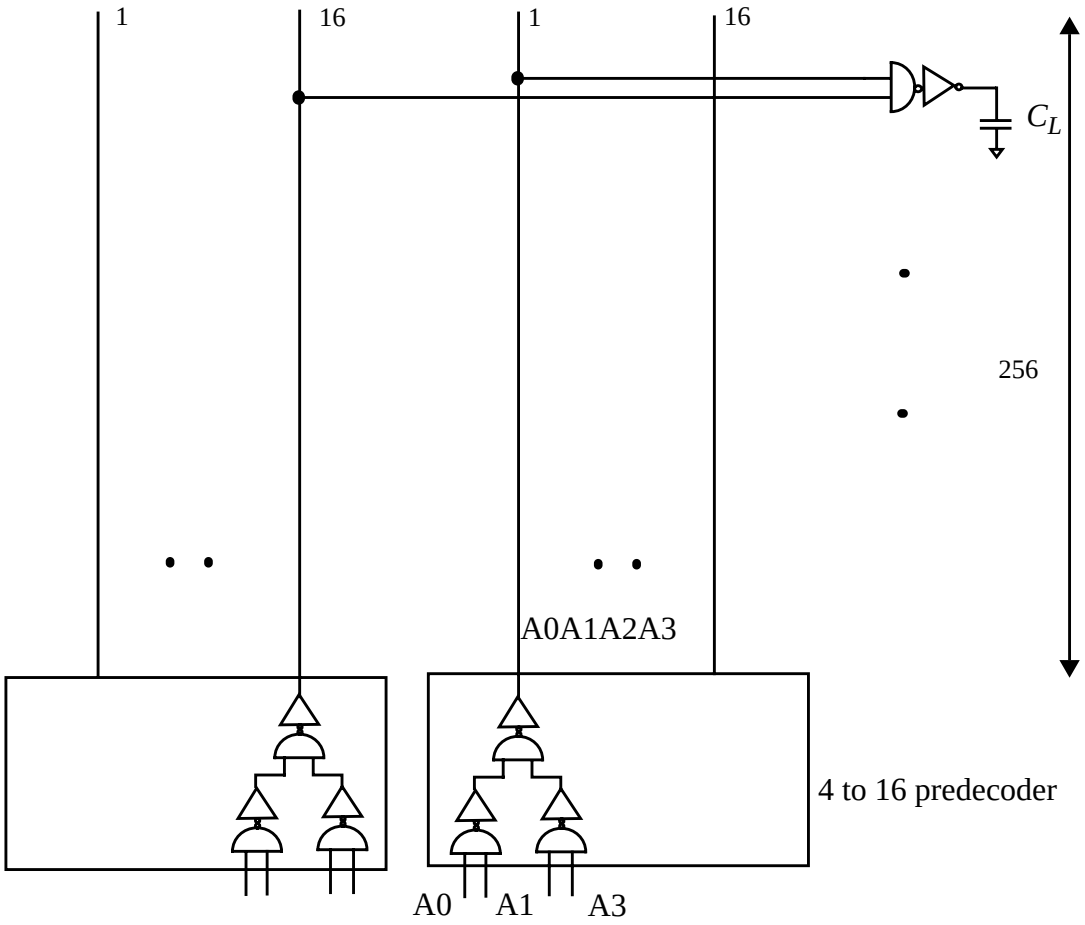


Figure 2.2: Schematic of a two-level 8 to 256 decoder

two groups of four address inputs and their complements (A0, $\overline{A0}$, A1, $\overline{A1}$, ...) are first decoded to activate one of the 16 predecoder output wires respectively to form the partially decoded products ($\overline{A0A1A2A3}$, A0A1A2A3, ...). The predecoder outputs are combined at the next level to activate the word line. The decoder delay consists of the gate delays in the critical path and the interconnect delay of the predecode and word line wires. As the wire RC delay grows as the square of the wire length, the wire delays within the decoder structure, especially of the word line, becomes significant in large SRAMs. Sizing of gates in the decoder allows for trade offs in the delay and power. Transistor sizing has been studied by a number of researchers for both high speed [12-14] and low power

[15-16]. The decoder sizing problem is complicated slightly due to the presence of intermediate interconnect from the predecode wires. We examine this problem in Chapter 3 and provide lower bounds for delay. We also evaluate some simple sizing heuristics for obtaining high speed and low power operation.

The decoder delay can be greatly improved by optimizing the circuit style used to construct the decoder gates. Older designs implemented the decode logic function in a

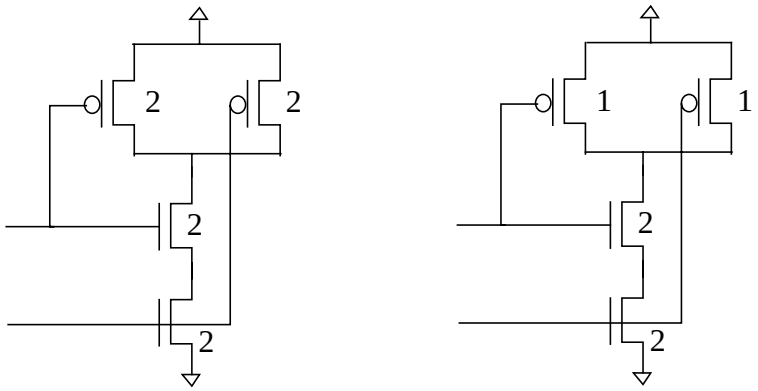


Figure 2.3: a) Conventional static NAND gate b) Nakamura's NAND gate [23]

simple combinational style using static CMOS circuit style (Figure 2.3a) [17-19]. In such a design, one of the 2^m word lines will be active at any time. If in any access, the new row address differs from the previous one, then the old word line is de-asserted and the new word line is asserted. Thus, the decoder gate delay in such a design is the maximum of the delay to de-assert the old word line and the delay to assert a new word line, and it is minimized when each gate in the decode path is designed to have equal rising and falling delays. The decode gate delay can be significantly reduced by using pulsed circuit techniques [20-22], where the word line is not a combinational signal but a pulse which stays active for a certain minimum duration and then shuts off. Thus, before any access all the word lines are off, and the decoder just needs to activate the word line for the new row

address. Since only one kind of transition needs to propagate through the decoder logic chain, the transistor sizes in the gates can be skewed to speed up this transition and minimize the decode delay. Figure 2.3b shows an instance of this technique [23] where the pMOS in the NAND gates are sized to be half that in a regular NAND structure. In the pulsed design, the pMOS sizes can be reduced by a factor of two and still result in the same rising delay since it is guaranteed that both the inputs will de-assert, thus reducing the loading of the previous stage and hence reducing the overall decoder delay. This concept is extended further in [20], where the de-assertion of the gate is completely decoupled from its assertion. Figure 2.4 shows an example of such a gate where the transistor sizes in the logic chain is skewed heavily to speed up the output assertion once the inputs are activated. The gate is then reset by some additional devices and made ready for the next access. By decoupling the assert and de-assert paths, the former can be optimized to reduce the decoder delay. In Chapter 3 we will compare the power

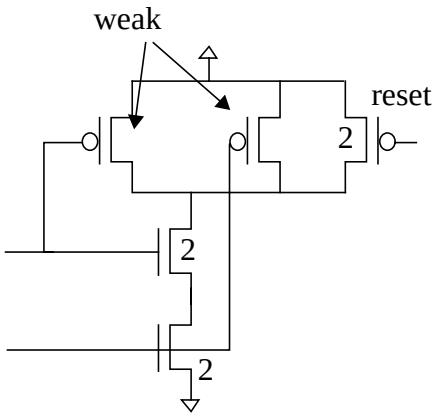
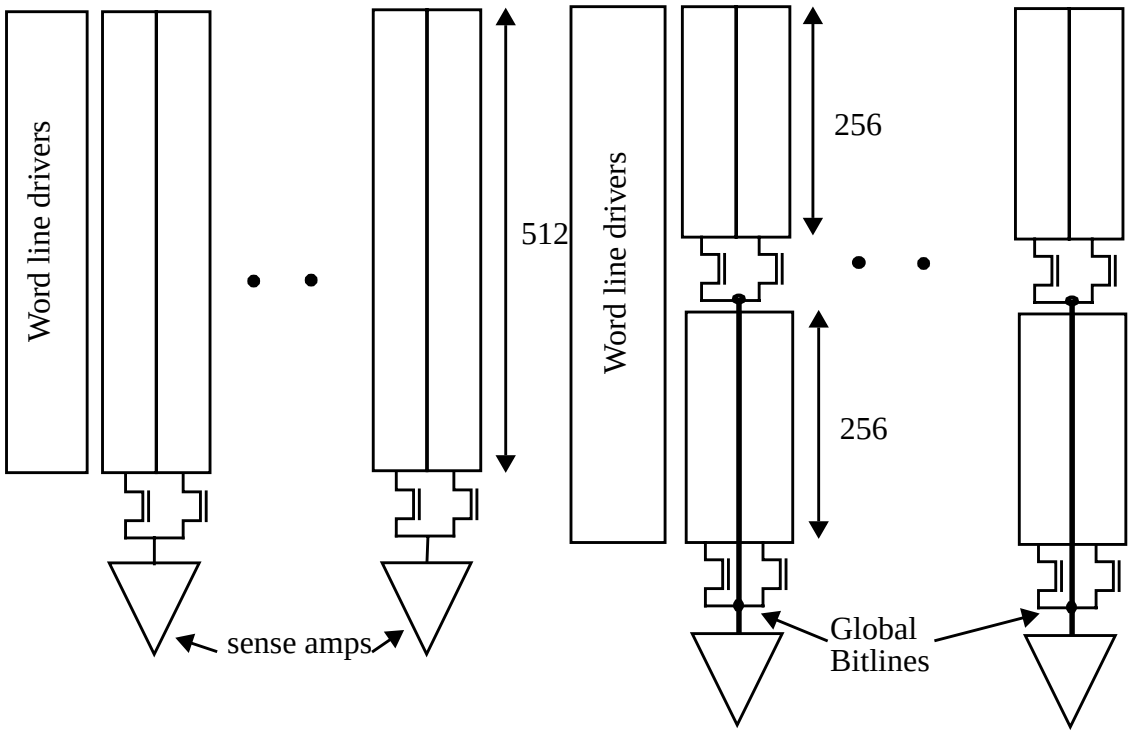


Figure 2.4: Skewed NAND gate

dissipation of using pulsed techniques in the decode path to the conventional static technique and show that the pulsed techniques reduce the delay significantly for only a modest increase in power. A pulsed decoder can also benefit a low power design in another way, viz. by reducing the power of the bitline path which we will explain shortly.

The SRAM data path logically implements a multiplexor for reads (and a demultiplexor for writes). In the simplest implementation, the multiplexor has only two levels: at the lowest level, the memory cells in a column are all connected together to a bitline and in the next level, a small number of these bitlines are multiplexed together through column pass transistors (Figure 1.1). When the bitline height is very large, it can be further partitioned to form multi-level bitline hierarchies, by using additional layers of metal [54]. In general, the multiplexor hierarchy can be constructed in a large number of ways ($2^{r-1}*2^c$ mux designs are possible for a 2^r x 2^{c+k} block with 2^r number of rows, 2^c number of columns and an access width of 2^k bits). Figure 2.5 shows two possible designs for a block with 512 rows. The schematic shows only the nMOS pass gates for a



a) Single Level Mux

b) Two Level Mux

Figure 2.5: Bitline mux hierarchies in a 512 row block

single-ended bitline to reduce the clutter in the figure, while the real multiplexor would use CMOS pass gates for differential bitlines, to allow for reads and writes. Figure 2.5a shows the single level mux design, where two adjacent columns with 512 cells each are multiplexed into a single sense amplifier. Figure 2.5b shows a two level structure in which the first level multiplexes two 256 high columns, the output of which are multiplexed in the second level to form the global bitlines, feeding into the sense amplifiers. Similarly hierarchical muxing can also be done in the I/O lines which connect the outputs of all the sense amplifiers to the I/O ports [60].

Due to its small size, a memory cell is very weak and limits the bitline slew rate during reads. Hence sense amplifiers are used to amplify the bitline signal so signals as small as 100mV can be detected. In a conventional design, even after the sense amplifier senses the bitlines, they continue to slew to eventually create a large voltage differential. This leads to a significant waste in power since the bitlines have a large capacitance. By limiting the word line pulse width, we can control the amount of charge pulled down by the bitlines and hence limit power dissipation [31-34]. In this thesis we propose a scheme to control the word line pulse width to be just wide enough, over a wide range of operating conditions, for the sense amplifiers to reliably sense, and prevent the bitlines from slewing further (Chapter 4).

A number of different sense amplifier circuits have been proposed in the past and they essentially fall into two categories: the linear amplifier type [24-25] and the latch type [20-22]. Figure 2.6 illustrates a simple prototype of each type. In the linear amplifier type (Figure 2.6a), the amplifier needs a DC bias current to set it up in the high gain region prior to the arrival of the bitline signal. To convert the small swing bitline signal into a full swing CMOS signal, a number of stages of amplification is required. These kinds of amplifiers are typically used in very high performance designs. Because they consume biasing power and since they operate over a limited supply voltage they are not preferred for low power and low voltage designs. In these designs the latch type designs are used

(Figure 2.6b). They consist of a pair of cross coupled gain stages which are turned on with

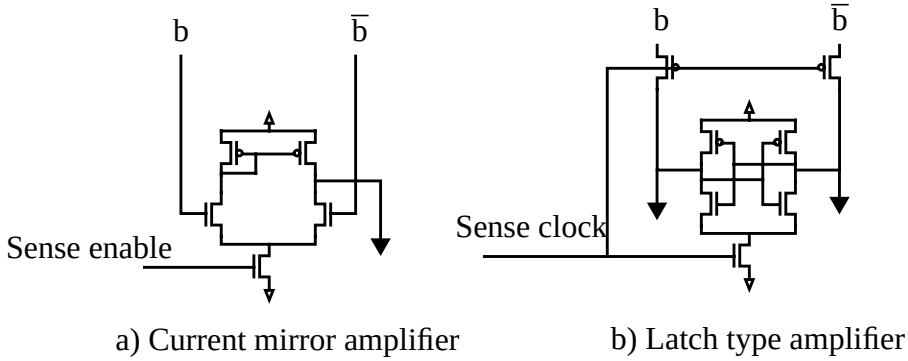


Figure 2.6: Two common types of sense amplifiers, a) current mirror type, b) latch type.

the aid of a sense clock when an adequate input differential is set up. The positive feedback in the latch leads to a full amplification of the input signal to a full digital level. While this type consumes the least amount of power due to the absence of any biasing power, they could potentially be slower since some timing margin is needed in the generation of the sense clock. If the sense clock arrives before enough input differential is set up, it could lead to a wrong output value. Typically the sense clock timing needs to be adjusted for the worst case operating and process condition, which in turn slows it down for the typical conditions due to the excess timing margins. In this thesis we will look at some timing circuits which track the bitline delay and which are used to generate a sense clock with a reduced timing overhead (Chapter 4).

In large SRAMs, another level is added to the data path hierarchy by connecting the outputs of the sense amplifiers onto the I/O lines (Figure 2.1). The I/O lines transport the signal between the RAM I/O ports to the memory blocks. In large access width SRAMs, the power dissipation of these lines can also be significant and hence the signalling on these is also via small swings [26]. In Chapter 4 we will apply the low swing bitline technique to the I/O lines too, to reduce the I/O line power.

Chapter 2: Overview of CMOS SRAMs

Chapter 3

Fast Low Power Decoders

As was described in Chapter 2, a fast decoder often uses pulses rather than level signals [20-22]. These pulse mode circuits also help reduce the power of the bitline access path [27-34]. This chapter explores the design of fast low power decoders.

The critical path of a typical three level decode hierarchy (based on the DWL technique) is shown in Figure 3.1. The path starts from the address input, goes through the

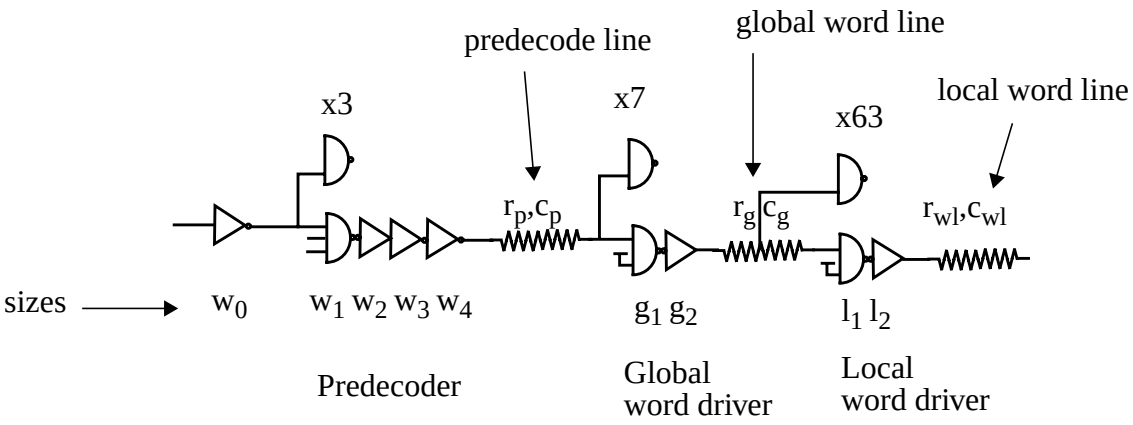


Figure 3.1: Critical path of a decoder in a large SRAM

predecoder gates which drive the long predecode wire and the global word driver, which in turn drives the global word line wire and the local word drivers and finally ends in the local word line. The decoder design problem has two major tasks: determining the optimal

circuit style and decode structure and finding the optimal sizes for the circuits and the amount of buffering at each level. The problem of optimally sizing a chain of gates for optimal delay and power is well understood [12-16]. Since the decode path also has intermediate interconnect, we will analyze the optimal sizing problem in this context. The analysis will lead to some formulae for bounding the decoder delay and allow us to evaluate some simple heuristics for doing the sizing in practical situations (Section 3.1). We will then look at various circuit techniques that have been proposed to speed up the decode path and analyze their delay and power characteristics (Section 3.2). This will eventually enable us to sketch the optimal decode structures to achieve fast and low power operation (Section 3.3).

3.1 Optimum Sizing and Buffering

The critical path of a typical decoder has multiple chains of gates separated by the intermediate interconnect wires. An important part of the decoder design process is to determine the optimal sizes and number of stages in each of these chains. When delay is to be minimized, the optimum sizing criterion can be analytically derived using some very simple models for the delay of the gates and the interconnect. In this analysis we will assume that the interconnect is fixed and independent of the decoder design. This is a valid assumption for 2 level and 3 level decode hierarchies as the intermediate interconnect serves the purpose of shipping the signals across the array. Since the array dimensions in most SRAM implementations are fixed (typically about twice the area occupied by the cells), the interconnect lengths are also fixed. When the objective function is more complicated like the energy-delay product or some weighted linear combination of energy and delay, then numerical optimization techniques have to be used to determine the optimal sizing. In practice, designers size either for minimum delay, or use some simple heuristics to back off from a minimum delay solution so as to reduce energy. In this section we will explore various sizing techniques and explain their impact on the delay,

energy and area of the decoder.

We introduce our notations by first reviewing the classic problem of optimally sizing a chain of inverters to drive a fixed output load. We will adopt terminology from the Logical Effort work of Sutherland and Sproull [35].

3.1.1 Review of Optimum Inverter Chain Design

Consider an inverter chain with *n* stages, driving an output load of C_L (Figure 3.2). Let

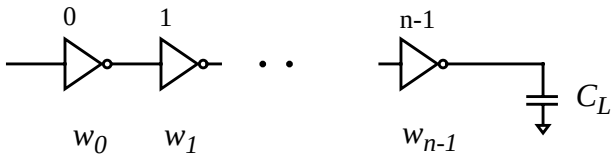


Figure 3.2: A chain of inverters to drive a load

the input stage have a fixed size of w_0 for the nfet and 2*w_0 for the pfet. Lets assume that the pfet needs to be twice as wide as an nfet in order to equalize the inverter rising and falling delays. Hence just one size parameter, the nfet width, is sufficient to fully characterize the inverter. We will define a unit sized inverter to have an nfet of size 1 unit. The sizing problem is to find the optimal number of stages, *n*, and the sizes for inverters w_1, to w_{n-1}, to minimize delay, power or area.

3.1.1.1 Sizing for Minimum Delay

We will use a simple RC model for the inverter [12,14], wherein the inverter is represented as having an input capacitance, an output conductance and an output parasitic capacitance, all proportional to its size (Figure 3.3). Let C_g be the gate capacitance per unit width, R_g be the output resistance for a unit width and C_j be the parasitic capacitance per unit width of an inverter. The inverter delay is then estimated to be the product of its

resistance times the total capacitance it is driving at its output. The delay for stage *i* is

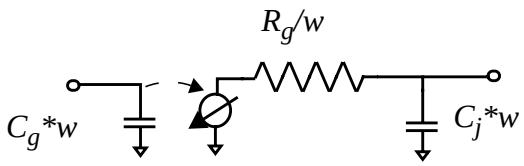


Figure 3.3: RC model of a inverter

given in Equation 1 and is the sum of two terms, the effort delay, (also referred to simply

$$D_i = \underbrace{\frac{R_g}{w_i} \cdot C_g \cdot w_{i+1}}_{\text{Effort delay}} + \underbrace{\frac{R_g}{w_i} \cdot C_j \cdot w_i}_{\text{Parasitic delay}} \qquad (1)$$

as effort), which is the product of the inverter resistance and the external load, and the parasitic delay which is a product of the resistance and the parasitic capacitance. The total delay of the inverter chain is the sum of delays of each stage in the chain as shown in Equation 2. The total delay is minimized when the gradient of the delay with respect to the

$$D_{path} = \sum_{i=0}^{n-1} D_i \qquad (2)$$

sizes is zero and is achieved when the effort delays for each stage is the same.

Lets denote the product R_g * C_g as τ, and call it the unit delay and represent the external load in terms of an inverter of size w_n such that $C_L = C_g * w_n$. Dividing both sides of Equation 2 by τ, and substituting Equation 1, gives the normalized delay equation for the total delay shown in Equation 3. In this equation, *p* is the ratio of C_j/Cg and is the ratio

of the drain junction capacitance of the output of the inverter with the gate capacitance of the inputs and is approximately independent of the inverter size. We will call the

$$D_{path}/\tau = \sum_{i=0}^{n-1}\left(\frac{w_{i+1}}{w_i} + p\right) \tag{3}$$

$$\frac{w_1}{w_0} = \frac{w_2}{w_1} \bullet\bullet = \frac{w_n}{w_{n-1}} = f \tag{4}$$

normalized effort delay of a stage as its stage effort, or simply effort if the context is clear. For minimum delay, all the stage efforts are equal to each other and we denote them as f in Equation 4. The product of all the stage efforts is a constant called the path effort and is the ratio of the output load and the size of the input driver (5). With the aid of Equations 4 and 5 total delay can be expressed as in (6). Differentiating Equation 6 with respect to f

$$f^n = \frac{w_n}{w_0} = \frac{C_L/C_g}{w_0} \tag{5}$$

$$D_{path}/\tau = (f + p)\cdot \ln(w_n/w_0)/\ln(f) \tag{6}$$

and setting the derivative to zero results in Equation 7. The solution to Equation 7 gives

$$\ln(f) - 1 - \frac{p}{f} = 0 \tag{7}$$

the optimum stage effort which will minimize the total delay [14]. When the parasitic delay p is zero, then $f = e$ is the optimum solution and is the classic result derived by Jaeger in [12]. In general p is not zero and Equation 7 needs to be solved numerically to obtain the optimal stage effort. For the base 0.25μm technology described in Appendix C, p = 1.33 and so f = 3.84 is the optimum effort per stage. In practice the delay is not very sensitive to the stage effort so to minimize power larger stage efforts are preferred.

3.1.1.2 Sizing for Minimum Power

The power dissipation in an inverter consists of three components, the dynamic power to switch the load and parasitic capacitance, the short circuit power due to the simultaneous currents in the pfet and nfet during a signal transition and the leakage power due to the subthreshold conduction in the fets which are supposed to be off. The dynamic power dissipation during one switching cycle in stage *i* of Figure 3.2 is the power to switch the gate load of stage *i+1* and self loading of stage *i* and is given in Equation 8, with V_{dd} rep-

$$P_{DY,\,i} = (C_g \cdot w_{i+1} + C_j \cdot w_i) \cdot V_{dd}^2 \cdot frequency \qquad (8)$$

resenting the supply voltage and *f,* the operating frequency. The short circuit current through the inverter depends on the edge rate of the input signal and the size of the output load being driven [36]. Cherkauer and Friedman in [16] estimate the short circuit power of stage *i* in the chain to be proportional to the total capacitive load, just as the dynamic power is. The subthreshold leakage current has been traditionally very small due to the high threshold voltages of the fets, but they are expected to become important in the future as the thresholds are scaled down in tandem with the supply voltage. The leakage current of stage *i* is proportional to the size w_i. Since all the three components are proportional to the size of the stage, the total normalized power dissipation can be simply expressed as the sum of the inverter sizes as in Equation 9. Power is reduced when the total width of tran-

$$P = \sum_{i=0}^{n} w_i \qquad (9)$$

sistors in the chain is minimized. The minimum power solution has exactly one stage, which is the input driver directly driving the load and is not very interesting due to the large delays involved. We will next look at sizing strategies to reduce power with some delay constraint.

3.1.1.3 Sizing for Minimum Delay and Power

In practical designs there is usually an upper bound on the delay specification of the chain and the power needs to be minimized subject to this constraint. The sizing problem in this case can only be solved numerically via some optimization program [15]. The main feature of the optimum solution is that the stage efforts increase as one goes down the chain.

A simpler approach to the sizing problem is to design each stage to have the same stage effort, but instead find the stage effort which minimizes energy, while meeting the delay constraint. Choi and Lee in [15] investigate this approach in the context of minimizing the power-delay product and they find that the solution requires stage efforts of about 6.5 and is within a few percent of the globally optimal solution. The larger stage effort leads to smaller number of stages as well as lower power solution compared to the minimum delay design. Further improvements in delay at minimal expense of power can be obtained by observing that the stages in the beginning of the chain provide as much delay as the latter ones, but consume less power, and so one can have stage efforts for minimum delay in the initial stages and larger efforts for the final stages, mirroring the globally optimal solution [37].

With this background on sizing gates, we will next turn our attention to the core problem of this section, viz., how to size a decoder optimally.

3.1.2 Sizing the Decoder

The three key features of the decode path which distinguishes it from a simple inverter chain is the presence of logic gates other than inverters, branching of the signal path at some of the intermediate nodes, and the presence of interconnect inside the path. Logic gates and branching are easily handled by using the concept of logical effort and branching effort introduced in [35].

A complex gate like an n-input NAND gate has *n* nfets in series, which degrades its speed compared to an inverter. To obtain the same drive strength as the inverter, the nfets

have to be sized *n* times bigger, causing the input capacitance for each of the NAND inputs to be *(n+2)/3* times bigger than that of an inverter having the same drive strength. This extra factor is called the logical effort of the gate. The total logical effort of the path is the product of logical efforts of each gate in the path. We note here that for short channel fets, the actual logical effort of complex gates is lower than that predicted by the previous simple formula since the fets are velocity saturated, and the output current degradation of a stack of fets will be lesser than in the long channel case.

In the decode path, the signal at some of the intermediate nodes is branched out to a number of identical stages, e.g. the global word line signal in Figure 3.1 is branched out to a number (*be*) of local word driver stages. Thus any change in sizes of the local word driver stage is has an impact on the global word line signal which is enhanced by the factor of *be*. The amount of branching at each node is called the branching effort of the node and the total branching effort of the path is the product of all the node branching efforts.

As an example, consider a simple 4 to 16 decoder having two 2-input NAND gates and two inverters, driving a load of C_L at the output (Figure 3.4). The size of the NAND gate is

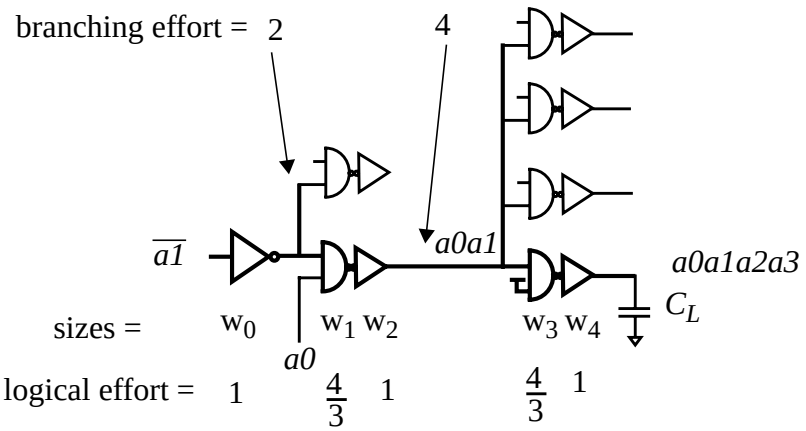


Figure 3.4: Critical path in a 4 to 16 predecoder. The path is highlighted by thicker lines. The sizes of the gates and their logical effort along with the branching efforts of two intermediate nodes is shown.

expressed in terms of that of an inverter having the same drive strength. The total effort delay of the path can then be expressed as the sum of effort delays of each of the five

stages in the critical path and is given as in Equation 10. To minimize this delay, the effort

$$D = \underbrace{\frac{\frac{4}{3}\cdot w_1 \cdot 2}{w_0}}_{\text{stage 0}} + \underbrace{\frac{w_2}{w_1}}_{\text{stage 1}} + \underbrace{\frac{\frac{4}{3}\cdot w_3 \cdot 4}{w_2}}_{\text{stage 2}} + \underbrace{\frac{w_4}{w_3} + \frac{C_L}{w_4}}_{\text{stage 4}} + parasitics \qquad (10)$$

Stage efforts for

$$\text{Total path effort equals} \quad f^5 = \frac{\left(\frac{4}{3}\right)^2 \cdot 2 \cdot 4 \cdot C_L}{w_0} \qquad (11)$$

of each stage needs to be the same (*f*), and can be obtained by relating it to the product of the stage efforts as in Equation 11. Since each NAND gate has a logical effort of 4/3, and two of the intermediate nodes have branching efforts of 2 and 4, the total path effort is enhanced by a factor of 128/9 in Equation 11.

In general for a r to 2^r decode, the total branching effort of the critical path from the input or its complement to the output is 2^{r-1} since each input selects half of all the outputs. The total logical effort of the path is the effort needed to build an r-input AND function. If the wire delays within the decoder are insignificant then the optimal sizing criterion remains the same as in the case of the inverter chain. The only difference is that the total path effort gets multiplied by the total logical effort and the total branching effort and hence affects the optimal number of stages in the path as shown in Equation 12. Theoreti-

$$f^n = \frac{outputLoad \cdot 2^{r-1} \cdot LogicalEffort(r - input - AND)}{w_0} \qquad (12)$$

cally, the optimum stage effort differs slightly from that for a inverter chain (since the parasitics of logic gates will be more than that of an inverter in general), but in practice the same stage effort of about 4 suffices to give a good solution.

The final distinguishing feature of the decode path, the presence of intermediate interconnect, can lead to an optimal sizing criterion which is different then that for the inverter chain. Ohbayashi, et. al. in [38] have shown that when the interconnect is purely capacitive, the optimal stage efforts for minimum delay reduces along the chain. In the next subsections we will derive this result as a special case of treating the interconnect more generally as having both resistance and capacitance. We will show that when the interconnect has both R and C, the optimal stage efforts for all the stages is still the same as in the inverter chain case, but this might require non integral number of stages in the intermediate sub chains, which is not physically realizable. We will also provide expressions for optimal efforts when the intermediate chains are forced to have some fixed integral number of stages. We will finally consider sizing strategies for reducing power and area along with the delay.

3.1.2.1 Sizing an Inverter Chain with Fixed Intermediate Interconnect for Minimum Delay

Without loss of generality, consider an inverter chain driving a fixed load C_L, and having two sub chains separated by interconnect having a resistance and capacitance of R_{wire} and C_{wire} respectively. To simplify the notations in the equations we will introduce two new normalized variables R and C which are obtained as $R = R_{wire}/R_g$ and $C = C_{wire}/C_g$. This will make all our delays come out in units of τ (= R_g*C_g). Again we will assume that the input inverter has a fixed size of w_0 (Figure 3.5). The first subchain has $n+1$ stages, with sizes w_0 to w_n and the second subchain has k stages with sizes x_1 to x_k. The goal is to find the number of stages n, k and the sizes of each stage such that the total delay is minimized.

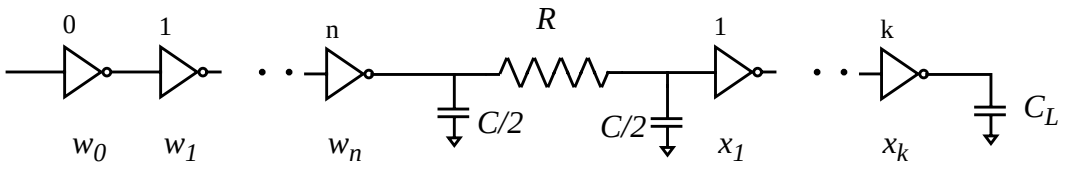


Figure 3.5: Buffer chain with intermediate interconnect

The interconnect is popularly modeled as a π section as shown in Figure 3.6 [39]. The delay of this section is approximated to be the inverse of the dominant pole [40] and is given as $R * (C/2 + x_1)$.

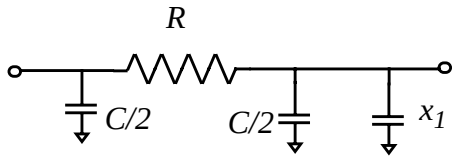


Figure 3.6: Modeling the interconnect delay

The total delay can be expressed as the sum of delays for each stage plus the delay of the interconnect as in Equation 13. Delay is minimized when the gradient of delay with

$$D = \underbrace{\frac{w_1}{w_0} + \frac{w_2}{w_1} \bullet\bullet \frac{(C + x_1)}{w_n}}_{\text{Delay of } 1^{st} \text{ chain}} + \underbrace{R \cdot (C/2 + x_1)}_{\text{Interconnect}} \underbrace{\bullet\bullet \frac{C_L}{x_k}}_{2^{nd} \text{ chain}} + \underbrace{(n + k + 1) \cdot p}_{\text{Parasitic delay}} \qquad (13)$$

respect to the sizes is zero and is achieved when conditions of Equations 14 and 15 are satisfied, viz., the efforts of all the gates in the first chain are equal (and denoted as f_1), and the efforts of all the gates in the second chain are equal (and denoted as f_2). The total path

$$\frac{w_1}{w_0} = \frac{w_2}{w_1} \quad = \frac{(C + x_1)}{w_n} = f_1 \qquad (14)$$

$$(R + 1/w_n) \cdot x_1 = \frac{x_2}{x_1} \bullet\bullet = \frac{L}{x_k} = f_2 \qquad (15)$$

$$D = (n + 1) \cdot f_1 + R \cdot (x_1 + C/2) + k \cdot f_2 + (n + k + 1) \cdot p \qquad (16)$$

delay for minimum delay can than be rewritten in terms of the optimal efforts as in Equation 16. By inspecting Equations 14 and 15, we can also obtain a relationship between f_1 and f_2 as in Equation 17. From this we can deduce that if $R=0$, $f_2 < f_1$, and is the case discussed by [38]. In this scenario, for minimum delay the stage effort for the second chain is less than the stage effort of the first chain. Similarly, when $C = 0$, $f_1 < f_2$, the exact dual occurs, i.e., the stage effort of the first chain is less than the stage effort of the second chain.

$$\frac{f_1}{f_2} = \frac{(C + x_1)/w_n}{(R + 1/w_n) \cdot x_1} \tag{17}$$

Analogous to the single chain case, the minimum delay in Equation 16 can be found by finding the optimum values for f_1, f_2, n *and* k and the detailed derivation is presented in Appendix A. The main result of this exercise is that for minimum delay when R and C are non-zero, $f_1 = f_2 = f$, and f is the solution of Equation 7, i.e., the same sizing rule which is used for the simple inverter chain, is also the optimal for the chain with intermediate interconnect. The appendix also shows that this result holds true even for chains with more than one intermediate interconnect stage.

An interesting corollary can be observed by substituting $f_1 = f_2 = f$ in Equation 17 to obtain the following necessary condition for optimum, viz., the product of wire resistance times the down stream gate capacitance is the same as the product of the driver resistance times the wire capacitance (18). The same condition also turns up in the solution of optimal repeater insertion in a long resistive wire [39].

$$C/w_n = R \cdot x_1 \tag{18}$$

The appendix also derives analytical expressions for optimum *n* and *k* and they are reproduced in Equations 19 and 20. If the interconnect delay is either too large or too

$$f^n = \frac{C/w_0}{2f} \cdot \left(1 + \sqrt{1 + \frac{4f}{RC}}\right) \quad (19)$$

$$f^k = \frac{RC_L}{2f} \cdot \left(1 + \sqrt{1 + \frac{4f}{RC}}\right) \quad (20)$$

small compared to the inverter delay, then we can simplify Equations 19 and 20 as follows:

A. RC/2 >> 2f

In this situation, the interconnect delay is much worse than the delay of a fully loaded inverter, (recall from Equation 3 that the optimal inverter delay is given as $f + p$, with $f \sim 4$ and $p \sim 1.33$ units of delay). Then Equations 19 and 20 can be approximated as in Equations 21 and 22. These equations indicate that when the interconnect delay is significant

$$f^n \approx \frac{C/w_0}{f} \quad (21)$$

$$f^k \approx \frac{RC_L}{f} \quad (22)$$

compared to the inverter delay, then the optimal way to size the two subchains is to treat them as two independent chains with one of them having the wire capacitance as its final load and the input inverter as its input drive and the other having the load capacitance as its final load and the wire resistance as its input drive.

B. RC/2 << 2f

When interconnect delay is negligible compared to the inverter delay, then Equations 19 and 20 can be rewritten as in Equations 23 and 24. The product of these two yields Equation 25 which relates the total number of stages to the ratio of the output load to input driver and is the same as what one would obtain when there is no intermediate intercon-

nect (see Equation 5). In many situations, the wire resistance can be neglible and only its

$$f^n \approx \frac{\sqrt{C/R}}{w_0\sqrt{f}} \tag{23}$$

$$f^k \approx \frac{C_L\sqrt{R/C}}{\sqrt{f}} \tag{24}$$

$$f^{n+k} \approx \frac{C_L/w_0}{f} \tag{25}$$

capacitance is of interest. From Equations 23 and 24 we see that when the wire resistance tends to zero, *k*, the number of stages in the second chain reduces, while, *n*, the number of stages in the first chain increases. Thus when the interconnect is mainly capacitive the interconnect capacitance should be driven as late in the buffer chain as possible for minimum delay, so as to get the maximum effect of the preceding buffers. The exact dual occurs when the interconnect capacitance is negligible, while the interconnect resistance is not. In this case, as *C* tends to zero, we see from Equations 23 and 24 that *n* reduces and *k* increases. This implies that when the interconnect is purely resistive the interconnect should be driven as early in the buffer chain as possible for minimum delay.

For extremely small values of *R* or *C*, Equations 23 and 24, will give values for *n* or *k* which could end up being less than 1 and even negative and will not make physical sense. In this scenario, a physically realizable solution will have $k = 1$ and the optimum stage effort of the second chain, f_2, will be smaller than the stage effort of the first chain, f_1.

We will next apply these results to develop simple sizing strategies for practical decoders.

3.1.2.2 Applications to a Two-Level Decoder

Consider a design where r row address bits have to be decoded to select one of 2^r word lines. Assume that the designer has partitioned the row decoder such that it has a hierarchy of two levels and the first level has two predecoders each decoding r/2 address bits to drive

one of $2^{r/2}$ predecode lines (in the next section we will look at how to optimally partition the decode hierarchy). The next level then ANDs two of the predecode lines to generate the word line. This is a typical design for small embedded SRAMs and is shown in Figure 3.7a. The equivalent critical path is shown in Figure 3.7b and is similar to the inverter chain case discussed in the previous subsection, except for the branching and logical efforts of the two chains. We label the branching effort at the input to the word line drivers as *be* (= $2^{r/2}$) and the logical effort of the NAND gate in the word line driver as *le*. Let f_1 and f_2 be the stage efforts for the two sub chains. With no constraint on *n, k,* from the previous subsection we need $f_1=f_2=f\sim4$ with *n, k* given as in Equations 19 and 20 for minimum delay given by Equation 16. Since *n* and *k* need not be integers in this solution, it might not be physically realizable and the delay value will serve only as a theoretical lower bound. Nevertheless these values for *n, k* can be used as a starting point for a real design.

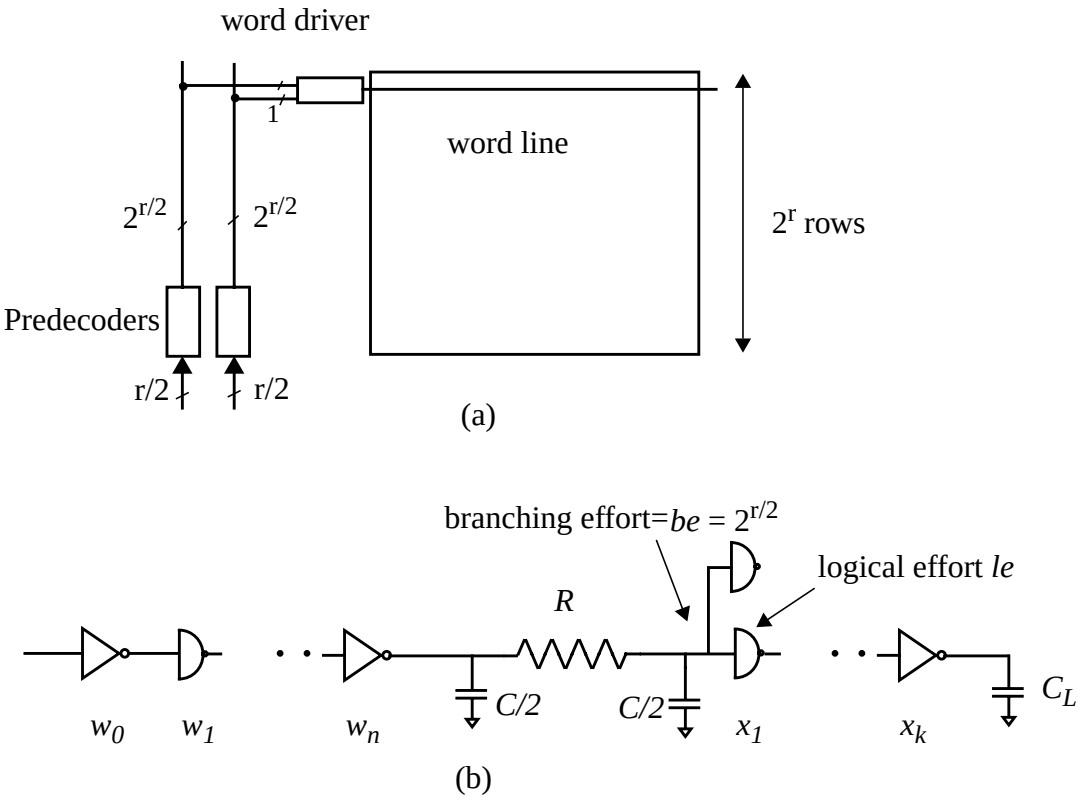


Figure 3.7: (a) Schematic of Embedded SRAMs (b) Equivalent critical path

Since the problem of finding a solution with integer *n* stages in the predecoder is the same as that for a simple buffer chain (see Equation 12), we will not discuss it here. The optimum number of integral stages in the word driver can be obtained by rounding *k* to the nearest even integer if the word driver implements the AND logical function of its input i.e. for *k* in *2i* to *2i+1*, round *k* to be *2i* and for *k* between *2i+1* to *2i+2*, round *k* to *2i+2*. Similarly, round *k* to the nearest odd integer if the word driver implements the AND function of the complement of the inputs. Since the number of stages in the word driver chain has now been constrained to be an integer, the optimal effort in this sub chain (f_2) need no longer be equal to *f(~4*). We present three alternatives to recompute this effort: the first (labeled as OPT) sets up an equation for delay in terms of f_2 and minimizes it to obtain the optimal solution for the specific integer number of stages in the word driver chain. The remaining two are heuristics which we will describe shortly.

We will next describe the approach to compute the optimum stage efforts with the number of stages in the word driver chain constrained to be an even integer (*ik*), close to the value of *k* obtained previously. Applying the condition for minimum delay, that the stage efforts in each of the sub chains be equal, we get Equations 26 and 27, which are a

$$\frac{w_1}{w_0} = \frac{w_2}{w_1} \bullet\bullet = \frac{C + be \cdot le \cdot x_1}{w_n} = f \qquad (26)$$

$$\left(R + \frac{1}{w_n}\right) \cdot be \cdot le \cdot x_1 = \frac{x_2}{x_1} \bullet\bullet = f_2 \qquad (27)$$

modification of Equations 14 and 15 and include the effects of the logical (*le*) and branching efforts (*be*). The effects of *le* and *be* cause the gate loading of the input of the word driver (x_1) to be enhanced by a factor of *be* * *le*. Substituting for $x_1 = C_L/f_2^{ik}$ and eliminat-

ing w_n between these, we can derive a relation for f_2 as in Equation 28. When R is

$$R \cdot \left(C + \frac{be \cdot le \cdot C_L}{f_2^{ik}} \right) + f = \frac{C \cdot f_2^{ik+1}}{be \cdot le \cdot C_L} + f_2 \qquad (28)$$

negligible and when ik=2, Equation 28 can be solved exactly, but for all other cases it has to be solved numerically. Table 3.1 lists the optimum efforts obtained from solving Equation 28 for a few different RAM sizes in the column labeled as OPT and the corresponding delay is expressed relative to the theoretical lower bound.

We consider two simple sizing heuristics to determine the effort f_2 of the word driver chain, which are an alternative to solving Equation 28 and compare their relative performance in the table. In one, labeled H1, the effort of the word driver chain is made such that it would present the same input load as in the theoretical lower bound design. Therefore the new effort is given as $f_2 = f^{k/ik}$. In the second heuristic, labeled H2, a fanout of about 4 is used in all the sub chains, i.e f_2 is also kept to be equal to f *(=3.84).* We see that solving

Rows	**Cols**	**From (19) *k***	***ik* = *even_int(k)***	**OPT f_2**	**H1 f_2**	**OPT relative delay**	**H1 relative delay**	**H2 ($f_1 = f_2 = 3.84$) relative delay**
32	32	0.04	2	3.12	1.03	1.04	1.22	1.05
64	64	0.6	2	3.14	1.5	1.03	1.11	1.03
64	512	2.1	2	3.9	4.1	1	1.00	1.00
256	64	1.1	2	3	2.1	1.02	1.04	1.03
256	512	2.7	2	4.6	6.1	1.01	1.03	1.02

Table 3.1: Efforts and delays for three sizing techniques relative to the theoretical lower bound. Fanout of first chain f_1 = 3.84 for all cases.

Equation 28 leads to delay values which are within 4% of the theoretical lower bound. But heuristic H2 also comes within 3% of the lower bound for large blocks though it is a bit slower than OPT for any given block size. This heuristic gives reasonable delays for small blocks too, while heuristic H1 doesn't perform well for small blocks. Hence H2, which

uses a simple fanout of 4 sizing in the entire path forms an effective sizing strategy for 2 level decode paths.

Large SRAMs typically use the Divided Word Line architecture which uses an additional level of decoding, and so we next look at sizing strategies for three-level decoders.

3.1.2.3 Application to a Three-Level Decoder

Figure 3.8 depicts the critical path for a typical decoder implemented using the Divided Word Line architecture. The path has three subchains, the predecode, the global word

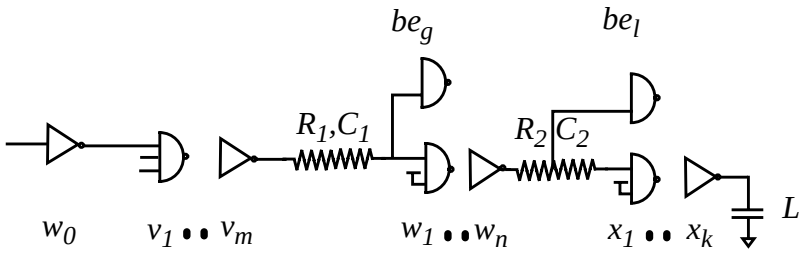


Figure 3.8: Critical path for a 3 level decoder

driver and the local word driver chains. Let the number of stages in these be *m, n and k* and the stage efforts be f_1, f_2 and f_3 respectively. Let be_g and be_l be the branching efforts as the input to the global and local word drivers and let *le* be the logical effort of the input NAND gate of these drivers. We will assume that the lengths of the global word lines are unaffected by the sizes in the local word driver chain, to simplify the analysis. Again from Appendix A, if there is no constraint on the number of stages (i.e. *m, n and k* can be any real numbers), then the lower bound for delay is achieved when $f_1=f_2=f_3=f \sim 4$ and the

number of stages in these subchains is given by Equations 29, 30 and 31. By rounding off

$$f^k = \frac{R_2 \cdot L \cdot b_l \cdot g}{2f} \cdot \left(1 + \sqrt{1 + \frac{4f}{R_2 \cdot C_2}}\right) \tag{29}$$

$$f^n = b_g \cdot g \cdot \frac{C_2}{C_1} \cdot \frac{\left(1 + \sqrt{1 + \frac{4f}{R_2 \cdot C_2}}\right)}{\left(\sqrt{\frac{4f}{R_1 \cdot C_1}} + 1 - 1\right)} \tag{30}$$

$$f^m = \frac{C/w_0}{2f} \cdot \left(1 + \sqrt{1 + \frac{4f}{R_1 \cdot C_1}}\right) \tag{31}$$

n and *k* to the nearest integers *in* and *ik* as discussed in the two level case, one can determine the optimum efforts for a real implementation. Similar to the two-level case, three alternatives exist: solving for exact values for f_2 and f_3 with these integral number of stages or use either of the two heuristics H1 or H2. The former approach of formulating a pair of equations akin to Equation 28 and solving them simultaneously is very cumbersome and we will not do that here. We will instead evaluate the performance of the two heuristics H1 and H2 discussed in the previous subsection and compare them to the theoretical lower bound.

Under H1, f_2 and f_3 are determined such that the input loading of the local and global word driver chains are kept the same as in the theoretical lower bound case. This is achieved by making $f_2 = f^{n/in}$ and $f_3 = f^{k/ik}$. Under H2, f_2 and f_3 are kept equal to f (~4). The relative delays of both these heuristics compared to the theoretical lower bound are shown in Table 3.2 for various SRAM sizes. The number of cells on the local word line is 128 for this table and we assume a 0.25µm technology (Appendix C). We observe that for large sizes, heuristic H1 performs better than H2. The main reason is that the wire delay due to the input loading of the word drivers in conjunction with the wire resistance becomes important for these sizes. Since H1 ensures that the input loading of the word

Size (kb)	from (30) n	from (29) k	in	ik	(H1) ($f_1 = 3.84$) f_2, f_3	H1 relative delay	H2 ($f_1 = f_2 = f_3 = 3.84$) relative delay
64	1.9	1.0	2	2	3.7, 2.0	1.04	1.04
256	2.6	1.6	2	2	5.6, 3.0	1.02	1.00
512	2.7	1.7	2	2	6.0, 3	1.02	1.01
1024	2.8	2.2	2	2	6.7, 4.4	1.03	1.07
4096	3.8	2.9	4	2	3.6, 7	1.02	1.05

Table 3.2: Relative delays with the two sizing heuristics compared to the theoretical lower bound.

drivers is the same as for the lower bound solution, it performs better. Conversely, by using wider wires, the impact of the wire resistance can be reduced to make the two heuristics give similar performance for large sized arrays. This can be seen in Table 3.3 which lists

Wire width (μm)	H1 relative delay compared to the lower bound	H2 relative delay compared to the lower bound	Wire delay (ns)
0.5	1.02	1.04	3.4
0.75	1.03	1.07	3.0
1.0	1.02	1.02	1.5

Table 3.3: Relative delays for a 1Mb SRAM with different wire sizes

the relative performances of H1 and H2 for a 1Mb SRAM with wire sizes of 0.5μm, 0.75μm and 1.0μm. For the 1.0μm wire width, H1 and H2 are identical. Thus if the wire delays are kept small, then the sizing rule of H2, which uses efforts in all the stages of about 4, provides a simple technique to size the decoders.

Minimum delay solutions typically burn a lot of power since getting the last bit of incremental improvement in delay requires significant power overhead. We will next look at some simple sizing heuristics which result in significant power savings at the cost of a modest increase in delay.

3.1.2.4 Sizing for Fast Low Power Operation

The main component of power loss in a decoder is the dynamic power lost in switching the large interconnect capacitances in the predecode, block select and word lines, as well as the gate and junction capacitances in the logic gates of the decode chain. Table 3.4 provides a detailed breakdown of the relative contribution from the different components to the total switching capacitance for two different SRAM sizes. The total switching capacitance is the sum of the interconnect capacitances (CI), the transistor capacitances internal to the predecoders (CP), the mosfet gate capacitance of the input gate of the global word drivers (CW1), the transistor capacitances internal to the global word drivers (CG), the mosfet gate capacitance of the input gate of the local word drivers (CX1) and the transistor capacitances internal to the local word driver (CL).

Size	Interconnect (CI)	Predecode Internal (CP)	Input of Global Word Driver (CW1)	Global Word Driver Internal (CG)	Input of Local Word Driver (CX1)	Local Word Driver Internal (CL)
16kb	11	20	20	10	30	7
1Mb	36	23	21	6	12.5	1.5

Table 3.4: Relative power of various components of the decode path in%

Size	Predecode (DP)	Predecode Wire (DRp)	Global Word Driver (DG)	Global Word Line (DRg)	Local Word Driver (DL)
16kb	56	2	26	1	15
1Mb	44	15.5	17.5	5	18

Table 3.5: Relative delay of various components of the decode path under H1 in %

Table 3.5 shows the relative breakdown of the total delay amongst the predecoder (DP), the predecode wire (DRp), the global word driver (DG), the global word line (DRg) and the local word driver (DL).

The two key features to note from these tables are that the input gate capacitance of the two word drivers contribute a significant fraction to the total switching capacitance due to

the large branching efforts, and that the delays of the two word drivers contribute a significant fraction to the total delay. In fact the input gate capacitance of the two word drivers are responsible for more of the decoder power than what is shown in the table as they also impact the sizing of the preceding stages. For example in the case of the 1Mb SRAM, by breaking down the power dissipation in the predecoders into two components, one directly dependent on the word driver sizes and the other independent on the word driver sizes, we find that 50% of the decoder power is directly proportional to the word driver input sizes. This suggests a simple heuristic to achieve a fast low power operation will be to reduce the input sizes of the two word drivers without compromising their speeds. This can be achieved by choosing minimum sized devices for the inputs of the word drivers (we use a minimum fet width which is 4 times the channel length), and then sizing each of the word driver chains independently to have the highest speed to drive their respective loads (which means they have a fanout of about 4 each). We label this heuristic as H3 in the following discussion. In order to evaluate the delay versus energy trade off obtained by this heuristic, we will compare it to the other two sizing heuristics and the optimum delay energy trade-off curve for a 14 input row decoder of a 1Mbit SRAM in Figure 3.9. All the data points in the figure are obtained from HSPICE [41] circuit simulation of the critical path with the respective device sizes. The optimum delay energy trade-off curve represents the minimum delay achievable for any given energy or equivalently, the minimum energy achievable for any given delay. It is obtained by numerically solving for the device sizes using the simple delay and energy equations described in the previous sections and is

described in more detail in Appendix B. Looking at the trade-off curve in Figure 3.9b, we

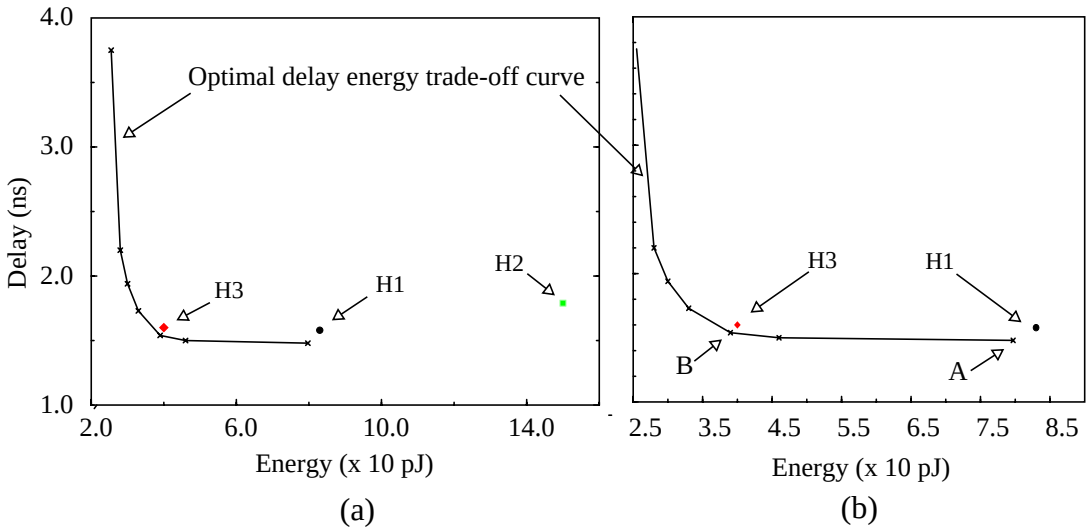


Figure 3.9: Delay energy performance of the three sizing heuristics and their comparison to optimal delay energy trade-off curve. (a) shows the full graph (b) shows the same graph magnified along the X axis.

can observe that device sizing allows for a trade off between delay and energy of over a factor of 2.5 along each dimension and hence is a great tool for finetuning the design. We note here that the fastest design point, labeled A, is very close to the theoretical lower bound. A factor of 2 savings in energy at only a 4% increase in delay can achieved by backing away from point A to point B on the trade-off curve. Heuristics H1-3 offer intuitive alternatives to the full blown numerical optimization required to generate points on the trade-off curve and their respective delay and energy are also shown in the figure. H1 comes within 4% in delay and energy of the fastest design point A. H2 uses small fan up ratios and hence consumes too much energy. H3 offers a reasonable fast low power design point and has 4% more delay and 2.5% more energy than point B.

3.2 Decoder Circuit Techniques

The total logical effort of the decode path is directly affected by the circuits used to construct the individual gates of the path. This effort can be reduced in two complementary ways, 1) by skewing the fet sizes in the gates and 2) by using circuit styles which implement the n-input logical AND function with the least logical effort. We first describe techniques to implement skewed gates in a power efficient way and then estimate the speed benefits of skewing and discuss sizing strategies for chains of skewed gates. We will then discuss ways of doing the n-input AND function efficiently and finally do a case study of three designs of a 4 to 16 predecoder, each progressively better than the previous one.

3.2.1 Reducing Logical Effort by Skewing the Gates

Since the word line selection requires each gate in the critical path to propagate an edge in a single direction, the fet sizes in the gate can be skewed to speed up this transition. By reducing the sizes for the fets which control the opposite transition, the capacitance of the inputs and hence the logical effort for the gate is reduced, thus speeding up the decode path. Separate reset devices are needed to reset the output and these devices are activated using one of three techniques: precharge logic uses an external clock, self resetting logic (SRCMOS) [20,42] uses the output to reset the gate, and delayed reset logic (DRCMOS) [28, 21, 43] uses a delayed version of one of the inputs to conditionally reset the gate.

Precharge logic is the simplest to implement, but is very power inefficient for decoders since the precharge clock is fed to all the gates so all gates are reset on every cycle. Since only a small percentage of these gates are activated for the decode, the power used to reset the gates of the precharge devices is mostly wasted. The SRCMOS and DRCMOS logic avoid this problem by activating the reset devices only for the gates which are active. In both these approaches, a sequence of gates, usually all in the same level of the decode hierarchy share a reset chain. In the SRCMOS approach, the output of this gate sequence

triggers the reset chain, which then activates the reset transistors in all the gates to eventually reset the output (Figure 3.10). The output pulse width is determined by the delay

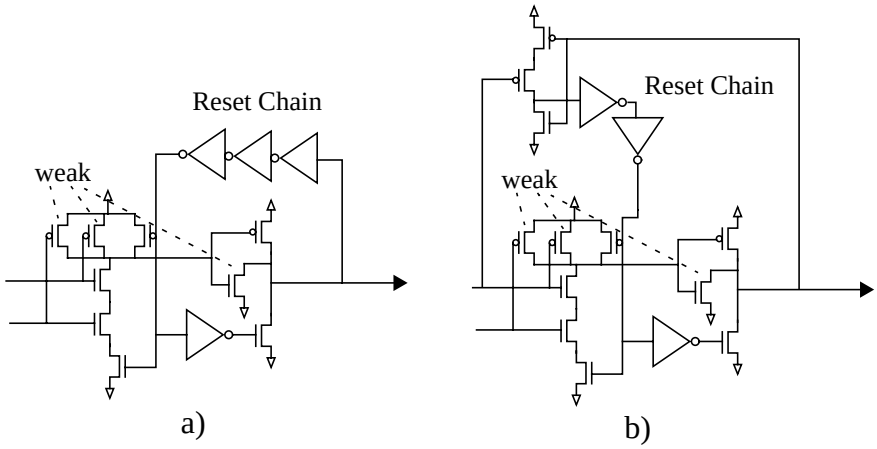


Figure 3.10: SRCMOS resetting technique, a) self-reset b) predicated self-reset

through this reset chain. If the delay of the reset chain cannot be guaranteed to be longer than the input pulse widths, then an extra series fet in the input is required to disconnect the pulldown stack during the reset phase, which will increase the logical effort of the gate. Once the output is reset, it travels back again through the reset chain to turn off the reset gates and get the gate ready for the next inputs. Hence, if the input pulse widths are longer than twice the delay of going around the reset chain, special care must to be taken to ensure that the gate doesn't activate more than once. This is achieved by predicating the reset chain the second time around with the falling input (Figure 3.10b) (another approach is shown in [42]).

The DRCMOS gate fixes the problem of needing an extra series nfet in the input gate by predicating the reset chain activation with the falling input even for propagating the signal the first time around the loop (Figure 3.11) (another version is shown in [43]). Hence the DRCMOS techniques will have the least logical effort and hence the lowest delay. The main problem with this approach is that the output pulse width will be larger than the input

pulse width and hence not many successive levels of the decode path can use this technique before the pulse widths exceed the cycle time.

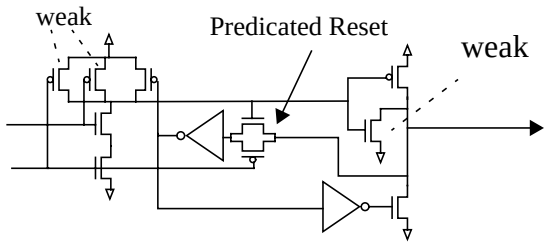


Figure 3.11: A DRCMOS Technique to do local self-resetting of a skewed gate

Thus, the strategy to build the fastest decoder will be to use DRCMOS gates broken occasionally by SRCMOS gates to prevent the pulse widths from becoming too large. Two external constraints might force a modification of this strategy. One is bitline power dissipation and the other is area overhead. Since the bitline power is directly proportional to the local word line pulse width, the latter must be controlled to limit the overall SRAM power. This implies that the final level of the decode tree has to be implemented in the SRCMOS style so that its output pulse width can be controlled independently. As these styles require a lot of fets for doing the reset, implementing the local word driver in this style will significantly add to the memory area, because of the large number of these drivers in the RAM. In a recent design described in [28], we found that the skewed gate implementation of the local word driver doubles its area and increases the total memory area by less than 4%, while it reduces the decode delay by only about 2/3 of a fanout 4 inverter delay. Hence it is preferable to implement the preceding decoder level (in the global word driver or the block select driver) in the SRCMOS style and implement the local word driver in the conventional unskewed style. Of the global word driver and the block select driver, the latter is a better candidate for the SRCMOS implementation since the reset chain for the block select driver can also be used to clock the sense amplifiers within the block (see Chapter 4). Also most SRAM implementations have asymmetric decoder hierarchies in which the number

of global word drivers is more than the number of block select drivers, and hence the latter can tolerate more area overhead.

We will next estimate the maximum achievable speedup and also develop sizing strategies for skewed gate decoders. Figure 3.12 shows a chain of maximally skewed inverters with the input for each connected only to either the nfet or the pfet. The logical effort of

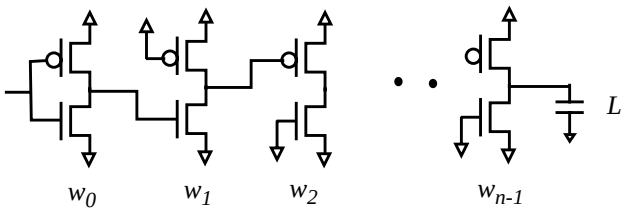


Figure 3.12: Chain of maximally skewed inverters

the nfet input gate is α (=1/3 in the above example) and for the pfet input gate is β (=2/3). We will assume that the skewed inverter has the same junction parasitics at its output as the unskewed inverter, because of the presence of the reset devices. The total delay can then be expressed as the sum of the stage and parasitic efforts of each gate as in Equation 32. The minimum delay condition occurs when the stage efforts are equal to *f* (Equation

$$D = \frac{\alpha w_1}{w_0} + \frac{\beta w_2}{w_1} \bullet\bullet \frac{L}{w_{n-1}} + n \cdot p \tag{32}$$

$$\frac{\alpha w_1}{w_0} = \frac{\beta w_2}{w_1} = \frac{L}{w_{n-1}} = f \tag{33}$$

$$f^n \sim \gamma^{n-1} \cdot L / w_0 \tag{34}$$

33) and the product of the stage efforts equals the total path effort (Equation 34) (this relation is approximate for odd *n*, but for large *n* the error is very small). Here γ is the geometric mean of α and β and equals 0.47. Let *q* be *f*/γ then the total delay can be rewrit-

ten as in Equation 35 and optimum q is the solution of Equation 36. We note here that

$$D/\gamma = n \cdot (q + p/\gamma) \tag{35}$$

$$\ln(q) - 1 - \frac{p/\gamma}{q} = 0 \tag{36}$$

Equations 34-36 are still valid when the arbitrary skewing strategy provided γ is the geometric mean of all the logical efforts. For p=1.33, Equation 36 results in q~4.8, and hence f ~ 2.4 and the delay per stage is about 30% lower than that for an unskewed chain. But the optimal number of stages is also smaller, giving a net reduction of about 40% for the total delay. For skewed chains with intermediate interconnect, Equations 19, 20, 29, 30 and 31 are still valid provided f is replaced with q and C, C_1, C_2 *and* L are replaced with C/γ, C_1/γ, C_2/γ and L/γ respectively and hence the sizing heuristic H3 can be used for high speed and low power skewed gate decoders too.

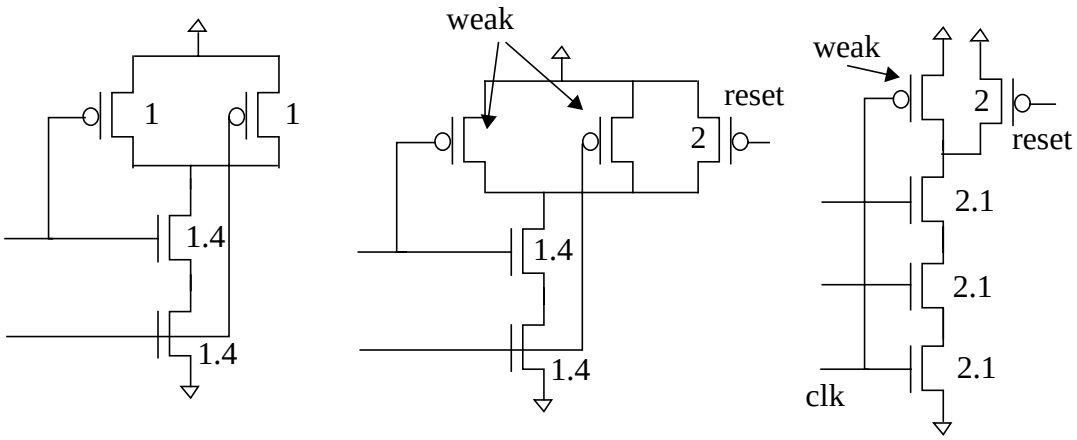
3.2.2 Performing an n-input AND Function with Minimum Logical Effort

The n-input AND function can be implemented via different combination of NANDs, NORs and Inverters. Since in current CMOS technologies, a pfet is at least two times slower than an nfet, NORs are inefficient and so the AND function is best achieved by a combination of NANDs and inverters.We will next discuss the three basic styles for building a NAND gate.

3.2.2.1 Conventional Series Stack NAND gate

Figure 3.13a shows the simplest implementation of a 2 input NAND gate in the series stack topology. In long channel devices, the nfets will be made twice as big as in an inverter, but in sub micron devices, because of velocity saturation in the inverter fet, its drive current will not twice as big as in the series stack. For our 0.25μm technology (Appendix C), the nfets need to be sized to be 1.4 times as large as the inverter nfet to

enable the gate to have the same drive strength as the inverter. The pfets are the same size as that in the inverter, thus resulting in a logical effort of 1.13 for this gate. In a pulsed decoder since both the inputs are guaranteed to fall low, the pfet sizes can be reduced by half and still maintain the same delay for the rising output [23]. This reduces the logical



(a) Non-skewed (b) Skewed (c) Clocked

Figure 3.13: Static 2 input NAND gate for a pulsed decoder: (a) Non-skewed (b) Skewed and (c) Clocked

effort of this gate to 0.8, down from 1.13 for the conventional static gate. Gates with more than 2 inputs will have a logical effort of at least 0.8 (for e.g. in a 3 input NAND gate, the nfet is sized to be 2.1 and pfet is 0.67 which gives a logical effort of 0.92), while a cascade of 2-input gates will have a total logical effort less than 0.8. Hence AND functions with 3 or more inputs are best built with cascades of 2-input gates to obtain the least logical effort in this style. Weakening the pfets further reduces the logical effort and for maximum skewing, when the input is not connected to the pfet at all, the logical effort goes down to 0.47.

In pulsed decoders the input stage of the decoder needs to be gated by a clock; we can take advantage of this by implementing the NAND function in the domino style, thus lowering the loading on the address inputs. Local resetting strategies discussed in the previous section can be applied here to reduce the clock load. The precharge pfet connected to the

clock can be weakened and a separate pfet for the precharge which will be much larger can be activated locally within the active gate.

This circuit style is very simple and robust to design in and is well suited for use in the predecoders. In power conscious designs, the input stage of the local and global word drivers will have minimum sized fets to reduce power and hence the gate cannot be skewed much. Since the 2-input NAND gate in this style does no worse than an inverter, it can also be used as the input stage of the word drivers without any delay penalty. But the NAND gates at this location can be constructed with fewer fets in the source coupled style which we will discuss next.

3.2.2.2 Source Coupled NAND gate

Two fets can be eliminated from the previous design by applying one of the inputs to the source of an nfet and is called the source coupled style by [24, 26]. Figure 3.14 shows a 2-input NAND implemented using only two fets. This gate can be at least as fast as an

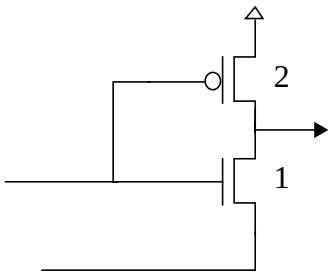


Figure 3.14: Source Coupled NAND gate

inverter if the branching effort of the source input is sufficiently large and if the total capacitance on the source line is much greater than the output load capacitance for the gate. To see this, lets compare the delay of the two paths shown in Figure 3.15. The path on the left (labeled A) has the NAND gate embedded within it and the path on the right (labeled B) has an identically sized inverter instead. Both the inputs to the NAND gate in A as well as the input to the inverter in B are driven by inverters of size w_0 through an

interconnect of capacitance *C* and have the same branching effort of *b.* The output load for

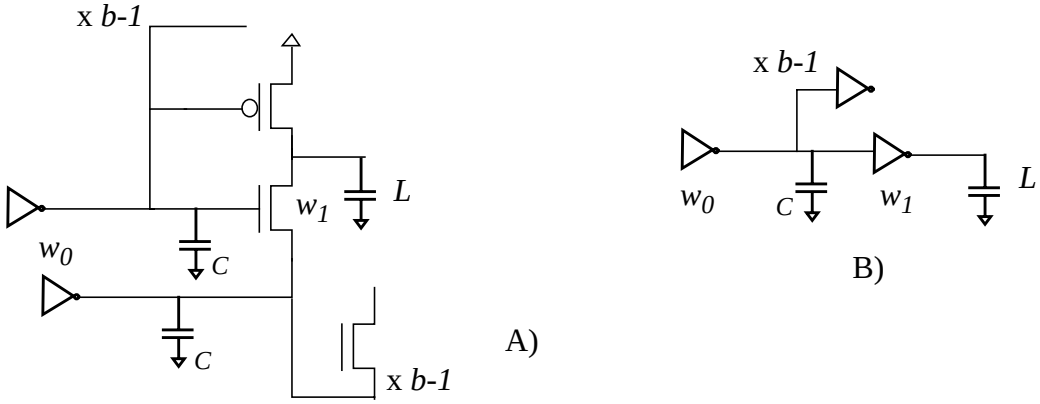


Figure 3.15: Comparison of delay of the source coupled NAND gate

both the paths is *L*. For path B, we can write the delay as in Equation 37. For path A, if the

$$D = \frac{C + b \cdot C_g \cdot w_1}{w_0} + \frac{L}{w_1} + \textit{parasitic delay} \qquad (37)$$

source input falls low much before the gate input and the source line capacitance is much larger than L, then it can be treated as a virtual ground and hence the delay of the gate will be the same as the inverter. If the gate input arrives much earlier than the source input, then the delay is determined by the source input and can be written as in Equation 38. The total

$$D = \frac{C + (b \cdot C_j/6 + C_j) \cdot w_1 + L}{w_0} + r \cdot \frac{L}{w_1} + \textit{parasitic delay} \qquad (38)$$

capacitance seen by the input inverter is the sum of the line capacitance, the junction capacitance of *b* nfets (since nfet size for a 3rd of the inverter size and since their source junctions are shared between adjacent rows their contribution to the loading is $C_j/6$), the junction capacitance of the drains of the nfet and the pfet of the gate and finally the capacitance of the load. Here we assume that the nfet source is shared to halve its junction

capacitance. Since the nfet source starts off at a high voltage and then falls low, the nfet threshold will also be high to begin with due the body effect and then will reduce which implies that the average drive current through this nfet will be slightly smaller than that in an inverter. This is captured in the factor *r* in the second term. To get some ball park numbers for *b*, lets substitute *r=1*, and $L=4C_g w_1$ in Equation 38 and compare it to Equation 37. We find that the NAND gate path will be faster than the inverter if $b > 6$. The branching effort and the input capacitance for the source input can be large in the case of the word drivers due to the interconnect and the loading from the other word drivers. The word drivers can be laid out such that the source area for adjacent drivers can be shared thus halving the junction capacitance [26].

3.2.2.3 NOR style NAND gate

Since a wide fanin NOR can be implemented with very small logical effort in the domino circuit style a large fanin NAND can be implemented doing a NOR of the complementary inputs (Figure 3.16) and is a candidate for building high speed predecoders. The rationale for this approach is that with increasing number of inputs, nfets are added in parallel, thus keeping the logical effort a constant, unlike in the previous two

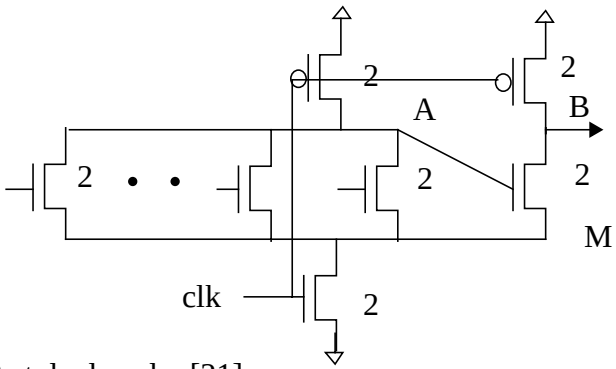


Figure 3.16: NOR style decoder [21]

styles. To implement the NAND functionality with NOR gates, Nambu et. al. in [21] have proposed a circuit technique to isolate the output node of an unselected gate from discharging and is reproduced in the figure. An extra nfet (M) on the output node B, shares the same source as the input nfets, but its gate is connected to the output of the NOR gate

(A). When clock (clk) is low, both nodes A and B are precharged high. When clock goes high, the behavior of the gate depends on the input values. If all the inputs are low, then node A remains high, while node B discharges and the decoder output is selected. If any of the inputs are high, then node A discharges, shutting off M and preventing node B from discharging and hence causing that output to remain unselected. As this situation involves a race between A and B, the gate needs to be carefully designed to ensure robust operation.

We will next compare this style with the conventional series stack style in the context of implementing predecoders.

3.2.2.4 Case Study of a 4 to 16 Predecoder

Lets consider the design of a 4 to 16 predecoder which needs to drive a load which is equivalent to 76 inverters of size 8 (using the convention of section 3.1). This load is typical when the predecode line spans 256 rows. We will do a design in both the series stack style and the NOR style, and for each consider both the non-skewed as well as the skewed versions. To have a fair comparison between the designs, we will size the input stage in

each such that the total input loading on any of the address inputs is the same across the designs.

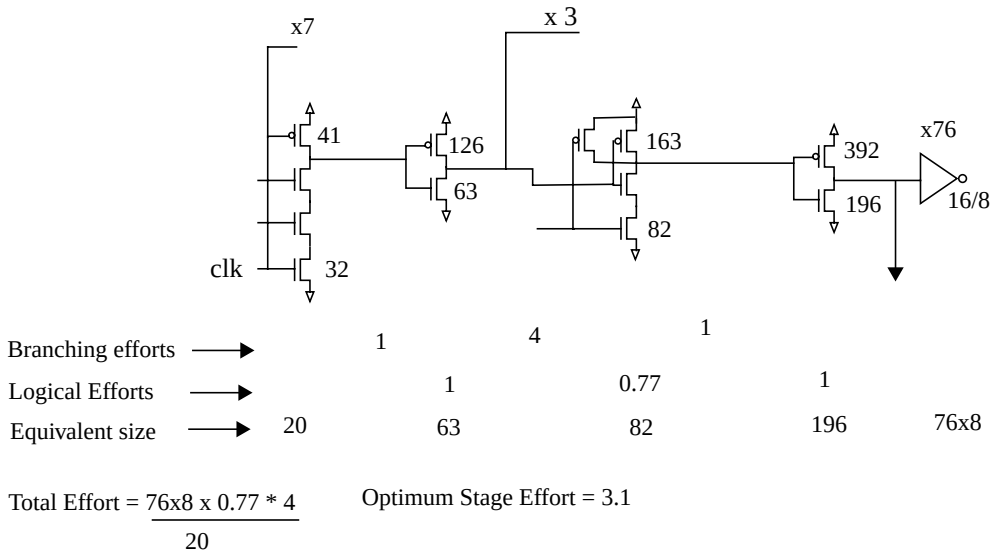


Figure 3.17: 4 to 16 predecoder in conventional series nfet style with no skewing

Figure 3.17 shows the design using the conventional series stack nfets and with no gate skewing. The decoding is done in two levels, first two address bits are decoded to select one of 4 outputs, which are then combined in the second level to activate one of 16 outputs. The fet sizes for the input stage are shown in the figure and result in an output drive strength which is equivalent to an inverter of size 20. There are 4 stages in total in the critical path and for minimum delay, the efforts of each stage are made equal to the fourth root of the total path effort. The optimal sizes for the stages is also shown in the figure. HSPICE simulations for this design lead to a delay of 316pS (which corresponds to about 3.5 fanout 4 loaded inverter delays) and power of 1.1mW at 2.5V and is summarized in Table 3.6.

Figure 3.18 shows the skewed gate version of the previous design, with all the gates skewed such that the fets not required during the output selection phase are made to be

minimum size and serve mainly to keep the voltage during the standby phase. The gates

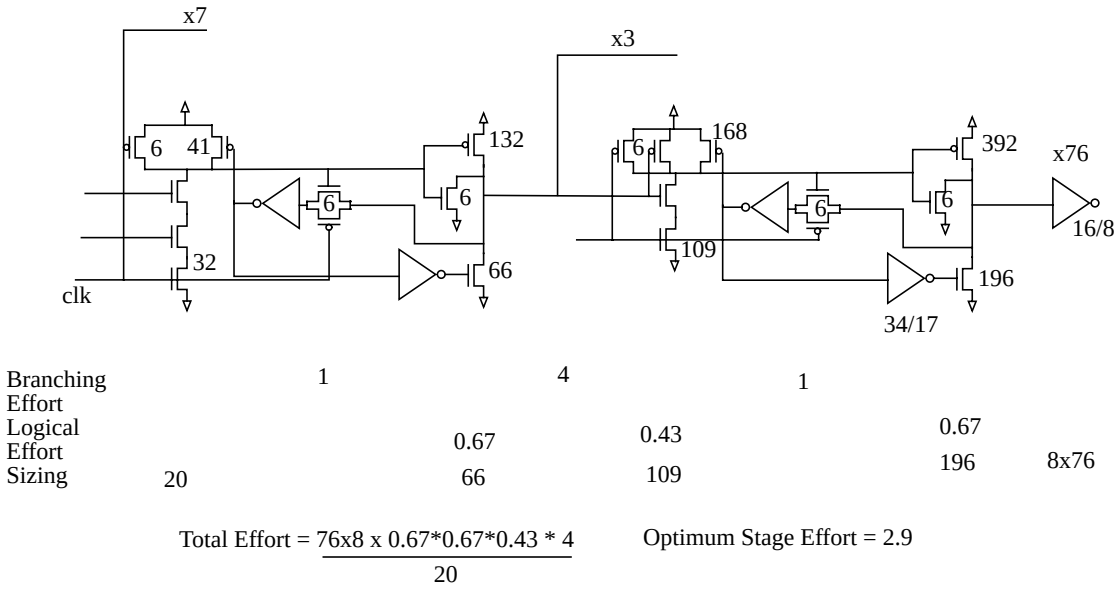


Figure 3.18: Critical path for 4 to 16 decoder in conventional static style with skewing

are locally reset in the DRCMOS style. Consequently, the logical effort of the two inverters is reduced to 0.67 and that for the second level NAND gate is reduced to 0.43, lowering the total path effort by a factor of 4. The input stage has the same size as before while all the other sizes are recalculated to yield the minimum delay, by equalizing the stage efforts and is shown in the figure. HSPICE simulated delay for this design is 234pS (which corresponds to 2.6 fanout 4 loaded inverters) and the power is 1.1mW. The delay is lower by 26% because of the skewing and the power is almost the same because the extra power loss in reset chain is compensated by the lower clock power.

Figure 3.19 shows the design using the NOR front end. There are 16 such units, one for each predecoder output. The branching factor of each address bit at the input stage is 8,

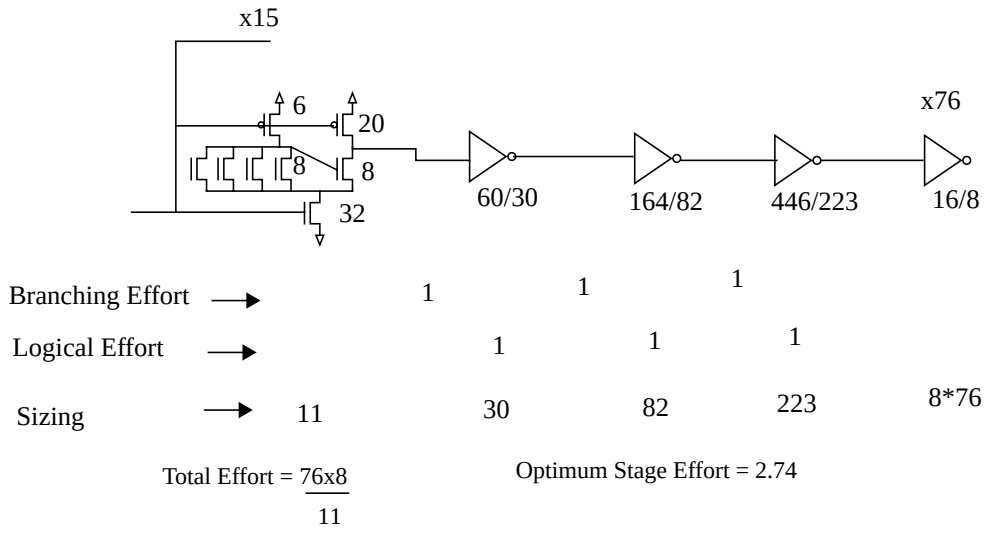


Figure 3.19: NOR style predecoder with no skewing

which is 4 times bigger than that for the previous two designs. Hence the input fets in the first stage have a size of only 8, to achieve the same total input loading. The evaluate nfet at the bottom is chosen to be 32 for good current drive strength at the output and is arrived at by simulations. The output drive of the first stage is equivalent to an inverter of size 11 and is about half as that in the previous two designs. The total path effort is lower than that in the first design by a factor of 2 as it saves a factor of 4 in the branching effort at the 3rd stage, but gives up a factor of 2 in the drive strength of the input stage. The total delay from HSPICE is 284pS (which corresponds to 3.2 fanout 4 loaded inverters), for a power dissipation of 1.2mW.

The final design shown in Figure 3.20 combines skewing and local resetting in the DRCMOS style. The total path effort is reduced by a further factor of 2.6 compared to the

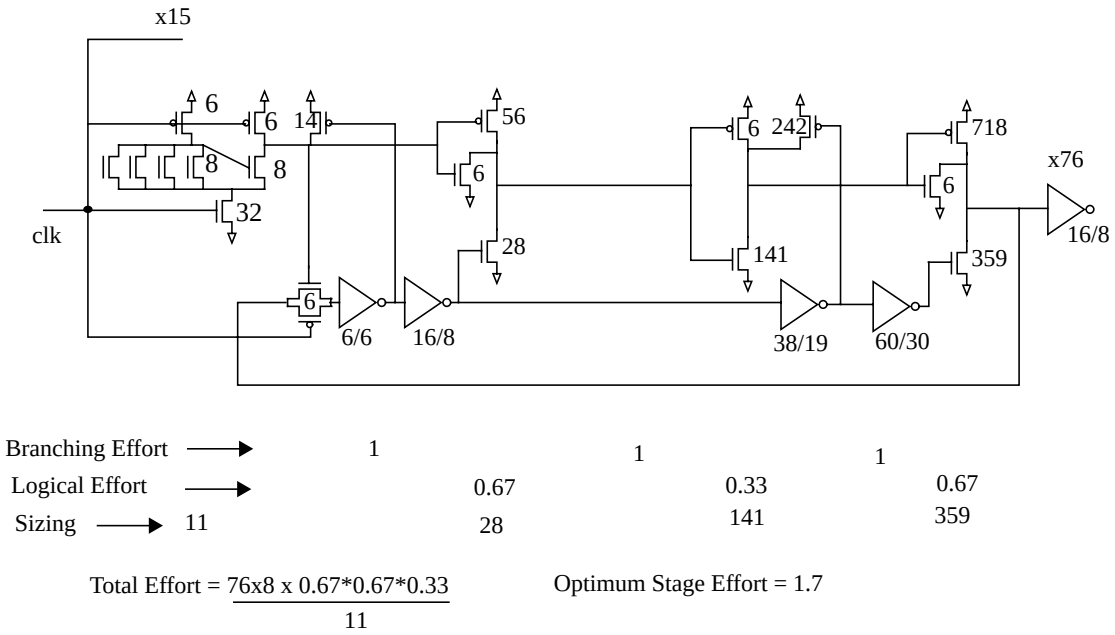


Figure 3.20: NOR style 4 to 16 predecoder with maximal skewing and DRCMOS resetting

skewed design of Figure 3.18, as the second skewed inverter has a logical effort which is 1.3 times lower than the skewed second level NAND gate. This results in the fastest design with a delay of 202pS (2.25 fanout 4 loaded inverters) which is about 36% lower than the non skewed version with series stack nfets. We note here that this number is almost the same as reported in [21], but we differ on what we ascribe the delay gains to. From the examples it is clear that the major cause for delay improvement in this style is because of the skewing which buys almost 26% of the reduction as seen row 2 of Table 3.6. The remaining 10% gain comes from using the NOR front end. Nambu et. al., have reversed this allocation of gains in their paper [21]. The power dissipation in the above design is kept to about 1.33mW, because of the DRCMOS reset technique (we include the power dissipation in the unselected NOR gates, which is not shown in the above figure for sake of clarity).

This design example illustrates the speed and power advantage of the NOR style design with skewing and local resetting, for implementing the predecoders. The relative advantage of this approach increases further with larger output loads and increasing number of address bits.

Circuit Style	Delay (pS)/ number of fanout 4 loaded inverters	Power (mW)
Series nfet without skewing (Fig. 16)	316 / 3.5	1.1
Series nfet with skewing (Fig. 17)	234 / 2.6	1.1
NOR style without skewing (Fig. 18)	284 / 3.2	1.2
NOR style with skewing (Fig. 19)	202 / 2.25	1.3

Table 3.6: Delay and Power Comparisons of Various Circuit Styles in 0.25μm process at 2.5V. Delay of a fanout 4 loaded inverter is 90pS.

3.3 Optimum Decode Structures - A Summary

Based on the discussions in the previous sections, we can now sketch out the optimal decoder structure for fast low power SRAMs (Figure 3.21). Except for the predecoder, all the higher levels of the decode tree should have a fan-in of 2 to minimize the power dissipation as we want only the smallest number of long decode wires to transition. This implies that the 2-input NAND function can be implemented in the source coupled style without any delay penalty, since it does as well as an inverter. The local word driver will have two stages in most cases, and have four when the block widths are very large. In the latter case, unless the applications demand it, it will be better to re-partition the block to be less wide in the interests of the word line RC delay and bitline power dissipation. Skewing the local word drivers for speed is very expensive in terms of area due to the large numbers of these circuits. Bitline power can be controlled by controlling the word line pulse width. This is easily achieved by controlling the block select pulse width. Hence the block select signal should be connected to the gate of the input NAND gate and the global word driver should be connected to the source. Both the block select and the global word line drivers

should have skewed gates for maximum speed. Both these drivers will have anywhere

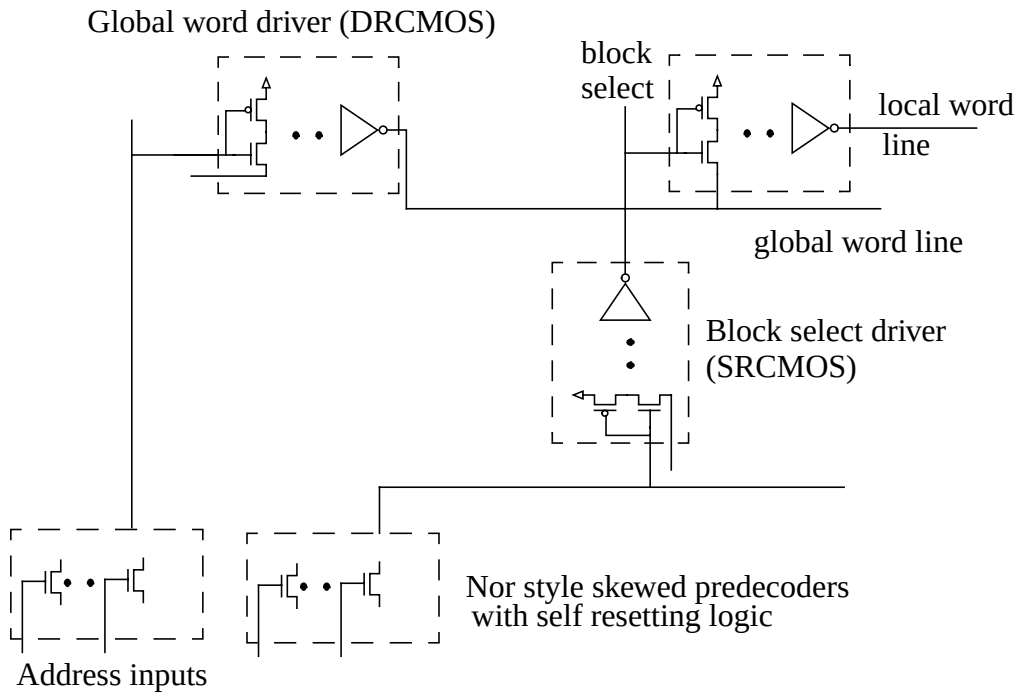


Figure 3.21: Schematic of fast low power 3 level decoder structure

from 2 to 4 stages depending on the size of the memory. The block select driver should be implemented in the SRCMOS style to allow for its output pulse width to be controlled independently of the input pulse widths. The global word driver should be made in the DRCMOS style to allow for generating a wide enough pulse width in the global word line to allow for sufficient margin of overlap with the block select signal. Since in large SRAMs the global word line spans multiple pitches, all the resetting circuitry can be laid out local to each driver. In cases where this is not possible, the reset circuitry can be pulled out and shared amongst a small group of drivers [21]. Predecoder performance can be significantly improved at no cost in power by skewing the gates and using local resetting techniques. The highest performance predecoders will have a NOR style wide fanin input stage followed by skewed buffers. When this is coupled with a technique like that presented in [21] to do a selective discharge of the output, the power dissipation is very reasonable compared to the speed gains that can be achieved. With the NOR style prede-

coder the total path effort becomes independent of the exact partitioning of the decode tree, between the block select and global word decoders. This will allow the SRAM designer to choose the best memory organization, based on other considerations.

Transistor sizing offers a great tool to trade off delay and energy of the decoders. Full blown numerical optimization techniques can be used to obtain the various design points. The simple sizing heuristic of keeping a constant fanout of about 4 works well as long as the wire delays are not significant. For large memories where the wire delay is significant, either the wire widths can be increased to reduce their impact or the alternative heuristic, which relies on calculating the optimal transistor sizes for the gates at the ends of these wires, can be used. For fast lower power solutions, the simple heuristic of reducing the sizes of the input stage in the higher levels of the decode tree allow for good trade-offs between delay and power.

Getting the decode signal to the local word lines in a fast and low power manner solves half of the memory design problem. We will next look at techniques to solve the other half, namely build a fast low power data path.

Chapter

4

Fast Low Power SRAM Data Path

4.1 Introduction

In an SRAM, switching of the bitlines and I/O lines and biasing the sense amplifiers consume a significant fraction of the total power, especially in wide access width memories. This chapter investigates techniques to reduce SRAM power without hurting performance by using tracking circuits to limit bitline and I/O line swing, and aid in the generation of the sense clock to enable clocked sense amplifiers.

Traditionally, the bitline swings during a read access have been limited by using active loads of either diode connected nMOS or resistive pMOS [26, 44]. These devices clamp the bitline swing at the expense of a steady bitline current. A more power efficient way of limiting the bitline swings is to use high impedance bitline loads and pulse the word lines [31-34]. Bitline power can be minimized by controlling the word line pulse width to be just wide enough to guarantee the minimum bitline swing required for sensing. This type of bitline swing control can be achieved by a precise pulse generator that matches the bitline delay. Low power SRAMs also use clocked sense amplifiers to limit the sense power. These amplifiers can be based on either a current mirror [24-25, 45-46] or a cross-coupled latch [20-22, 47-48]. In the former, the sense clock turns on the amplifier sometime before the sensing, to set up the amplifier in the high gain region. To reduce power, the amount of time the amplifier is ON is minimized. In the latch type amplifiers, the sense clock starts the amplification and hence the sense clock needs to track the bitline

delay to ensure correct and fast operation.

The prevalent technique to generate the timing signals within the array core essentially uses an inverter chain. This can take one of two forms - the first kind relies on a clock phase to do the timing (Figure 4.1a) [49] and the second kind uses a delay chain within the accessed block and is triggered by the block select signal (Figure 4.1b) [50]. The main

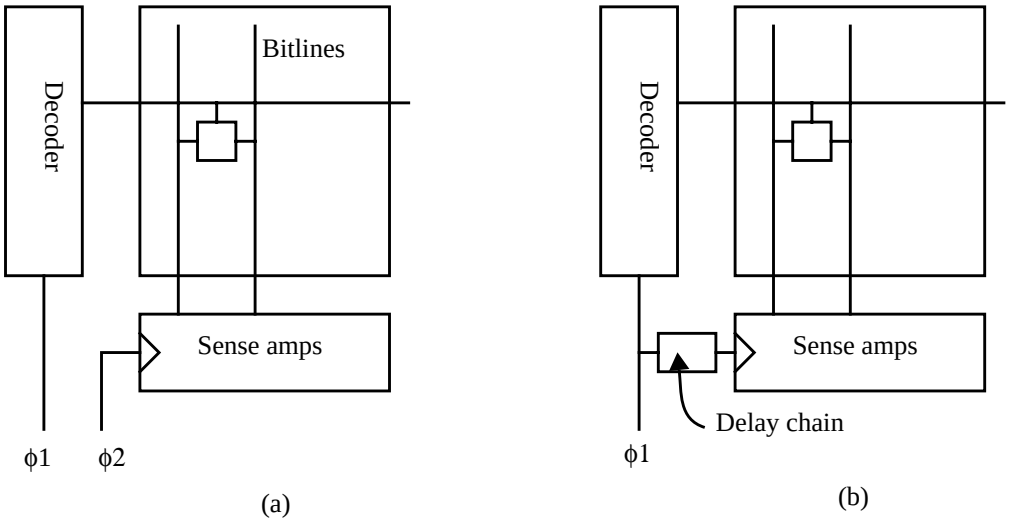


Figure 4.1: Common sense clock generation techniques

problem in these approaches is that the inverter delay does not track the delay of the memory cell over all process and environment conditions. The tracking issue becomes more severe for low power SRAMs operating at low voltages due to enhanced impact of threshold and supply voltage fluctuations on delays. If *T* is the delay of a circuit running

$$\frac{\sigma_T^2}{T^2} \propto \frac{\sigma_{Vdd}^2 + \sigma_{Vt}^2}{(Vdd - Vt)^2} \qquad (1)$$

off a supply of *Vdd* with transistor thresholds as *Vt*, then using a simple α-power model for

the transistor current [67], one can relate the standard deviations of these parameters as in Equation 1, which shows that percentage delay variations are inversely proportional to the gate overdrive. Figure 4.2 plots the ratio of bitline delay to obtain a bitline swing of 120mV from a 1.2V supply and the delay of an inverter delay chain. The process and temperature are encoded as XYZ where X represents the nMOS type (S = slow, F = fast, T = typical), Y represents the pMOS type (one of S, F, T) and Z is the temperature (H for 115C and C for 25C). The S and F transistors have a 2 sigma threshold variation unless subscripted by a 3 in which case they represent 3 sigma threshold variations. The process used is a typical 0.25μm CMOS process and simulations are done for a bitline spanning 64 rows. We can observe that the bitline delay to inverter delay ratio can vary by a factor of

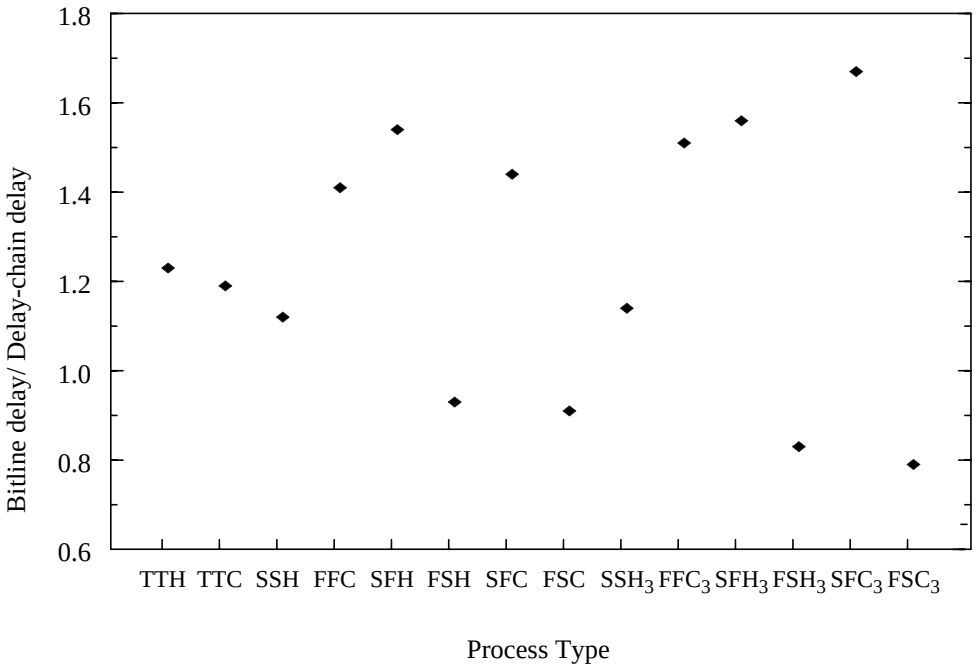


Figure 4.2: Delay matching of inverter chain delay stage with respect to bitline delay

two over these conditions, the primary reason being that while the memory cell delay is mainly affected by the nMOS thresholds, the inverter chain delay is affected by both nMOS and pMOS thresholds. The worst case matching for the inverter delay chain occurs for process corners where the nMOS and the pMOS thresholds move in the opposite

direction. There are two more sources of variations that are not included in the graph above and make the inverter matching even worse. The minimum sized transistors used in memory cells are more vulnerable to *delta-W* variations than the non-minimum sized devices used typically in the delay chain. Furthermore, accurate characterization of the bitline capacitance is also required to enable a proper delay chain design.

All the sources of variations have to be taken into account in determining the speed and power specifications for the part. To guarantee functionality, the delay chain has to be designed for worst case conditions which means the clock circuit must be padded in the nominal case, degrading performance. The following two sections look at using replicas of the bitline delay to control both the sense timing and the word line pulse width. These circuits track the bitline delay much better [± 10%] and improve both power and access time especially in high density SRAMs with long bitlines. Section 4.4 then describes experimental results of SRAMs using replica timing generation described in the previous sections. These techniques reduce the read power, but leave the write power unchanged. Section 4.5 looks at a technique of doing low swing write in an SRAM and discusses some experimental results.

4.2 Replica delay element based on capacitance ratioing

Memory timing circuits need a delay element which tracks the bitline delay but still provide a large swing signal which can be used by the subsequent stages of the control logic. The key to building such a delay stage is to use a delay element which is a replica of the memory cell connected to the bitline, while still providing a full swing output. This section uses a normal memory cell driving a short bitline, while Section 4.3 uses a number of memory cells connected to a replica of the full bitline. The short bitline's capacitance is set to be a fraction of the main bitline capacitance. The value is determined by the required bitline swing for proper sensing. For the clocked voltage sense amplifiers we use (Figure

4.3), the minimum bitline swing for correct sensing is around a tenth of the supply. An

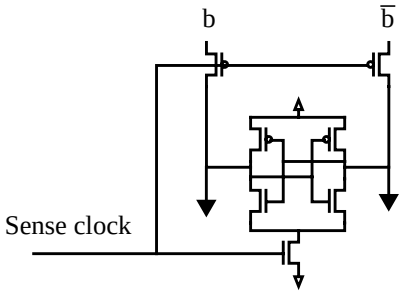


Figure 4.3: Latch type sense amplifier

extra column in each memory block is converted into the replica column by cutting its bit-line pair to obtain a segment whose capacitance is the desired fraction of the main bitline (Figure 4.4). The replica bitline has a similar structure to the main bitlines in terms of the wire and diode parasitic capacitances. Hence its capacitance ratio to the main bitlines is

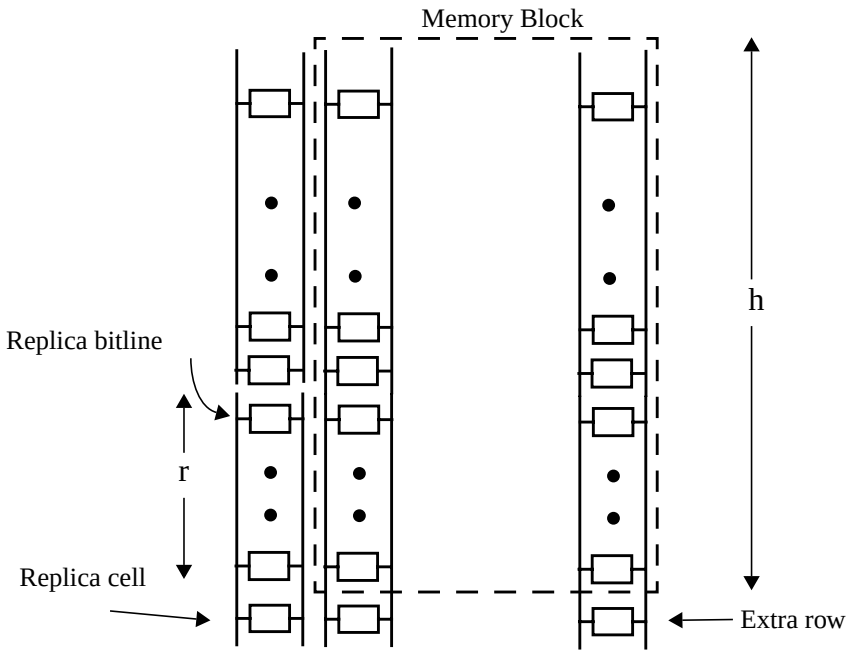


Figure 4.4: Replica column with bitline capacitance ratioing

set purely by the ratio of the geometric lengths, *r/h*. The replica memory cell is programmed to always store a zero so that, when activated, it discharges the replica bitline.

The delay from the activation of the replica cell to the 50% discharge of the replica bitline tracks that of the main bitline very well over different process corners (Figure 4.5 -circles).

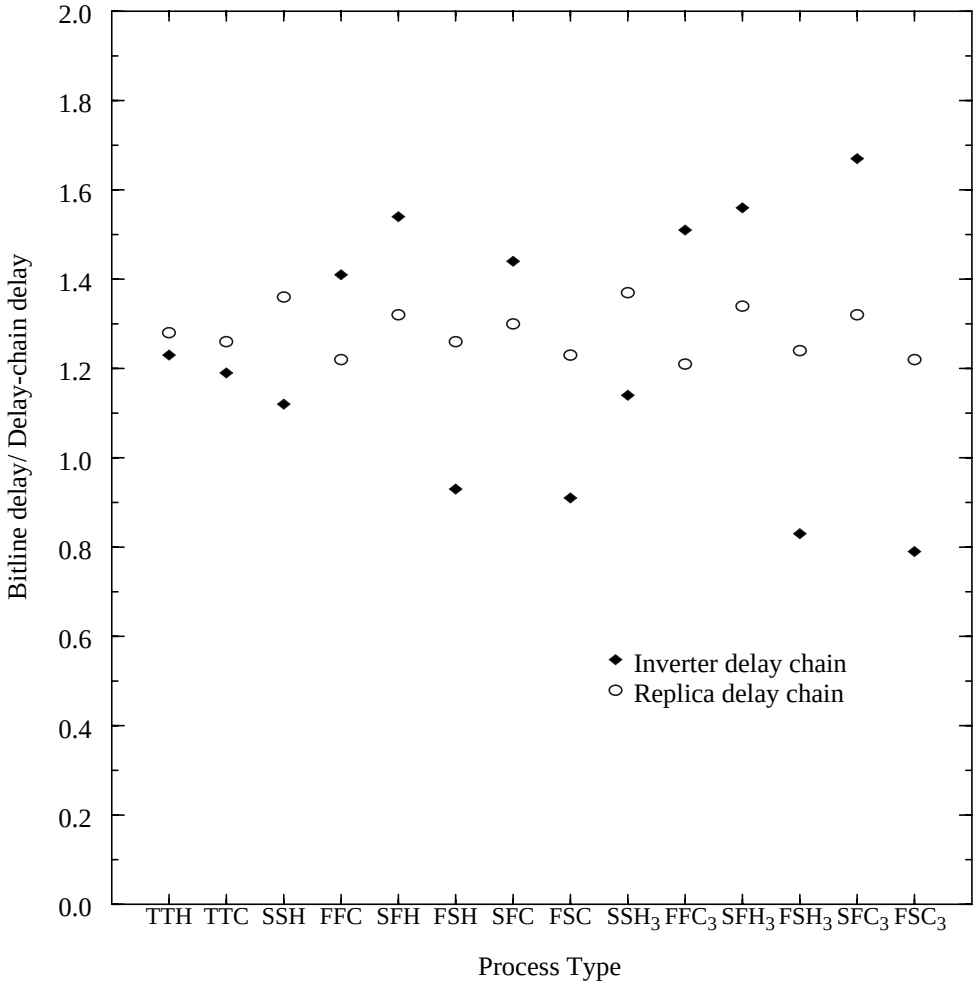


Figure 4.5: Comparison of delay matching of replica versus inverter chain delay elements

The delays can be made equal by fine tuning of the replica bitline height using simulations. The replica structure takes up only one additional column per block and hence has very little area overhead.

The delay element is designed to match the delay of a nominal memory cell in a block. But in an actual block of cells, there will be variations in the cell currents across the cells in the block. Figure 4.6 displays the ratio of delays for the bitline and the delay elements for varying amounts of threshold mismatch in the access device of the memory cell, compared to the nominal cell. The graph is shown only for the case of the accessed cell being

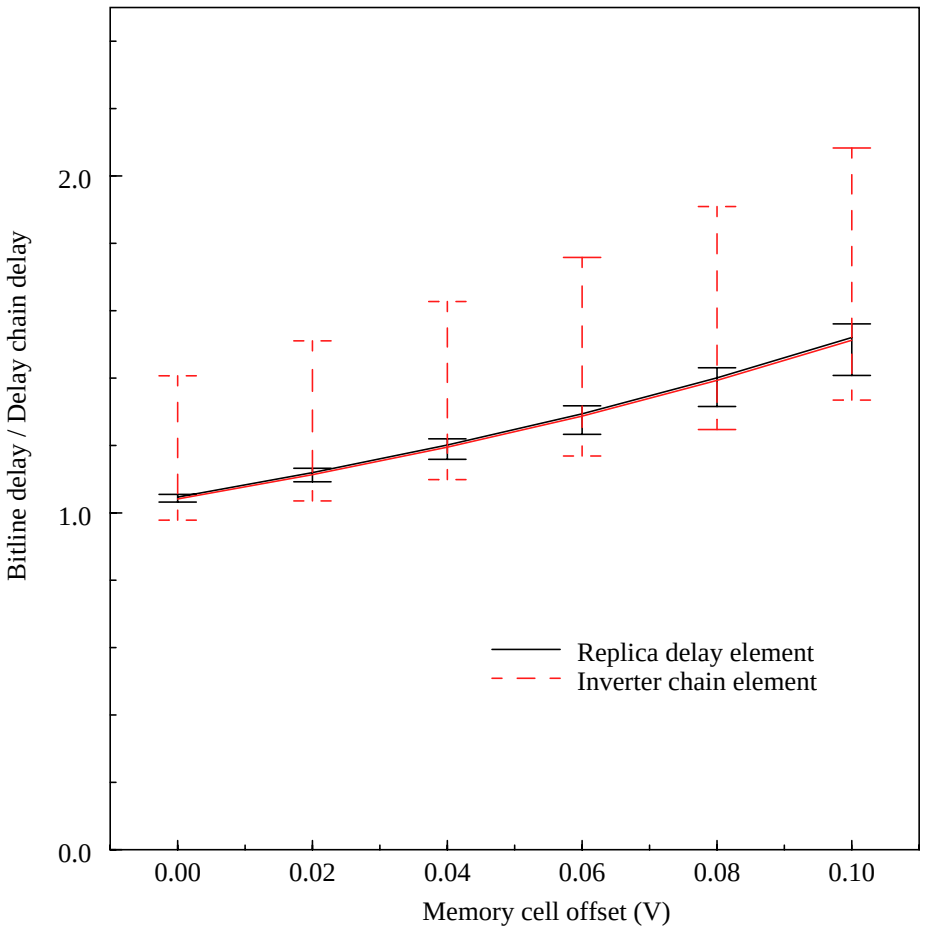


Figure 4.6: Matching comparisons over varying cell threshold offsets in a 0.25um @ 1.2V

weaker than the nominal cell as this would result in a lower bitline swing. The curves for the inverter chain delay element (hatched) and the replica delay element (solid) are shown with error bars for the worst case fluctuations across process corners. The variation of the delay ratio across process corners in the case of the inverter chain delay element is large even with zero offset in the accessed cell and grows further as the offsets increase. In the

case of the replica delay element, the variation across the process corners is negligible at zero offsets and starts growing with increasing offsets in the accessed cell. This is mainly due the adverse impact of the higher nMOS thresholds in the accessed cell under slow nMOS conditions. It can be noted that the tracking of the replica delay element is better than that of the inverter chain delay element across process corners, even with offsets in the accessed memory cell.

Though the output of the replica delay stage is a full swing signal, it has a very slow

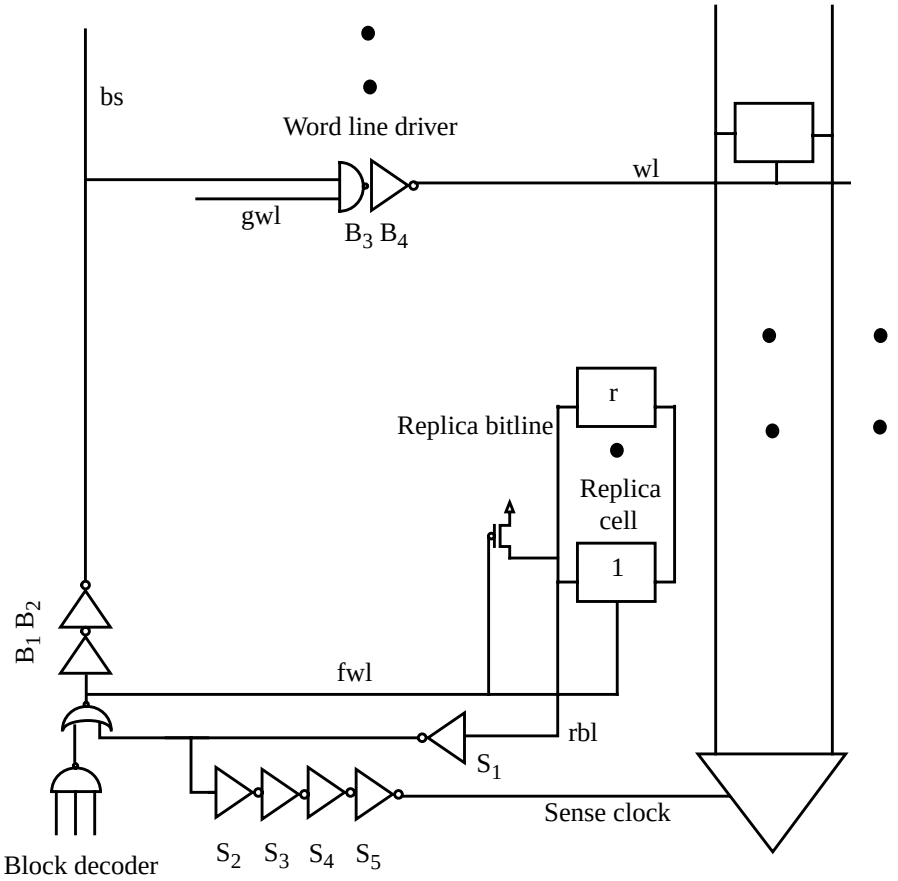


Figure 4.7: Control circuits for sense clock activation and word line pulse control

slew rate which is limited by the memory cell current. Hence this signal needs to be buffered up before it can be used by the subsequent logic stages. The consequent buffer delay

will need to be compensated since it will not track the bitline delay. One approach to deal with this extra delay is to try to minimize it by carefully partitioning the blocks such that the load that needs to be driven by the replica signal is minimized and is explained further in Section 4.4.2. An alternative approach is to hide this extra delay by pipelining the access to the delay circuit and overlapping it with the access to the word line. The circuits to do this are shown in Figure 4.7. The block decoder activates the replica delay cell (node *fwl*). The output of the replica delay cell is fed to a buffer chain to start the local sensing and is also fed back to the block decoder to reset the block select signal. Since the block select pulse is ANDed with the global word line signal to generate the local word line pulse, the latter's pulse width is set by the width of block select signal. It is assumed that the block select signal doesn't arrive earlier than the global word line. The delay of the buffer chain to drive the sense clock is compensated by activating the replica delay cell with the unbuffered block select signal. The delay of the five inverters in the buffer chain, S_1 to S_5, is set to match the delay of the four stages, B_1 to B_4, of the block select to local word line path (the sense clock needs to be a rising edge). The problem of delay matching has now been pushed from having to match bitline and inverter chain delay to having to match the delay of one inverter chain to a chain of inverters and an AND gate. The latter is

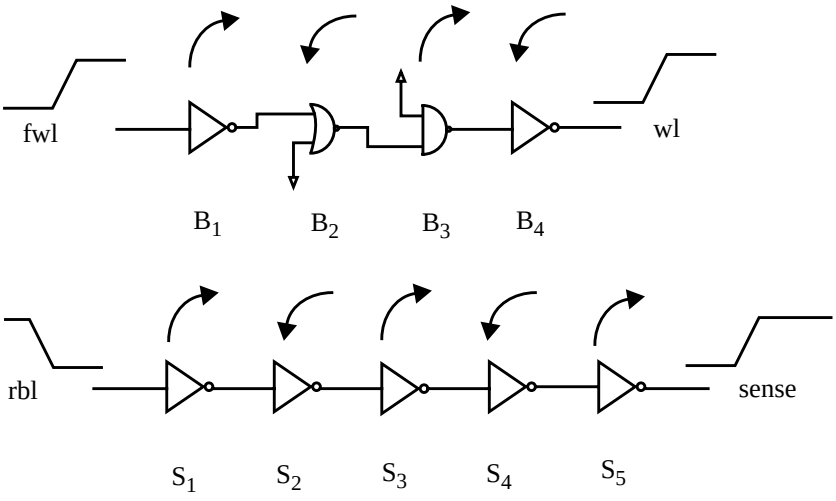


Figure 4.8: Delay matching of two buffer chains

easier to tackle, especially since the matching for only one pair of edges needs to be done. A simple heuristic for matching the delay of a rising edge of the five long chain, S_1 to S_5, to the rising delay of the four long chain, B_1 to B_4, is to ensure that sum of falling delays in the two chains are equal as well as the sum of rising delays [51] (Figure 4.8). The S chain has three rising delays and two falling delays while the B chain has two rising and falling delays. This simple sizing technique ensures that the rising and falling delays in the two chains are close to each other giving good delay tracking between the two chains over all process corners. The delay from *fwl* (see Figure 4.6) to minimum bitline swing is $t_{Bchain} + t_{bitline}$ and the delay to the sense clock is $t_{replica} + t_{Schain}$ delay. If $t_{bitline}$ equals $t_{replica}$ and t_{Bchain} equals t_{Schain}, then the sense clock fires exactly when the minimum bitline swings have developed.

We next look at two design examples, one for a block of 256 rows and 64 columns and the other for a block with 64 rows and 32 columns. The number of rows in these blocks are typical of large and small SRAMs respectively. For each block the replica based implementation is compared to an inverter chain based one. Table 4.1 summarizes the simulated performance of the 256 block design over various process corners.Five process corners are considered along with a 10% supply voltage variation at TT corner. The delay elements are designed to yield a bitline swing of around 100mV when the sense clock fires, under all conditions with the weakest corner being the SF_3 corner with slow nMOS and fast pMOS (since the memory cell is weak and inverters are relatively faster). For each type of delay element, the table provides the bitline swing when the sense clock fires and the maximum bitline swing after the word line is shut off. An additional column notes the “slack” time of the sense clock with respect to an ideal sense clock, as a fraction of a fanout of 4 gate delay. This time indicates the time lost in turning ON of the sense amp compared to an ideal sense clock generator which would magically fire under all conditions exactly when the bitline swings are 100mV, and directly adds to the critical path delay for the SRAM. The last column shows the excess swing of the bitlines for the inverter chain case relative to the replica case as a percentage and represents the excess

bitline power for the former over the latter. The last two rows show the performance at

Table 4.1: Block parameters: 256 rows, 64 columns

	Replica Delay Element (replica bitline height = 29)			Inverter Chain Element			
Process, Supply(V)	Bitline Swing (mV)	Max Bitline Swing (mV)	tSlack (as fraction of fo4del)	Bitline Swing (mV)	Max Bitline Swing (mV)	tSlack (as fraction of fo4del) a.u.	Relative Max Bitline swing (%)
TTH, 1.2	128	136	1.67	139	158	2.24	16.4
FF3, 1.2	114	133	0.64	181	187	4.2	41
SS3, 1.2	136	138	2.37	126	147	1.63	6
SF3, 1.2	97	102	-0.3	101	110	0	7
FS3, 1.2	167	176	2.17	240	251	5.25	43
TTH, 1.08	121	127	1.55	114	134	0.93	5.6
TTH, 1.32	135	145	1.72	172	187	3.64	30

$\pm$10% of the nominal supply of 1.2V in typical conditions. Considering all the rows of the table, we note that the slack time for the replica case is within 2.4 gate delays while that of the inverter case goes up to 5.25 gate delays, indicating that the latter approach will lead to a speed specification which is at least 3 gate delays slower than the former. The bitline power overhead in the inverter based approach can be up to 40% more than the replica based approach. If we were to consider only correlated threshold fluctuations for the nMOS and the pMOS, then the delay spread for both the approaches is lower by one gate delay, but the relative performance difference still exists. The main reason for the variation in the slack time for the replica approach is the mismatch in the delays of the buffer chains across the operating conditions. This comes about mainly due to the variation of the falling edge rate of the replica bitline. In the case of the inverter based design, the spread in slack time comes about due to the lack of tracking between the bitline delay and the

inverter chain delay as discussed in the earlier section. To study the scalability of the replica technique, designs for a 64 row block are compared in Table 4.2. The small bitline

Table 4.2: Block parameters: 64 rows, 32 columns

	Replica Delay Element (replica bitline height = 6)			Inverter Chain Element			
Process, Supply(V)	Bitline Swing (mV)	Max Bitline Swing (mV)	tSlack (as fraction of fo4del)	Bitline Swing (mV)	Max Bitline Swing (mV)	tSlack (as fraction of fo4del) a.u.	Relative Max Bitline swing (%)
TTH, 1.2	101	198	0.24	123	192	0.38	-3
FF3, 1.2	100	210	0	162	207	0.85	-1
SS3, 1.2	125	193	0.42	102	177	0	-8
SF3, 1.2	101	155	0	100	145	0	-7.5
FS3, 1.2	110	211	0.1	194	211	1.1	0
TTH, 1.08	113	176	0.24	88	162	-0.24	-8
TTH, 1.32	122	228	0.37	162	223	0.83	-2

delay for short bitlines, is easy to match even with an inverter chain delay element and there is only a slight advantage for the replica design in terms of delay spread while there is not much difference in the maximum bitline swings. The maximum bitline swings are considerably larger than the previous case, mainly due to the smaller bitline capacitance.

This technique can be modified for clocked current mirror sense amplifiers, where the exact time of arrival of the sense clock is not critical as long as it arrives sufficiently ahead to setup the amplifiers in the high gain stage by the time the bitline signal starts developing. Delaying the sense clock to be as late as safely possible, minimizes the amplifier static power dissipation and can be readily accomplished using the replica techniques. To illustrate this, consider the inverter delay variations in the graph of Figure 4.5. In order to ensure high performance operation under all process corners, the amplifier

delay chain must be fast enough for the FSC_3 corner. But then in the other process corners, the amplifier will be turned on much earlier than it needs to and it will be wasting biasing power unnecessarily. If one references the turn on time of the amplifier with respect to the bitline signal development, then for low power operation across process corners, the amplifiers need to turn on at some fixed number of gate delays before the bitline swings develop. This can be achieved via the replica scheme by merely trimming the delay of the S chain with respect to the B chain to ensure that the sense clock turns on a fixed number of gate delays before the bitlines differentiate.

While this replica technique generates well controlled sense clock edges, it will be unable to do the same for the word line pulse widths which turn out to be longer than what is desired for minimum bitline swings. This is mainly because the word line reset signal has to travel back from the replica bitline through the B chain which is usually optimized to speed up the set signals to assert the local word line in order to reduce the access times. In the next section we will describe an alternative replica scheme which allows for feeding the replica signal directly into the word line driver. This will then allow the forward path to assert the local word lines to be optimized using skewed gates [20] without hampering the local word line pulse width control.

4.3 Replica delay element based on cell current ratioing

An extra row and column containing replica memory cells can be used to provide local resetting timing information for the word line drivers. The extra row contains memory cells whose pMOS devices are eliminated to act as current sources, with currents equal to that of an accessed memory cell (Figure 4.9). All their outputs are tied together and they simultaneously discharge the replica bitline. This enables a multiple of memory cell current to discharge the replica bitline. The current sources are activated by the replica word line which is turned on during each access of the block. The replica bitline is identical in structure to the main bitlines with dummy memory cells providing the same

amount of drain parasitic loading as the regular cells. By connecting *n* current sources to

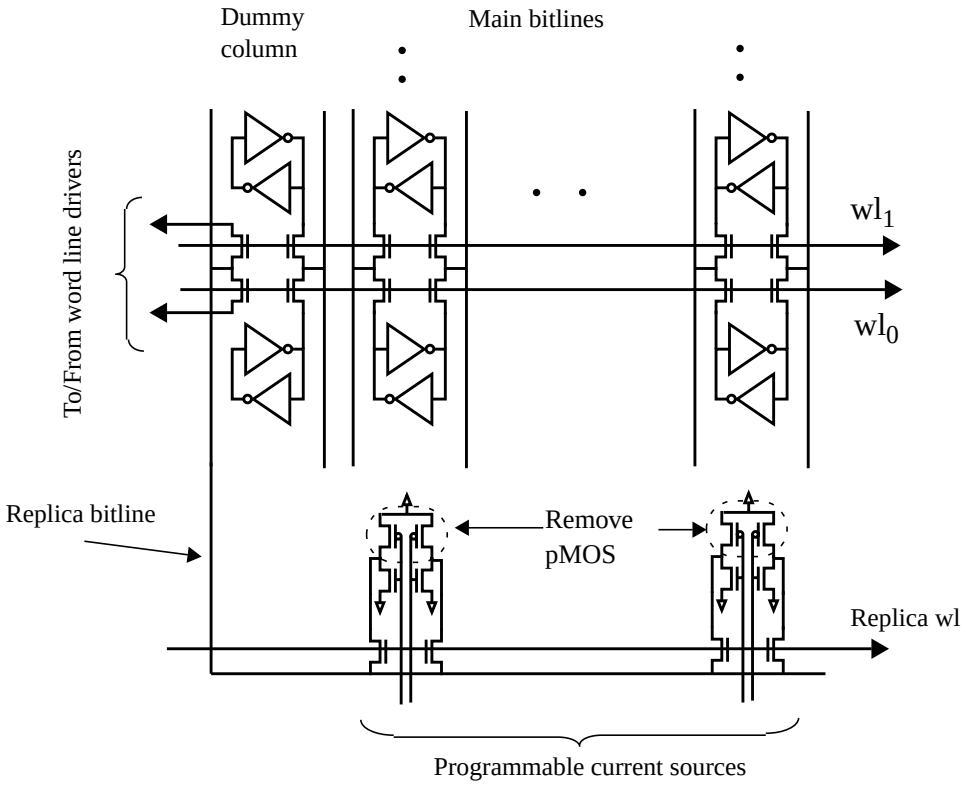


Figure 4.9: Current Ratio Based Replica Structure

the replica bitline, the replica bitline slew rate can be made to be *n* times that of the main bitline slew rate achieving the same effect as bitline capacitance ratioing described earlier.

The local word line drivers are skewed to speed up the rising transition and they are reset by the replica bitline as shown in Figure 4.10. The replica bitline signal is forwarded into the word line driver through the dummy cell access transistor M_a. This occurs only in the activated row since the access transistor of the dummy cell is controlled by the row word line *wl*, minimizing the impact of the extra loading of F_1 on the replica bitline. The forward path of the word line driver can be optimized for speed, independent of the

resetting of the block select or global word line by skewing the transistors sizes.

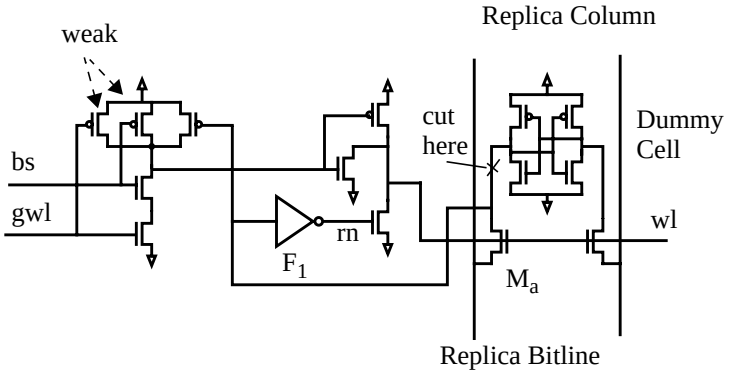


Figure 4.10: Skewed word line driver

The control circuits to activate the replica bitline and the sense clock are shown in Figure 4.11. The dummy word line driver is activated by the unbuffered block select, *fwl*. The replica bitline is detected by F1 and buffered to drive the sense clock signal. If the

Table 4.3: Block parameters: 256 rows, 64 columns

	Replica Delay Element (replica current sources = 8)		
Process, Supply(V)	Bitline Swing (mV)	Max Bitline Swing (mV)	tSlack (as fraction of fo4del)
TTH, 1.2	120	155	1.2
FF3, 1.2	106	170	0.3
SS3, 1.2	124	155	1.5
SF3, 1.2	94	153	-0.5
FS3, 1.2	144	155	1.53
TTH, 1.08	115	144	1.1
TTH, 1.32	124	172	1.2

delay of the replica bitline is matched with the bitline delay and the delay of F1, S2, S3 is

made equal to B1, B2, then the sense clock fires when the bitline swing is the desired amount. Also if the delay of B1, B2 is equal to the delay of generating *rn* (Figure 4.10) from the replica bitline, then the word line pulse width will be the minimum needed to generate the required bitline swing. The performance for a 256 row block with 64 columns implementing this replica technique, is summarized in Table 4.3 The slack in activating the sense clock is less than 1.5 gate delays and the maximum bitline swing is within 170

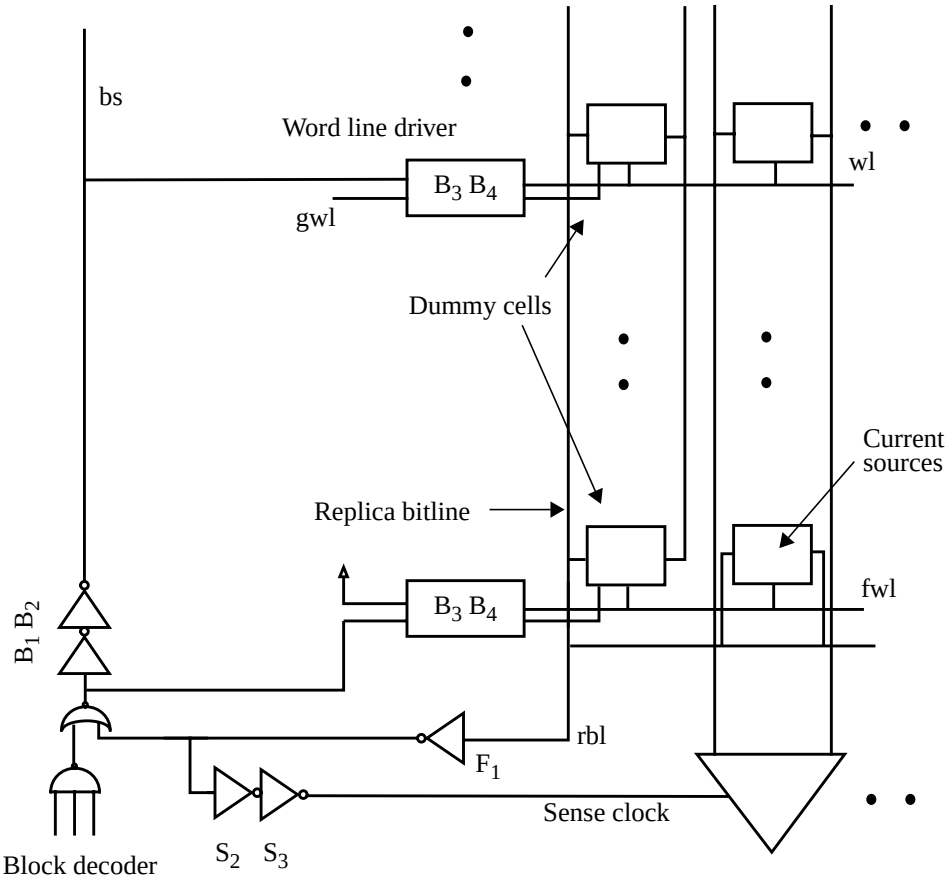


Figure 4.11: Control circuits for current ratio based replica circuit

mV. When compared with the implementation based on capacitance ratioing discussed in the previous section, this design is faster by about 2/3rd of a gate delay, due to the skewing of the word line driver.

The power dissipation overhead of this approach is the switching power for the replica bitline which has the same capacitance as the main bitline. The power overhead becomes small for large access width SRAMs. The area overhead consists of 1 extra column and row and the extra area required for the layout of the skewed local word line drivers.

4.4 Measured Results

We designed and tested two prototype chips with both kinds of replica delay element.The first chip is a 8Kbit RAM built in a MOSIS 1.2µm process and has the capacitance ratioing replica besides some other low power techniques and will be described next. Following that we will discuss the second prototype chip which was built in a 0.25µm process from TI and includes the replica based on cell current ratioing.

4.4.1 SRAM test chip with replica feedback based on capacitance ratioing

The replica feedback technique based on capacitive ratioing was implemented in a 1.2µm process as part of a 2k x 32 SRAM [27-28]. The SRAM array was partitioned into eight blocks each with 256 rows and 32 columns. Making the block width equal to the access width ensures that the bitline power is minimized since only the desired number of bitline columns swing in any access. Two extra columns near the word line drivers and two extra rows near the sense amps are provided for each block with the replica column being the second one from the word line drivers and the replica cell being the second cell in the column (Figure 4.12). The extra row and column surrounding the replica cell contain dummy cells for padding purposes, so that the replica cell doesn’t suffer from any processing variations at the array edge. The bitlines in the replica column are cut at a height of 26 rows, yielding a capacitance for the replica bitline which is one tenth that of the main bitline. Further control for the replica delay is provided for testing purposes by utilizing part of the replica structure above the cut to provide an adjustable current source.

This consists of a pMOS transistor whose drain is tied to the replica bitline and whose gate

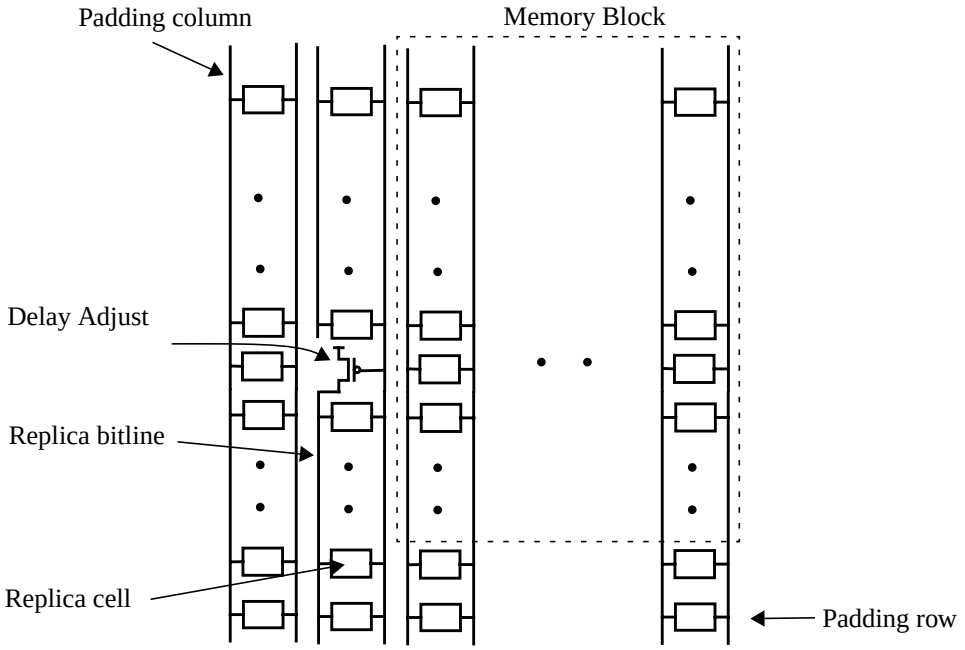


Figure 4.12: Replica Structure Placement

is controlled from outside the block. Since the replica bitline runs only part way along the column, the bitline wiring track in the rest of the column can be used for running the gate control for this transistor. By varying the gate voltage, we can subtract from the replica cell pull down current, thus slowing the replica delay element if needed. The inverters S1, S2 and F1 (Figure 4.7) are laid close to replica cell and to each other to minimize wire loading.

Clocked voltage sense amplifiers, precharge devices and write buffers are all laid on one side of the block as shown in Figure 4.13. Since the bitline swings during a read are significantly less than that for writes, the precharge devices are partitioned in two different groups, with one activated after every access and the other activated only after a write, to reduce power in driving the precharge signal. Having the block width equal to access width, requires that all these peripheral circuitry be laid out in a bit pitch. The precharge

and write control are locally derived from the block select, write enable and the replica feedback signals.

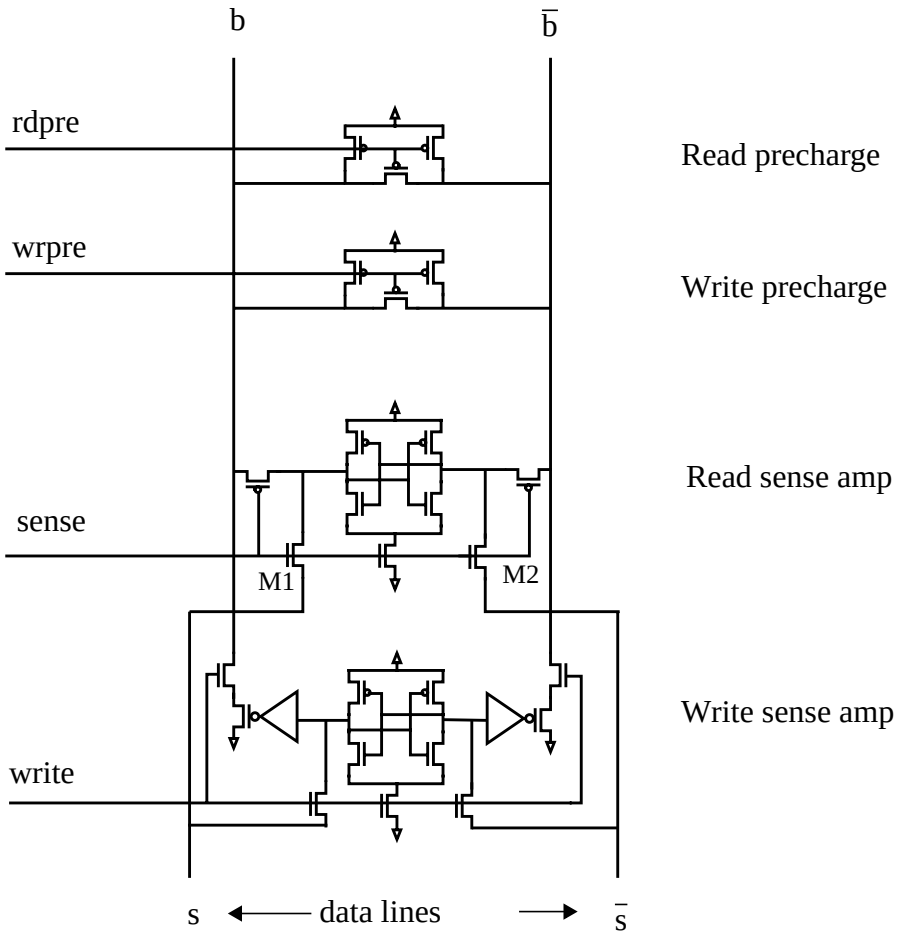


Figure 4.13: Column I/O circuits

An 8 to 256 row decoder is implemented in three stages of 2-input AND gates. The transistor sizes in the gates are skewed to favor one transition as described in [20], to speed up the clock to word line rising delay. The input and output signals for the gates are in the form of pulses. In that design, the gates are activated by only one edge of the input pulse and the resetting is done by a chain of inverters to create a pulse at the output. While this approach can lead to a cycle time faster than the access time, careful attention has to be paid to ensure sufficient overlap of pulses under all conditions. This is not a problem for

pulses within the row decoder as all the internal convergent paths are symmetric. But for the local word line driver inputs, the global word lines (the outputs of the row decoder) and the block select signal (output of the block decoder) converge from two different paths. To ensure sufficient margins for their overlap, the global word line signal is prolonged by the modifying the global word line driver as shown in Figure 4.14. Here the

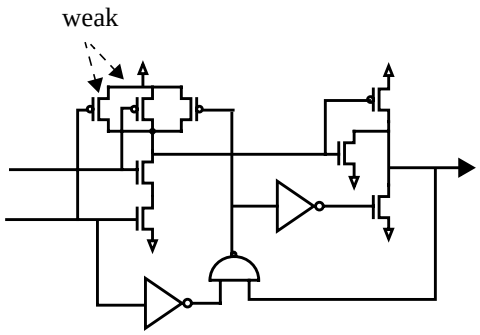


Figure 4.14: Pulsed Global word line driver

reset signal is generated by ANDing the gate's output and one of the address inputs to increase the output pulse width. Clearly, for the gate to have any speed advantage over a static gate, the extra loading of the inverter on one of the inputs must be negligibly small compared to the pMOS devices in a full NAND gate. The input pulses to the row decoder are created from level address inputs by a "chopper" circuit shown in Figure 4.15, which

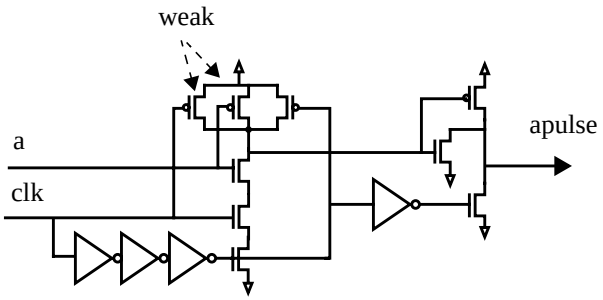


Figure 4.15: Level to pulse converter

can be merged with a 2 bit decode function.

The data lines in the SRAM, which connects the blocks to the I/O ports, is implemented as a low swing, shared differential bus. The voltage swings for reads and writes are limited by using a pulsing technique similar to that described in Section IV. During reads, the bitline data is amplified by the sense amps and transferred onto the data lines through devices M1 and M2 (see Figure 4.13). The swing on the local data lines is limited by pulsing the sense clock. The sense clock pulse width is set by a delay element consisting of stacked pull down devices connected to a global line which mimics the capacitance of the data lines. The data line mimic line also serves as a timing signal for the global sense amps at the end of the data lines and has the same capacitance as that of the worst case data bit. The global sense amps, write drivers and the data lines precharge are shown in Figure 4.16. The same bus is also used to communicate the write data from the

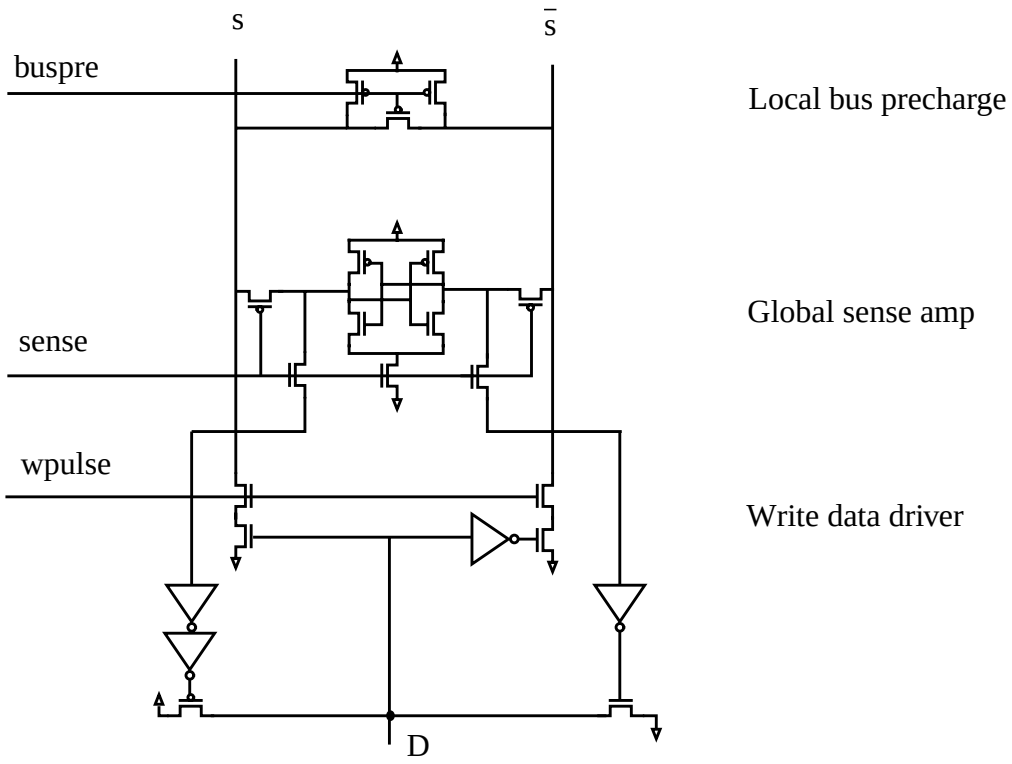


Figure 4.16: Global I/O circuits

I/O ports to the memory blocks during writes. The write data is gated with a pulse, whose

width is controlled to create low swings on the data lines. The low swing write data is then amplified by the write amplifiers in the selected block and driven onto the bitlines (Figure 4.13).

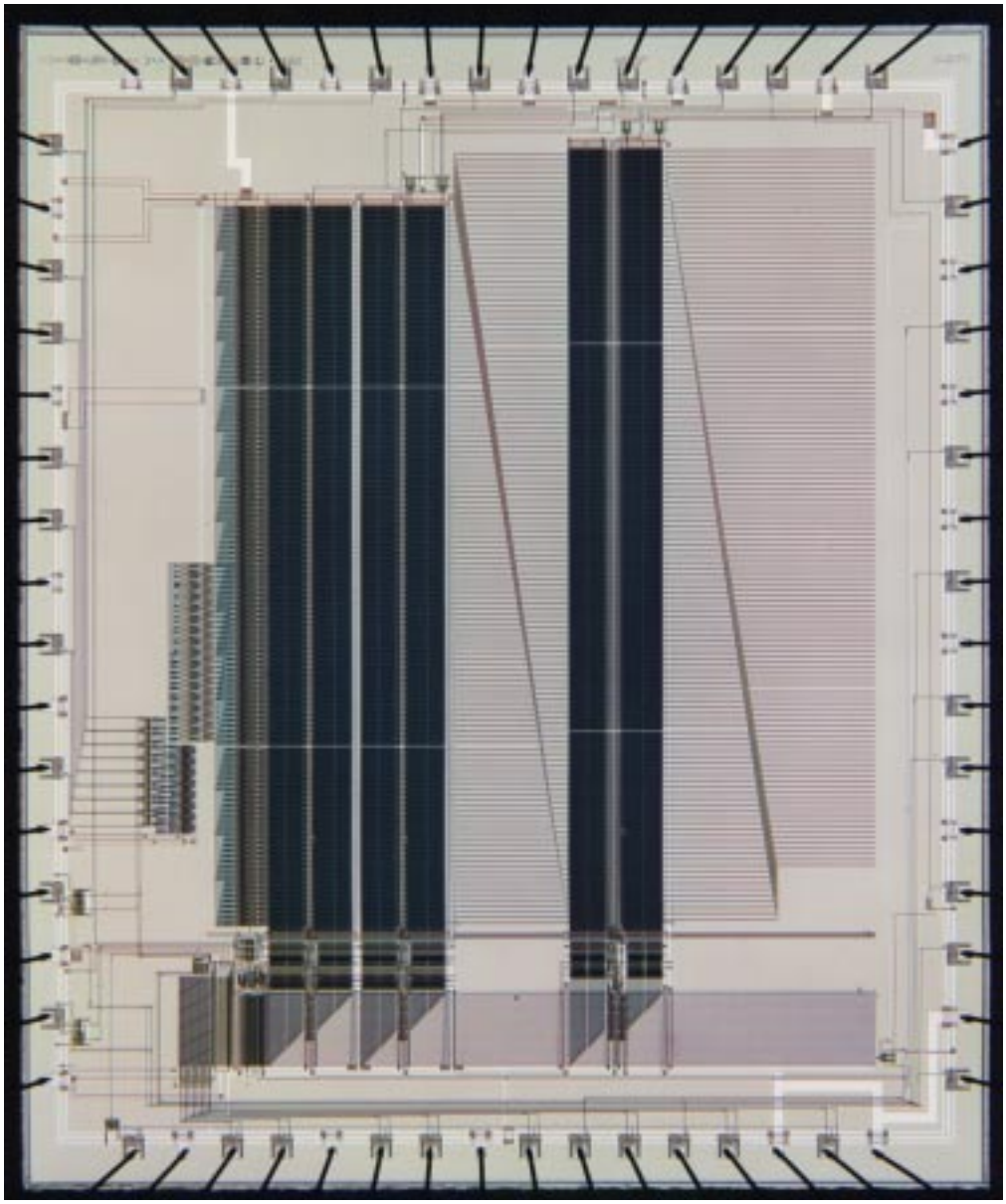


Figure 4.17: Die photo of a 1.2µm prototype chip

Figure 4.17 displays the die photo of the chip. Only two blocks are implemented in the prototype chip to reduce the test chip die area, but extra wire loading for the global word lines and I/O lines is provided to emulate the loading in a full SRAM chip. Accesses to these rows and bits are used to measure the power and speed. The bitline waveforms were probed for different supply voltages to measure the bitline swing. Figure 4.18 displays the

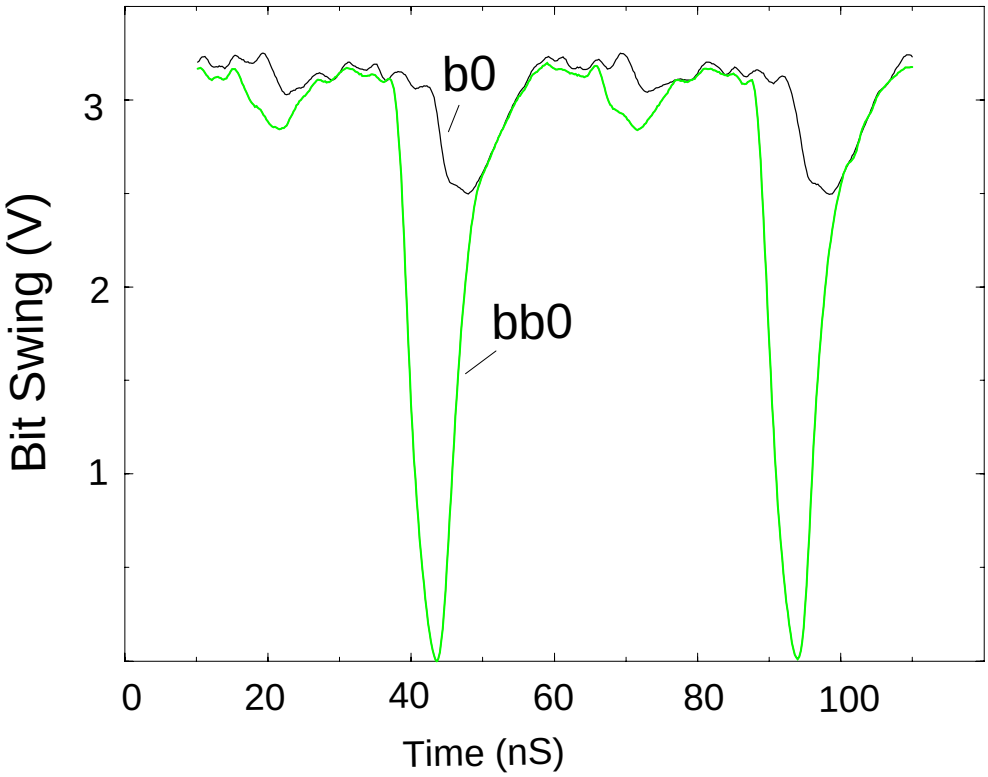


Figure 4.18: Waveforms on a bitline pair during an alternating read/write access pattern, obtained via direct probing of the bitline wires on the chip.

on-chip measured waveform of a bitline pair at 3.5V supply with an alternating read/write pattern. The bitline swings are limited to be about 11% of the supply at 3.5V due to the action of the replica feedback on the word line. Table 4.4 gives the measured speed, power, bitline swing and I/O line swing for the chip at three different supply voltages of

1.5V, 2.5V and 3.5V. The access times at these voltages is equivalent to the delay of a

Table 4.4: Measurement Data for the 1.2µm prototype chip

Supply (V)	$\Delta I/O^w$ (%)	$\Delta I/O^r$ (%)	ΔB^r (%)	Tacc/ Fanout 4 Inverter units	Power (mW)
1.5	16	17	16	60.0ns / 21.84	2.88@10MHz
2.5	12	16	12	21.1ns / 21.53	31.1@37MHz
3.5	11	19	11	12.8ns / 20.7	105@40MHz

chain of about 21 fanout 4 loaded inverters (the delay of a fanout 4 loaded inverter was obtained from simulations using the process data for this wafer run). The on-chip probed waveforms for the data lines are shown in Figure 4.19 for a consecutive read and write

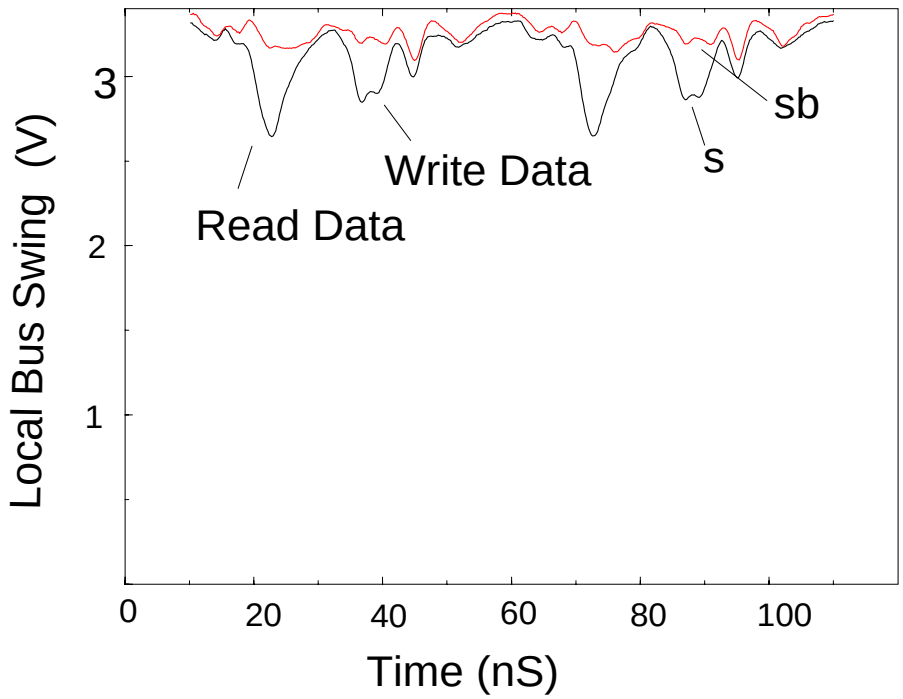


Figure 4.19: On-chip probed data line waveforms for an alternating read/write pattern

operation. The I/O bus swings for writes are limited to 11% of the supply at 3.5V, but are 19% of the supply for reads. The rather large read swings is due to improper matching of the sense amplifier replica circuit with the read sense amplifiers.

4.4.2 SRAM test chip with replica feedback based on cell current ratioing

Cell current ratioing based replica technique was implemented in a 0.25µm process (with wire pitch based on a 0.35µm process) from Texas Instruments. A 2KB RAM is partitioned into two blocks of 256 rows by 32 columns. Two extra rows are added to the top of each block, with the topmost row serving as a padding for the array and the next row containing the replica cells.

The prototype chip (Figure 4.20) has 2 blocks each 256 rows x 32 bits to form a 8Kbit memory. The premise in this chip was that the delay relationship between the global word line and the block select is unknown, i.e, either one of them could be the late arriving signal and hence the local word line could be triggered by either. This implies that the replica bitline path too needs to be activated by either the local block select or a replica of the global word line. The replica of the global word line is created by mirroring the critical path of the row decoders as shown in Figure 4.21. This involves creating replica predecoders, global word drivers, replica predecode and global word lines with loading identical to the main predecode and global word lines [53]. The replica global word line and the block select signal in each block generate the replica word line which discharges the replica bitline as described in the previous section. Thus the delay from the address inputs to the bitline is mirrored in the delay to the replica bitlines. But the additional delay of buffering this replica bitline to generate the sense amp signal cannot be cancelled in this approach. Hence the only alternative to minimize this overhead in our prototype was to distribute the replica bitline structure throughout the array. The test chip has one replica bitline structure per eight bitline columns, with just one sense clock buffer stage (Figure

4.21). The large swing replica bitline is shielded from the low swing bitlines by cell

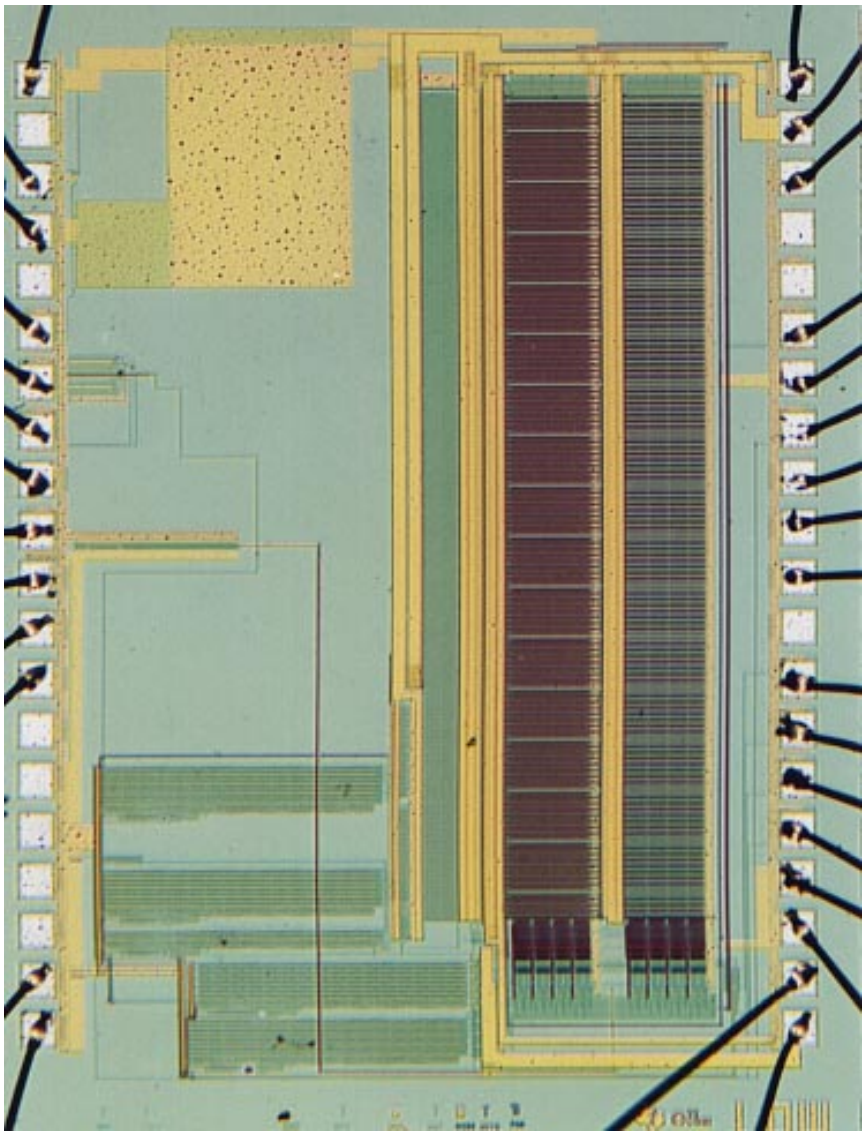


Figure 4.20: Die photo of a prototype chip in 0.25µm technology

ground lines running parallel to it. The coupling is further reduced by twisting the bitlines.

The row decoder is implemented in the postcharge style for high speeds. The global word

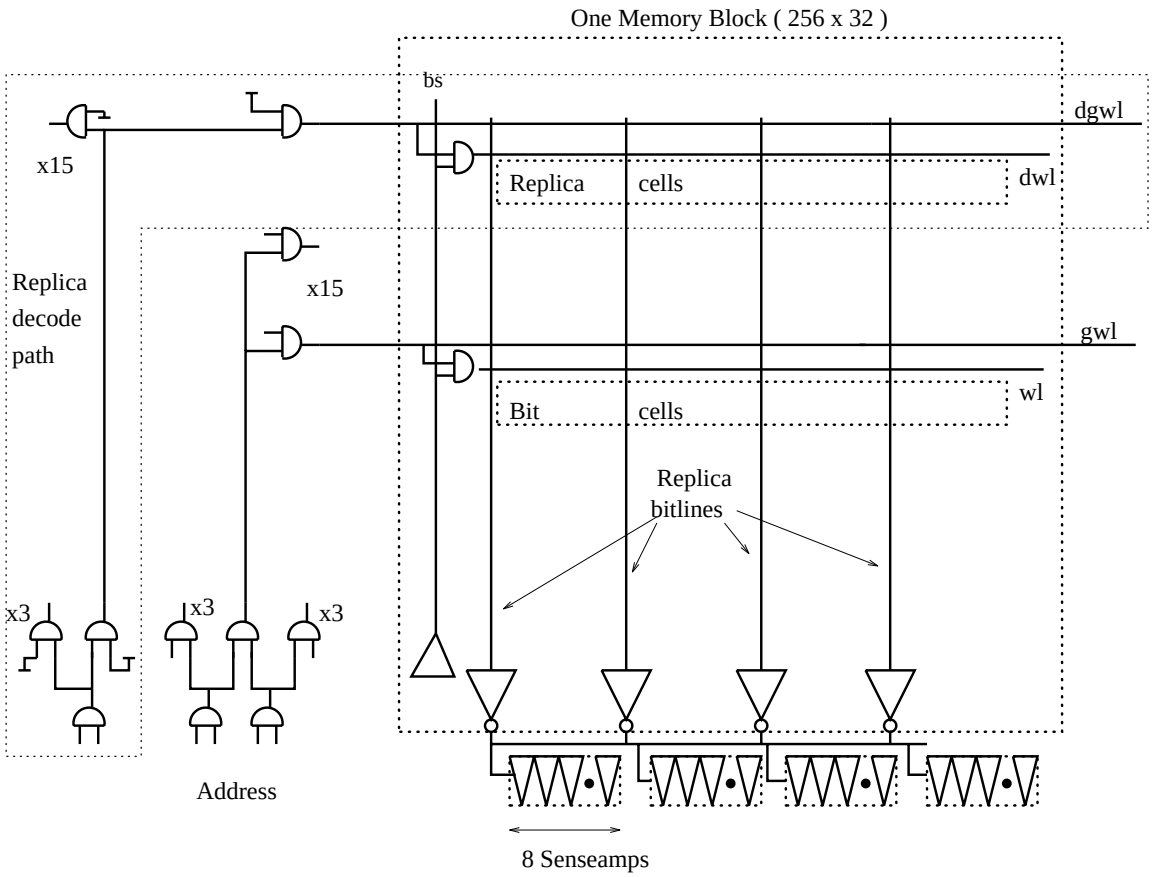


Figure 4.21: Architecture of a prototype chip in 0.25µm CMOS

line drivers shown are modified from Figure 4.14 to have two fewer transistors and yet

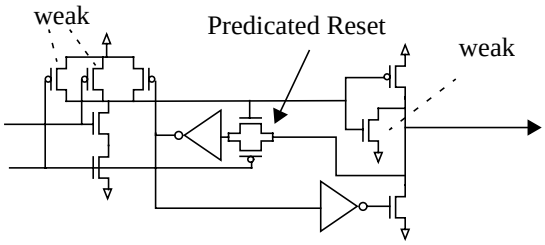


Figure 4.22: Modified Global Word line Driver

achieve the same functionality of ensuring a fast rising edge with a long pulse at the output

(Figure 4.22).

Table 4.5 shows the measured results for the chip. The test chip operates down to 0.4V

Table 4.5: Measured data for the 0.35µm prototype

Supply (V)	Tcyc (ns) / FO4 inverter delays	Power
1.0	6.6 / 18.9	3.64mW
0.75	14.1 / 18.1	0.99mW
0.5	138 / 19	50µW
0.45	350 / 21.1	17.15µW
0.4	1020 / 23	5.24µW

at 980KHz and 5.2uW dissipation. At 1V the chip was measured to dissipate 3.6mW@150MHz. In an SRAM, except for the bitlines and the sense amps, all the other circuits are completely digital and hence are expected to have a large operating range. The use of latch type sense amps (Figure 4.3) coupled with replica based generation of their sense clock allows for the wide operating range for the whole SRAM path, as we observe in the table. The energy per access is plotted against the supply voltage in Figure 4.23. The graph also includes a quadratic curve fit on two of the measured points (the 0.5V and the 1V points). The measured energy (black squares) follows the quadratic curve indicating that the replica circuits are controlling the bitline swings to be proportional to the supply. The table also calibrates the SRAM delay in terms of the delay of a fanout 4 loaded inverter which was measured from the same die. We find that the SRAM delay is about 19 inverter delays for voltages above 0.5V and increases to about 23 gate delays when the supply voltage approaches the transistor threshold voltage.

The replica predecode structure was simulated to consume 15% of the total power in the test chip and the replica bitlines paths (4 per block) was simulated to consume 9% of the chip power. By putting the constraint that the block select cannot arrive earlier than the

global word line, the replica decode path and the multiple replica bitline paths can be eliminated to achieve a savings of 20% in the power dissipation. Extrapolating from test chip measurements, we estimate that a typical cache RAM of 16KB will dissipate 7mW@150MHz at 1V and 10.4µW@980KHz at 0.4V.

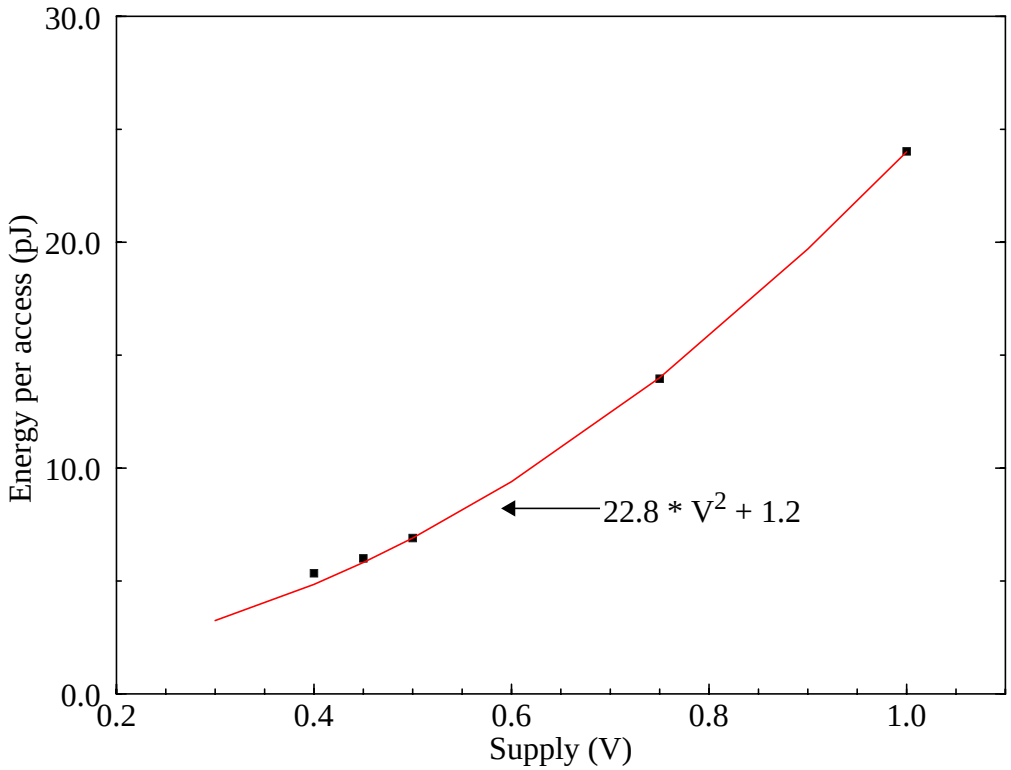


Figure 4.23: Measured energy (black squares) versus supply. Also shown is the quadratic curve fit on the 0.5V and the 1V points.

4.5 Low Power Writes

The bitline power to perform a write operation can significantly exceed the read power due to the large signal swings required to write into the memory cell. Typically the bitline is discharged almost all the way to ground before the write is achieved. If the bitline swings during reads are controlled to be around 200mV, then the bitline read power will be

only about a factor of 1/5 of that of the bitline write power, for a 1.2V operation. The total read power will then be typically a factor of 2 smaller than the total write power. For mainstream computing applications, the number of write accesses is only half as many as read accesses [52], and so the average read power will be about the same as the average write power. But for certain applications like video decompression or ATM switching, the writes to the SRAM are as frequent as the reads. Write power can be reduced by partitioning the bitline into small segments with very small capacitance [54] or by using small voltage swings in the bitlines to do the writes. In this section we will explore the latter.

The key problem in achieving a write using small swings is to be able to overpower the memory cell flip flop with a small bitline differential voltage while ensuring that the cell will not erroneously flip its state during a read. Larsson et. al. in [29] propose a technique where they bias the bitlines near the ground voltage. When a small bitline differential is set up for the write and the word line is activated, the internal nodes of the memory cell are quickly discharged to the values of the bitlines through the access nfets. When the word line is turned off, the small differential voltage between the cell internal nodes is amplified by the cross coupled pfets in the cell. To prevent the reads from over writing the cell, the authors propose to reduce the word line voltage for the read operation, thus weakening the access nfets and preventing a spurious discharge of the cell internal nodes. The main weakness of this approach is that read accesses become slower now since the word line voltage is smaller. Mori et. al. in [30] use a similar concept, except that they use a bitline reference of Vdd/2 which is easy to generate and incurs significantly lower read access penalty. A write is achieved by discharging one of the bitlines to ground. Since the write bitline swings are halved, a factor of four savings in the bitline write power is achieved. To improve the robustness of reads, the cell voltage is boosted above Vdd.

In this section we will explore the possibility of using small voltage swings in the bitlines while keeping the bitline reference close to Vdd. These small swings are amplified by operating the memory cell as a latch sense amplifier, accomplished by modifying the ground connection to the cell as shown in Figure 4.24a. All the cells in a given row share a ground line, called the virtual ground which is driven by an inverter. During reads and

standby, the virtual ground is driven low, and the SRAM operates as in the conventional case. During writes, the virtual ground line is first driven high to reset the contents of the cells. The write data is put onto the bitlines in the form of small voltage swings. The word line is pulsed high and the small swing bitline signal is the transferred into the internal nodes of the cell after which the virtual ground line is driven low, causing the cell to amplify and latch the data. Since the cell access transistors are nfets, the word line high voltage needs to be at least an nMOS threshold above bitline high voltage, to enable a full reset of the cell (see waveform schematics in Figure 4.24b). Thus instead of a number of

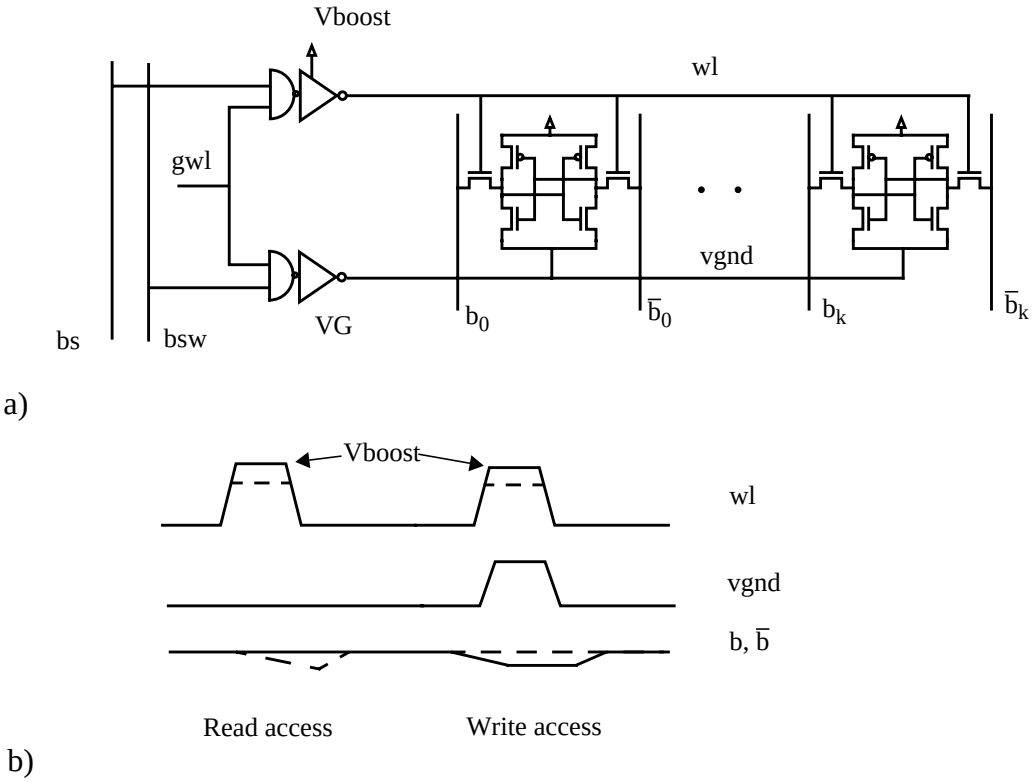


Figure 4.24: a) Schematic of cells for low swing writes b) waveforms

bitlines swinging full rail, this scheme enables all of them to have only small swings at the expense of an additional full swing line in the word line direction.

There are a couple of limitations to this approach. Since boosted word lines are used, the access nfets will have a higher current drive which also makes the cell less stable during reads. To achieve a robust cell, either the pull down nfets within the cell have to be increased in size, the supply voltage to the cell should also be boosted, or the word line voltage should not be boosted for reads. A more significant problem is that the read cell currents from all the accessed cells in a row flow through the pull down device of the inverter VG, causing the virtual ground line, vgnd, to have a small voltage drop. This drop needs to be limited by carefully sizing the pull down device in VG, to ensure that the memory cells have sufficient read noise margins. This extra ground resistance can also slow down the read access since the read cell currents will be reduced. Also, the area of the word driver increases because of the extra inverter. But this area increase is partially offset by a reduction in the area for the write drivers and the precharge devices for the bitlines as the write swing is reduced.

One limitation of the approach is that since the cells along a word line share the virtual ground line, all of these will get over written. This implies that the block width needs to be the same as the write word width. In order to allow for write widths which are smaller than the block width, the write operation can be modified to first read the cell to transfer their data on the bitlines, and then writing new data onto the desired subset of bitlines and proceeding with the write as above. Thus though all the cells are over written, only the desired subset is written with the new data while the rest are over written with their own data.

Another limitation of the approach is the requirement of boosted voltage for the word lines during writes. High performance SRAMs are already operated with supplies close to the upper limit prescribed for the process and there might not be any more head room to boost the voltages further. An alternative will be to use pfet access devices in the cell and use a negative going word line. This approach also has the advantage that the same word line voltage of 0 can be used for both the reads and the writes. The main disadvantage of this approach is that the pfets are about a factor of three slower than the nfets and hence the reads become slower.

One of the prototype chips we designed included a memory block with this circuit technique. The area overhead over the conventional memory block was 20% and came about mainly due to the extra gates in the word driver and the area overhead in the memory cell to accommodate for individual ground lines as opposed to shared ground lines in the conventional design, though in modern processes this might not be an issue due to availability of a large number of metal layers. Figure 4.16 shows a die photo of a prototype implementation, which has three blocks each of 256 by 32 in the 1.2µm technology. The taller block in the photo is modified to implement the low swing writes while the other two blocks have conventional write circuitry. Circuit simulations indicated that the total write power was reduced from 2x of the read power to 1.3x of the read power. The read access time slows down by about one inverter delay. Due to an error in the connections of the sense amplifiers to the local I/O bus, the read and write operations to the block could not be fully verified. But we could probe the bitline waveforms and the measured data for consecutive read and write accesses is shown in Figure 4.24. The voltage bumps on the virtual

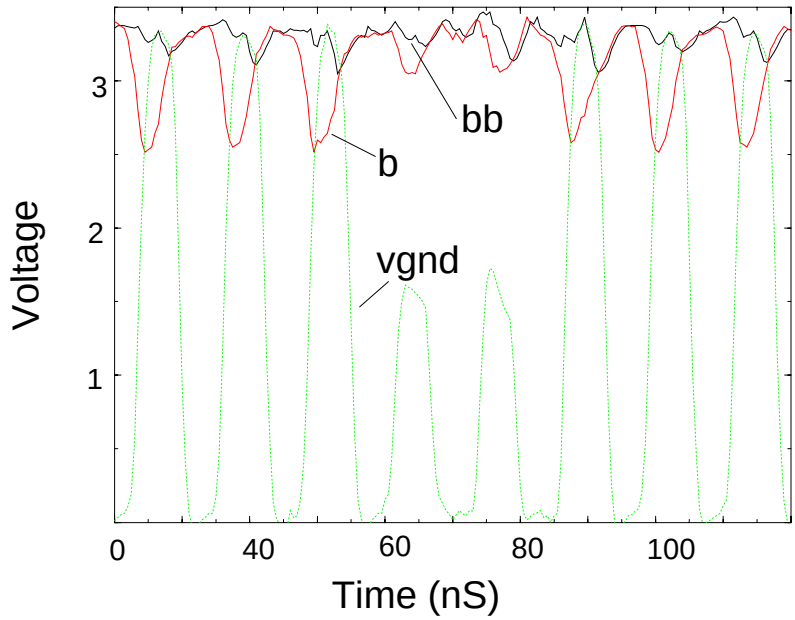


Figure 4.25: On chip probed wave forms of bitlines for low swing writes alternating with reads.

ground node is about 1.5V during the read access indicating that the pull down device width connected to this line is not sufficiently wide. For reliable read operations the device width needs to be increased by a factor of four, which in turn will increase the total area overhead by an additional 5%. The area and read access penalty inherent in this approach might not be acceptable to a large number of applications in return for the lower write power.

4.6 Summary

The key to low power operation is to reduce the signal swings in the high capacitance nodes like the bitlines and the data lines. Clocked voltage sense amplifiers are essential for obtaining low sensing power, and accurate generation of their sense clock is required for high speed operation. An energy efficient way to obtain low voltage swings in the bitlines is to limit the word line pulse width, by controlling the pulse width of the block select signal. The pulse widths are regulated by the aid of a replica delay element which consists of a replica memory cell and a replica bitline and hence tracks the delay of the memory cell over a wide range of process and operating conditions. Two variations of the technique were discussed and their use to generate the sense clock was also demonstrated. We believe that this is a very simple and robust technique with very low area overhead and is easily applicable to a wide variety of SRAM designs. The chapter also demonstrates the application of the technique to achieve low swing data line operation for both the read and write accesses, by pulsing the control signal which gates the drivers of these lines. The pulse width is controlled by discharging a replica line which has the same capacitance as the worst case data line. This replica line is also used to trigger the amplifiers at the receiving end of these lines. Finally we presented a technique to achieve low bitline write power, by using small voltage swings on the bitlines during the write operation. The memory cell structure is modified such that the cells can be used as latch type sense amplifier to amplify and store the small swing write data presented on the bitlines. This

technique requires boosted word lines and 20% larger area and hence is suitable only for certain niche applications.

In the next chapter we will take a step back from detailed circuit design issues and look at scaling trends of delay and power in SRAMs as a function of their size and technology. We will assume that the SRAMs have been optimized for speed and power by using some of the techniques discussed in this chapter.

Chapter 4: Fast Low Power SRAM Data Path

Chapter

5

Speed and Power Scaling of SRAMs

This chapter analyzes the scaling of delay and power of SRAMs with size and technology using some simple analytical models for the various components of the SRAM. Exploring the large design space using conventional SPICE circuit simulation would be extremely time consuming and hence simplified analytical models are very valuable. Such models not only help in designing SRAMs for the current generation, but can also be used to forecast future trends.

Analytical models for delay, area and energy have been developed separately by a number of authors [56-59]. Wada et. al. in [56] and Wilton and Jouppi in [57], develop delay models for the decoder and the bitline path and use it to explore the impact of various cache organizations on the access time. Evans and Franzon develop analytical models in [58-59] for the energy consumption of an SRAM as a function of its organization. This chapter extends the delay models of [56] and combines them with the energy and area models for the SRAM. The delay models are modified to include the effects of interconnect resistance and more complex partitioning schemes. We allow for multi-level hierarchical structures for the bitline and data line muxes [54, 60], which is an additional degree of freedom in the organization not considered by [56-59]. The models are then used to estimate the delay, area and energy of SRAMs of various capacities and in different technology generations.

With feature size shrinking by a factor of 2 every 18 months, two effects stand out: the interconnect delay is getting worse compared to the transistor delay and the threshold mismatches between transistors are not scaling with the supply voltage [61-62]. One expects

both these effects to have a significant influence on SRAMs, since SRAMs require information to be broadcast globally across the whole array and part of the signal path within the array uses small signal swings followed by sense amplification. The chapter investigates both these effects with the aid of the analytical models.

As explained in Chapter 2, a lot of attention is paid to organizing and partitioning the cell array to manage the delay and power. Hence our analysis incorporates this by examining a large number of different organizations to determine the optimal ones. Section 5.1 explains our notation for capturing each partition. To keep the analysis tractable, we make certain simplifying assumptions and discuss the main ones in Section 5.2. An extensive list of all the other assumptions is provided in Appendix C. Using these assumptions we then develop models for delay, area and energy for the key components of the SRAM. We then apply these models to estimate the delay and power for SRAMs and investigate the scaling trends with densities and technology in Section 5.3.

5.1 SRAM Partitioning

At the top level, any partitioning can be captured by three variables: the number of macros (*nm*) which comprise the array, the block width (*bw*) and block height (*bh*) of each of the sub blocks which make up a macro. Figure 5.1 shows an example of partitioning a 1024x1024 array of cells of a 1Mb SRAM for a 64 bit access. The array is broken into 4 macros all of which are accessed simultaneously, each providing 16 bits of the accessed word. Each macro is further subdivided into four blocks of 512 rows and 128 columns and with one of the blocks furnishing the 16 bit sub word.

As discussed in Chapter 2, partitioning of the RAM incurs area overhead at the boundaries of the partitions. Hence we estimate area as an integral part of the analysis. The next

section details the assumptions made for the analysis and describes the area, delay and power models developed for the RAM.

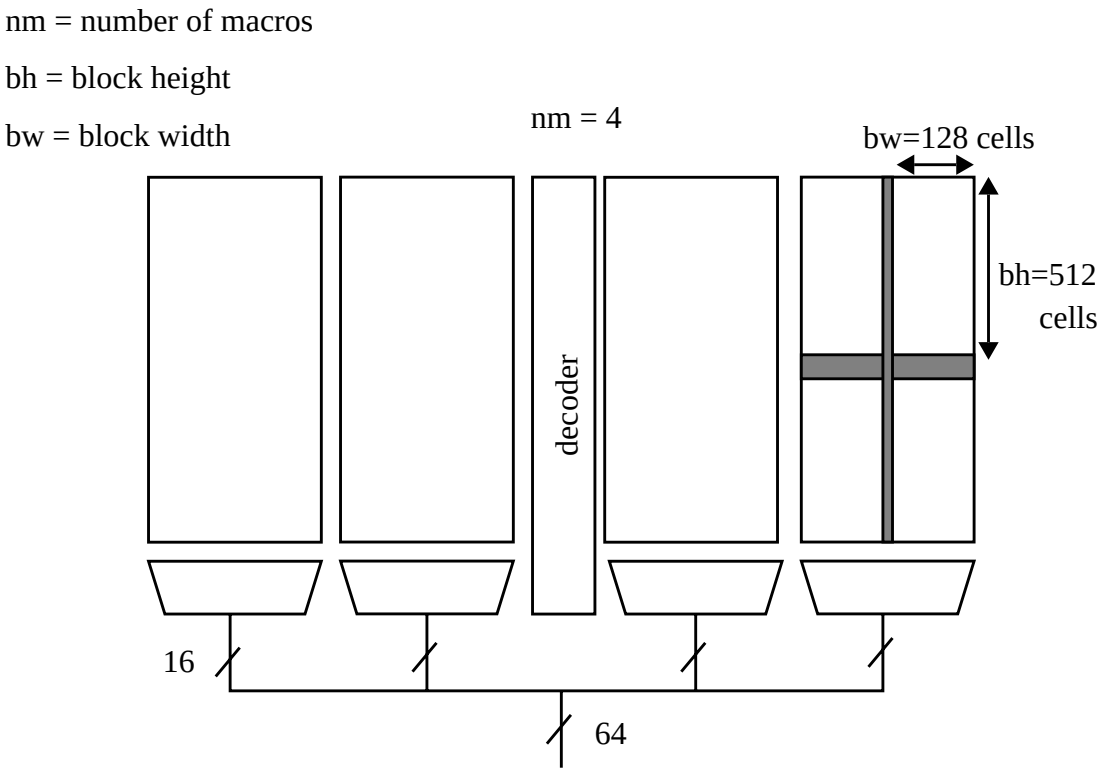


Figure 5.1: Array partitioning example

5.2 Modeling of the SRAM

In order to explore the large SRAM design space in a tractable manner, we make some simplifying assumptions about certain aspects of the design. We outline and justify the key assumptions in the next sub section and list all the assumptions in Appendix C. As discussed in Chapter 2, the SRAM path can be broken into two components, the row decoder and the RAM data path. We develop simple analytical models for delay, area and power for these and the verify these against HSPICE circuit simulations.

5.2.1 Assumptions

The base technology used for this analysis is a 0.25μm CMOS process. A convenient process independent unit of length called λ is used to describe geometric parameters in the chapter. λ is equal to half the minimum feature size for any technology. We assume that all the device speeds and dimensions scale linearly with the feature size. The supply scales as in SIA road map [55] and the wires scale as described by Bakoglu et. al. in [39] with copper metallurgy from 0.18μm generation onwards.

Mizuno et. al. show in [62] that the dominant source of threshold variations in closely spaced transistors in deep sub micron geometries, is the random fluctuations of the channel dopant concentrations. They also show that this portion of the threshold mismatch remains constant with process scaling (see also [61]). So we assume a constant mismatch of 50mV in the thresholds of the input differential pair in the sense amplifiers, irrespective of the process generation.

We model the delay and energy of the RAM core and ignore the external loads as they are a constant independent of the internal organization. Since the read access results in the critical timing path for the RAM, only the delay and power consumption of the read operation is modeled. While ignoring the write power doesn't alter the main conclusions of this chapter, it does affect the optimal organization for low power and we will examine this at the end of the chapter.

We assume a static circuit style for all the gates in the RAM to simplify the modeling task. The pMOS portion of the gate is sized to yield the same delay as the nMOS portion, and hence the gate can be characterized by a single size parameter. Since high speed SRAMs commonly skew the transistor sizes in the decoder gates to improve the critical path we will quantify its impact on our delay analysis.

There is a large diversity in the circuits used for the sense amplifiers. In this chapter, we will assume a latch style amplifier which consists of a pair of cross coupled gain stages which are activated by a clock [21, 30, 20, 27-28]. In these structures, the amplifier delay is proportional to the logarithm of the required voltage gain [63], and hence if the sense

clock timing is well controlled, they lead to the fastest implementations. They also consume very low power since they are inherently clocked. We will assume that the sense clock timing is perfectly controlled but will quantify the impact of non ideal sense clock generation on our delay estimates.

When the number of wiring levels is limited, space has to be allocated in between blocks to route the data lines. This significantly adds to the overall memory area, especially when the block size is very small. Since the number of available wiring levels has been steadily growing [55], we will assume in this chapter that the data lines can be routed vertically over the array if required. Thus extra routing space for a horizontal bus is required only once at the bottom of the array.

Transistor sizing offers another technique to trade-off delay, area and power. In Chapter 3 we found that for high speed designs, sizing each gate to yield a delay of that of a fanout of 4 loaded inverter works pretty well as long as the wire delays are not significant. We assume this sizing rule in this chapter to simplify the analysis. While this assumption does not affect minimum delay solutions, it causes the low energy and low area solutions to be sub optimal and we examine this further at the end of the chapter.

We use the simple RC models of Chapter 3 to model the delay of the gates and the wires. In an energy efficient SRAM design, the dynamic power dominates the total power dissipation, and so we only model the dynamic energy required to switch the capacitances in the decoder and the output mux. We will next discuss in detail the models for the decoder and the output mux.

5.2.2 Decoder model

The decoder delay with the fanout of 4 sizing rule is summarized in Equations 1a,b and is the sum of the extrinsic delays of each gate in the path (each of which is equal to the extrinsic delay of the fanout 4 inverter, τ_{fo4x}), their intrinsic delays and the wire delays. In these equations N is the number of gates in the path, p_i is the parasitic delay of gate i, G is the gate loading due to all the global word drivers connected to the predecode wire and L

is the gate loading of all the local word drivers connected to a global word line. Since the

$$D_{decoder} = N \cdot \tau_{fo4x} + \sum_{i=0}^{N} p_i + D_{wire} \quad (1a)$$

$$D_{wire} = r_p \cdot (c_p/2 + G) + r_g \cdot (c_g + L)/8 + r_{wl} \cdot c_{wl}/8 \quad (1b)$$

local and global word lines are driven from their center by their respective drivers, the RC delay of these wires is reduced by a factor of 4.

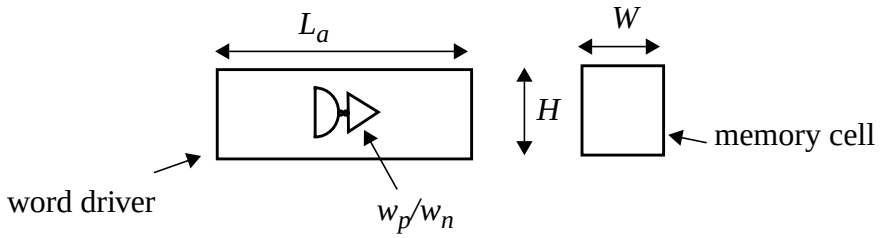
Since bitline delay depends on the local word line rise time, we estimate the edge rate at the end of the local word line. From circuit simulations the rise time was found to be 1.4 times the delay of the final stage of the word driver and is summarized in Equation 2.

$$rise\ time = (\tau_{fo4} + r_{wl} \cdot c_{wl}/8) \times 1.4 \quad (2)$$

Since the final stage is sized to have a fanout of 4, the total delay of the stage is the sum of a fanout of 4 inverter delay (τ_{fo4}) and the RC delay of the local word line ($r_{wl} * c_{wl} / 8$ assuming that the word drivers drive the local word line from the center of the line).

The gate and wire capacitances in the signal path are added up to give an estimate of the decoder energy. Decoder area is calculated by estimating the area of the local and global word drivers and the area required for the predecode wires. The area of the word drivers is modeled as a linear function of the total device widths inside the drivers (Equation 3). The constants for this function have been obtained by fitting it to the areas obtained from the layout of six different word drivers [28, 30]. The total device width within the driver is estimated to be 1.25 times the size of the final buffer as the active area of the pre driver can approximated to be a quarter of the final inverter when fanout 4 sizing is used for the gates. The area for running the vertical predecode and block select wires is also added to the total decode area. As an example, the increase in the SRAM array width

due to the 12 to 4096 decoder of Figure 5.2 is accounted by the areas for 64 local word drivers, 1 global word driver, and vertical wiring tracks for 16 predecode wires and 64 block select wires.

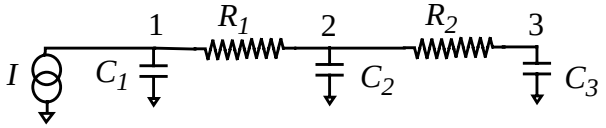


$$L_a*H = 24.05*1.25 * (w_p + w_n) + 497 \quad (in\ \lambda^2) \qquad (3)$$

Figure 5.2: Area estimation for the word drivers

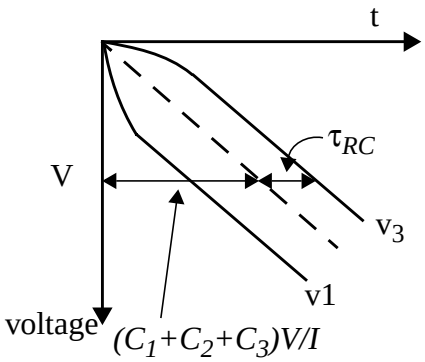
5.2.3 Output Mux

The output mux consists of the bitline mux which routes the cell data into the sense amplifiers, and the data line mux which routes data from the sense amplifiers to the output. Since the signal levels in both these muxes are small (~100mV), the input signal source for both these muxes can be considered as ideal current sources.



$$\tau_{RC} = R_1*(C_1)*(C_2+C_3)/(C_1+C_2+C_3) + R_2*(C_1+C_2)*(C_3)/(C_1+C_2+C_3) \qquad (4)$$

a)



b)

Figure 5.3: a) Current source driving a RC π network. b) Sketch of the node waveforms

The degradation of the delay through a RC network for a current source input is different from that for a voltage source input. Consider an ideal current source driving a RC π

network as shown in Figure 5.3a. The voltage waveforms of the nodes 1 and 3 are

$$D = \frac{(C_1 + C_2 + C_3) \cdot V}{I} + \tau_{RC} \qquad (5)$$

sketched in Figure 5.3b along with the waveform when the resistance is 0 (dashed line). The time constant τ_{RC} of the network is evaluated as in Equation 4 and is easily general-

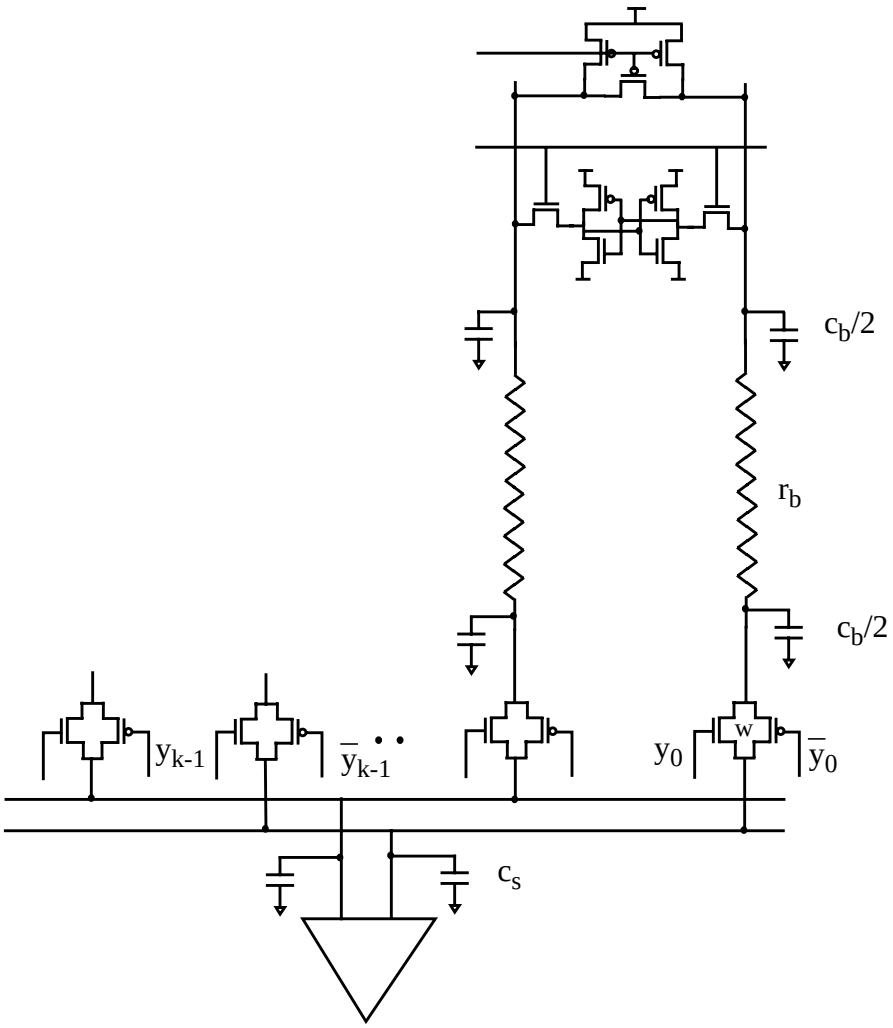


Figure 5.4: Bitline circuit

ized for an arbitrary RC chain as the sum of the product of each resistance with a

capacitance which is obtained by considering all the downstream capacitance lumped together, in series with all the upstream capacitance lumped together. In steady state (t >> τ_{RC}), nodes 1, 2 and 3 slew at the same rate, and the delay to obtain a swing of V at node 3

$$D_b = (c_b + j_{mux} \cdot (k + 1) + c_s) \cdot \frac{\delta v}{I_{cell}} + \alpha \cdot T_r + \tau_{RC} \qquad (6)$$

c_b: Bitline capacitance

j_{mux}: Unit junction capacitance of the mux switch

k: Number of columns multiplexed into a single sense amplifier

c_s: Input capacitance of the sense amplifier

δv: Voltage swing at the input of the sense amplifier

I_{cell}: Memory cell current

T_r: Local word line rise time

α: Proportionality constant determined from HSPICE

τ_{RC}: Time constant of the bitline RC network

can be approximated by Equation 5, which is the delay when there is no resistance plus the time constant of the network. This formula is used for estimating the delay of both the bitline and data line muxes.

A single level bitline mux is shown in Figure 5.4 and is modeled as an ideal current source driving a RC network as in Figure 5.3. Local and global bitline wires and the mux switches contribute to the capacitances and resistances in the network. The bitline delay to obtain a signal swing of δv by Equation 7 is the sum of the delay to generate the voltage swing with no resistance and the time constant of the RC network (Equation 6). Long local word lines can have slow rise times because of the line resistance. Since the rise time affects the cell delay, we need to include it in the delay model. The effect of the rise time (T_r) can be captured by adding an additional term to the delay equation which is proportional to it [57]. The proportionality constant, α, depends on the ratio of the threshold voltage of the access device in the cell, to the supply voltage and we find it from simulations to be about 0.3 for a wide range of block widths. The RC time constant, τ_{RC}, in the

bitline delay equation is estimated as in Equation 4. Figure 5.5 graphs the estimated and

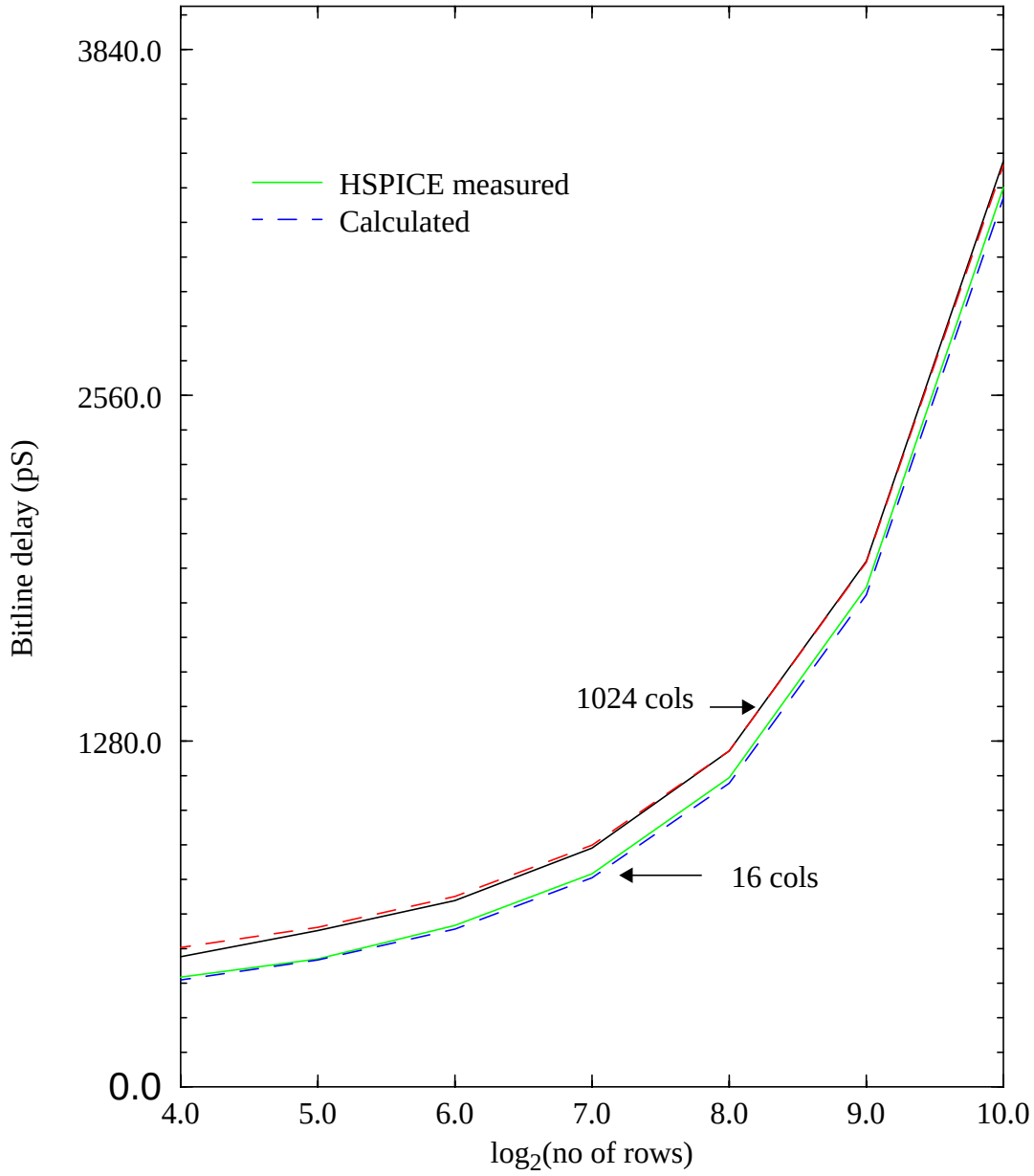


Figure 5.5: Bitline delay versus column height; 0.25µm, 1.8V and 4 columns multiplexing

HSPICE measured delay through the local word driver and the resistive word line and bit-

line, up to the input of the sense amps. The estimated delay is within 2.4% of the HSPICE delay when the bitline height is at least 32 rows, for both short word lines (16 columns) and long word lines (1024 columns).

The sense amplifier buffer chain is shown in Figure 5.6 and consists of the basic cross

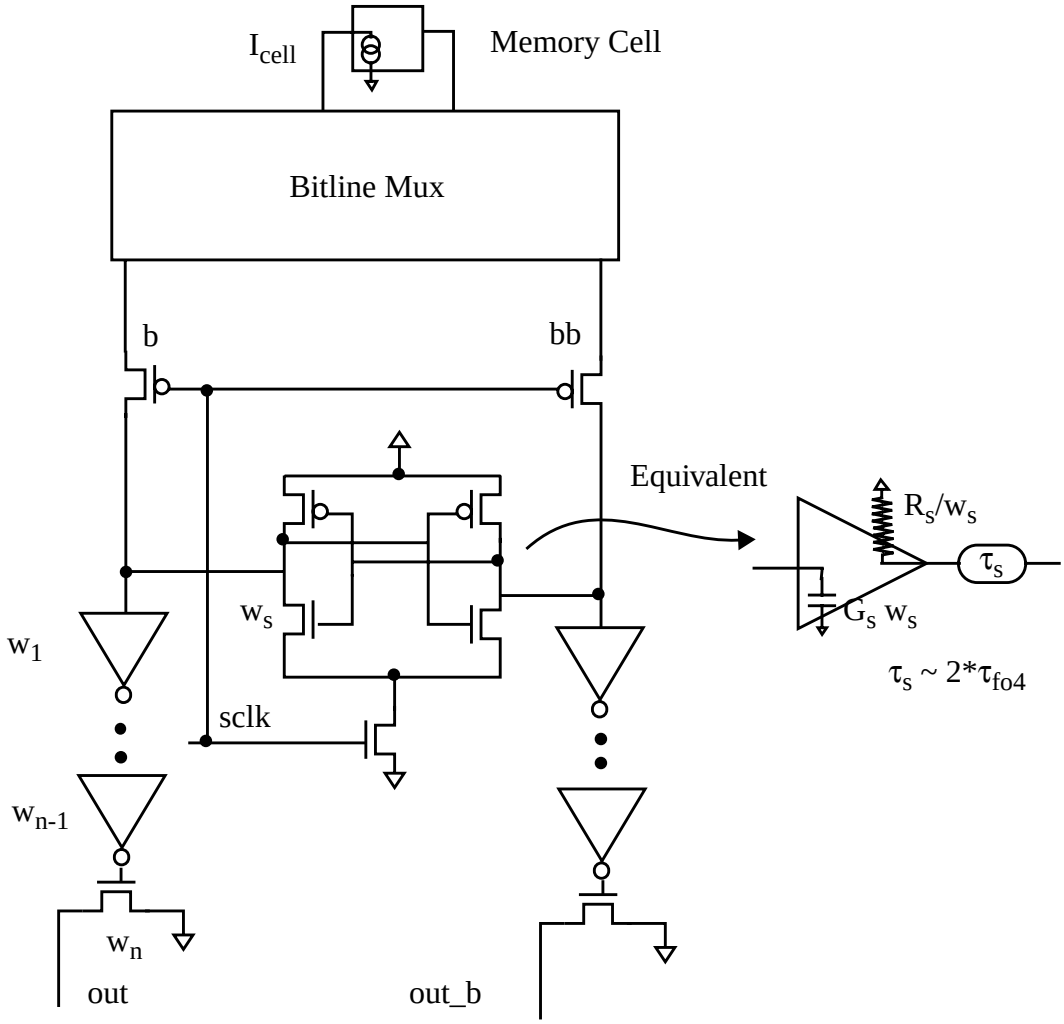


Figure 5.6: Local sense amplifier structure

coupled latch followed by a chain of inverters and a pair of nMOS drivers [26, 28]. The latch converts the small swing input signal to a full swing CMOS signal and is used for both the local and global sense amplifiers. In the case of the local sense amplifiers, the latch output is buffered by the inverter chain and driven onto the gates of the output nMOS

drivers. These nMOS transistors create a small swing voltage signal at their outputs by discharging a previously precharged data lines (analogous to the memory cell discharging the precharged bitlines). The delay of the sense amplifier structure is the sum of the delay of the latch amplifier, τ_s, and the delay to buffer and drive the outputs. τ_s is proportional to the logarithm of the desired voltage gain and the loading of the amplifier outputs [63]. For a gain of about 20 with only the self loading of the amplifier, τ_s is found to be about 2 τ_{fo4} by both calculations and circuit simulations. If we assume that all the transistors of the latch are scaled in the same proportion, then its output resistance and input capacitance can be expressed as simple functions of the size of the cross coupled nMOS in the latch, w_s, as shown in Figure 5.6. The nMOS drivers are modeled as current sources, with their

$$T = D_b(w_s) + \tau_s + \frac{R_s \cdot 3 \cdot C_g \cdot w_1}{w_s} + \frac{R_g \cdot 3 \cdot C_g \cdot w_2}{w_1} \cdots \frac{R_g \cdot C_g \cdot w_n}{w_{n-1}} + \frac{C \cdot \delta v}{I_n \cdot w_n} \quad (7)$$

$$+ \textit{other constants}$$

$$D_b(w_s) \approx \frac{G_s \cdot w_s \cdot \delta v}{I_{cell}} \quad (8)$$

$\tau_s = 2\ \tau_{fo4}$; amplification delay of the latch sense amp

G_s = 3.8fF/λ: sense amp input capacitance per unit width in 0.25μm process

R_s = 36kΩ-λ: sense amp output resistance per unit width

w_s: size of sense amp

R_g, C_g: output resistance and input capacitance per unit width of a 2:1 inverter

I_n = 37.5 μA/λ: current per unit width of nMOS

C: Capacitance of the data line mux

current output proportional to their size w_n. As in the decoders, optimal sizes w_s, w_1,.. w_n are determined to minimize the total output mux delay. Equation 7 captures the relevant portions of the output mux delay needed for doing this optimization, and is the sum of the delays of the bitline mux, the latch sense amp, the buffers and the nMOS drivers. To sim-

plify the procedure for finding the optimal sizes, impact of the latch sense amp size on the bitline mux time constant is ignored and only the cell delay is considered (Equation 8). Similarly, we ignore the effect of the nMOS junction capacitance on the data line RC time constant. Both these factors have little influence on the optimal sizing, but we include them for the final delay calculations. The minimum delay through the sense amp structure occurs when each term in Equation 7 is equal to the extrinsic delay of a fanout of 4 loaded inverter. The delay of the global sense amp is estimated in a similar fashion, except that the buffering delay to drive the output load is not considered in this analysis.

With technology scaling if the transistor threshold mismatch in the sense amplifier doesn't scale, then the delay of the output mux has a component which doesn't scale. This component is the delay of the memory cell to generate the bitline swing of δv_m, which is the input offset voltage of the sense amplifier. Hence for delay estimations in future technologies, we keep this component a constant.

For low power operation the signals on high capacitance nodes like the bitlines and the data lines are clamped to have small voltage swings [28]. Bitlines are clamped by pulsing the word lines, resulting in a total signal swing of about $2 * \delta v$ (the data lines are clamped in an analogous fashion to have similar signal swings). Hence the energy of the bitline and data line mux is computed as $C * V_{dd} * 2 * \delta v$, where C is the capacitance on the line and includes the wire, junction and input gate capacitances and V_{dd} is the supply voltage. The energy of a unit sized sense amp is obtained from simulations to be $12fJ/\lambda$ for the 0.25μm process and it is scaled up by w_s to obtain the sense amp energy.

The area of the switches in the bitline mux and the circuitry of the sense amplifier, precharge and write drivers add to the vertical area of the SRAM array (Figure 5.7). We base

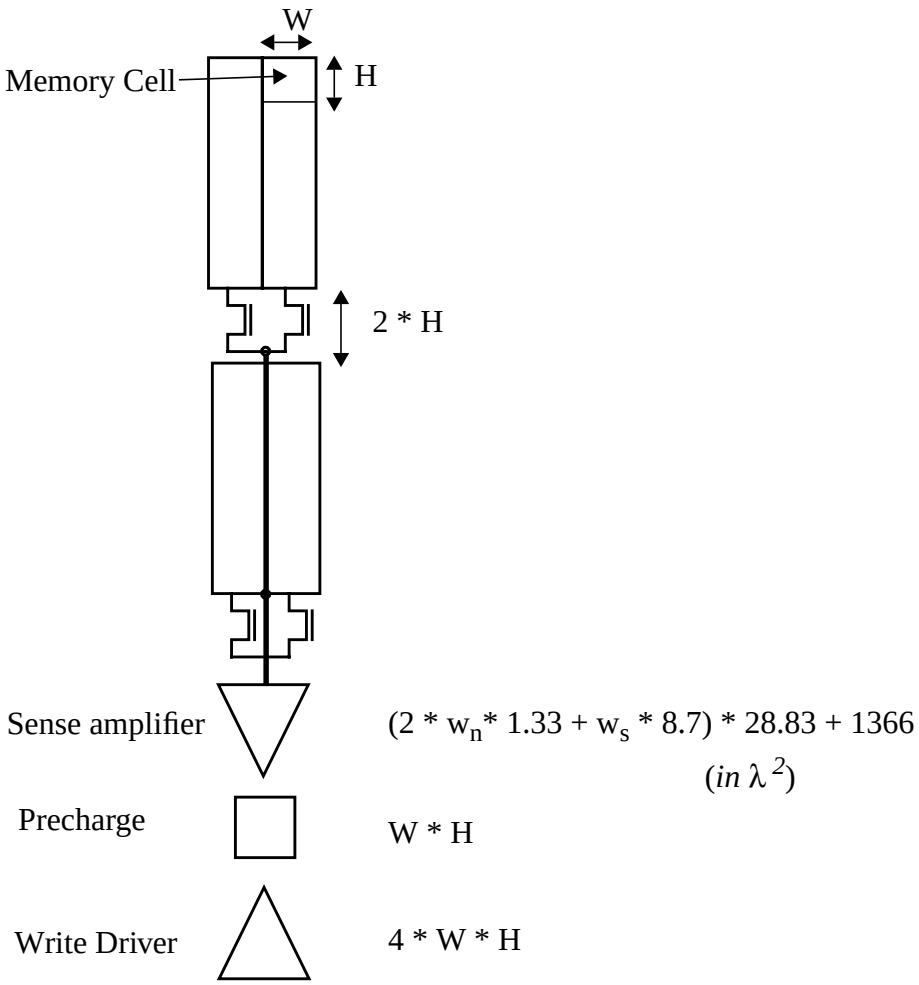


Figure 5.7: Area estimation of the output mux

the area estimates of these components on data from a previous design [30]. Since the write driver, precharge and mux transistors are not optimized, we add a fixed overhead of 4, 1 and 2 memory cells respectively. The area of the local sense amps is modeled as a linear function of the total device width within the sense amp. The parameters to the model are obtained by fitting it to the data obtained from five different designs [28, 30] and is shown in Figure 5.7. The total device width within the sense amp structure is itself esti-

mated from the size parameters w_s, w_n & w_p. The sum of all the device widths within the latch is estimated as w_s * 8.7, where the factor of 8.7 is obtained for the latch design in [30]. With fanout of 4 sizing, the active area of the buffers prior to each nMOS output driver is no more than 1/3 of the driver width w_n. Hence the active area of two nMOS drivers and their respective buffers is given by 2 * w_n * 1.33.

We will next describe the results obtained by using these models to analyze many RAM organizations of different sizes in various technology generations.

5.3 Analysis Results

We enumerate all the RAM organizations and estimate the area, delay and energy of each using the simple models described previously. This allows us to determine the optimal organizations which minimize a weighted objective function of delay, area and energy

$$\underset{\text{[over all partitions]}}{\textit{minimize}} \quad (1-\beta-\alpha)\cdot Delay + \alpha \cdot Area + \beta \cdot Energy \tag{9}$$

(Equation 9). The trade-off curves are also obtained between these, by varying the weight values α and β between 0 and 1.

Figure 5.8 plots the delay of SRAMs organized for minimum delay ($\alpha = \beta = 0$ in Equation 7), with and without wire resistance, for sizes from 64Kb to 16Mb with access width of 64 bits, in the 0.25μm technology. The delay of the SRAM without wire resistance, is about 15 τ_{fo4} for a 64Kb design, and is proportional to the log of the capacity as observed in [56]. The delay increases by about 1.2 τ_{fo4} for every doubling of the RAM size and can be understood in terms of the delay scaling of the row decoder and the output path. The delays for both of these are also plotted in the same graph and are almost equal in an optimally organized SRAM. In the case of the row decoder, each address bit selects half the array, and hence the loading seen by the address bit is proportional to S/2, where S is the total number of bits in the array. With fanout 4 sizing rule, the number of stages in

the decoder will be proportional to the logarithm to base 4 of the total load, with each stage having approximately the delay of one τ_{fo4}. Hence, each doubling in number of bits adds about half a τ_{fo4} delay. In the case of the output path, the wire capacitance in the data

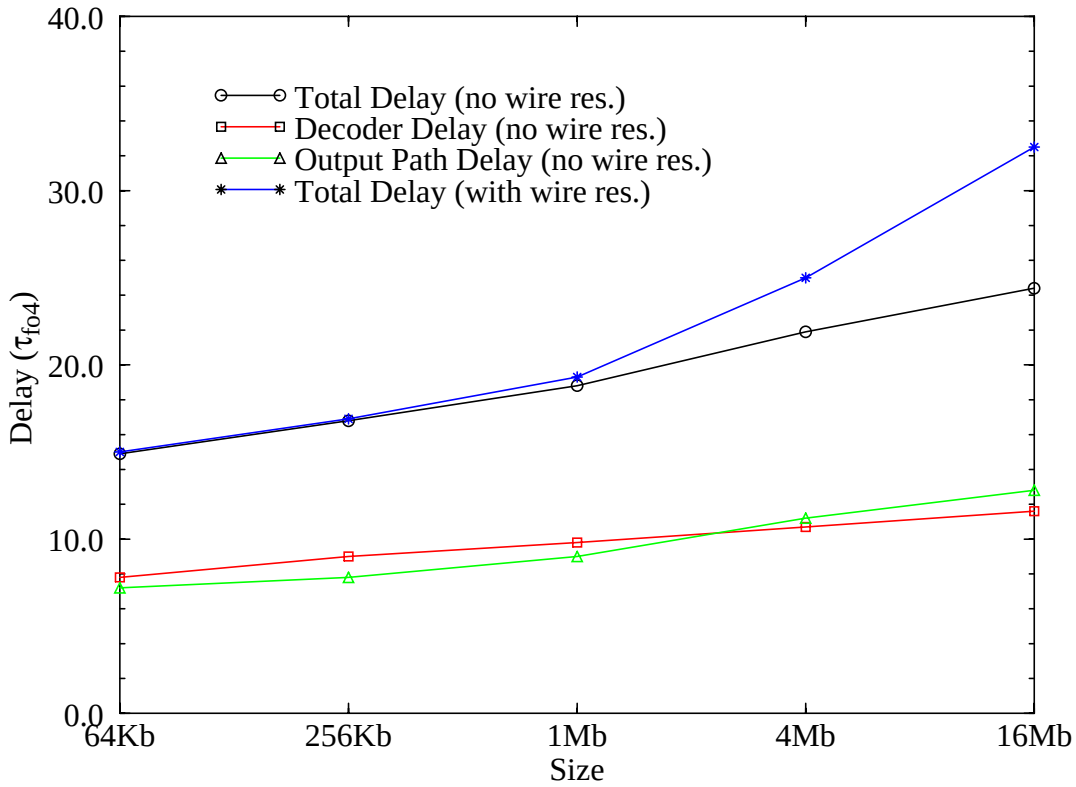


Figure 5.8: Delay scaling with size in 0.25µm process

line mux increases by about 1.4 for every doubling of the size, since it is proportional to the perimeter of the array and hence the delay of the local sense amps increase by about 0.25 τ_{fo4}. The remaining increase comes about due to the doubling of the multiplexor size for the bitline and the data line mux and its exact value depends on the unit drain junction capacitance and the unit saturation current of the memory cell and the nMOS output drivers.

The final curve in Figure 5.8 is the SRAM delay with wire resistance. The global wires for this curve are assumed to have a width of 10λ, (7.5 ohms/mm). Since wire RC delay

grows as the length of the wire, the wire delay for global wires in the SRAM scales as the size of the SRAM and becomes dominant for large sized SRAMs.

Wire width optimization can be done to reduce the impact of interconnect delay. Figure 5.9 shows the total delay for the 4Mb SRAM for two different wire widths in four different technology generations. It is assumed that the metallization in 0.18um and below is in copper. The lowest curve plots the delay when the wire resistance is assumed to be

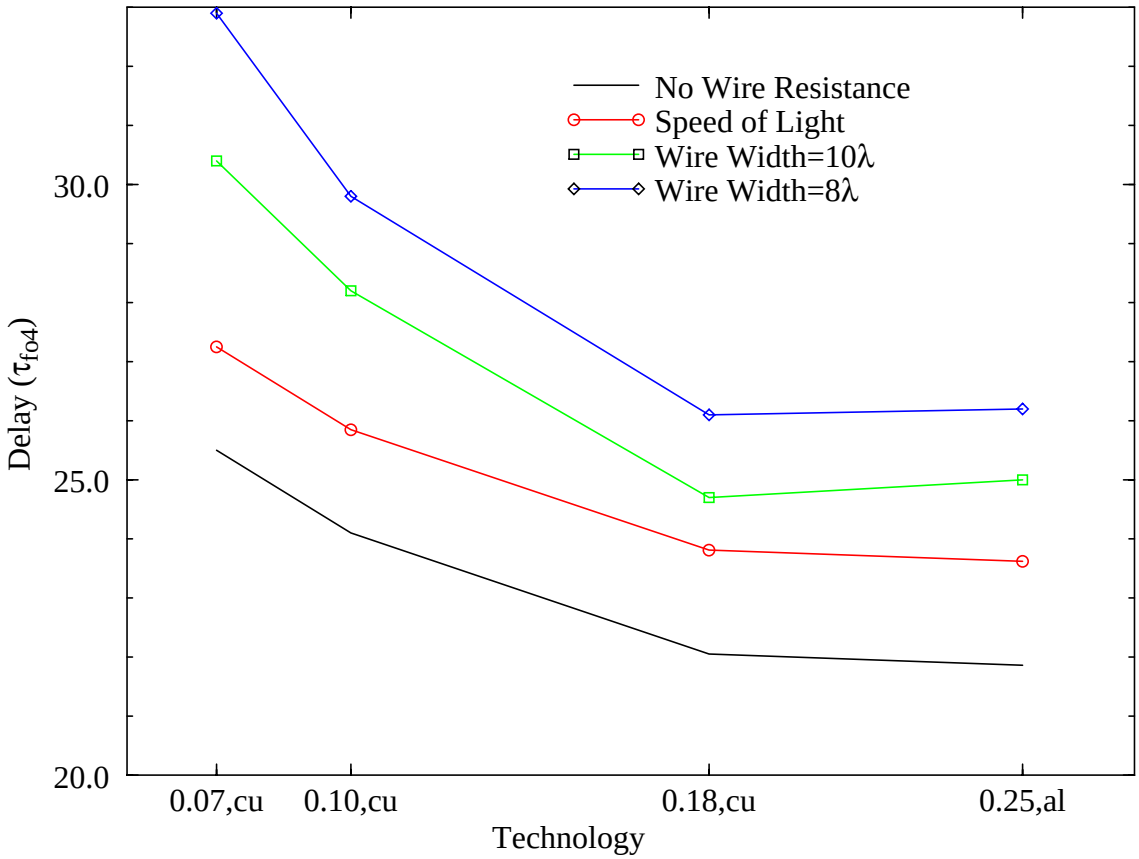


Figure 5.9: Delay versus technology for different wire widths for a 4Mb SRAM

zero. Since the threshold voltage mismatch remains constant with technology scaling, the bitline and data line signal swing do not scale in proportion to the supply voltage and hence their delays will get worse relative to the rest of the RAM. As seen in the figure, the delay of the RAM increases by about 2.2 τ_{fo4} for the 0.1µm and by 3.6 τ_{fo4} for the 0.07µm, when interconnect delay is ignored. The second curve adds the round trip signal

delays around the access path assuming a speed of light propagation of 1mm/6.6pS and gives the lower bound for interconnect delay. The speed of light interconnect delay is about 1.75 τ_{fo4} for the 4Mb SRAM, independent of the technology and doubles for every quadrupling of RAM size. The two curves above it graph the delay with wire resistance being non-zero for two different wire widths of 8λ and 10λ. Significant reduction in wire delay is possible when fat wires are used for global wiring. Going from 0.25μm with aluminum wiring to 0.18μm copper wiring, essentially leaves the delays (in terms of τ_{fo4}) unchanged, but with further shrinks of the design, the delay for any particular wire width worsens, since the wire RC delay doesn't scale as well as the gate delay. But by widening the wires in subsequent shrinks, it is possible to maintain the same delay (in terms of τ_{fo4}) across process generations. A wire width of 10λ brings the delay within a τ_{fo4} of speed of light limit at the 0.25μm and 0.18μm generations, while wider wires are needed in the 0.1μm and 0.07μm generations. The larger pitch requirements for these fat wires can be easily supported when the divided word line structure in the decoders and column multiplexing in the bitlines are used.

We will next look at some ways in which performance of actual SRAM implementations might differ from those predicted by the previous curves. Large SRAMs typically incorporate some form of row redundancy circuits in the decode path. This usually takes the form of a series pass transistor in the local word driver and will cause the delay curves to shift up by about 1/2 to 1 τ_{fo4}. Fanouts larger than 4 in the word line driver, commonly done to reduce area, will also shift the delay curves up by about 1/2 to 1 τ_{fo4}. High performance SRAMs don't use static circuit style in the entire decode path, but skew the gates in the predecoders and the global word drivers to favor a fast word line activation [20, 21], causing the delay curves to shift down. As discussed in Chapter 3, the delay of a skewed gate is about 70% that of a non skewed gate and grows slower with increasing load. When applied to the predecoder and the global word driver the delay of the decoder in the 64Kb RAM reduces to about 6 τ_{fo4} instead of the 8 τ_{fo4} for the static implementation leading to a net decoder gain of up to 1 τ_{fo4} when combined with the losses due to redundancy and smaller local word driver. But with every doubling of the RAM size, the decoder delay

will increase by about 0.3 τ_{fo4} instead of 0.5 τ_{fo4} for the static case. In the output path, the sense clock for the local sense amplifiers is usually padded with extra delay, to enable operation over a wide range of process conditions [21, 28] which incurs an additional delay of up to 1 τ_{fo4}, when bitlines are short. With these changes, we would expect the total delay to start off at about the same point in Figure 5.8 but grow at a slower rate with size than predicted, due to better decoder performance. The curves in Figure 5.8 indicate that the decoder and output path delays are about equal. As we shall see later, for these points, the memory has been partitioned to have the smallest block possible, to minimize the local bitline delay and the output path delay cant be reduced further by partitioning. Thus with dynamic decoders, the output path delay will not be able to keep pace with the decoder delay at least for the circuits we have assumed.

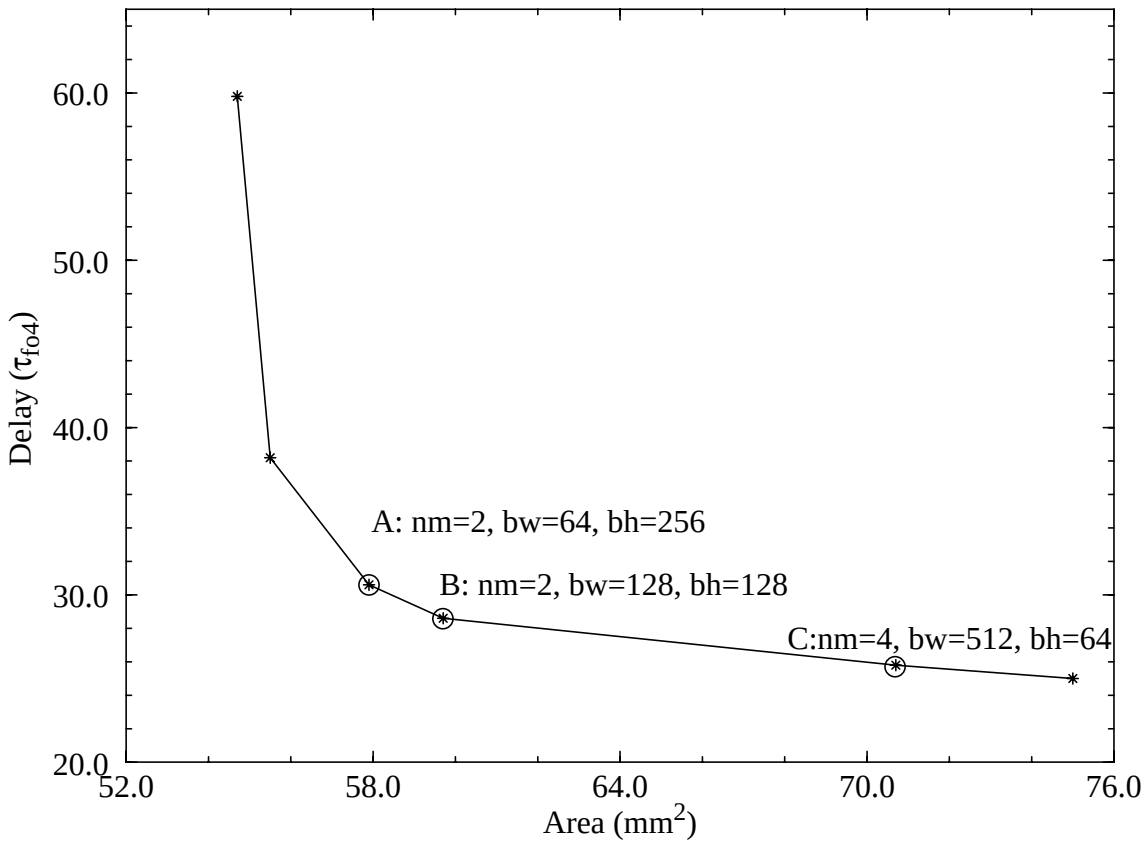


Figure 5.10: Delay versus Area for a 4Mb SRAM in 0.25µm process

Partitioning allows for a trade-off between delay, area, and power. Trade-off curves can be obtained by solving Equation 9, with various values for the parameters α and β. When β equals zero, the delay-area trade-off is obtained and the curve for a 4Mb SRAM in 0.25μm process is shown in Figure 5.10. Any point on this curve represents the lowest area achievable via RAM reorganization for the corresponding delay. Starting from a minimum delay design which is finely partitioned, significant improvements in the area is possible by reducing the amount of partitioning and incurring a small delay penalty, while subsequent reduction in partitioning results in decreasing improvements in area for increasing delay penalty. Partitioning parameters for three points A, B and C are shown in the figure. Points A & B are in the sweet spot of the curve, with A being about 22% slower and 22% smaller area and B being 14% slower and 20% smaller area when compared to the fastest implementation.

Of the various organization parameters, the RAM delay is most sensitive to the block height and fast access times are obtained by using smaller block heights. Figure 5.11 shows the delay and area for a 4Mb SRAM for various block heights, while using optimal values for the remaining organization parameters. Small block heights reduce the delay of the bitlines, but increase the delay of the global wires since the RAM area is increases due to the overhead of bitline partitioning. For very large block heights, the slow bitline delay limits the access time. Hence an optimum block height exists and is 32 rows for the example above. Increasing the block height to 128 rows incurs a delay penalty of about 8%

while the area can be reduced by 7.6%, illustrating the area - delay trade-off that are possible via partitioning.

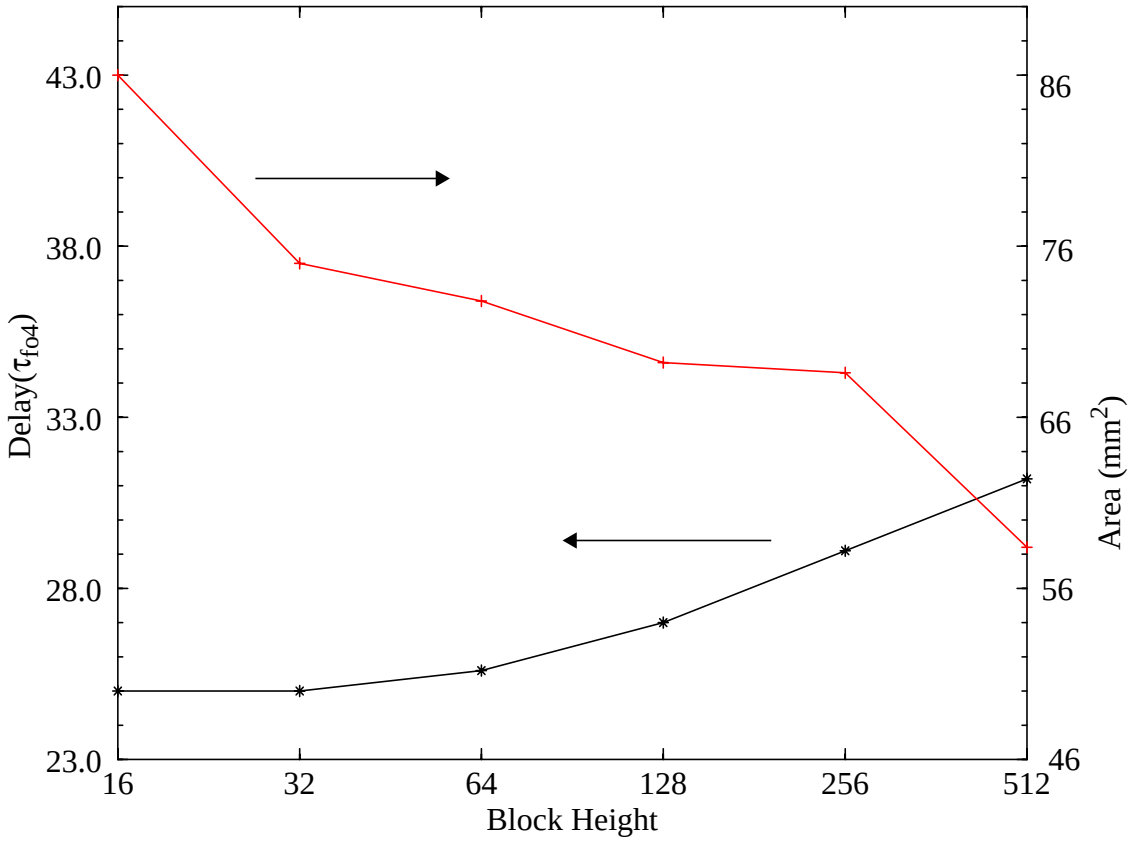


Figure 5.11: Delay and Area versus block height for a 4Mb SRAM in a 0.25µm process

By setting α equal to 0 in Equation 9, one can obtain the delay-energy trade-off through partitioning, with no constraints on the area, and is shown in Figure 5.12. The unit used on the left hand vertical axis is the energy consumed to switch the gate of a 16λ/32λ sized inverter (Eunit = 72fJ). Partitioning allows for a large trade-off between energy and delay as noted in [58-59]. The figure also indicates the optimal degree of column multiplexing (cm) and the block height (bh) required to obtain the corresponding delay and energy for some of the points. We find that for low energy solutions, the column multiplexing is one, i.e. the block width is equal to the access width, since this enables only the minimum number of bitline columns to switch. Since we do sizing optimization to mini-

mize delay, the final transistors in the output of the local sense amps become large and consequently have a large capacitance associated with their drain junction capacitance. Hence in the low energy designs, it is advantageous to have large block heights as noted in [58-59], since this allows most of the multiplexing to be done in the bitline mux where the junction capacitances from the memory cell's access transistor are very small compared to

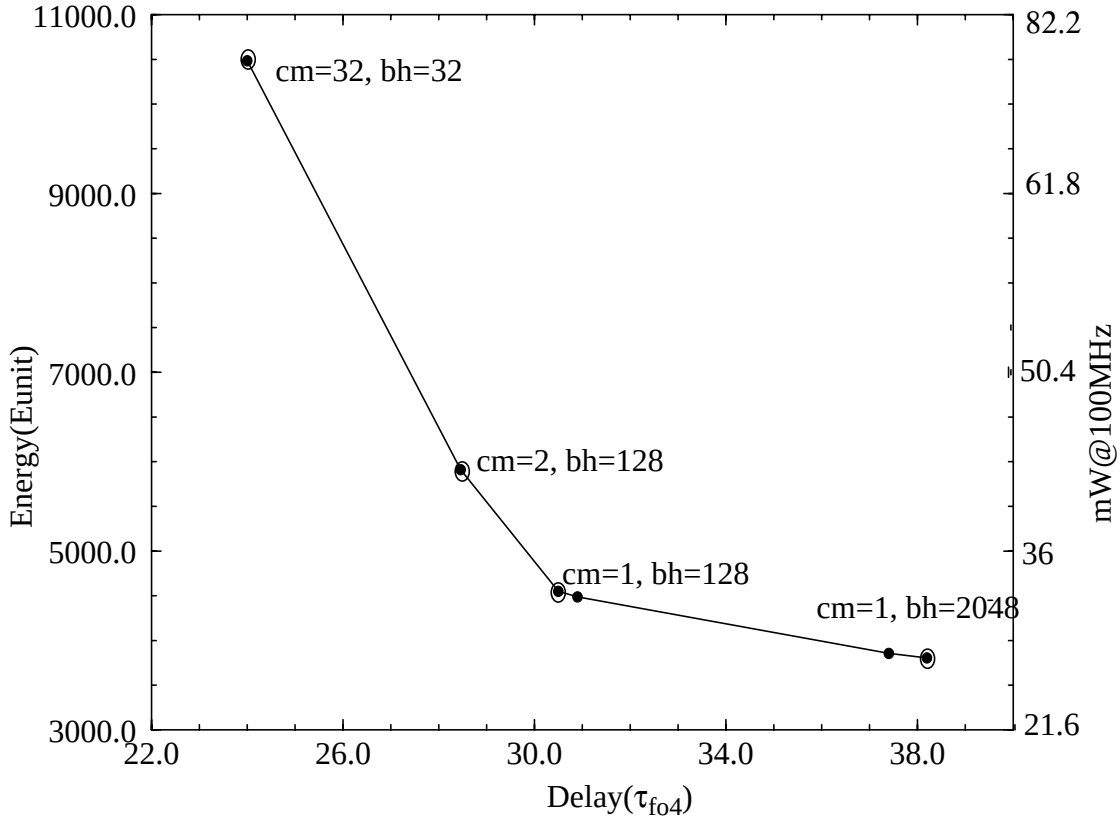


Figure 5.12: Energy versus Delay for a 4Mb SRAM in 0.25µm process

the junction capacitances in the data line mux. We also find that the energy consumption in optimally organized SRAMs, can be expressed as a sum of two components. One is independent of the capacity and depends only on the access width, and is due to the local word line, the precharge signal, local and global sense amps etc. The other component scales as the square root of the capacity as observed in [58-59] and is related to the power dissipation in the global wires and the decoders.

The trade-off curves of Figures 5.10 and 5.12 can be improved upon if one were to also perform transistor size optimization along with partition optimization. We would expect the optimal trade-off curves to have the same high speed end point as those shown in the figures, but rotated towards the origin (since the sizing is done for fastest speed the corresponding end point is already optimal). We would also expect the optimal trade-off curve to support smaller block sizes since the area and energy overhead of these will be offset by the smaller transistor sizes.

5.4 Conclusions

Analytical models for delay, area and energy, allows one to explore a range of design possibilities in a very short span of time. These models are used to study the impact of SRAM partitioning and it is found that a substantial trade-off between area, delay and energy can be obtained via the choice of SRAM organization.

The models are also used to predict the scaling trends of delay with capacity and process technology. The delay of SRAM can be broken into two components, one is due to the transistors in the technology (gate delay) and the other is due to the interconnect (wire delay). The gate delay increases by about 1.2 τ_{fo4} for every doubling of the RAM size, starting with 15 τ_{fo4} for a 64Kb RAM, when a static circuit style is used to design the decoders.

Non-scaling of threshold mismatches with process scaling, causes the signal swings in the bitlines and data lines also not to scale, leading to an increase in the gate delay of an SRAM across technology generations. For an optimally organized 4Mb SRAM, the increase in delay is about 2 τ_{fo4} in the 0.1μm and 3.6 τ_{fo4} in the 0.07 μm generations, and is worse for other organizations. This delay increase for most SRAM organizations can be mitigated by using more hierarchical designs for the bitline and data line paths, and using offset compensation techniques like in [60, 64].

The wire delay starts becoming important for RAMs beyond the 1Mb generation. Across process shrinks, the wire delay becomes worse and wire redesign has to be done to

keep the wire delay in the same proportion to the gate delay. Divided word line structure for the decoders, and column muxing for the bitline path, opens up enough space over the array for using fat wires, and these can be used to control the wire delay for 4Mb and smaller designs across process shrinks. The wire delay is lower bounded by speed of light and this is about 1.75 τ_{fo4} for the 4Mb SRAM, and doubles with every quadrupling of capacity. Thus for high performance RAM designs at the 16Mb and higher level, the RAM architecture needs to be changed to use routing of address and data (see for example [65]), instead of the current approach where the signals are broadcast globally across the array.

Chapter 6

Conclusions

In this thesis we looked at design trade offs for fast low power SRAMs. The SRAM access path is split into two portions: the row decoders and the read data path. Techniques to optimize both of these were investigated. In Chapter 3 we sketched out the optimal decoder structure for fast low power SRAMs. Optimal decoder implementations result when the decoder, excluding the predecoder, is implemented as a binary tree. This minimizes the power dissipation as only the smallest number of long decode wires transition. Also this doesn't introduce any delay penalty if the 2-input NAND function is implemented in the source coupled style, since in this style the delay is almost the same as that of an inverter. Except for the final word driver in the decoder, all the other stages should be implemented using skewed circuit techniques with self resetting. The highest performance predecoders will have a NOR style wide fanin input stage followed by skewed buffers. When this is coupled with a technique like that presented by Nambu et.al. in [21] to do a selective discharge of the output, the power dissipation is very reasonable compared to the speed gains that can be achieved. With the NOR style predecoder the total path effort becomes independent of the exact partitioning of the decode tree, which will allow the SRAM designer to choose the best memory organization, based on other considerations.

Transistor sizing offers a great tool to trade off delay and energy of the decoders. The presence of RC limited interconnect within the decode structure makes the decoder sizing problem a little different from the sizing of only logic gates. Full blown numerical optimization techniques can be used to obtain the various design points. The simple sizing heuristic of keeping a constant fanout of about 4 works well for high speed designs, as long as the wire delays are not significant. For fast lower power solutions, the heuristic of

reducing the sizes of the input stage in the higher levels of the decode tree allows for good trade-offs between delay and power. Getting the decode signal to the local word lines in a fast and low power manner solves half of the memory design problem.

The key to low power operation in the SRAM data path is to reduce the signal swings in the high capacitance nodes like the bitlines and the data lines. Clocked voltage sense amplifiers are essential for obtaining low sensing power, and accurate generation of their sense clock is required for high speed operation. An energy efficient way to obtain low voltage swings in the bitlines is to limit the word line pulse width, by controlling the pulse width of the block select signal. The pulse widths are regulated by the aid of a replica delay element which consists of a replica memory cell and a replica bitline and hence tracks the delay of the memory cell over a wide range of process and operating conditions. Two variations of the technique were discussed and their use to generate the sense clock was also demonstrated. We believe that this is a very simple and robust technique with very low area overhead and is easily applicable to a wide variety of SRAM designs. This method can also be used to design a low swing data line operation for both the read and write accesses, by pulsing the control signal which gates the drivers of these lines. The pulse width is controlled by discharging a replica line which has the same capacitance as the worst case data line. This replica line is also used to trigger the amplifiers at the receiving end of these lines. We finally presented a technique to achieve low bitline write power, by using small voltage swings on the bitlines during the write operation. The memory cell structure is modified such that the cells can be used as latch type sense amplifier to amplify and store the small swing write data presented on the bitlines. This technique requires boosted word lines and 20% larger area and hence is suitable only for certain niche applications.

We finally looked at scaling trends of delay and power in SRAMs as a function of their size and technology. Simple analytical models for delay, area and energy are used to explore a range of design possibilities. These models are used to study the impact of

SRAM partitioning and it is found that a substantial trade-off between area, delay and energy can be obtained via the choice of SRAM organization.

The models are also used to predict the scaling trends of delay with capacity and process technology. The delay of SRAM can be broken into two components, one is due to the transistors in the technology (gate delay) and the other is due to the interconnect (wire delay). The gate delay increases by about 1.2 τ_{fo4} for every doubling of the RAM size, starting with 15 τ_{fo4} for a 64Kb RAM, when a static circuit style is used to design the decoders.

Non-scaling of threshold mismatches with process scaling, causes the signal swings in the bitlines and data lines also not to scale, leading to an increase in the delay of an SRAM across technology generations. For an optimally organized 4Mb SRAM, the increase in delay is about 2 τ_{fo4} in the 0.1μm and 3.6 τ_{fo4} in the 0.07 μm generations, and is worse for other organizations. This delay increase for most SRAM organizations can be mitigated by using more hierarchical designs for the bitline and data line paths, and using sense amplifier offset compensation techniques like in [60, 64].

The wire delay starts becoming important for RAMs beyond the 1Mb generation. Across process shrinks, the wire delay becomes worse and wire redesign has to be done to keep the wire delay in the same proportion to the gate delay. A divided word line structure for the decoders, and column muxing for the bitline path, opens up enough space over the array for using fat wires, and these can be used to control the wire delay for 4Mb and smaller designs across process shrinks. The wire delay has a lower bound set by the speed of light and this is about 1.75 τ_{fo4} for the 4Mb SRAM, and doubles with every quadrupling of capacity. Thus for high performance RAM designs at the 16Mb and higher level, the RAM architecture needs to be changed to use routing of address and data (see for example [65]), instead of the current approach where the signals are broadcast globally across the array. Wire delay is also directly proportional to the cell area, and hence cell designs with smaller area will win out for large RAMs, even if the cells are weaker. Thus

the DRAM cell, multi-valued cells, TFT based cells, and other novel cell designs will be worth investigating for designing future high performance high capacity RAMs.

Appendix A

Optimum sizing of buffer chains with intermediate interconnect

In this appendix we will derive the conditions for optimum fan up of chains of buffers with intermediate interconnect. For a simple chain of inverters driving some fixed load C_L, the optimum fan up, f, is obtained by solving (1). In this equation p is the ratio of drain

$$\ln(f) - 1 - \frac{p}{f} = 0 \qquad (1)$$

junction capacitance to the gate capacitance. We will first solve the two subchain case and then use the results to solve the three subchain case.

A.1 Chain of buffers with one stage of intermediate interconnect

Consider a chain which has two sub chains of buffers which are separated by an interconnect having a resistance *R* and capacitance *C* (Figure 1). The total delay can be

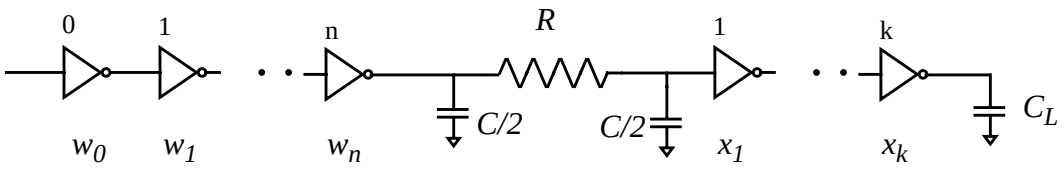


Figure 1: Buffer chain with intermediate interconnect

expressed as the sum of delays for each stage plus the delay of the interconnect as in (2).

$$D = \underbrace{\frac{w_1}{w_0} + \frac{w_2}{w_1} \bullet\bullet \frac{(C + x_1)}{w_n}}_{\text{Delay of } 1^{st} \text{ chain}} + \underbrace{R \cdot (C/2 + x_1)}_{\text{Interconnect}} \underbrace{\bullet\bullet \frac{C_L}{x_k}}_{2^{nd} \text{ chain}} + \underbrace{(n + k + 1) \cdot p}_{\text{Parasitic delay}} \qquad (2)$$

Delay is minimized when the gradient of delay with respect to the sizes is zero and is achieved when conditions (3), (4) are satisfied, viz., the efforts of all the gates in the first chain are equal (and denoted as f_1), and the efforts of all the gates in the second chain are equal (and denoted as f_2). The total path delay for minimum delay can than be rewritten in

$$\frac{w_1}{w_0} = \frac{w_2}{w_1} \quad = \frac{(C + x_1)}{w_n} = f_1 \qquad (3)$$

$$(R + 1/w_n) \cdot x_1 = \frac{x_2}{x_1} \bullet\bullet = \frac{C_L}{x_k} = f_2 \qquad (4)$$

$$D = (n + 1) \cdot f_1 + (RC)/2 + (k + 1) \cdot f_2 - \frac{x_1}{w_n} + (n + k + 1) \cdot p \qquad (5)$$

terms of the optimal efforts as in (5). Also any $w_i = w_0 * f_1^{\,i}$ and $x_i = C_L/f_2^{(k-1-i)}$. The optimum values of f_1, f_2, n and k can now be found by taking the gradient of D in (5) with respect to these and setting it to 0. To simplify the description we will call x_1/w_n as A (6) and write the natural logarithm of f_1 and f_2, i.e. $ln(f_1)$ and $ln(f_2)$, as l_{f1} and l_{f2}. A differential

$$A = \frac{x_1}{w_n} = \frac{C_L/w_0}{f_1^n \cdot f_2^k} \tag{6}$$

change in A is related to the differential changes in f_1, f_2, n, k as shown in (7). The differen-

$$\Delta A = -A \cdot \left[l_{f1} \cdot \Delta n + \frac{n}{f_1} \cdot \Delta f_1 + l_{f2} \cdot \Delta k + \frac{k}{f_2} \cdot \Delta k \right] \tag{7}$$

tial change in D can then be expressed as in (8). Since the chain has only two subchains,

$$\Delta D = [f_1 + A l_{f1} + p]\Delta n + [f_2 + A l_{f2} + p]\Delta k + \left[n + 1 + A\frac{n}{f_1} \right]\Delta f_1 + \left[k + 1 + A\frac{k}{f_2} \right]\Delta f_2 \tag{8}$$

there are only two independent variables in the problem and we choose them to be f_1 and f_2. Hence we can express Δn and Δk in (8) in terms of Δf_1 and Δf_2. Expanding the last term of (3) and the first term of (4) we get (9) and (10). Taking a differential of these we get

$$C + \frac{C_L}{f_2^k} = f_1^{n+1} \cdot w_0 \tag{9}$$

$$R \cdot C_L + \frac{C_L}{f_1^n} = f_2^{k+1} \cdot w_0 \tag{10}$$

(11) and (12), where A is as defined in (6). We can now use these two equations to express

$$f_1 \cdot l_{f1} \cdot \Delta n + (n+1) \cdot \Delta f_1 + A \cdot l_{f2} \cdot \Delta k + A \cdot \frac{k}{f_2} \cdot \Delta f_2 = 0 \qquad (11)$$

$$f_2 \cdot l_{f2} \cdot \Delta k + (k+1) \cdot \Delta f_2 + A \cdot l_{f1} \cdot \Delta n + A \cdot \frac{n}{f_1} \cdot \Delta f_1 = 0 \qquad (12)$$

Δn and Δk in terms of Δf_1 and Δf_2. Doing (11) * f_2 - (12) * A leads to an expression for Δn in (13) (similarly for Δk in (14)). Putting (13) and (14) in (8) and simplifying leads to an

$$\Delta n = \frac{A \cdot \Delta f_2}{(f_1 \cdot f_2 - A^2) \cdot l_{f1}} - \frac{(n+1) \cdot f_2 - A^2 \cdot \frac{n}{f_1}}{(f_1 \cdot f_2 - A^2) \cdot l_{f1}} \cdot \Delta f_1 \qquad (13)$$

$$\Delta k = \frac{A \cdot \Delta f_1}{(f_1 \cdot f_2 - A^2) \cdot l_{f2}} - \frac{(k+1) \cdot f_1 - A^2 \cdot \frac{k}{f_2}}{(f_1 \cdot f_2 - A^2) \cdot l_{f2}} \cdot \Delta f_2 \qquad (14)$$

expression of ΔD only in terms of Δf_1 and Δf_2 as shown in (15). Here C_{f1} and C_{f2} are

$$\Delta D \cdot (f_1 \cdot f_2 - A^2) \cdot l_{f1} \cdot l_{f2} = \Delta f_1 \cdot C_{f1} + \Delta f_2 \cdot C_{f2} \qquad (15)$$

expressed as in (16) and (17). Inspecting (16) and (17) we see that the sufficient condition

$$C_{f1} = (A^2 l_{f2} n - (n+1) f_1 f_2 l_{f2}) \cdot \left(1 + \frac{p}{f_1} - l_{f1}\right) + A l_{f1} f_2 \cdot \left(1 + \frac{p}{f_2} - l_{f2}\right) \qquad (16)$$

$$C_{f2} = (A^2 l_{f1} k - (k+1) f_1 f_2 l_{f1}) \cdot \left(1 + \frac{p}{f_2} - l_{f2}\right) + A l_{f2} f_1 \cdot \left(1 + \frac{p}{f_1} - l_{f1}\right) \qquad (17)$$

for ΔD to be zero is that (18) and (19) be satisfied and both of these have the same solution as (1) for the single buffer case.

$$1 + \frac{p}{f_1} - l_{f1} = 0 \tag{18}$$

$$1 + \frac{p}{f_2} - l_{f2} = 0 \tag{19}$$

Thus $f_1 = f_2 = f$ (which is about 4). Substituting this in (9) and (10) we can solve for n and k and are given in (20) and (21). Multiplying these two equations and rearranging the

$$f^n = \frac{C}{2fw_0} \cdot \left[1 + \sqrt{1 + \frac{4f}{RC}}\right] \tag{20}$$

$$f^k = \frac{RC_L}{2f} \cdot \left[1 + \sqrt{1 + \frac{4f}{RC}}\right] \tag{21}$$

terms we get (22) which relates the total number of stages in the chain to the output load.

$$f^{n+k+1} = \frac{C_L}{\frac{w_0}{\frac{RC}{4f} \cdot \left(1 + \sqrt{1 + \frac{4f}{RC}}\right)^2}} \tag{22}$$

Notice that this equation is very similar to the single chain case except that the input driver size w_0 is now derated by some factor which is only a function of the interconnect parasitics. Again rearranging the terms in (22) we get an expression for A which is dependent only on the intermediate interconnect parasitics as shown in (23). Hence the total delay of

$$A = \frac{x_1}{w_n} = \frac{C_L / w_0}{f^{n+k}} = \frac{4f^2RC}{\left(1 + \sqrt{1 + \frac{4f}{RC}}\right)^2} \tag{23}$$

the chain can be rewritten from (5) as in (24) and is very similar to the delay of a simple chain except for some additional terms independent of the final load. An equivalent buffer

$$D = (n + k + 1) \cdot (f + p) + \text{terms independent of the final load} \qquad (24)$$

chain with the same delay is shown in Figure 2 and consists of a chain of buffers preceded by an ideal delay element. The input driver size of the buffer chain is modified according to (22). We will use this fact to simplify the solution of the three subchain case in the next

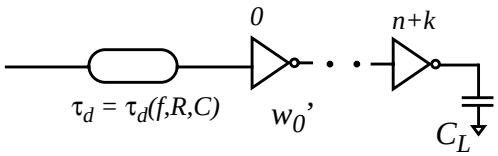


Figure 2: Equivalent model of the buffer chain used for delay calculation

section.

A.2 Chain of buffers with two stages of intermediate interconnect

Consider the chain shown in Figure 2 with two stages of intermediate interconnect. We

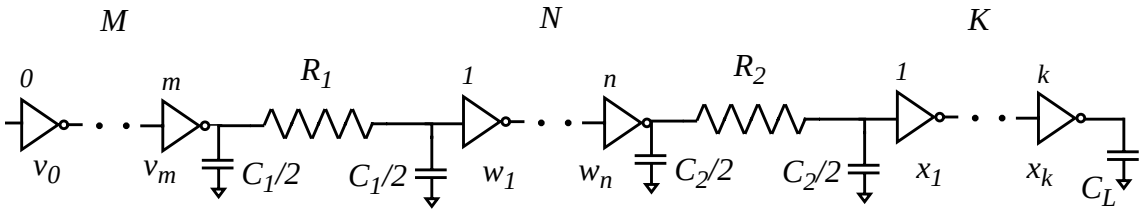


Figure 2: Buffer chain with two stages of intermediate interconnect

need to find the optimum number of stages and the effort for each of the three subchains, which we label as M, N and K subchains. For any size x_1 at the input of the K chain, the problem of optimally sizing M and N chains is the same as discussed in the previous sec-

tion and the optimum design has a stage effort of f and the number of stages and delay given by (21), (22) and (24) respectively. Hence the M and N subchains can be replaced by a single chain with no intermediate interconnect, but with an altered input driver v_0' as shown in Figure 4. From the previous section, the optimum stage effort for the K chain is

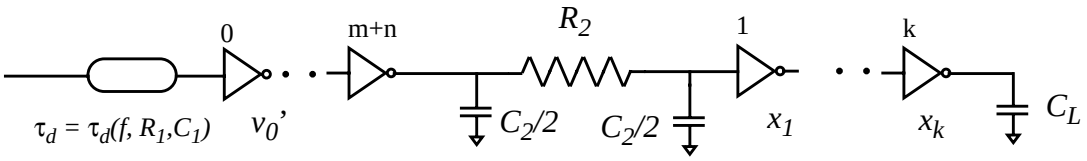


Figure 4: Equivalent model for delay calculation purposes

also *f*. Hence even for the three subchain case, the optimum stage efforts is *f* for all the stages. From (21) and Figure 4, we can write the expression for *k* as in (25). This can be

$$f^k = \frac{R_2 C_L}{2f} \cdot \left[1 + \sqrt{1 + \frac{4f}{R_2 C_2}}\right] \tag{25}$$

used to calculate the size of x_1 as C_L/f^k to give the loading for the first two subchains as C_2+x_1. Again using (20) and (21) for the M and N chains with this output load yields the expressions for *m* and *n* as in (26) and (27).

$$f^n = \frac{C_2}{C_1} \cdot \frac{\left(1 + \sqrt{1 + \frac{4f}{R_2 C_2}}\right)}{\left(\sqrt{\frac{4f}{R_1 C_1}} + 1 - 1\right)} \tag{26}$$

$$f^m = \frac{C_1 / v_0}{2f} \cdot \left(1 + \sqrt{1 + \frac{4f}{R_1 C_1}}\right) \tag{27}$$

Appendix A Optimum sizing of buffer chains with intermediate interconnect

Appendix B

Delay and Energy Trade-off in Row Decoders

The delay energy trade-off curve with respect to transistor widths can be obtained by minimizing the weighted objective function of the delay and energy as shown in (1). In

$$min_{w_1 \ldots w_n} D(w_1 \ldots w_n) + \lambda \cdot E(w_1 \ldots w_n) \quad (1)$$

this Equation, λ is a lagrangian parameter which is a positive real number. When λ is 0, the optimization leads to the fastest delay solution and when it is a large positive number, the optimization leads to the lowest energy solution and these form the two end points of the delay energy curve. Intermediate points on this curve are obtained by optimizing for a range of values for λ. Any point on this curve represents the fastest delay achievable via transistor sizing for the corresponding energy and equivalently, the lowest energy for the

corresponding delay. Consider a generic chain of *n+1* logic gates as shown in Figure 1.

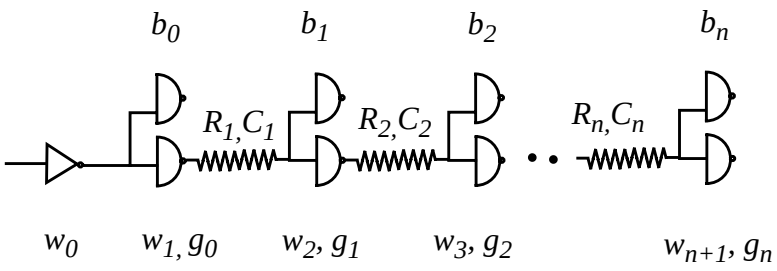


Figure 1: Schematic of a critical path

Each gate has a size w_i, logical effort g_{i-1} and a branching effort of b_i at its output. Its output also passes through an interconnect with resistance R_i and C_i. The delay of stage i in this chain is then given by (2) and is the delay of the gate to drive the output load consisting of the wire loading and the loading of the next stage plus the delay of the interconnect

$$i=0..n \qquad D_i = \frac{C_i + b_i \cdot g_i \cdot w_{n+1}}{w_i} + R_i \cdot C_i/2 + R_i \cdot b_i \cdot g_i \cdot w_{i+1} + p \qquad (2)$$

(here p is the self loading factor and is the ratio of the drain capacitance to the gate capacitance). The energy dissipation is the sum of the energies to switch all the capacitances at its output (3). The total objective function of (1) is then the sum of delay and energy of all

$$i=0..\,n \qquad E_i = C_i + b_i \cdot g_i \cdot w_{i+1} + p \cdot w_i \qquad (3)$$

the n+1 stages. For any given λ, the objective function is minimized by setting its gradient to 0. This leads to n equations which can then be simultaneously solved for each different

λ to obtain the *n* sizes, and a typical equation is shown in (4). The solution to these equa-

for i = 1.. n,

$$\frac{b_{i-1} \cdot g_{i-1}}{w_{i-1}} + R_{i-1} \cdot b_{i-1} \cdot g_{i-1} + \lambda \cdot (b_{i-1} \cdot g_{i-1} + p) - \frac{C_i + b_i \cdot g_i \cdot w_{i+1}}{w_i^2} = 0 \quad (4)$$

tions is guaranteed to be the minima due to the posynomial nature of the objective function [66]. We choose to solve the *n* simultaneous equations using a iterative procedure. This is done by starting off with an initial guess for the sizes and then by repeatedly using equation (5) which is a reorganization of (4), a sequence of estimates for the sizes is obtained.

$$w_{i,new} = \sqrt{\frac{C_i + b_i \cdot g_i \cdot w_{i+1}}{R_{i-1} \cdot b_{i-1} \cdot g_{i-1} + \frac{b_{i-1} \cdot g_{i-1}}{w_{i-1}} + \lambda \cdot (b_{i-1} \cdot g_{i-1} + p)}} \quad (5)$$

The procedure is terminated when a new estimate in this sequence differs by less than 1% from the previous one. The sizes are then used in a HSPICE simulation of the critical path to obtain the final delay and energy value for the point.

Appendix B Delay and Energy Trade-off in Row Decoders

Appendix C

Technology Assumptions

In this appendix we will describe all the assumptions that we have made for developing the analytical models of Chapter 5. The technology described here is also the one used in Chapter 3 to compare various decoder architectures and is based on an actual industry process.

Technology

The base technology is assumed to be a 0.25µm CMOS process and the relevant process details are shown in Table C.1.

Parameter	Value	Comments
C_g	1.29 fF/µm	Unit gate capacitance of an inverter (p-width = 2 x n-width)
C_j	1.72 fF/µm	Unit parasitic junction capacitance of an inverter
R_g	4.4 KΩ/µm	Unit output Resistance of an inverter
τ_{fo4}	90pS@2.5V	Delay of an inverter driving a load four times its size.
M1 width	0.45µm	Minimum width of metal 1
M1 aspect ratio	1.5	Aspect ratio (height/width) of metal 1 cross section.

Table C.1: Features of the base 0.25µm technology

Parameter	Value	Comments
M1 cap	0.21 fF/μm	Capacitance of metal1
M1 res	0.1 Ω/μm	Resistance of metal1, (aluminum)

Table C.1: Features of the base 0.25μm technology

The key features for four different generations are shown in Table C.2. Copper metallurgy

L (μm)	C_g (fF/μm)	τ_{fo4} (pS)	Supply (V)	M1 min. Width (μm)	M1 aspect ratio	M1 cap (fF/μm)	M1 res, type (Ω/μm)
0.25	1.29	90	2.5	0.45	1.5	0.21	0.1, al
0.18	1.29	65	1.8	0.324	1.77	0.23	0.12, cu
0.10	1.29	36	1.2	0.18	2.37	0.29	0.29, cu
0.07	1.29	25	0.9	0.126	2.83	0.33	0.49, cu

Table C.2: Technology scaling of some parameters

is assumed from 0.18μm generation onwards.

Higher level metals are designed as “fat” wires, namely, their heights are also scaled along with their widths to yield a larger cross section, but the heights are increased only by the square root of the factor of increase of the widths [39]. For example, a higher level metal layer with twice the minimum width of the metal 1 layer, has a height which is 1.4 times the metal 1 height, thus resulting in a resistance which is a factor of 3 smaller than the metal 1 resistance. We assume that the wiring pitch is twice the wire width for all the global wires.

Architecture

The SRAM is synchronous, i.e., a clock starts off the access, though the results can be easily extended to asynchronous SRAMs, by adding the power and delay to generate the ATD (address transition detection) signal.

An embedded SRAM structure is assumed, viz., all the data bits of the accessed word

come out of the memory core in close physical proximity to each other, unlike in stand-alone SRAMs, where the data I/O port locations are optimized for external pad connections. Since this optimization adds a constant offset to the delay and power of the SRAM core, the conclusions of this study are applicable even to stand-alone SRAMs.

The RAM cell size used for the analysis is 19.2λ x 32.8λ as in [21], and the cell area is typical of high density six transistor CMOS layouts.

Circuit Style

The RAM is designed for high speed operation, with low power pulsed techniques which reduce energy loss without affecting speed as discussed in [27]. The local wordlines are pulsed to control the bitline swings and small swings are used in the datalines to reduce power. Since these techniques do not affect the speed of the RAM, our analysis results pertaining to delay scaling is applicable to any speed optimized SRAM design.

A latch style sense amplifier, with perfect timing control is assumed for the sense amplifier as this consumes the least power and is the fastest. Hence our analysis results will be of relevance to both high speed and low power SRAMs. For the 0.25μm process, the optimal input swing which minimizes the senseamp delay is found from simulations to be 100mV, of which 50mV is the input offset.

The transistors in the bitline mux have a fixed size of 16λ and those in the dataline mux are sized to be 50λ wide, to simplify the analysis. Circuit simulations indicate that the RAM delay is only weakly sensitive to the sizes of these transistors.

Energy Modeling

The swings in the bitlines and I/O lines are limited for low power operation. While ideally they should be limited to be exactly that required for optimum detection by the sense amps, in practical designs, there is some slack in how tightly they can be controlled [27] and hence are assumed to be twice the optimum signal swing. Thus for the 0.25μm

process, these swing by about 200mV since the optimal swing for the senseamps is about 100mV.

Bibliography

[1] P. Barnes, “A 500MHz 64b RISC CPU with 1.5Mb On-Chip Cache”, *1999 IEEE International Solid State Circuits Conference, Digest of Technical Papers*, pp. 86-87.

[2] S. Hesley, et. al., “A 7th-Generation x86 Microprocessor”, *1999 IEEE International Solid State Circuits Conference, Digest of Technical Papers*, pp. 92-93.

[3] Special issue on low power electronics. *Proceedings of the IEEE*, vol. 83, no. 4, April 1995.

[4] S. Subbanna, et. al., “A High-Density 6.9 sq. µm embedded SRAM cell in a High-Performance 0.25µm-generation CMOS Logic Technology”, I*EDM Technical Digest*, pp. 275-278, 1996.

[5] G.G. Shahidi, et. al., “Partially-depleted SOI technology for digital logic”, *ISSCC Digest of Technical Papers*, Feb. 1999, pp. 426-427.

[6] A.P. Chandrakasan, et. al., “Low-Power CMOS Digital Design”, *IEEE Journal of Solid State Circuits*, vol. 27, no. 4, p. 473-484, April 1992.

[7] W. Lee, et. al., “A 1V DSP for Wireless Communications”, *1997 IEEE International Solid State Circuits Conference, Digest of Technical Papers*, pp. 92-93.

[8] M. Izumikawa, et. al., “A 0.25-μm CMOS 0.9-V 100-MHz DSP Core”, *IEEE Journal of Solid State Circuits*, vol. 32, no. 1, pp. 52-61, Jan. 1997.

[9] K. Ishibashi, et. al., “A 1V TFT-Load SRAM using a Two-Step Word Voltage Method”, *1992 IEEE International Solid State Circuits Conference, Digest of Technical Papers*, pp. 206-207.

[10] H. Yamauchi, et. al., “A 0.5V / 100MHz Over-Vcc Grounded Data Storage (OVGS) SRAM Cell Architecture with Boosted Bit-Line and Offset Source Over-Driving Schemes”, *1996 IEEE International Symposium on Low Power Electronics and Design, Digest of Technical Papers*, pp. 49-54.

[11] K. Itoh, A.R.Fridi, A. Bellaouar and M.I. Elmasry, “A deep sub-V, single power-supply SRAM cell with multi-Vt, boosted storage node and dynamic load”, *1996 Symposium on VLSI Circuits, Digest of Technical Papers*, pp. 132-133.

[12] R. C. Jaeger, “Comments on ‘An optimized output stage for MOS integrated circuits’,” *IEEE Journal of Solid State Circuits*, vol. SC-10, no. 3, pp. 185-186, June 1975.

[13] C. Mead and L. Conway, *Introduction to VLSI Systems*, Reading, MA, Addison-Wesley, 1980.

[14] N. C. Li, et. al., “CMOS tapered buffer”, *IEEE Journal of Solid State Circuits*, vol. 25, no. 4, pp. 1005-1008, August 1990.

[15] J. Choi, et. al., “Design of CMOS tapered buffer for minimum power-delay product”, *IEEE Journal of Solid State Circuits*, vol. 29, no. 9, pp. 1142-1145, September 1994.

[16] B. S. Cherkauer and E. G. Friedman, “A unified design methodology for CMOS tapered buffers”, *IEEE Journal of Solid State Circuits*, vol. 3, no. 1, pp. 99-110, March 1995.

[17] M. Yoshimoto, et. al., “A 64kb CMOS RAM with divided word line structure”, *I983 IEEE International Solid State Circuits Conference, Digest of Technical Papers*, pp. 58-59.

[18] O. Minato, et. al., “2Kx8 bit Hi-CMOS static RAMs”, *IEEE Journal of Solid State Circuits*, vol. SC-15, no. 4, pp. 656-660, August 1980.

[19] Y. Kohno, et. al., “A 14-ns 1-Mbit CMOS SRAM with variable bit organization”, *IEEE Journal of Solid State Circuits*, vol. 29, no. 9, pp. 1060-1065, October 1988.

[20] T. Chappell, et. al., “A 2-ns Cycle, 3.8-ns Access 512-Kb CMOS ECL SRAM with a Fully Pipelined Architecture”, *IEEE Journal of Solid State Circuits,* vol. 26, no. 11, pp. 1577-1585, Nov. 1991.

[21] H. Nambu, et. al., “A 1.8ns Access, 550MHz 4.5Mb CMOS SRAM”, *1998 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 360-361.

[22] G. Braceras, et. al., “A 350MHz 3.3V 4Mb SRAM fabricated in a 0.3µm CMOS process”, 1*997 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 404-405.

[23] K. Nakamura, et. al., “A 500MHz 4Mb CMOS pipeline-burst cache SRAM with point-to-point noise reduction coding I/O”, 1*997 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 406-407.

[24] K. Sasaki, et. al., “A 15-ns 1-Mbit CMOS SRAM”, *IEEE Journal of Solid State Circuits,* vol. 23, no. 5, pp. 1067-1071, October 1988.

[25] K. Sasaki, et. al., “A 7-ns 140-mW 1-Mb CMOS SRAM with current sense amplifier”, *IEEE Journal of Solid State Circuits,* vol. 27, no. 11, pp. 1511-1517, November 1992.

[26] M. Matsumiya, et. al., “A 15-ns 16-Mb CMOS SRAM with interdigitated bit-line architecture”, *IEEE Journal of Solid State Circuits,* vol. 27, no. 11, pp. 1497-1502, November 1992.

[27] B. Amrutur and M. Horowitz, “Techniques to Reduce Power in Fast Wide Memories”, *1994 IEEE Symposium on Low Power Electronics, Digest of Technical Papers,* p. 92-93.

[28] B. Amrutur and M. Horowitz, “A replica technique for word line and sense control in low-power SRAMs”, *IEEE Journal of Solid State Circuits,* vol. 33, no. 8, pp. 1208-1219, August1998. pp. 1208-1219.

[29] K.W. Mai, et al. “Low-power SRAM design using half-swing pulse-mode techniques”, *IEEE Journal of Solid State Circuits,* vol. 33, no. 11, pp. 1659-1671, November 1998.

[30] T. Mori, et. al., ““A 1 V 0.9 mW at 100 MHz 2 k*16 b SRAM utilizing a half-swing pulsed-decoder and write-bus architecture in 0.25 µm dual-Vt CMOS”, 1998 *IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 354-355.

[31] O. Minato, et. al., “A 20ns 64K CMOS SRAM”, *1984 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 222-223.

[32] S. Yamamoto, et. al., “256k CMOS SRAM With Variable Impedance Data-Line Loads”, *IEEE Journal of Solid State Circuits,* vol. SC-20, pp. 924-928, October 1985.

[33] K. J. Schultz, et. al., “Low-Supply-Noise Low-Power Embedded Modular SRAM for Mixed Analog-Digital ICs”, *Proceedings of 1992 IEEE Custom Integrated Circuits Conference,* p. 7.1/1-4.

[34] P. Reed, et. al., “A 250MHz 5W RISC Microprocessor with On-Chip L2 Cache Controller”, *1997 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 412-413.

[35] I. E. Sutherland and R. F. Sproull, “Logical effort: Designing for speed on the back of an envelope”, Advanced research in VLSI, pp. 1-16, 1991.

[36] Veendrick, H.J.M. et al. “Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits”, *IEEE Journal of Solid State Circuits,* vol. SC-19, no. 4, pp. 468-473, August 1984.

[37] S. R. Vemuru and A. R. Thorbjornsen, “Variable-taper CMOS buffer”, *IEEE Journal of Solid State Circuits,* vol. 26, no. 9, pp. 1265-1269, September 1991.

[38] S. Ohbayashi, et. al., “A study of fanout optimization of SRAM decoder with a line capacitance”, *The Transactions of the IEICE,* vol. E 73, no. 11, pp. 1855-1857, November 1990.

[39] H. B. Bakoglu and J. D. Meindl, “Optimal interconnects for VLSI”, *IEEE Transactions on Electronic Devices,* vol. ED-32, no. 5, pp. 903-909, May 1985.

[40] W. C. Elmore, “The transient response of damped linear networks with particular regard to wide-band amplifiers”, *Journal of Applied Physics,* vol. 19, no. 1, pp. 55-63, January 1948.

[41] *HSPICE,* Meta-Software, Inc., February 1996.

[42] H. C. Park, et. al., “A 833Mb/s 2.5V 4Mb double data rate SRAM”, 1998 *EEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 356-357.

[43] R. Heald and J. Holst, “A 6-ns cycle 256kb cache memory and memory management unit”, *IEEE Journal of Solid State Circuits,* vol. 28, pp. 1078-1083, November 1993.

[44] S. Murukami, et. al., “A 21-mW 4-Mb CMOS SRAM for Battery Operation”, *IEEE Journal of Solid State Circuits,* vol. 26, no. 11, pp. 1563-1569, Nov. 1991.

[45] T. Hirose, et. al., “A 20nS 4Mb CMOS SRAM with Hierarchical Word Decoding Architecture”, 1*990 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp.132-133.

[46] S. Flannagan, et. al., “8ns CMOS 64kx4 and 256kx1 SRAMs”, *1990 IEEE International Solid State Circuits Conference, Digest of Technical Papers,* pp. 134-135.

[47] J.S. Caravella, “A Low Voltage SRAM For Embedded Applications”, *IEEE Journal of Solid State Circuits,* vol. 32, no. 3, pp. 428-432, March 1997.

[48] A.R. Pelella, et. al., “A 2ns Access, 500MHz 288Kb SRAM MACRO”, *1996 Symposium of VLSI Circuits, Digest of Technical Papers,* pp.128-129.

[49] S. Tachibana, et. al., “A 2.6-ns Wave-Pipelined CMOS SRAM with Dual-Sensing-Latch Circuits”, *IEEE Journal of Solid State Circuits,* vol. 30, no. 4, pp. 487-490, April 1995.

[50] S. E. Schuster, et. al., “A 15-ns CMOS 64K RAM”, *IEEE Journal of Solid State Circuits,* vol. sc-21, no. 5, p. 704-711, October 1986.

[51] M. Shoji, “Elimination of Process-Dependent Clock Skew in CMOS VLSI”, *IEEE Journal of Solid State Circuits,* vol. sc-21, no. 5, p. 875-880, October 1986.

[52] J. L. Hennessy and D. A. Patterson, *Computer Architecture a Quantitative Approach*, Morgan Kaufmann, 1996.

[53] A. L. Silburt, et. al., “A 180-MHz 0.8-µm BiCMOS Modular Memory Family of DRAM and Multiport SRAM”, *IEEE Journal of Solid State Circuits,* vol. 28, no. 3, p. 222-231, March 1993.

[54] K. Osada, et. al., “A 2ns Access, 285MHz, Two-Port Cache Macro using Double Global Bit-Line Pairs”, *ISSCC Digest of Technical Papers*, pp. 402-403, February 1997.

[55] 1997 National Technology Roadmap for Semiconductor.

[56] T. Wada, S. Rajan and S. A. Przybylski, “An Analytical Access Time Model for On-Chip Cache Memories”, *IEEE Journal of Solid State Circuits*, vol. 27, no. 8, pp. 1147-1156, August 1992.

[57] S. J. E. Wilton and N. P. Jouppi, “An Enhanced Access and Cycle Time Model for On-Chip Caches”, *WRL Research Report 93/5*, June 1994.

[58] R. J. Evans and P. D. Franzon, “Energy Consumption Modeling and Optimization for SRAMs”, *IEEE Journal of Solid State Circuits*, vol. 30, no. 5, pp. 571-579, May 1995.

[59] R. J. Evans, “Energy consumption modeling and optimization for SRAMs”, *Ph.D. dissertation*, Dept. of Electrical and Computer Engineering, North Carolina State Univ., July 1993.

[60] K. Seno, et. al., “A 9-ns 16-Mb CMOS SRAM with offset-compensated current sense amplifier”, *IEEE Journal of Solid State Circuits*, vol.28, no. 11, Nov. 1993.

[61] J.D. Meindl et. al., “The impact of stochastic dopant and interconnect distributions on gigascale integration”, *1997 IEEE Int. Solid-State Circuits Conf., Dig. Tech. Papers*, pp. 232-233.

[62] T. Mizuno, et. al., “Experimental Study of Threshold Voltage Fluctuation Due to Statistical Variation of Channel Dopant Number in MOSFET’s”, IEEE Transactions of Electron Devices, Vol. 41, No. 11, p. 2216-2221, Nov. 1994.

[63] C. L. Portmann, et. al., “Metastability in CMOS library elements in reduced supply and technology scaled applications”, *IEEE Journal of Solid-State Circuits*, vol.30, no.1, p. 39-46.

[64] K. Ishibashi, et. al., “A 6-ns 4-Mb CMOS SRAM with offset-voltage-insensitive current sense amplifiers”, *IEEE Journal of Solid State Circuits*, vol. 30, no. 4, April 1995.

[65] T. Higuchi, et. al., “A 500MHz Synchronous Pipelined 1Mbit CMOS SRAM”, *Technical Report of IEICE*, May 1996, pp. 9-14, *(in Japanese)*.

[66] Fishburn, J.P. et al. “TILOS: a posynomial programming approach to transistor sizing” Held: Santa Clara, CA, USA 18-21 Nov. 1985. *IEEE International Conference on Computer-Aided Design:* ICCAD-85. Digest of Technical Papers

[67] Sakurai, T. et al. “Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas” *IEEE Journal of Solid State Circuits* April 1990. vol.25, no.2, pp. 584-94.