University of Alberta

Hybrid Multi-Layered Read-Only-Memories

by

Tyler Lee Brandon ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of *Master of Science*

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2003

Your file Votre référence
ISBN: 0-612-95896-5
Our file Notre référence
ISBN: 0-612-95896-5

Contents

List of Figures

List of Tables

List of Symbols

λ	Scalable length unit for IC process design rules
Λ	Scalable area unit for building devices in various manufacturing processes
layout	Mask-Layout, the physical representation of a circuit
IC	Integrated Circuit
LSB	Least Significant Bit
MSB	Most Significant Bit
RAM	Random-Access Memory
ROM	Read-Only Memory

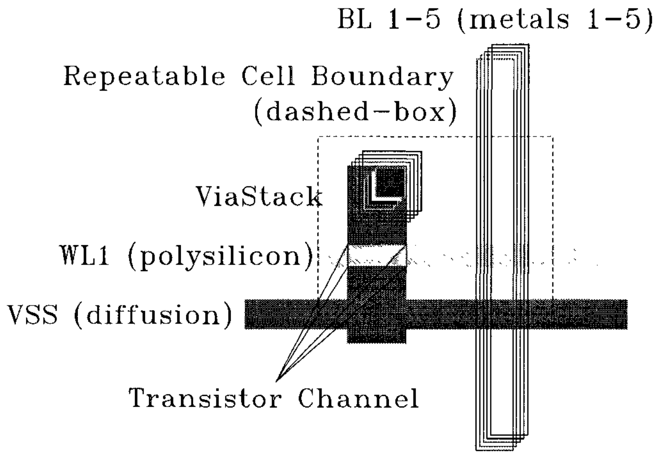

Figure 1: Layout Legend

Light Gray	Polysilicon
Dark Gray	Diffusion
Dashed Box	Repeatable cell boundary
Un-Filled Rectangle	Metal layer
Filled Square	Via

Chapter 1

Introduction

Read-only-memories (ROMs) are among the densest storage structures available in integrated circuits. This thesis will present conventional and new ROM cell architectures and analyze their densities across generations of manufacturing processes. We will provide both schematic and layout designs of each cell along with equations that describe the relationship between the cell density and the number of interconnect layers available within a process. It will be shown that using additional interconnect layers in specific configurations will result in a higher density ROM cell.

The behavior of a ROM is quite simple: it accepts an address and returns the fixed data associated with that address. Structurally, a typical ROM consists of a two-dimensional array of storage elements, wordline (row) decoders, data (column) decoders and sense-amplifiers (see Figure 1.1). Wordline decoder circuitry, built on one side of the array, is designed to input the row subfield of the input address and activate one row of storage cells at a time. The information stored within the cells traverses the bitlines to the sense-amplifiers, where the analog voltages from the bitlines are converted into digital signals. Data decoders relay the requested data to external circuitry based on the column subfield of the input address.

The data contents of a ROM are typically static, being determined at the time of manufacturing. Integrated circuit ROMs can exist in a number of physical states (i.e. connect

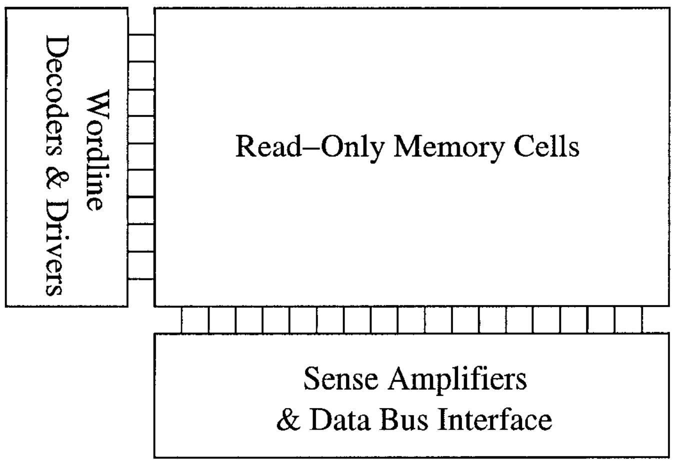

Figure 1.1: ROM Block Diagram

to the bitlines or not connected). The physical structure of the ROM cell depends on the information stored. Through the presence or absence of layers within the layout masks, manufacturers create the different physical structures that encode the information. There are exceptions, however, in which the contents of a ROM can be modified at a later point in time after manufacturing [5]. One current example of ROMs being used today, are the lookup tables used in certain complex calculations in micro-processors (e.g. fast division or transcendental functions).

Until now, reported ROMs have used only one metal layer for each of the bitlines or wordlines. By using multiple layers for additional bitlines, a higher-density ROM can be constructed. As will be shown, the cost of additional bitlines, in terms of area, is small. We will explore architectures that maximize the density of the storage cell. We will then estimate their impact on the area of the the peripheral circuitry to the memory array (which includes wordline decoder and drivers, as well as, sense-amplifiers and column decoders).

While many techniques may exist to compact a design in specific process technologies, we will not focus on these techniques but will instead strive for a deeper, process independent, understanding of ROM architecture. Specifically, there are key principles by which ROM storage density can be increased. These principles are outlined in Chapter 3.

Still, the following question naturally arises: “Why use a ROM when flash EEPROMs or other non-volatile memories are available?”. While non-volatile memory can be altered after manufacturing, this capability comes at the cost of decreased density and increased manufacturing complexity. If an integrated circuit design truly requires a static data set, a ROM will cost less and operate at a higher speed.

1.1 Technology-Independent Area Assessment

To facilitate the generalized analysis of various ROM architectures in relation to density, we will use a common metric across manufacturing processes. Every manufacturing process is different, which means that a common absolute measurement unit, based solely on physical design rules within the processes, will not exist. By abstracting each process to meet a common set of layout or design conditions, it is possible to construct a scalable relative measurement unit that allows us to compare ROM architectures across technologies. In other words, these results should remain valid as process technology continues to scale (as long as no radical shifts occur in the way IC devices are built). We will denote this measurement unit as Λ (upper case lambda).

Compared to λ (lambda), which is understood to be a common metric for *design rules* across various manufacturing processes [13], Λ is defined to be a metric for *devices* across manufacturing processes. Unlike λ, which is one-dimensional, Λ represents a unit of area rather than a unit of length.

The physical size of Λ is grown in each given manufacturing process until a number of common devices can be built without violating any manufacturing design rules. For example, a small transistor can fit into an area of 2Λ and contact/via structures in 1Λ. While using Λ does not necessarily produce the densest possible layout, it does provide a method by which layout architectures can be compared across multiple manufacturing process technologies. To design and build an architecture that would provide the greatest possible density would require the disclosure of the manufacturing rules upon which the design is based; manufacturers typically forbid such disclosures. Furthermore, such designs and claims would only be valid in the specific process for which they were built. It is our intention to investigate architectures that provide higher-density ROMs across many processes when compared with other conventional ROM architectures.

1.2 Our Contributions

The main contributions of this work are:

- Using multiple layers within a ROM and RAM-ROM architecture to form multiple bit-lines or wordlines
- Combining two or more ROM cell types to form hybrid cells
- Multi-Layered SRAM-ROMs that require no extra transistors to implement the ROM
- Analysis using Λ, a proposed process-independent unit of area
- Designed and implemented a test chip based on an early version of a hybrid multi-layered ROM cell

1.3 Dissertation Organization

In Chapter 2, we will start by describing the current state-of-the-art in ROM technology. In Chapter 3, we describe a generalized analysis of the principles behind ROMs. In Chapter 3, we provide original analysis of various ROM cell architectures. At the end of Chapter 3, we will compare the architectures, including novel hybrid ROMs, based on relative density and compare the compacted architectures in a 0.13-μm CMOS process. Chapter 4 will discuss RAM-ROMs and will introduce an architecture for increasing RAM-ROM density. The impact of the hybrid architectures on periphery area and circuits will be discussed in

Chapter 5. In Chapter 6, we will describe a test chip that was designed. Finally, some conclusions will be drawn in Chapter 7.

Chapter 2

Background

In this chapter we will present the conventional ROM cell architectures relevant to those new architectures that will be presented in Chapter 3. A description of the operation and design of the cell architecture will be provided for the following cell types: NOR, NAND, multi-valued, non-volatile, multi-bitline and multiple transistor ROM cells. The number of bits stored and the density of each cell will be analyzed in Chapter 3.

Storage density, or simply density, is defined as the amount of information divided by the area required to represent that information. For a ROM, the area includes decoders, sense-amplifiers and an external interface (periphery circuitry) as well as the storage elements. For the time being, we'll ignore the periphery area and calculate cell density as the number of bits stored in the cell divided by the area of just the cell. In this chapter we'll look at some common ROM architectures. In the following chapter we'll move on to examine some new ROM cell architectures. (Note that the multi-layer techniques presented in the following chapter can be used to enhance all of the memories discussed here.)

2.1 NOR ROM Cell

The NOR ROM cell uses one transistor to represent one bit. Wordlines (e.g. WL0 and WL1 in Figure 2.1) run along the row of cells. A one or zero in a cell is represented through the presence or absence of the cell transistor or connection to the bitline (BL in Figure 2.1).

Some ROMs are built such that the conditional connection to the bitlines is made in the highest layer of metal. This facilitates the "writing" of the ROM data contents at a later date, in a final stage of manufacturing. Bitlines are either actively pre-charged between reads or are connected to a weak pull-up resistive device. Figure 2.1 shows two NOR ROM cells connected to a bitline along with a pre-charge device. In this case, if wordline zero (*WL0*) were to turn on, the bitline would be pulled low via the connection to V_{SS} (ground potential). On the other hand, if *WL1* were turned on the bitline would remain in its pre-charged state. The number of cells attached to a bitline depends on the density and speed that the designer wishes to achieve. Typical bitlines have from 32 to 256 cells in a column.

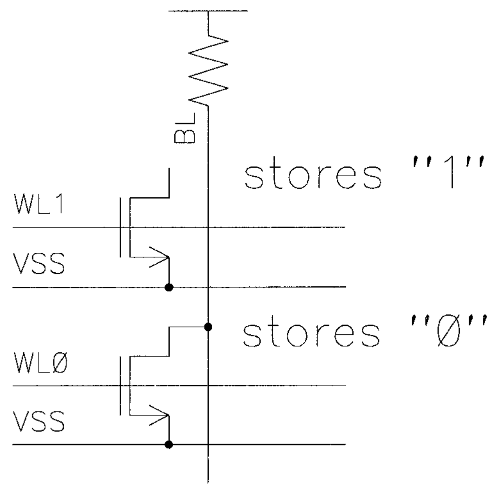

Figure 2.1: NOR ROM Schematic

A read operation proceeds as follows: The wordline decoder decodes part of the address to activate one wordline (wordline decoders are found in all memories). The wordline turns "on" the access transistors in a row, which exist at a cell location only if the corresponding cell data is a '0'. Alternatively, the absence of a connection between the transistor drain and the bitline can represent a '0'. Depending on the presence or absence of the transistor (or bitline connection) within the cell, the associated bitline will either stay at a high voltage or will be pulled low from a pre-charged level. The voltage swing or difference in current on the bitlines is sensed by a sense-amplifier using a reference signal to produce a "one" or "zero". Data, the "ones" and "zeros" from the sense-amplifiers, is multiplexed out to produce the output of the ROM. The wordline is then turned "off" and the bitlines are then pre-charged in preparation for the next read.

2.2 NAND ROM Cell

Unlike the NOR ROM, where cells within a column are connected to the bitline in parallel, the NAND ROM connects the cells, within a column, in series to form the bitline.

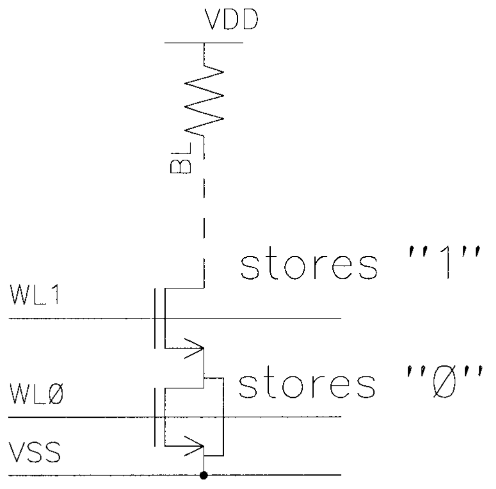

Figure 2.2: NAND ROM Schematic

During normal operation all the wordlines are "on", with the exception of the addressed wordline, which is turned "off". If a bypass connection (from the source to the drain of the transistor) exists, a "zero" is stored in the cell. Here the act of turning "off" the cell transistor has no effect on the state of the bitline and the other series transistors will pull the bitline low. On the other hand, if a bypass connection does not exist, the transistor will turn off, the bitline will charge high and a value of "one" will be read.

Figure 2.2 shows two NAND ROM cells in series. A bypass connection exists in the cell that is attached to *WL0*. If *WL1* were to turn "off" (remember that all wordlines in a NAND ROM are by default "on") the bitline would charge high. If *WL0* were to turn "off" the bitline would remain low. Due to the series connection between cells, a large amount of series resistance will be present in the bitlines, thus limiting the speed of operation for the NAND ROM cell architecture.

2.3 Multi-Valued ROM Cell

Multi-valued ROMs work in much the same way as a NOR ROM except that multi-valued cells store information by modifying the storage transistor (e.g. by changing the width or length of the transistor, as shown in Figure 2.3). This change in the transistor's dimensions directly changes the current the transistor is able to source. The resulting variations in sourcing current can be sensed relative to reference currents to identify the the transistor's size. Essentially, the current being sensed will fall into one of a predefined set of categories, also referred to as one-hot encoding. One-hot encoding is where only one state, out of many states, can be selected. The one-hot result can then be converted by peripheral circuitry to a binary value.

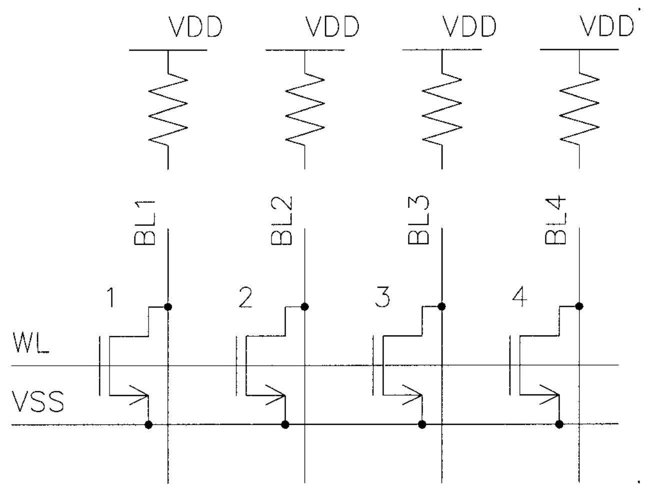

Figure 2.3: Multi-Valued Schematic

In Figure 2.3, the relative widths of the transistor are given by a number above the transistor symbol. Shown are four variations on the size of the transistor, which would effect the drive strength. The underlying cell configuration is that of a NOR network, where the cells in a column are connected in parallel to the bitline. While more information can be stored using the multi-valued cell, it is more difficult to reliably sense the variations in sourcing current.

2.4 Programmable Non-Volatile Memories

Programmable non-volatile memories employ many different techniques: floating gate, charge-trapping, ferro-electric and magnetic devices. In principle, floating gate and charge-trapping non-volatile memories work by altering the threshold of a transistor by injecting charge into the region between the transistor gate and the channel [5]. This charge can be stored in a conducting layer (using a so-called floating gate) or a non-conduction layer (using charge trapping). Depending on the technique used, densities can approach those of DRAM. Programmable non-volatile memories, however, typically lag behind by one or two lithography generations [5].

ROMs and programmable non-volatile memories address separate markets. Essentially, non-volatile memories allow the memory to be modified. For those applications, that do not require the memory to be modified, ROMs hold a clear advantage in that they require no modification to the logic manufacturing process and they typically have a smaller cell size. There are other types of programmable non-volatile memories that we will not cover, such as FeRAM and MRAM.

2.4.1 Multi-Valued Flash EEPROMs

By finely controlling the amount of charge injected on to the floating gate, more than two signal levels can be stored in a cell [4]. Both 4-level [4] and 16-level [8] EEPROMs have been reported. EEPROMs benefit from their re-programmability and that they can hold their charge for over ten years [8].

2.5 Multiple Bitline ROM Cell

The use of multiple bitlines in a ROM is a relatively new technique [6] when compared with multiple bit ROM storage devices.

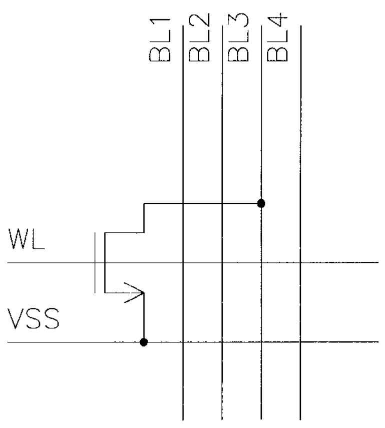

Figure 2.4: Multiple Adjacent Bitlines Schematic

Dvir in “Read-only-memory (ROM) having a Memory Cell that Stores a Plurality of Bits of Information” (US Patent 6,002,607) [6] describes a method by which the drain of a transistor in a ROM cell can connect to one of many adjacent bitlines (see Figure 2.4). The multiple bitlines allows a single transistor to store multiple bits. Also presented is a method by which x bits can be stored using x+1 bitlines and x/2 transistors (presented in the following section).

It is also claimed that for a 2-bit cell, that the bitline capacitance is reduced by 75% approximately. The bitline capacitance reduction is due to reduced number of transistors connected to each bitline. This claim is questionable, however, as such a large reduction only seems possible if one ignores the parasitic wiring capacitance of the bitlines.

During a read operation, the addressed wordline is pulled high connecting one and only one of the bitlines to the ground potential V_{SS}. The other bitlines remain high. Sense-amplifiers connected to the bitlines determine which, if any, bitline was pulled low. The output of the sense-amplifiers is one-hot encoded (only one of the bit pattern from the sense-amplifiers will be zero). The one-hot code can then be converted to a binary representation to be outputted. For example, the output of a multi-bitline cell in Figure 2.4 can be represented by a binary value according to Table 2.1. A zero in the table under a bitline (BL) means that bitline is connected to the transistor and the other bitlines are not. When the wordline (WL) is activated one (or none) of the bitlines is pulled low. This corresponding pattern of "ones" and "zeros" can then be represented using a binary value (see Chapter 5 for more information on determining the binary representation of one-hot encoding).

BL4	BL3	BL2	BL1	Binary
1	1	1	1	000
1	1	1	0	001
1	1	0	1	010
1	0	1	1	011
0	1	1	1	100

Table 2.1: Representing a One-Hot Encoded Value in Binary.

2.6 Multiple Transistor ROM Cell

If multiple bitlines are present, it is possible to use multiple transistors to connect to the bitlines, thus using two or more transistors together to represent multiple bits. The theoretical number of bits that can be represented is the "log base two" of the total number of combinations of connections between the transistors and the bitlines (including the absence of connections).

Figure 2.5 shows an example of using two transistors to connect to a combination of two of the bitlines. It is possible for these two transistors to connect to one, two or no bitlines. During a read operation, one of the wordlines in the memory array is asserted causing one cell in each column to pull-low one, two or no bitlines. For those cells we are interested in, a sense-amplifier is connected to each bitline. If the bitline has been pulled

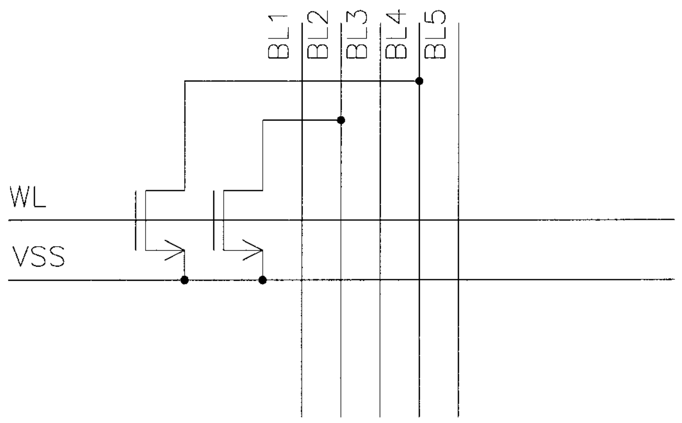

Figure 2.5: Multiple Transistor Schematic

low, the sense-amplifier produces a “zero” output; otherwise, a “one” output is produced. The resulting pattern of “ones” and “zeros” from the bitlines from the sense-amplifiers can then be converted to binary.

Chapter 3

High-Density ROM Cell Architecture

In this chapter we will start by identifying the underlying principles of a ROM, then move on to review conventional and to introduce new and new hybrid ROM cell architectures. We analyze the density of a number of novel ROM cell architectures. Each architecture will include a schematic, a layout and an equation relating the cell density to the number of inter-connect layers available in the manufacturing process. Cell density will be given in bits per normalized area, where the normalized area is based on the NOR ROM cell. At the end of the chapter we compare the cell densities based on Λ (our process-independent unit of area). In addition, we compare the cell density after compacting them in a 0.13-μm CMOS process.

As much of this chapter will be based on the multi-layer technique, we need to answer the question, "Why multiple layers?". Simply put, multiple metal layers will exist in a logic process whether we use them or not. We will show how they can be used to increase the number of bitlines and wordlines attached to each cell. The more bitlines that can be attached to a cell, the more information that the cell can be made to store. The real question is, "Does more bitlines mean more density?". We will show that the answer is a conditional "yes".

3.1 Principles of ROM

ROM storage mainly involves the following three principles: re-using physical structures, creating unique paths to or from the cell, and electrical differentiation. When programming the cell in Figure 3.1, values stored in the cell will determine that the gate of the transistor is connected to one or none of the wordlines and the drain of the transistor is connected to one or none of the bitlines.

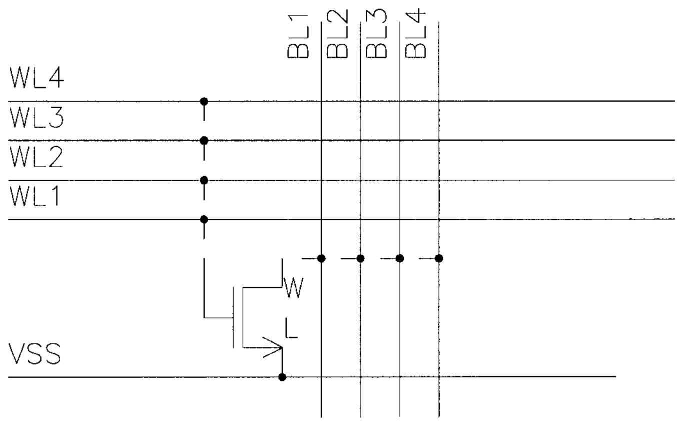

Figure 3.1: Principles of a ROM

Figure 3.1 shows how a ROM cell can have multiple wordlines, bitlines and variations in underlying transistor dimensions (channel width *W* and length *L*). Note that, only one wordline or bitline can be connected to the transistor (more than one causes multiple bitlines or wordlines to become shorted together). Also note that the transistor can potentially be left unconnected (to create an additional state). A ROM cell can use the multiple bitlines and wordlines as a way of creating more identifiable conditions (which directly equates to storing more information in the cell). The wordlines would all be connected to separate wordline drivers, which would be control by the wordline decoders. The bitlines would be attached to a number of cells within the column as well as pre-charge circuitry, and sense-amplifiers (access to sense-amplifiers is provided through column decoders).

Now let's discuss these aspects in relation to ROMs. The activation of one of the wordlines (rows) at different points in time allows the ROM to re-use the same physical structures (primarily the bitlines and sense-amplifiers) when accessing information. An example of creating unique paths *to* the storage cell is the use of multiple wordlines for a single row of storage elements. Multiple wordlines allows various sets of cells along the row to be activated at different times (through multiple wordlines). Every cell in a row, depending on what it stores can be connected to one of the multiple wordlines. We determine which wordline a cell is connected to by asserting the wordlines in turn, while monitoring (sensing) the bitline to determine when the bitline is pulled low. The bitline being pulled low tells us which wordline the cell is connected to. Using multiple bitlines is an example of creating unique paths *from* the cell. Adding unique paths both *to* and *from* the cell increases the amount of information stored by the cell. These extra paths, in essence, add more identifiable conditions that the cell can produce. By "identifiable condition" we mean a state which can be reliably sensed as a "one" or "zero" by circuitry commonly found in memory sense-amplifiers. Uniquely addressable activation paths *to* a common storage element(s) can also be referred to as time-multiplexed storage in that accessing information in the cell requires the assertion of each wordline at separate points in time (due to a single bitline for the output of the cell). The same principle of time-multiplexed storage is commonly found in all memory arrays and can be observed in a column of cells with a shared bitline. Each cell in the column is attached to a different wordline, as is the case in a typical ROM, and each cell must in turn be accessed at different points in time. Typically, all the memory cells in a row share the same wordline, but as we will show, this does not need to be so (the same applies for bitlines).

The amount of information that can be stored in a ROM depends on the number of unique conditions that the ROM cells can generate. The use of multiple bitlines per storage cell is one example of creating unique paths *from* the cell to the sense-amplifier (and thus creating more identifiable conditions). Creating more identifiable conditions by creating more paths differs from electrical storage, which creates identifiable conditions along a path (or bitline, as is commonly the case). Multiple metal layers within a manufacturing process allow the designer to connect a ROM cell to one of many bitlines running over the cell (instead of the conventional single bitline). These potential connections to the bitlines increases density by increasing the number identifiable conditions that a cell can generate. By contrast, electrical storage relies on the ability to identify conditions along a bitline. By varying the current drive of a storage cell by altering a transistor width or length, it is possible to store not only "one" and "zero" but also additional levels in between (these are levels are represented by one-hot encoding).

3.2 Method

In theory, it is possible to design storage elements that can store an arbitrary numbers of bits. When it comes to the actual design, however, what looked good on paper can often be very inefficient or impractical in the real world. Approaching the design problem from the physical point of view (manufacturing process technology parameters) helps refine the feasibility of a design by eliminating theoretical designs that only work on paper. If the predictions in the International Technology Roadmap for Semiconductors [3] hold true, then the following analysis in the next sections should remain valid over the next five to ten years.

The use of a process-independent measurement (Λ) further helps to reduce the process-specific design possibilities to a set of design possibilities that can be implemented across many process technologies. In effect, we are comparing generalized design techniques and ignoring specific process "tricks" and "hacks" used to increase density. Such "tricks" can always be used to enhance a more efficient design architecture once the architecture has been identified. A program was written to create Λ from the process technology design rules available in every process technology. Process technology design rules determine the minium size, width, spacing, enclosure and overlap of the process layers relative to themselves and the other layers. Λ is the area required to build a set of structures (that can contain multiple process layers), unlike λ which is a unit length based on a single process layers (such as the polysilicon width or metal spacing). λ does not contain information relating the various layers together to form useful structures. The structures required to build circuits, based on Λ, include: the substrate contacts, transistor source and drain regions, a via stack, and the pitch of the metal layers. Once Λ is defined it also acts as a grid to limit the placement of these structures. The cell architectures are entered into the mask layout editor using Λ to constrain each structures placement. As a result, the cell designs have an identical layout, in term of Λ, across process technologies; although the mask layout in each process technology is quite different. Because Λ was generated using the process technology design rules each cell will automatically be free from any design rule violations. To determine the physical area of a cell, it is a simple matter of multiplying the total Λ the cell covers by the physical size of Λ generated by the program.

It is my experience, in the design of ROMs, that the asymptotic analysis of schematic architectures, considered in isolation, does not yield accurate results. Only by examining physical structures within a manufacturing process can one obtain conclusive information about an architecture.

In the next sections we will explore various ROM cell architectures. Each ROM cell is constructed using Λ layout rules, allowing for equal design effort in the creation of each cell. Manhattan layout is used for all cells and all cells are compared on a relative basis to the NOR ROM cell. To clarify the layouts, the repeatable cell tile is surrounded by a dashed box.

Before moving on to the individual cell architectures, we will cover three layout techniques. These techniques can increase the storage density, and yet have simple (or nonexistent) schematic representations. We will describe each of these techniques before we present designs that make use of these structures.

3.3 Layout Techniques

It is the layout that drives the design of a high-density ROM architecture (and in turn the schematic representation). Schematics do not generally contain a mechanism by which area/density information can be represented. It is only through the layout that the area savings and its effect on density can be shown. Area savings are important because they directly relate to the manufacturing cost of a design. The proposed techniques deal with the multi-layer interconnect aspect of manufacturing processes and how these layers can be used to increase storage density. The reason we are presenting these techniques here is that they are used to construct most of the "conventional", "new" and "new hybrid" ROM layouts.

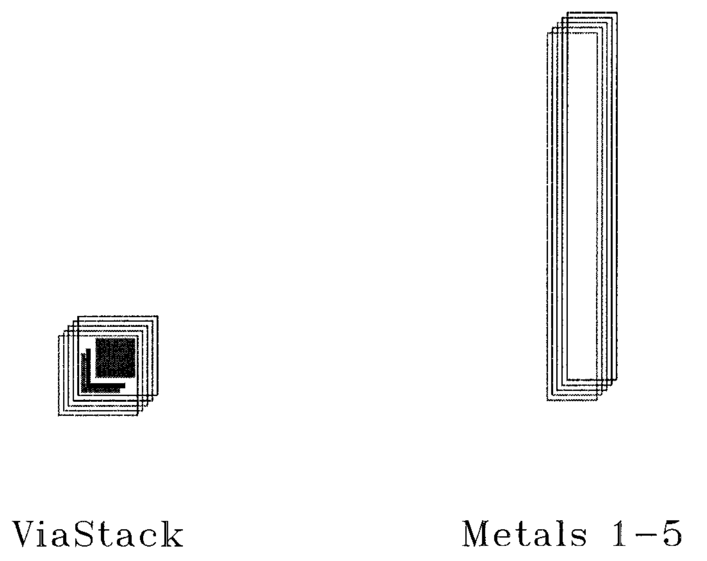

Figure 3.2: Multi-Layer Layout Structures: (left) Via-stack, (right) Multiple Stacked Metal Layer Paths

It is important to gain an understanding of the constructs being used to represent the ROM architectures in the following sections. The via-stack, to the left in Figure 3.2, shows the vertical layering of multiple vias and occupies an area of one Λ. Each via is surrounded by the metal layers it connects. This via-stack is connecting metal1 to metal5 (where metal5 is physically the highest layer in the stack and metal1 the lowest). To the right, are shown five metal layers, not connected, but overlapping each other.

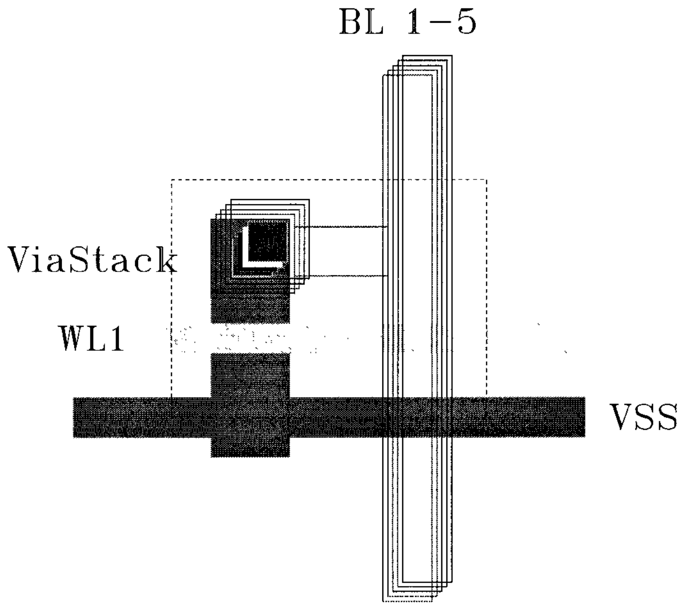

Figure 3.3: Multi-Bitline ROM Cell Layout

Figures 3.3, 3.4 and 3.5 show examples of how these structures would be typically used. The via-stacks allow a connection from the drain of the transistors to any of the five metal bitlines running vertically over a cell. Note that for a minimum-sized transistor the drain and source occupy an area of one Λ each. At this point we would also like to point out a few of the other constructs used to represent this and later layouts. In Figures 3.3 and 3.4, the dashed box surrounding part of the layout is the boundary of the repeatable ROM cell layout. Wordlines (typically implemented in a poly-silicon layer; light grey in the figure) are represented by "WL0", "WL1", etc. The ground potential is referred to as "V_{SS}" and the dark layer, to which "V_{SS}" is next to, is diffusion. Where a wordline crosses diffusion, a transistor is formed. Figure 3.3 shows an example connection to the second metal layer in the bitline stack from the via-stack. To avoid confusion, no connections are

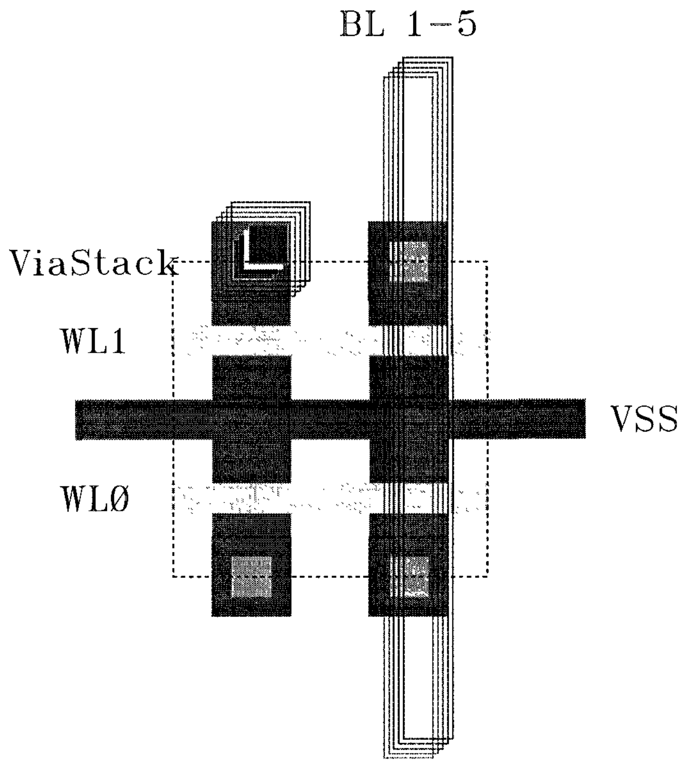

Figure 3.4: Hybrid 3 NOR to 1 Multi-Layer ROM Cell Layout

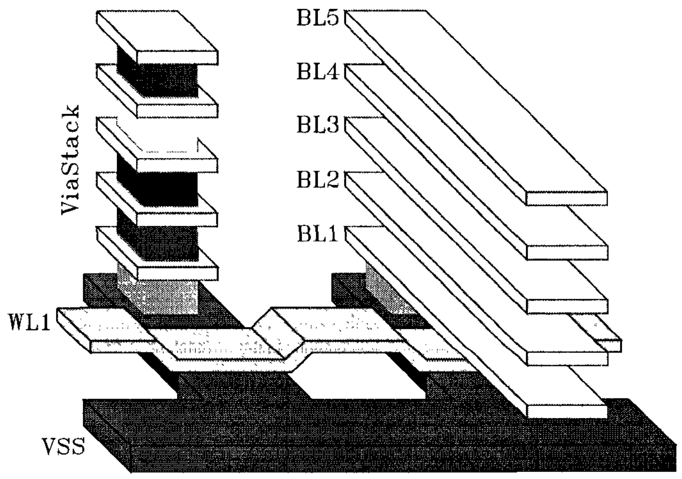

Figure 3.5: Hybrid 3 NOR to 1 Multi-Layer ROM Cell Cross-Sectional Layout

shown between the via-stacks and the multiple bitlines in the following layout figures. The cell's un-programmed state allows us to show the simplest representation of the layout. When programmed, the connections between the via-stacks and the bitlines would depend on the information being stored in the ROM.

3.3.1 Multi-Layer Technique

The multi-layer technique uses existing process metal layers to increase cell density. In the past few years the number of metal layers that have become available to the IC designer has increased from two in 1985 [9] to eight in 2002 [3]. It is interesting to note that the number of metal layers is predicted to reach eleven by 2016 [3].

Most designers and IC manufacturers utilize the multiple metal layers as inter-connect. We, on the other hand, see the potential to encode information within these layers. For example, multiple bitlines can be built in multiple layers, stacked one above another. Increasing the number of bitlines per cell increases the number of states that the cell can represent. Although the connections between metal layers are not easily modified after fabrication, the multi-bitline technique remains quite suitable to ROMs (in all their forms).

3.3.2 Short Via-Stack Layout Technique

The short via-stack technique extends the multi-layer technique. When using multiple columns of via-stacks within the same cell to connect to multiple bitlines, restricting the height of one or more columns of via-stacks allows additional bitlines to run over top of these "short" stacks. Thus more information can be stored within the same cell area.

Consider the example in Figure 3.6, which shows a set of multiple bitlines with via-stacks on either side. If we shorten the stack on the left hand side by one layer, it is possible to run another bitline over the stack. The via-stack on the right-hand side then has the option to connect to one of the multiple bitlines, as described earlier, or to the bitline running over the short stack in the cell to its right.

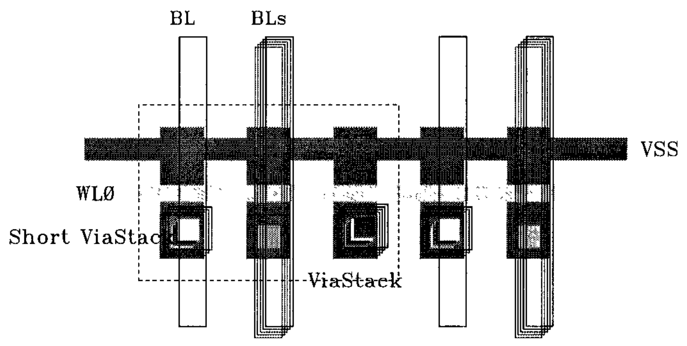

Figure 3.6: Short Via Stack Layout Technique

3.3.3 Cross-Over Layout Technique

The cross-over layout technique allows the designer to use multiple transistors together, at the cost of one bitline in a column of bitlines.

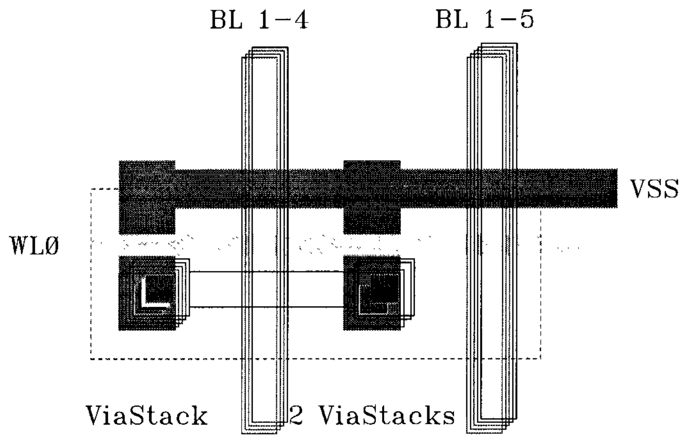

Figure 3.7: Hybrid Cross-Over Layout Technique

In Figure 3.7, the top bitline in the first column of bitlines (left bitline column) is removed. Removing the top bitline allows a cross-over metal layer (metal5 in this case) to be used to access the second column of bitlines (on the right). In this way, both multi-layered cells have access to both columns of bitlines. Thus any two bitlines from either column can be connected to both of the underlying transistors.

3.4 Conventional ROM Cells

In this section we will present schematic ROM cells designed by others, along with our best layout architectures for these cells (using the layout techniques mentioned above).

3.4.1 NOR ROM Cell

Let's start with the NOR ROM (Figure 3.8), which by definition has a cell density of 1-bit per Λ, as shown in the layout in Figure 3.9. This structure consists of a transistor that shares both the source and the drain connections with the cells above and below. In this case, information is stored by the presence or absence of the fully-formed transistor (the diffusion crossing polysilicon). Bitlines are routed in metal1 vertically over the array. Wordlines, which run horizontally, are routed in polysilicon and are strapped in metal2. Note that strapping is a technique for reducing the effective resistance of a polysilicon line by including periodic connections to a parallel metal line. Grounding for the pull-down NMOS transistors is provided on metal1 at predetermined multiples of the ROM cells. Note that the actual density of the ROM array is less than 1-bit per Λ when the wordline contacts along with the ground and substrate contacts are taken into account. In Figure 3.9, the surrounding dashed box represents the boundary for the repeatable cell layout.

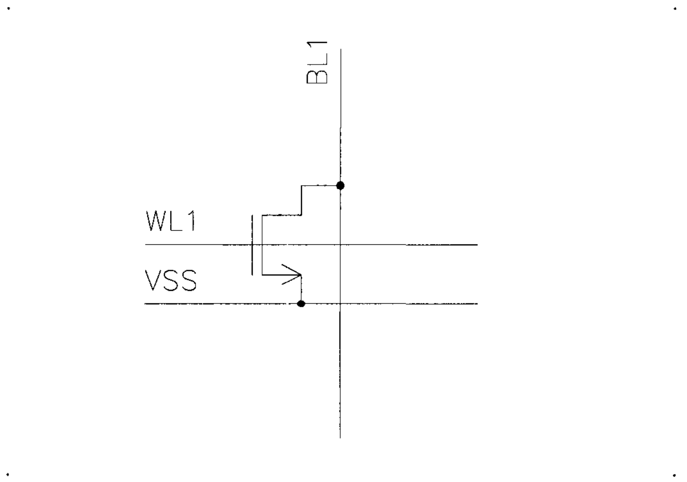

Figure 3.8: Conventional NOR ROM Cell Schematic

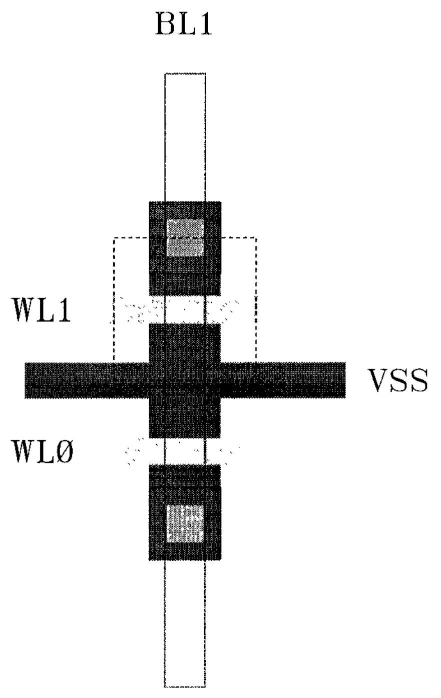

Figure 3.9: Conventional NOR ROM Cell Layout, 1-bit per Λ

3.4.2 NAND ROM Cell

Similar to the NOR ROM, the NAND ROM (Figure 3.10) has a density of 1-bit per Λ. The NAND ROM, however, aligns its cells in *series* to form the bitlines, instead of having the cells attach in *parallel* to a bitline. The value in the cell is determined by the presence or absence of a metal connection between the source and drain, which has the effect of bypassing the transistor. In Figure 3.11 we can see two metal1 "jumpers". A read occurs, when one of the wordlines goes low, shutting off a transistor in the column. If no bypass exists, the bitline will remain high; otherwise, the bitline is pulled low through the series of transistors. For a large number of transistors in series, the read operation can be very slow relative to a similarly-sized NOR ROM (due to the resistance in the transistor channel). In Figure 3.11, the dashed box represents the repeatable boundary for the cell.

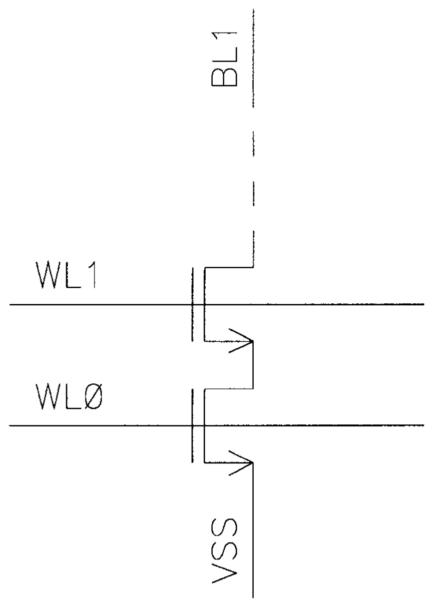

Figure 3.10: Conventional NAND ROM Cell Schematic

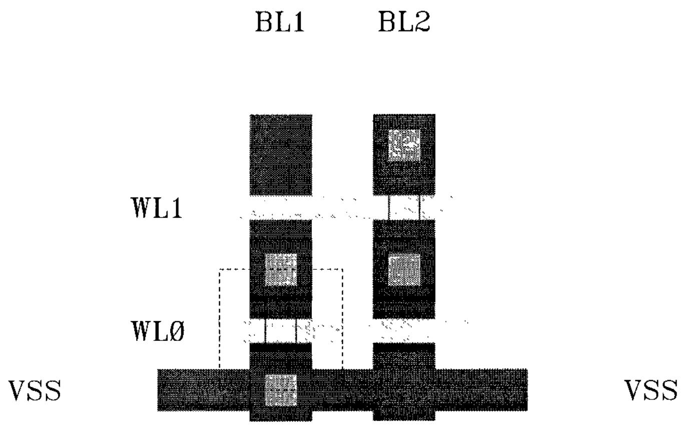

Figure 3.11: Conventional NAND ROM Cell Layout, 1-bit per Λ

3.4.3 Multi-Valued ROM Cell

The multi-valued cell read operation relies on sensing the effects of different transistor drive strengths. Variations in drive strength can be accomplished by altering the width or length of a transistor.

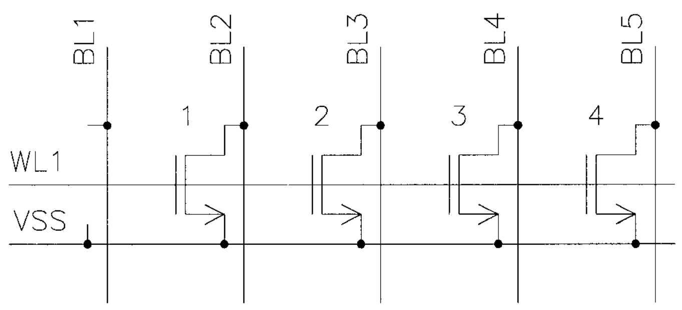

Figure 3.12: Multi-Valued ROM Cell Schematic

In Figure 3.12 the relative widths of the transistors are indicated by the number above the transistors.

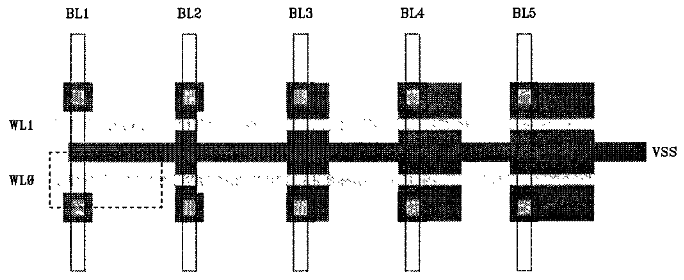

Figure 3.13: Multi-Valued ROM Cell Layout, Two Rows of Cells (with Five Distinct Drives), 1.16-bits per Λ

In Figure 3.13, the different widths can be seen. In this case the transistor can be manufactured in one of five possible states. Specifically, four states correspond to four different transistor sizes and the fifth state is the absence of a transistor (open circuit). In this case the transistors are sized from the weakest drive strength to the strongest, going from left to right. Note that the size of the cell is the same for all the transistor sizes. By fixing the cell size, the cells can be easily abutted to form arrays.

The cell density (for Figure 3.13) is given by:

$$BitsPer\Lambda = log_2(5)/2\Lambda \tag{3.1}$$

$$BitsPer\Lambda = 1.16 \quad \text{for 5 drive states}$$

There are five possible signal states (or symbols) in each cell, which are observable on the associated bitline. Each state is capable of sourcing a different amount of current. By comparing the source current to four reference currents, it is possible to determine which one of five states the cell represents. Log base two of the possible states gives the number of bits stored. The cell area is two Λ. The density of the cell can be determined by dividing the number of bits by the area measured in Λ.

3.4.4 Multiple Bitline ROM Cell

The only reference we found that describes a multiple bitline ROM cell structure is a US patent [6] and it only describes "adjacent" bitlines in terms of a schematic description. In this reference [6], no references were found describing the use of multiple metal layers in the design of such a multi-bitline cell.

As manufacturing processes make more layers of metal interconnect available, it is beneficial to incorporate these layers into the design of the ROM. We will introduce the multi-layer multi-bitline structure using Figures 3.14 and 3.15. The multi-layer multi-bitline structure relies on via-stacks and multiple metal layer bitlines. The via-stack allows a cell to connect to one of many bitlines (or none). The number of bitlines is determined by the number of metal layers available in the manufacturing process. By sensing which bitline, if any, has been driven by the storage cell it is possible to decode the signals on multiple bitlines into multiple bits.

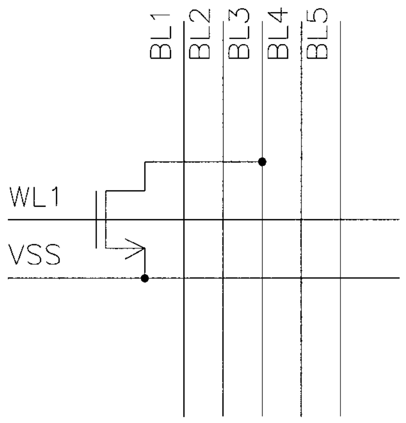

Figure 3.14: Multi-Bitline ROM Cell Schematic

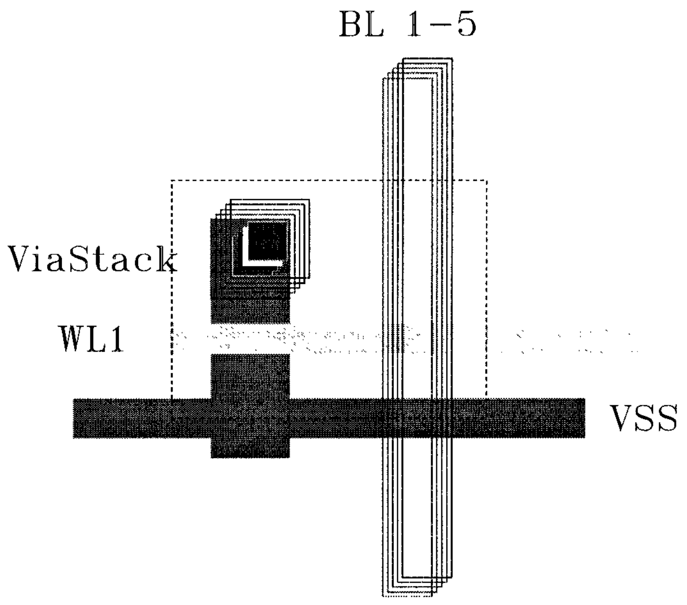

Figure 3.15: Multi-Bitline ROM Cell Layout, 0.86-bits per Λ

The following equation gives the cell density in terms of bits per Λ:

$$BitsPer\Lambda = log_2(n+1)/3\Lambda \tag{3.2}$$

for $n = 5$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.86$$

where *n* is the number of metal layer bitlines available to attach to in the column of bitlines. Note that,

$$n = m - 1 \tag{3.3}$$

for $m = 6$ metal layers:

$$n = 5$$

where *m* is the number of metal layers available in the process. One of the metal layers is used to contact the polysilicon wordline at "N" ROM cell intervals (also referred to as "strapping the wordline"). Grounding (V_{SS}) for the pull-down NMOS transistors is provided on metal1 strips (parallel to the bitlines) at intervals of "N" ROM cells. We assume that "N" is relatively large and neglect it in our cell area calculations.

The multi-bitline cell in Figure 3.15 has an area of three Λ (the repeatable bounding box covers an area of three Λ). The log base two of the total number of bitlines that the transistor can possibly connect to, plus one (for no-connection), results in the number of bits the cell can store. To determine density we take the bits and divide by the area (in terms of Λ) of the cell.

For the purpose of comparison, the multi-bitline architecture (Figure 3.15) requires seven layers of metal for the bitlines in order to achieve a density of 1-bit per Λ (NOR ROM cell). Later, in the section on Hybrid ROMs, we will describe architectures which achieve densities greater than 1-bit per Λ using the multi-layer technique.

3.4.5 Multiple Transistor ROM Cell

It is possible to use multiple transistors in combination to store information (see Figure 3.16). In Figure 3.17 both transistors have access to the multiple bitline columns using the cross over technique.

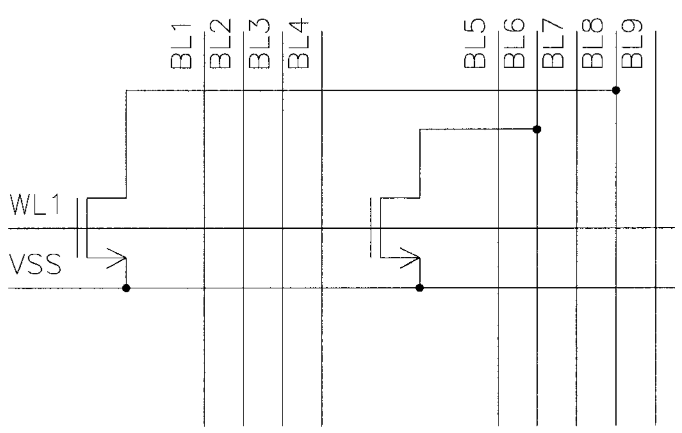

Figure 3.16: Multiple Transistor ROM Cell Schematic

In the schematic (Figure 3.16) the transistor on the left connects to the set of bitlines on the right. Alternatively, the transistor on the right could connect to the bitlines on the left. Basically, using both transistors, any combination of the bitlines (from both sets) can be chosen.

Figure 3.17 shows the layout for the multiple-transistor ROM cell. This layout uses the cross-over technique described in the Layout Techniques section.

If n is the number of bitlines available, then the following equation gives the cell density for the two-transistor cell.

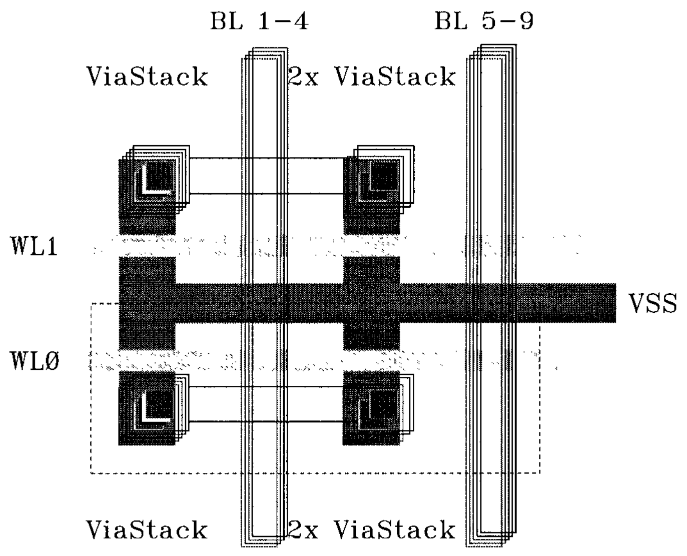

Figure 3.17: Multiple Transistor ROM Cell Layout, 0.92-bits per Λ

$$BitsPer\Lambda = log_2(C(2n-1,2) + C(2n-1,1) + C(2n-1,0)/6\Lambda \tag{3.4}$$

for $n = 5$ in a 6 metal layer process:

$BitsPer\Lambda = log_2(36 + 9 + 1)/6\Lambda$

$BitsPer\Lambda = 0.92$

and

$$n = m - 1 \tag{3.5}$$

for $m = 6$ metal layers:

$n = 5$

where m is the number of metal layers in the process. The function *C(n,r)* represents the number of combinations of n choose r. The multiple-transistor cell in Figure 3.17 has an area of six Λ. There are $2n$-1 bitlines. We can choose any combination of two, one or none of the bitlines. A sense-amplifier attached to a bitline, typically, outputs a "one" or "zero" based on two inputs; in this case the two inputs are a bitline and a reference signal. Essentially, the sense-amplifier answers the question, "has the bitline been pulled lower than the reference signal?". If the bitline is lower than the reference the sense-amplifier produces a "zero" output; otherwise, it produces a "one" output. In the case of a multiple transistor cell, as is described above, the resulting bit-pattern from the sense-amplifiers connected to the bitlines can contain two, one, or no "zeros". This resulting bit-pattern can then be converted to bits, a binary representation (see Chapter 5). Thus to determine density we take the number of bits producable by the cell and divide by the area (in terms of Λ).

Simple Decode

An alternative technique uses a simple decoding scheme [6] where x transistors can represent $2x$ bits. This scheme requires $2x$+1 bitlines. The transistors can select none or one bitline. The first $2x$ bitlines represent the binary output. The extra bitline inverts the binary output, this allows all possible binary states to be represented. Using the simple decoding scheme, the following equation gives the cell density in terms of bits per Λ (assume x is two):

$$BitsPer\Lambda = (2x)/(1.5x + 3)\Lambda \tag{3.6}$$

for $x = 2$ and $n = 5$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.67$$

provided that,

$$n = 2x + 1 \tag{3.7}$$

for $x = 2$ transistors:

$$n = 5$$

where x is the number of transistors in the cell and n is the number of bitlines available to attach to any transistor.

Note that due to the decoding scheme, the amount of information stored in the cell directly depends on the number of transistors (as each transistor represents two bits). As a result, when the limit as x goes to infinity is taken of the density equation (above), this design asymptotically approaches a maximum of 1.33 bits per Λ, provided enough metal layers are available to implement it using the multi-layer technique. If enough metal layers are not available then the cell would have to expand its footprint, which, in turn, would change the density equation.

3.5 New ROM Cells

In this section we will present new ROM architectures we designed. Table 3.1 summarizes the following sections, outlining the techniques and cells used to create the hybrid cells. For comparison purposes, the storage density in bits per Λ is also listed.

Note that the *bits per* Λ depends on the number of metal layers in the process. As such, the relative density of the cells can be expected to change for various manufacturing processes.

3.5.1 Multiple Wordline ROM Cell

Multiple wordlines facilitate information storage by accessing each wordline at different points in time (through changes in the address supplied to the wordline decoders) in the

Name	bits perΛ	Multi Layer	Short Stack	Cross Over	NOR	NAND	Multi BL	Multi Value	Multi Trans	Multi WL	Vert WL
NOR	1.00				x						
NAND	1.00					x					
Multi-BL (ML)	0.86	x					x				
Multi-Trans. (MT)	0.92	x			x		x		x		
Multi-Value (MV)	1.16				x			x			
Multi-WL (MW)	0.65	x			x					x	
Vert WL (x=4)	0.55	x					x				x
3-NOR to 1-ML	1.33	x			x		x				
4-NOR to 2-MT	1.26	x	x		x		x		x		
1-NOR to 2-MT	1.11	x	x		x		x		x		
1-NOR to 3-MT	1.07	x			x		x		x		
2-NOR to 2-MT	1.14	x		x	x		x		x		
2-MV to 1-ML	1.68	x			x		x	x			
1-MV to 1-ML	1.46	x			x		x	x			
1-NAND to 2-MW	0.75	x				x	x			x	

Table 3.1: Techniques Used to Construct the ROM Cell (Bits per Λ is Calculated Using a Six Metal Layer 0.13-μm CMOS Process).

row of wordlines to determine which one controls the gate of the transistor. Information is stored within a cell by selectively connecting the cell gate to one of the multiple wordlines or by removing the transistor.

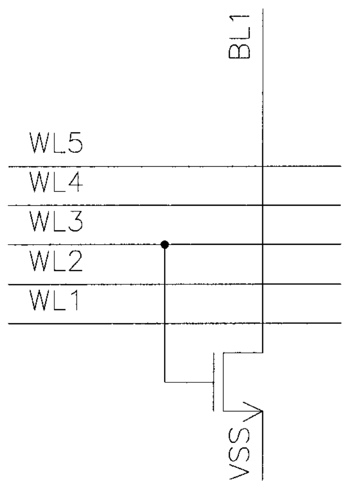

Figure 3.18: Multiple Wordline ROM Cell Schematic

As can be seen in Figure 3.18, the gate of the transistor connects to one of the wordlines. Connecting the gate to multiple wordlines would result in a short between those wordlines for all the cells in a row (which is something we try to avoid).

Figure 3.19 shows the layout for the multiple wordline cell. Each ROM cell can con-

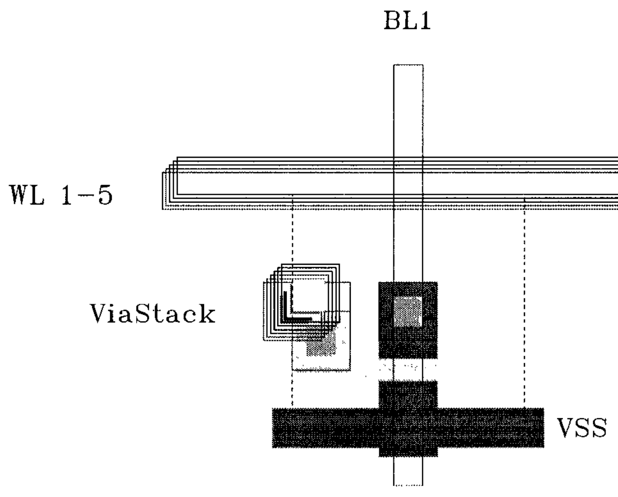

Figure 3.19: Multiple Wordline ROM Cell, Four Transistors and Four Bitlines, 0.65-bits per Λ

nect to one of five wordlines running horizontally over the row of cells. Proper operation requires that each of the wordlines, in turn, go high, making the memory access time proportional to the number of wordlines available to each cell. The assertion of each wordline eventually reveals which wordline the cell is attached to (or reveals the absence of a transistor).

The cell density is given by:

$$BitsPer\Lambda = log_2(n+1)/4\Lambda \tag{3.8}$$

for $n = 5$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.65$$

where n is the number of wordlines in the row of wordlines. The area of the cell is four Λ. Log base two of the possible connections returns the number of bits that can be stored in the cell. Density is given by dividing the bits by the area (in terms of Λ).

3.5.2 Vertical Wordline Gate ROM Cell

In a typical ROM, the wordlines run horizontally over the row of memory cells. This cell uses short vertical wordline gates that branch out vertically from the horizontal metal wordline. The primary advantage of is that the short vertical wordlines can control two or more base cells in the vertical direction.

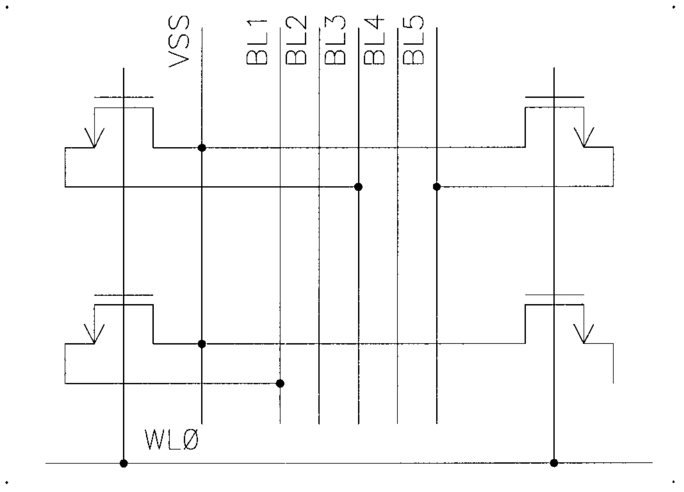

Figure 3.20: Vertical Wordline Gate ROM Cell Schematic

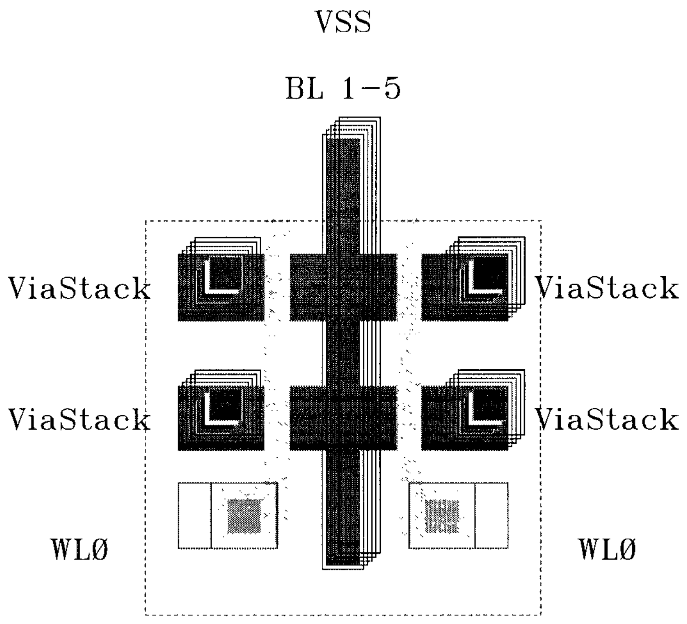

Figure 3.21: Vertical Wordline Gate ROM Cell Layout, 0.55-bits per Λ

The equation for maximum cell density, in terms of bits per Λ is given by (let x equal to four) :

$$BitsPer\Lambda = log_2(\sum_{i=0}^{x} C(n,i))/(1.5x+3)\Lambda \tag{3.9}$$

$$BitsPer\Lambda = log_2(C(n,4)+C(n,3)+C(n,2)+C(n,1)+C(n,0))/9\Lambda$$

for $n = 5$ in a 6 metal layer process:

$$BitsPer\Lambda = log_2(5+10+10+5+1)/9\Lambda$$

$$BitsPer\Lambda = 0.55$$

where n is the number of metal layer bitlines available to attach to the column of bitlines and x is the number of transistors in the cell. Note that,

$$n = m - 1 \tag{3.10}$$

for $m = 6$ metal layers:

$$n = 5$$

where m is the number of metal layers available in the process. One of the metal layers is used as the horizontal metal wordline. The number of states that can be represented by the cell is a combination of bitlines and transistors (i.e. n-bitlines choose x-transistors). Log base two of the number of states returns the number of bits stored in the cell. The cell area grows with the number of transistors, x. Note that x is assumed to be an even number. As the transistors share the source connections, an odd number of transistors would leave one transistor sitting by itself (and would result in a hole in the cell layout).

3.6 New Hybrid ROM Cells

By combining the two or more cell types it is possible to achieve cell densities beyond those of the individual cell types alone. To further increase density, the resulting hybrid cells also make use of the multi-layer, short via-stack and cross-over techniques where possible. In a manufacturing process where *m* metal layers are available, *n* metal layers are used for the multi-layered column bitlines. The number of bitlines in a column is represented by:

$$n = m - 2 \tag{3.11}$$

We assume that one metal will be used for the NOR or multi-value ROM cell bitline and one will be used for wordline strapping. A value of six was used for *m* in all of the following hybrid ROM cell figures. Thus,

$$\text{for } m = 6 \text{ metal layers:}$$
$$n = 4$$

Setting *n* equal to four will allow us to present the density of the architecture in a six metal layer (six interconnect layer) process.

3.6.1 Hybrid 3 NOR to 1 Multi-Layer ROM Cell

This hybrid cell combines three NOR cells and one multi-layered ROM cell. In Figure 3.23, we see the layout for a 1.33-bit per Λ ROM cell (built in a six metal layer process). This cell uses a multi-layer, multi-bitline structure. The hybrid multi-layer design technique mixes three 1-bit NOR cells with one multi-bitline cell, over a four Λ area. The multi-bitline cell can connect to one of four bitlines (in this case) or have its transistor removed; thus providing five unique possibilities (or 2.3-bits). The NOR cells, under the same column of multiple bitlines, attach to the first bitline (metal1 layer) and represent a "zero" or "one" by the presence or absence of the transistor. The NOR cell that shares the via-stack of the multi-bitline cell represents a "zero" by the presence of the connection to one of the multiple bitlines (which one does not matter), and represents a "one" by the absence of the transistor.

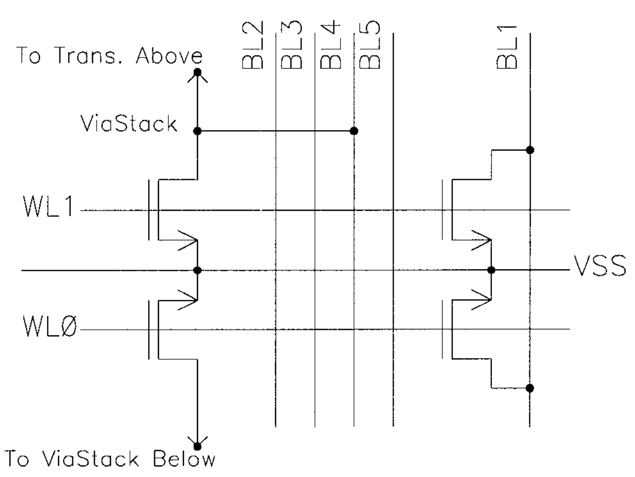

Figure 3.22: Hybrid 3 NOR to 1 Multi-Layer ROM Cell Schematic

The two transistors, to the right in Figure 3.23 under the stack of bitlines, connect directly to a metal1 bitline. They each store 1-bit of information, depending on the presence or absence of the transistor (in Figure 3.23 both transistors are present). The multi-bitline cells occur on odd wordlines and are attached to a via-stack. The connection to one of n-bitlines or the absence of the transistor selects one of n+1 unique conditions. The 1-bit cells on even wordlines that share the drain contact with the multi-bitline cell represent their data through the presence or absence of the transistor. They do not affect which of the bitlines the multi-bitline cell will connect to. Instead, during the sensing of a row on an even wordline, the n-bitlines are "AND'ed" together. In other words, any one of the bitlines can be pulled low to indicate a value of "zero" stored in the 1-bit NOR cell.

During a read operation, a wordline is asserted based on the input address. In this case, let us assume it is *WL0*. The first bitline will be pulled low if the lower-right NOR cell

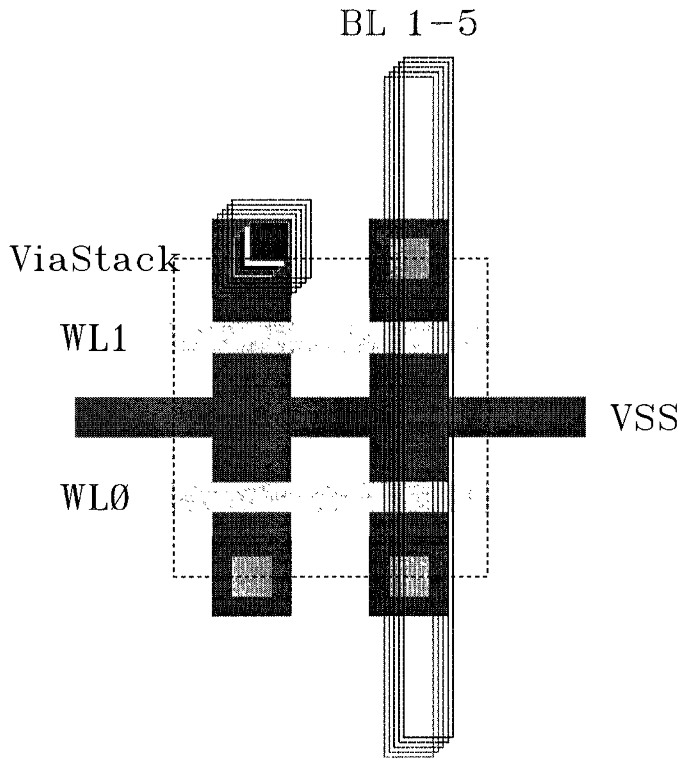

Figure 3.23: Hybrid 3 NOR to 1 Multi-Layer ROM Cell Layout, 1.33-bits per Λ

is connected to the first bitline (and the output will be read as a "zero"), otherwise it will remain high (and the output will be read as a "one"). If the lower-left transistor exists, one of the remaining four bitlines will be pulled low, otherwise they will all remain high. The sense-amplifiers attached to the bitlines check the state of the bitlines and output the associated "one" "zero" bit-pattern. As we know the wordline address is even, a "zero" is decoded if any of the four bitlines is "zero", otherwise a "one" output. If the wordline address is odd, the upper-right NOR cell behaves identical to the lower-right NOR cell, however the upper-left cell must be treated differently than the lower-left cell. When the address is odd, information is decoded from the position of a "zero" on one of the four bitlines (from more information on decoding see Chapter 5).

The following equation gives the cell density in terms of bits per Λ:

$$BitsPer\Lambda = (log_2(n+1)+3)/4\Lambda \tag{3.12}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = 1.33$$

where n is the number of metal layer bitlines available to attach to in the column of bitlines. The "three bits" comes from the three base NOR cells. The "log base two" bits are generated by the possible connections to the multiple bitlines from the fourth multi-bitline cell. The total area for the cell is four Λ. Thus density is given by the sum of the "three bits" plus the "log base two bits", divided by the area.

For the multi-bitline cells, the drain capacitance on the bitlines is reduced, on average, as the average bitline is only connected to one third (in this case) the number of cells it would have otherwise been connected to. Stacking the bitlines over a column of cells, however, increases the capacitance of each bitline.

3.6.2 Hybrid 1 NOR to 2 Multi-Transistor ROM Cell

Figures 3.24 and 3.25 show the schematic and layout, respectively, for a hybrid cell that uses a combination of one NOR and two multi-layered cells. The pair of multi-layered cells (identified by the via-stack on the drain of the transistors) work in conjunction to store multiple bits. The multi-layered cells on either side of the multiple bitlines can connect to two of n-bitlines. The value stored in these cells is determined by pulling the bitlines low and is generated as follows: no-bitline is pulled low (absence of both transistors or bitline connections), one bitline is pulled low (absence of one of the transistors or one bitline connection) or two of n-bitlines are pulled low. The NOR cell is located under the column of bitlines and connects to the metal1 bitline. Also of interest is the use of a short via-stack to make room for an additional bitline.

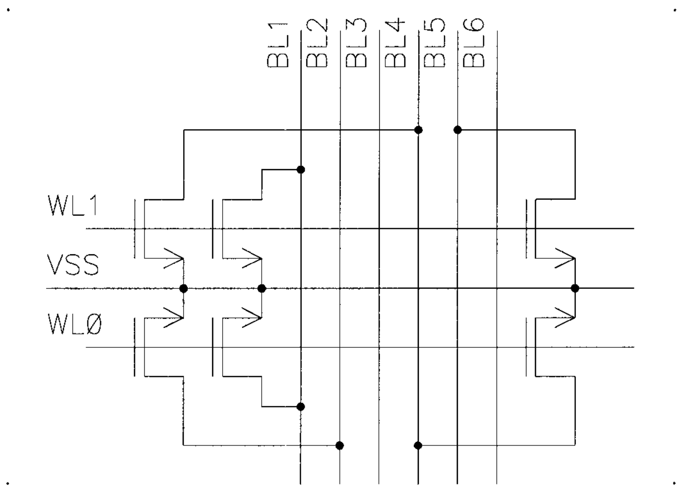

Figure 3.24: Hybrid 1 NOR to 2 Multi-Transistor ROM Cell Schematic

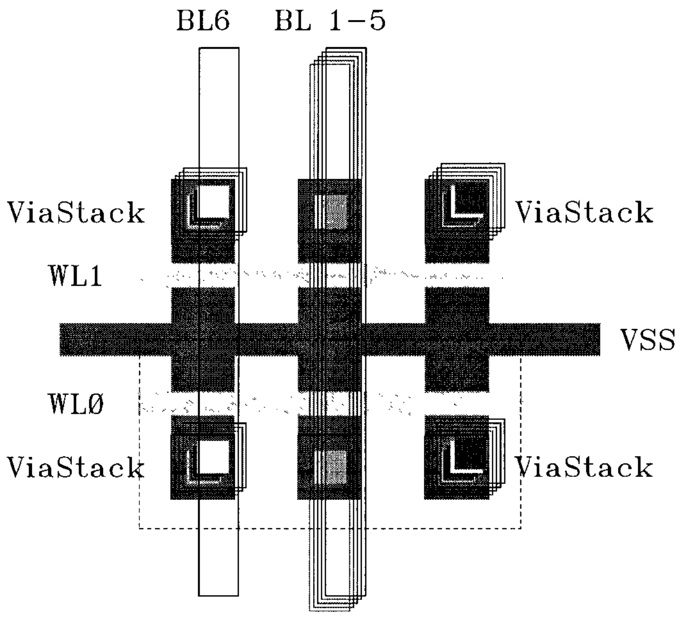

Figure 3.25: Hybrid 1 NOR to 2 Multi-Transistor ROM Cell Layout, 1.11-bits per Λ

Note that the drain contacts are not shared but rather have a separate via-stack for each of the multi-layered transistors. The use of a multi-transistor cell prohibits the sharing of the drain contacts with a NOR cell (see the hybrid 3 NOR to 1 multi-bitline cell): because this cell has the ability to connect to any two of the n-bitlines, it would make it impossible to tell which "shared" NOR cell the information is coming from.

The resulting density, in bits per Λ, is given by:

$$BitsPer\Lambda = (log_2(C(n+1,2)+C(n+1,1)+C(n+1,0))+1)/4.5\Lambda \tag{3.13}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = (log_2(10+5+1)+1)/4.5\Lambda$$

$$BitsPer\Lambda = 1.11$$

where n+1 is the number of bitlines available to attach to in the column of bitlines (in this case, five). The "plus one bit" comes from the NOR cell. The other two NOR cells can connect to two (n-bitlines choose two possibilities), one (n possibilities) or no (one possibility) bitlines. The log base two of these possible connections returns the bits stored within the multi-transistor cell. Thus density is the sum of the bits stored in the NOR cell and multi-transistor cell, divided by the cell area of four and a half Λ. The result, in this case, is 1.11-bits per Λ.

3.6.3 Hybrid 1 NOR to 3 Multi-Transistor ROM Cell

Extending the multi-transistor design, a multi-transistor cell (consisting of three multi-layered cells) and one NOR cell are used to form the hybrid cell. In addition, the short stack technique is used to increase the number of bitlines. Figure 3.26 shows the schematic design.

In Figure 3.27 note how the two short via-stacks allow two extra bitlines to run over top of the cell. The via-stack under BL6 (metal5) goes up to metal3 and the via-stack

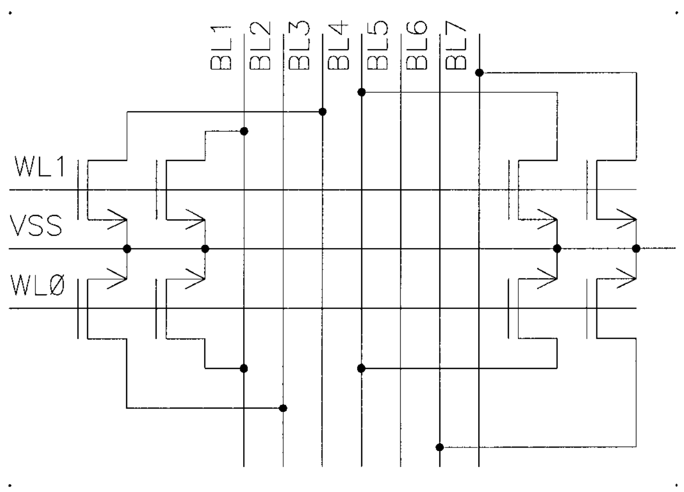

Figure 3.26: Hybrid 1 NOR to 3 Multi-Transistor ROM Cell Schematic

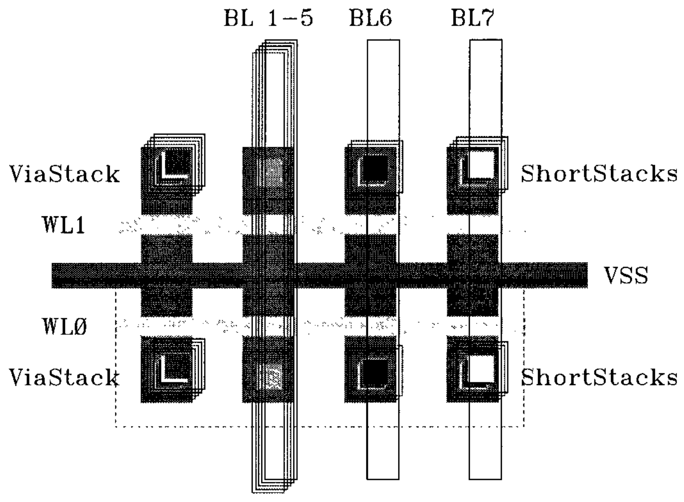

Figure 3.27: Hybrid 1 NOR to 3 Multi-Transistor ROM Cell Layout, 1.07-bits per Λ

under BL7 (metal5) goes up to metal4 (in this case the normal height of a via-stack is up to metal5).

The density, in bits per Λ, is given by:

$$BitsPer\Lambda = (log_2(C(n+2,3)+C(n+2,2)+C(n+2,1)+C(n+2,0))+1)/6\Lambda \tag{3.14}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = (log_2(20+15+6+1)+1/6\Lambda$$

$$BitsPer\Lambda = 1.07$$

where n+2 is the number of metal layer bitlines available to attach to in the column of bitlines. There are four NOR cells that make up this hybrid cell. One is a basic NOR cell, which contributes one bit to the density equation. That leave three NOR cells that can choose from n+2 bitlines. The possible combinations consist of n+2 choose three, n+2 choose two, n+2 choose one (equates to n+2), or none (n+2 choose none, which is one possibility). The log base two of the combinations returns the number of bits. Density is arrived at by adding the bits together, then dividing by the cell area of six Λ.

3.6.4 Hybrid 2 NOR to 2 Multi-Transistor ROM Cell

This hybrid cell consists of two NOR cells and one multi-transistor cell (which itself consists of two multi-layered cells). In previous cell designs, the multi-transistor cell was limited to a single bitline column. Here we demonstrate how to use the cross-over technique to gain access to multiple bitline columns.

In Figure 3.29 the top bitline in the first column of bitlines (left column) is removed. The removal of the top bitline allows a cross-over (the horizontal metal strip) to be used to access a second column of bitlines (on the right). In this way, the multi-layered cells can choose any two bitlines in either column of bitlines. A double via-stack is used to access

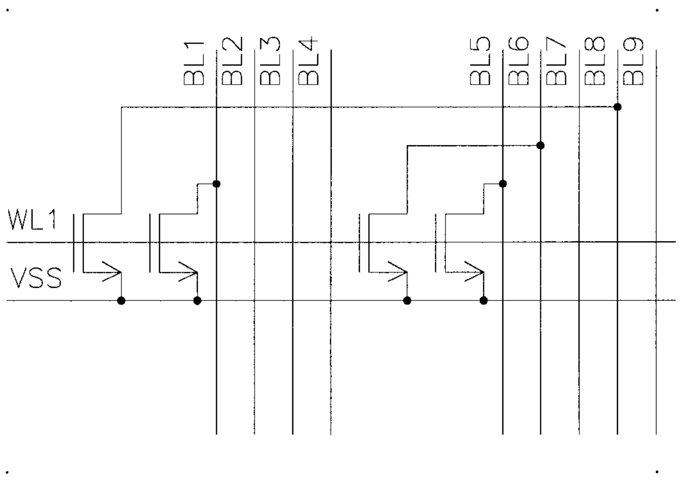

Figure 3.28: Hybrid 2 NOR to 2 Multi-Transistor ROM Cell Schematic

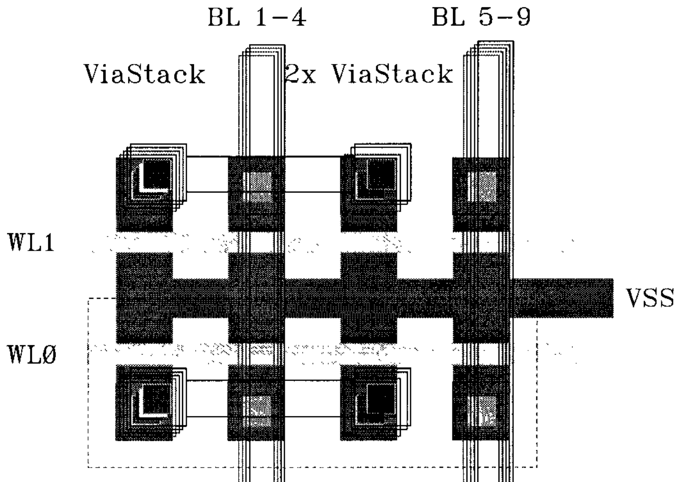

Figure 3.29: Hybrid 2 NOR to 2 Multi-Transistor ROM Cell Layout, 1.14-bits per Λ

the desired bitlines. These via-stacks do not connect to each other, the lower via-stack brings the signal from the underlying NOR cell up. The upper via-stack brings the signal from the cross-over down to access the bitlines in the right most column. In certain cases, depending on which bitlines need to be connected to, the cross-over may or may not be needed for any given cell in the array.

The cell density (for Figure 3.29) is given by:

$$BitsPer\Lambda = (log_2(C(2n-1,2) + C(2n-1,1) + C(2n-1,0)) + 2)/6\Lambda \qquad (3.15)$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = (log_2(21+7+1) + 2)/6\Lambda$$

$$BitsPer\Lambda = 1.14$$

where n is the number of metal layer bitlines available to attach to in the column of bitlines. The number of bits stored is calculated by the adding the two bits from the NOR cells with the "log base two" of the possible combinations generated by the multi-transistor cell and the two sets of bitline columns. As we have seen before, the number of combinations is arrived at by choosing two, one or zero combinations from the total number of bitlines. In this case, however, we have almost doubled the total number of bitlines (2*n-1). Given a cell area of six Λ, the density is simply the total bits divided by the area.

3.6.5 Hybrid 2 Multi-Valued to 1 Multi-Layered ROM Cell

In this cell, the multi-valued cell is combined with the multi-layer technique. The result is a hybrid cell with the greatest density out of all conventional, new and hybrid cells.

Figures 3.30 and 3.31 show the schematic and layout, respectively. The alternating via-stacks between "WL0" and "WL1" was done to balance the number of bits in each row. The multi-valued cells which "own" a via-stack contain more information in the form of which bitline that the information comes out on. The multi-value cells which do not

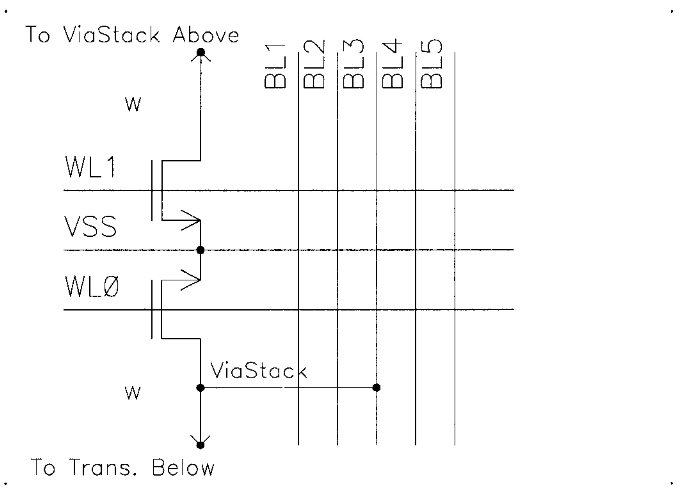

Figure 3.30: Hybrid 2 Multi-Valued to 1 Multi-Layered ROM Cell Schematic

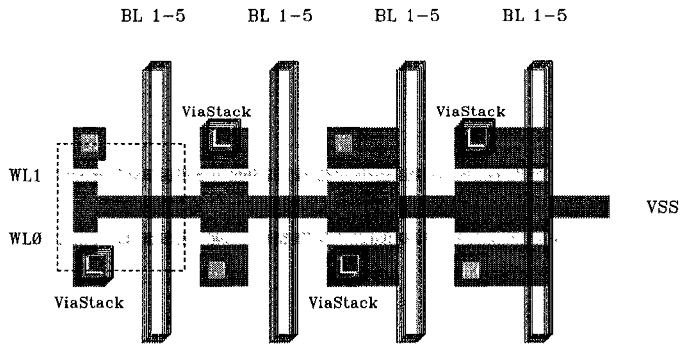

Figure 3.31: Hybrid 2 Multi-Value to 1 Multi-Layer ROM Cell Layout, 1.68-bits per Λ

"own" a via-stack, but rather "share" one, are sensed through all bitlines in the column (an effective "OR" operation on the sensed result of all the bitlines). In other words, for these "shared" cells no information is contained in *which* bitline the result comes out on. Note that the via-stack must connect to one of the bitlines, otherwise the "shared" cell will not be able to output its data. As such, each multi-valued cell can have five possible states (including the no transistor case). Thus the density, in bits per Λ, is given by:

$$BitsPer\Lambda = (log_2((d-1)(n+1)+1)+log_2(d))/4\Lambda \tag{3.16}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = (log_2(4(n+1)+1)+log_2(5)/4\Lambda$$

$$BitsPer\Lambda = 1.68$$

where n+1 is the number of metal layer bitlines available to attach to in the column of bitlines. And d (five in this case) is the number of drive-states the multi-value transistor can produce. Hence the multi-valued cell that "owns" the via-stack can represent "n+1 by d-1" states; and the "share" cell, d states. The reason we use d-1 drives for the cell that "owns" the via-stack is that one of the states is "no drive" and thus the bitline attached to the via-stack would not be able to be distinguished from the unattached bitlines. The log base two of the states returns the number of bits represented. To conclude, the sum of the bits divided by the area of four Λ yields the density of the hybrid cell.

For this cell, we assumed a five drive state multi-valued cell. Note that the density equation will not always scale proportionately with the number of drive-states. The physical design must be taken into account to accurately modify the density equation. Essentially, the number of drive-states will affect the cell area.

3.6.6 Hybrid 1 Multi-Valued to 1 Multi-Layered ROM Cell

This hybrid cell is similar to the previous hybrid multi-value cell except each cell has its own via-stack. Figures 3.32 and 3.33 show the schematic and layout, respectively.

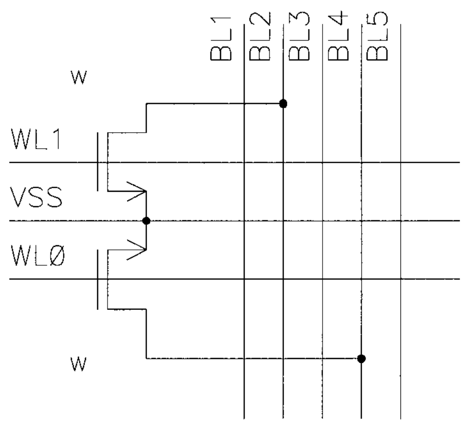

Figure 3.32: Hybrid 1 Multi-Valued to 1 Multi-Layered ROM Cell Schematic

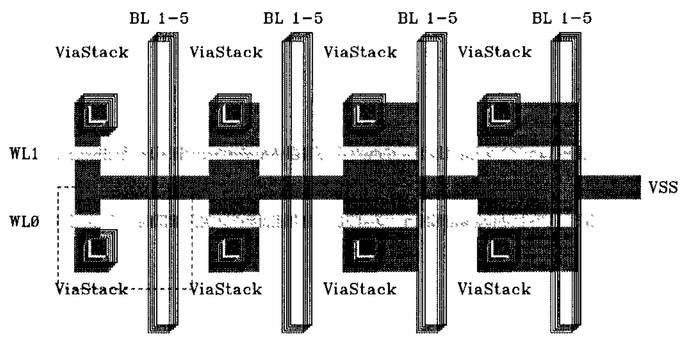

Figure 3.33: Hybrid 1 Multi-Value to 1 Multi-Layer ROM Cell Layout, 1.46-bits per Λ

Each of the multi-value cells can represent five states (or 2.3-bits), by changing the transistor width (w). In addition each cell can connect to one bitline in the column of bitlines.

The density, in bits per Λ, is given by:

$$BitsPer\Lambda = log_2(4(n+1)+1)/3\Lambda \tag{3.17}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = 1.46$$

where n+1 is the number of metal layer bitlines available to attach to in the column of bitlines. There are five drive states, one of which is “no drive”, thus we multiply the four active drives by the number of bitlines in the column (n+1) and add one more state for the fifth,“non-driving”, state. Log base two of the total possible states returns the number of bits stored in the cell. Density is given by dividing the bits by three Λ.

3.6.7 Hybrid 1 NAND to 1 Multi-Wordline ROM Cell

The multiple wordline structure works best with the NAND ROM cell architecture. Remember, the NAND architecture works by placing the cells in series to form the bitline. Through the presence or absence of a “jumper” over the transistor, the cell is able to store a logical zero or one. In this cell a basic NAND cell and a multiple wordline cell are combined (Figure 3.34).

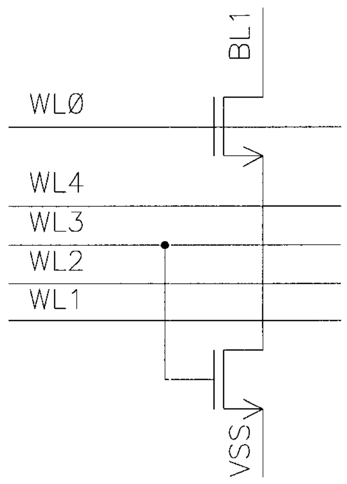

Figure 3.34: Hybrid 1 NAND to 1 Multi-Wordline ROM Cell Schematic

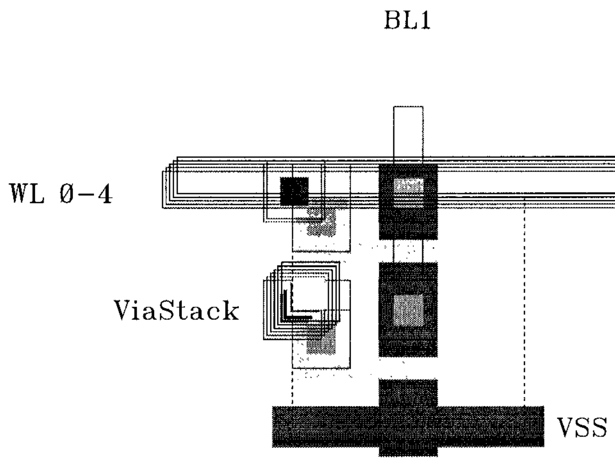

Figure 3.35: Hybrid 1 NAND to 1 Multi-Wordline ROM Cell Layout, 0.75-bits per Λ

Wordlines zero to four are in metal two to six (in this case). Metal1 is used for the bitline "jumpers". "WL0" (metal2) connects to the basic NAND cell, while the remainder of the wordlines can be connected to by the multi-wordline cell.

The cell density is given by:

$$BitsPer\Lambda = (log_2(p-1)+1)/4\Lambda \tag{3.18}$$

for $p = 5$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.75$$

where p is the number of metal layer wordlines available to attach to in the row of wordlines. p is derived from the total metal layers available, such that:

$$p = m - 1 \tag{3.19}$$

for $m = 6$ metal layers:

$$p = 5$$

where m is the number of metal layers in the process. One metal is reserved for the bitlines. Note that one wordline is used to attach to the basic NAND cells. The others form the multiple wordline structure. Thus there are p-1 possible connections for the multiple wordline, as the first wordline connects to the basic NAND cell. The "one" bit comes from the basic NAND cell. Log base two of the possible wordline connections returns the number of bits stored in multiple-wordline structure. Summing the bits from the NAND and multi-wordline cells, then dividing by the cell area of four Λ, gives the cell density.

3.7 Comparison

Figure 3.36 compares the normalized cell density of the different ROM architectures versus the number of metal layers in various manufacturing processes (based on the density Equations 3.1 to 3.18 presented in the previous sections). Cell densities have been normalized to the NOR ROM cell, which has a density of 1-bit per Λ. Data cell area was measured in Cadence. The NOR cell and multi-valued cell are unaffected by the number of metal layers, while the other cell densities increase with the number of metal layers. Those hybrid cells with a low ratio of NOR/multi-valued to multi-layered cells increase at a greater rate with the number of metal layers than those with a higher ratio. For a small number of metal layers, however, the higher ratio of NOR/multi-valued to multi-layered cells results in more compact architectures (and thus higher densities). From Figure 3.36 the benefit of using a 2 multi-value to 1 multi-layer ROM cell is clearly evident with a 68% increase in cell density over that of the NOR ROM cell (for six metal layers).

Table 3.2 and Figure 3.37 shows how the minimum width of a metal layer expands over the range of available metal layers [14]. However, metal width is of little interest when working with ROMs. Of more interest is metal pitch and even more so, the metal pitch with a via, as a ROM may require a via connection at every cell. The metal pitch with a via is the largest pitch associated with a metal layer and thus the determining pitch for the hybrid multi-layer ROM cells. Manufacturing design rules often require a metal enclosure of the via which expands the metal beyond its minimum width. Thus the pitch of the metal,

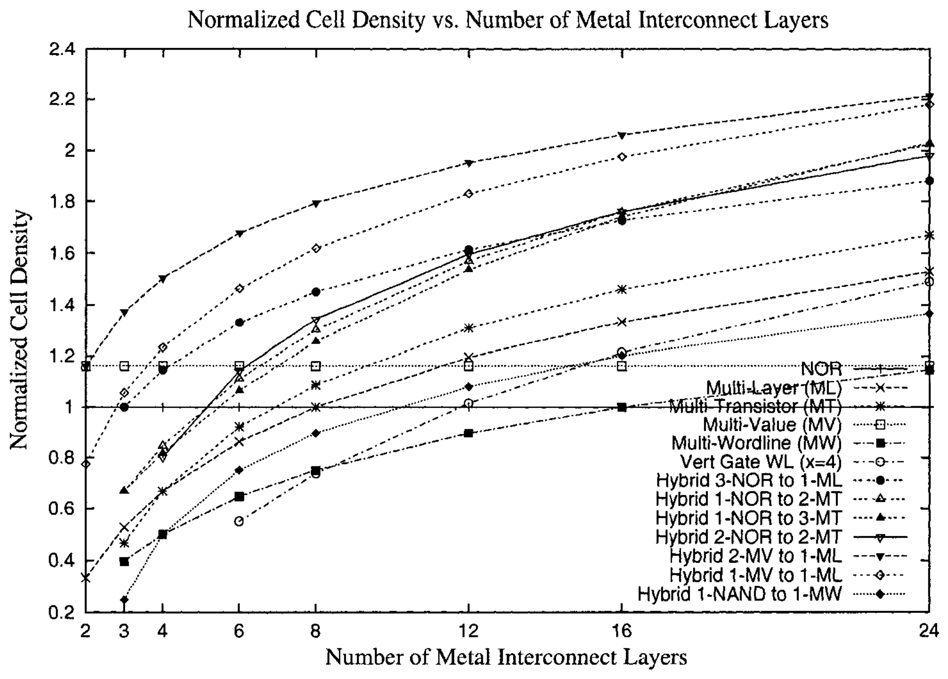

Figure 3.36: Normalized Cell Density vs. Number of Metal Layers.

including the via, becomes a limiting factor of how close the cells can be manufactured.

Process	P1	P2	P3	P4	P5	P6	P7	P8
0.13-μm CMOS	1.00	1.00	1.00	1.00	1.00	1.00	1.67	1.67
0.18-μm CMOS	1.00	1.00	1.00	1.00	1.00	1.67	na	na
0.25-μm CMOS	1.00	1.00	1.00	1.00	1.33	na	na	na
0.35-μm CMOS	1.00	1.00	1.67	na	na	na	na	na

Table 3.2: Process Metal vs. Relative Width.

Note how the metal layers above the first metal layer remain the same width until the upper layer(s) are reached. The constant pitch of the middle metal layers bodes well for their use in multiple bitlines.

Table 3.3 shows how the pitch of the metal grows with the increase in metal layer position [14]. While metal pitch may be a good gauge for routing problems, in a ROM, the metal pitch with a via is of more importance (see Table 3.4). Again, note how the metal pitch remains constant for the "middle" metal layers.

Table 3.4 and Figure 3.39 shows how the minimum pitch of a metal with a via increases

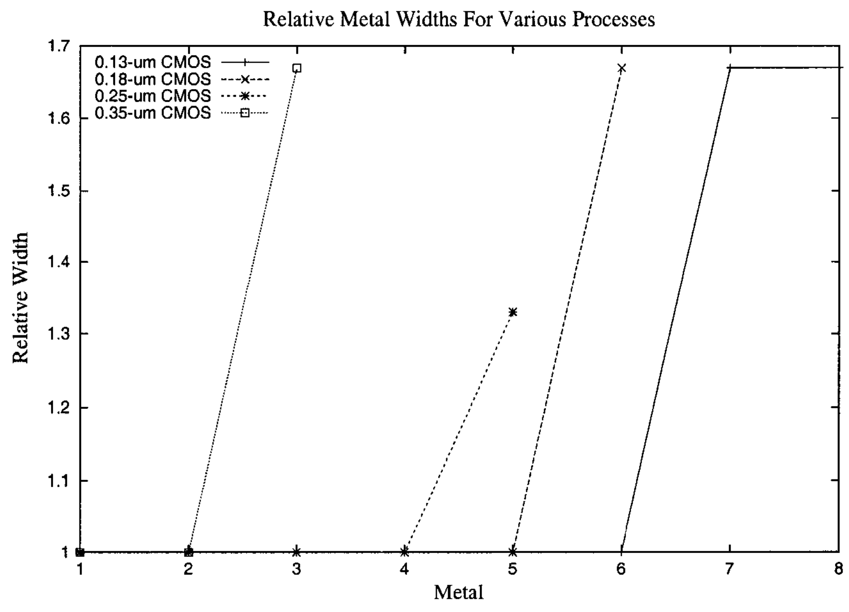

Figure 3.37: Process Metal vs. Relative Width.

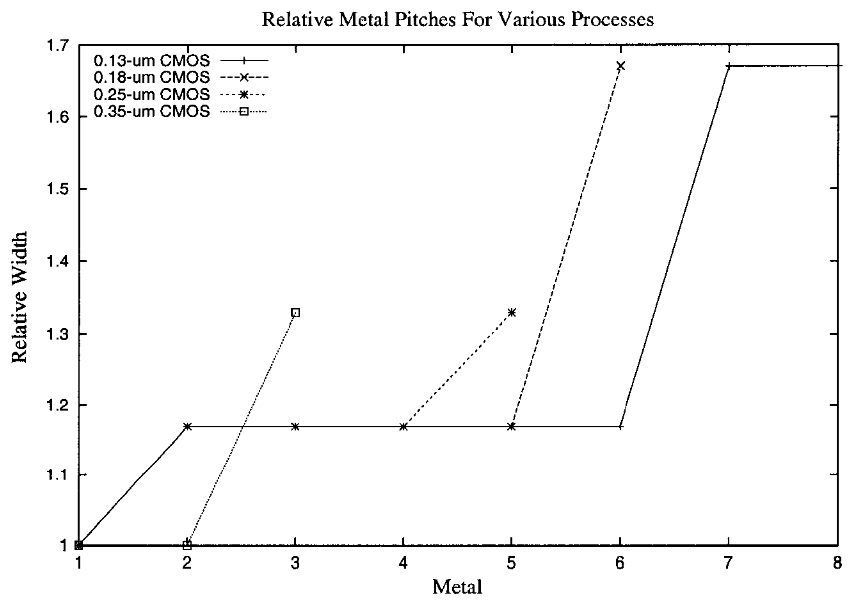

Figure 3.38: Process Metal vs. Relative Metal Pitch.

Process	P1	P2	P3	P4	P5	P6	P7	P8
0.13-μm CMOS	1.00	1.17	1.17	1.17	1.17	1.17	1.67	1.67
0.18-μm CMOS	1.00	1.17	1.17	1.17	1.17	1.67	na	na
0.25-μm CMOS	1.00	1.17	1.17	1.17	1.33	na	na	na
0.35-μm CMOS	1.00	1.00	1.33	na	na	na	na	na

Table 3.3: Process Metal vs. Relative Metal Pitch.

with the metal layer [14]. Remember, relative pitch represents how much larger the metal pitch is on a higher level metal relative to metal1. These numbers are representative of the effect an increase in metal size and spacing would have on the density of a ROM. The metal width (Table 3.2) and pitch (Table 3.3) are only half of the story. In a ROM, the metal layers need to connect to the drain of the transistors at regular locations. These connections cannot be "shuffled" to save space (as would be done in a routed digital design). Thus it is necessary to examine the metal pitch with a via.

Process	P1	P2	P3	P4	P5	P6	P7	P8
0.13-μm CMOS	1.00	1.14	1.14	1.14	1.14	1.14	1.71	1.71
0.18-μm CMOS	1.00	1.14	1.14	1.14	1.14	1.71	na	na
0.25-μm CMOS	1.00	1.14	1.14	1.14	1.42	na	na	na
0.35-μm CMOS	1.00	1.00	1.42	na	na	na	na	na

Table 3.4: Process Metal vs. Relative Metal Pitch with Lower Via.

From Table 3.4 it can be seen that multiple metal layers can be used with little impact on the size of the ROM. For example, in 0.13-μm CMOS by taking a small area penalty it is possible to use six layers of metal to increase the density up to 68% by using a hybrid ROM cell. The area penalty comes from the increase in metal pitch (with a via) for the metal layers above the first layer (the exact value is confidential). On the other hand, using eight layers of metal would increase the cell size prohibitively. In essence, the benefit gained from using the extra two metals (metal7 and metal8), in terms of "extra bitlines", is far out-weighed by the increase in cell area.

Table 3.5 and Figure 3.40 show how the lower layer metals do not have an impact

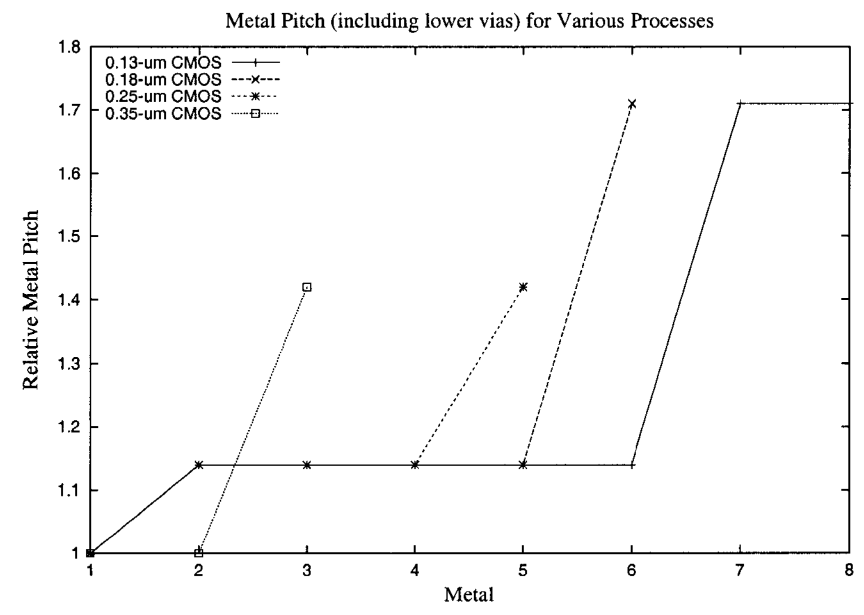

Figure 3.39: Process Metal vs. Relative Metal Pitch with the Via Below.

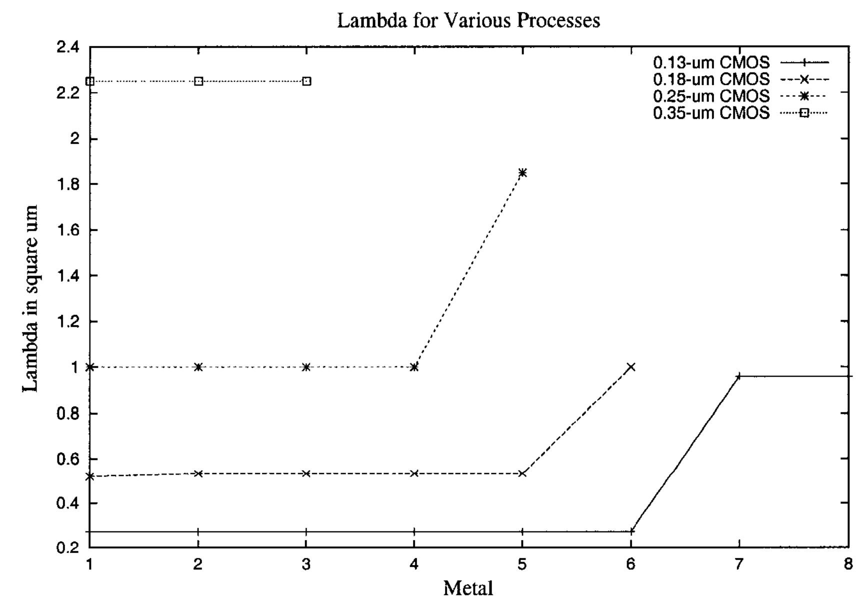

Figure 3.40: Λ Size for Various Manufacturing Processes.

Process	**MG1 (um2)**	**MG2 (um2)**	**MG3 (um2)**	**MG4 (um2)**	**MG5 (um2)**	**MG6 (um2)**	**MG7 (um2)**	**MG8 (um2)**
0.13-μm CMOS	0.27	0.27	0.27	0.27	0.27	0.27	0.96	0.96
0.18-μm CMOS	0.52	0.53	0.53	0.53	0.53	1.00	na	na
0.25-μm CMOS	1.00	1.00	1.00	1.00	1.85	na	na	na
0.35-μm CMOS	2.25	2.25	2.25	na	na	na	na	na

Table 3.5: Lambda vs. Transistors and Metal Layers within a Process.

on Λ as the smallest transistor size is larger than the required metal pitch of those metal layers. However, the higher level metal layers pitch expands Λ beyond the dimensions of the smallest transistor and can severely impact the area of the ROM (as can be seen from Figure 3.40 when using 8 metal layers in 0.13-μm CMOS).

Figure 3.36 shows the cell density in terms of BitsPerΛ. Figure 3.39 and Table 3.4 shows how the physical size of Λ changes with the number of metals used. Together, these figures give us the un-compacted cell density of the various ROM cell architectures in each manufacturing process.

To give the reader a physical feel for the amount of information that can be stored, we will calculate cell storage densities for the different manufacturing processes. Table 3.5 and Figure 3.40 lists Λ sizes for each technology based on the metal pitches with a via and the minimum transistor sizes. In Figure 3.41, we see the theoretical cell storage over a 1-mm^2 area for various processes (each containing a different number of metal layers and a different Λ size). The storage over a 1-mm^2 area is a best case number as we are neglecting to include the support structures, such as wordline strapping and V_{SS} connection points, inside the memory array and the peripheral circuity needed in every memory (such as the wordline decoders and sense-amplifiers). The following equation is used to calculate the amount of cell storage:

$$storage = (1mm^2/\Lambda) * BitsPer\Lambda \qquad (3.20)$$

Note that only the size of the storage cells is taken into account; at this time the area of the periphery is ignored.

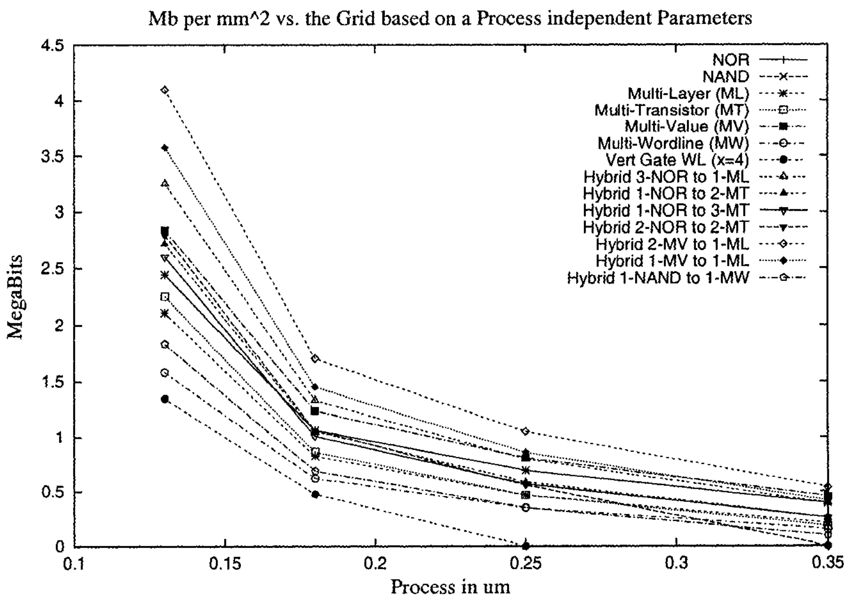

Figure 3.41: Theoretical Cell Storage Capacity for Various Manufacturing Technologies in a 1-mm^2 Area, Based on Λ.

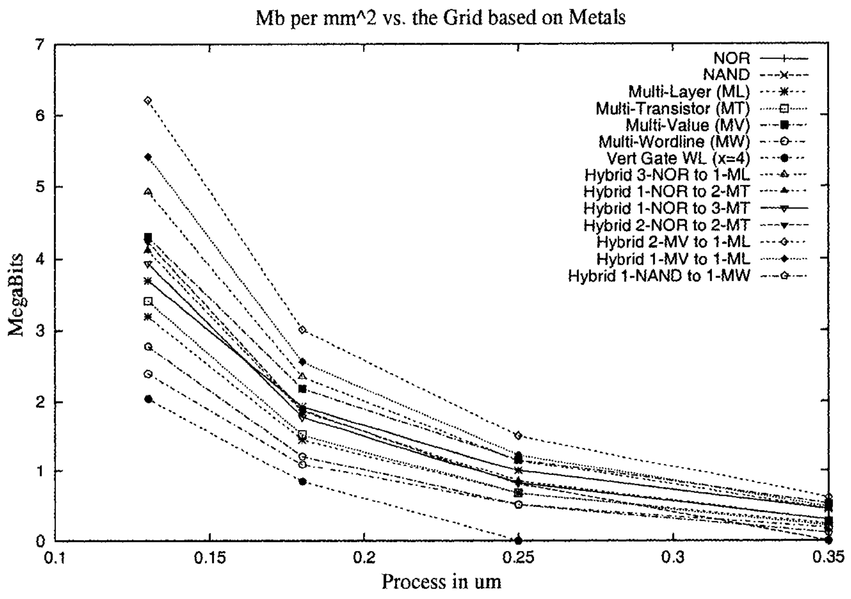

Figure 3.42: Theoretical Cell Storage Capacity for Various Manufacturing Technologies in a 1-mm^2 Area, Based on Metal Layers and Minimum Transistor Sizes.

Figures 3.41 and 3.42 plot cell storage over a 1-mm^2 area using Equation 3.20 which is varied over four manufacturing technologies. Each manufacturing technology is listed by its feature size in μm. Figure 3.41 replaces Λ in Equation 3.20 with a area based solely on the metal layer. In both figures, note how the hybrid multi-layered cell densities increase at a greater rate than the conventional cells when moving to a smaller process technology. This greater increase in density is attributed to the effect of the extra metal layers available in the smaller process technologies.

As can be seen, in Figure 3.41, the greatest effect on density is the decrease in Λ (which is dependent on the manufacturing process); the second most important factor is ROM cell architecture.

3.8 Comparison in 0.13-μm CMOS

In the previous section we compared the architectures in a process-independent way. In this section, we have hand-compacted each of the ROM architectures, in Cadence, to minimize the cell area, subject to the design rules in a six metal layer 0.13-μm CMOS process.

Table 3.6 shows the normalized bits per area (relative to the NOR cell) before and after compacting the cells in a 0.13-μm CMOS process. Each ROM cell has been compressed to a minimum size. The higher the value, the higher the density. In the ratio column, we see the impact of compression. Those cells whose ratio is less than one did not compact as well as the NOR ROM cell. On the other hand, those cells whose ratio is greater than one compacted more than the NOR ROM cell.

It is interesting to note the effect that compaction had on the normalized bits per area in 0.13-μm CMOS. The maximum variation was 20%, the average was 1% and the standard deviation was 9%. These statistics supports the utility of the grid-based (process-independent) area analysis.

With these new compacted sizes we can calculate the storage over a 1-mm^2 area (Table

ROM Cell	**Not Compacted Normalized Density**	**Compacted 0.13-μm CMOS Normalized Density**	**Ratio**
NOR	1.00	1.00	1.00
NAND	1.00	1.04	1.04
Multi-Layer (ML)	0.86	0.89	1.03
Multi-Transistor (MT)	0.92	1.10	1.20
Multi-Value (MV)	1.16	1.28	1.10
Multi-Wordline (MW)	0.65	0.58	0.89
Vert Gate WL (x=4)	0.55	0.49	0.89
Hybrid 3-NOR to 1-ML	1.33	1.28	0.96
Hybrid 1-NOR to 2-MT	1.11	1.07	0.96
Hybrid 1-NOR to 3-MT	1.07	1.01	0.95
Hybrid 2-NOR to 2-MT	1.14	1.14	1.00
Hybrid 2-MV to 1-ML	1.68	1.85	1.10
Hybrid 1-MV to 1-ML	1.46	1.69	1.16
Hybrid 1-NAND to 1-MW	0.75	0.72	0.96

Table 3.6: Normalized Cell Density Before and After Compaction in Six Metal Layer 0.13-μm CMOS.

3.7) based on the number of bits stored in a cell, multiplied by a 1-mm^2 area, divided by the compacted area of the cell. The values in Table 3.7 are based on memory cell size and do not include sense-amplifiers or other peripheral components. Similar to Table 3.6 the *Ratio* shows how well the cell compacted relative to the NOR cell (a higher number means it compacted better than the NOR cell). We include this information for the purposes of comparisons to other memory cell architectures, which was not discussed herein.

3.9 Summary

It is beneficial, in terms of cell density, to use multi-layered bitlines in the design of a ROM. It is important to balance the use of extra metal layers against the potential increase in ROM size when determining the number of metal layers to use.

From Figures 3.38 and 3.39 is is interesting to note how the middle layers of metal tend to have the same dimensions. Thus making the use of extra metal layers, through the hybrid ROM cells, clearly beneficial.

ROM Cell	**Process Independent 0.13-μm CMOS (Mb)**	**Compacted 0.13-μm CMOS (Mb)**	**Ratio**
NOR	2.44	3.86	1.00
NAND	2.44	4.01	1.04
Multi-Layer (ML)	2.10	3.43	1.03
Multi-Transistor (MT)	2.25	4.26	1.20
Multi-Value (MV)	2.83	4.94	1.10
Multi-Wordline (MW)	1.58	2.22	0.89
Vert Gate WL (x=4)	1.34	1.89	0.89
Hybrid 3-NOR to 1-ML	3.25	4.95	0.96
Hybrid 1-NOR to 2-MT	2.71	4.12	0.96
Hybrid 1-NOR to 3-MT	2.60	3.90	0.95
Hybrid 2-NOR to 2-MT	2.79	4.41	1.00
Hybrid 2-MV to 1-ML	4.10	7.14	1.10
Hybrid 1-MV to 1-ML	3.57	6.54	1.16
Hybrid 1-NAND to 1-MW	1.83	2.79	0.96

Table 3.7: Theoretical Cell Storage in Mb for 0.13-μm CMOS Manufacturing Technology in a 1-mm^2 Area.

Compacting has little relative effect on the density of the ROM cells (Table 3.6). In other words, compacting the various architectures does not significantly benefit one cell more than another. Compacting, in general, does have a major effect on the density of all the ROM architectures. For our purposes, however, it demonstrates the accuracy of a process-independent normalized area when comparing various ROM architectures.

From our results, the hybrid NOR multi-valued multi-layer cell offers the highest density for processes with less than 10 metal layers.

Chapter 4

RAM-ROM

We have shown how information can be stored in the multiple layers above the transistors. Extending this result, it is possible to overlay a ROM on top of a RAM. Using the multi-layer technique, we could build a ROM above a Flash EEPROM, SRAM or any other type of storage cell. By identifying which bitline the information comes out on, it is possible to simultaneously read both the value stored in the memory cell and the value of the ROM. The key benefit of our combined RAM-ROM cells is that the ROM requires no extra transistors beyond those already used in the RAM. The combined RAM-ROM also means no extra wordline decoding circuitry or drivers are required. The primary negative effect on the potential speed of the RAM-ROM array would be the inter-bitline capacitive coupling along with the extra decoding circuitry. Aiding the operational speed, however, is the reduced gate drain capacitance of the access transistors as only a fraction of access transistors in a column would be connected to any one bitline.

4.1 SRAM-ROM

Static Random Access Memory (SRAM) can be found in virtually every microprocessor produced today in the form of a memory cache. SRAM stores values so long as power is supplied to the circuit. Typically each SRAM cell consists of six transistors: four in the form of back to back inverters and two access transistors. We have designs a number of

SRAM-ROM circuits that can store two or more bits using only the six original transistors found in the SRAM (1 SRAM bit and 1 or more ROM bits). It will become apparent, however, that the presented technique can be easily applied to any other type of RAM and is not exclusive to SRAM. As a result, designers could store ROM-based look-up tables overtop of their RAMs without any cell area penalty.

4.2 Conventional SRAM-ROM

Previous SRAM-ROM designs [2][15][7] typically add extra transistors to the SRAM memory array to implement the ROM. These designs primarily benefit from the sharing of the SRAM's bitlines, sensing and column decode circuitry. Other designs use special structures, such as fuses built into the SRAM cell [10][11], so that the SRAM cell can be turned permanently into a ROM cell.

Matsumura et al. [12] presents an SRAM-ROM that implements the ROM by choosing to connect the power and ground of each SRAM cell, during manufacturing, to one of two power lines and one of two ground lines. By controlling the voltages on these lines it is possible to have the SRAM cell output the contents of the ROM (which are based on the "power and ground" connections inside the SRAM cell).

4.3 Hybrid SRAM-ROM

An SRAM typically contains two bitlines that run overtop of the SRAM cell ("true" and "complementary"). These bitlines are used for both reading and writing the memory cell. In one case of our SRAM-ROM, where there are multiple "true" and "complementary" bitlines, the SRAM cell connects to a pair of "true" and "complementary" bitlines. This way, when the SRAM cell is read, either the associated "true" or "complementary" bitline will be pulled low (we do not have fore-knowledge of which one, it depends on the value stored in the SRAM). By detecting on which pair of bitlines the information arrived, we can deduce the contents of the ROM. The information in the ROM is programmed when we choose which pair of "true" and "complementary" bitlines to connect to the SRAM cell. When writing to the SRAM, all the "true" bitlines are forced to one value and the "complementary" bitlines to the opposite (complementary) value. Driving all the bitlines removes the need to know which pair of bitlines the SRAM cell is connected to. Connecting the SRAM to a pair of bitlines also allows us to read the ROM at the same time as we read the SRAM cell. Alternatively, when the SRAM can connect to any "true" or "complementary" bitline, we must read then write back the opposite value to the SRAM cell in order to determine what bitlines are connected to the cell. The benefit of such a configuration would be increased ROM storage at the cost of increased access time.

In the following sections, the layouts will assume a six metal layer process. Thus the number n of bitline layers in a column will be defined as,

$$n = m - 2 \tag{4.1}$$

$$\text{for } m = 6 \text{ metal layers:}$$

$$n = 4$$

where m is the number of metal layers (interconnect layer) in the manufacturing process. One layer is reserved for a wordline and another for use in constructing the SRAM cell. The size of the cells is based on the ROM structure implemented above the SRAM. The density equations in the following cells include 1-bit from the SRAM cell. We will present a variety of ROM type structures and sizes. Given an SRAM size, one can choose the ROM structure that would best fit the SRAM.

4.3.1 SRAM-ROM 2 Bitline Columns

Figure 4.1 shows an SRAM-ROM that uses two columns of bitlines (one column for the "true" bitlines and one for the "complementary" bitlines).

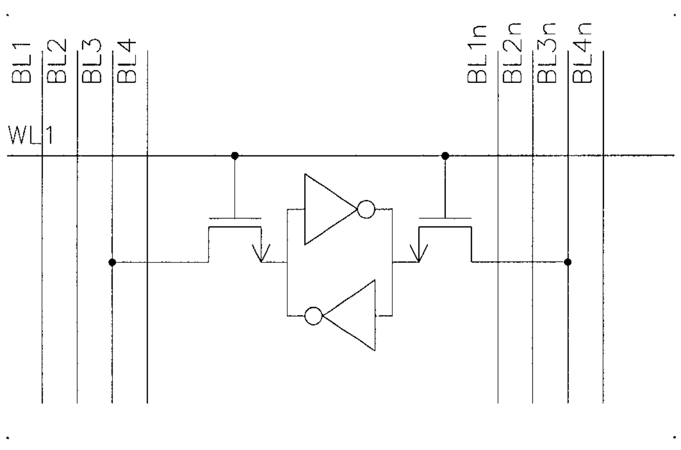

Figure 4.1: SRAM-ROM 2 Bitline Column Cell Schematic

The dashed box in Figures 4.2 and 4.3 shows the area required to implement the layout of the ROM. It is assumed that the SRAM can fit inside the area, surrounded by the dashed box, below the ROM. Figure 4.2 shows a cell with no wordline connection. Here it is also assumed that the wordline is strapped at multiples of the cell width.

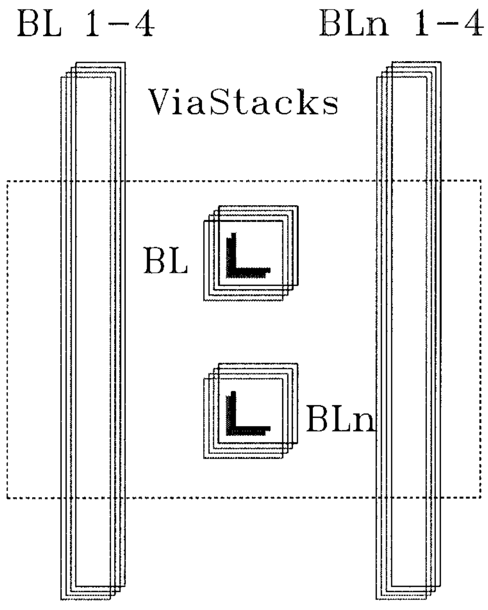

Figure 4.2: SRAM-ROM 2 Bitline Column Cell Layout, 0.50-bits per Λ

The density, in bits per Λ, is given by:

$$BitsPer\Lambda = (log_2(n) + 1)/6\Lambda \tag{4.2}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.50$$

where n is the number of metal layer bitlines available to attach to in the column of bitlines (in this case four). Log base two of the total possible states (n) returns the number of bits stored in the ROM cell. One bit is added to account for the SRAM cell storage. Density is obtained by dividing the bits by the area of six Λ.

Alternatively, Figure 4.3 shows a cell where the wordline is strapped at every cell. The density of the wordline strapped cell, in bits per Λ, is given by:

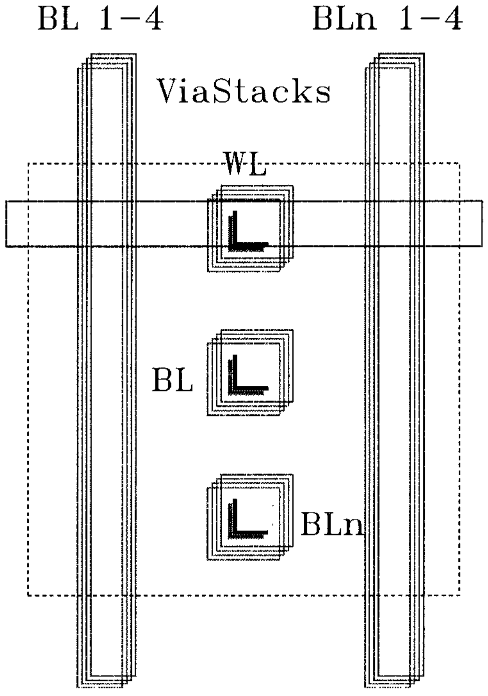

Figure 4.3: SRAM-ROM 2 Bitline Column Cell with Wordline Via Layout, 0.33-bits per Λ

$$BitsPer\Lambda = (log_2(n) + 1)/9\Lambda \tag{4.3}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.33$$

where n is the number of metal layer bitlines available to attach to in the column of bitlines. Log base two of the total possible states (n) returns the number of bits stored in the ROM cell. One bit is added to account for the SRAM cell storage. Density is given by dividing the bits by the area of nine Λ. Essentially, this cell can only store two-thirds the information when a wordline connection is provided in every cell.

4.3.2 SRAM-ROM 1 Bitline Column

Here we examine an SRAM-ROM single bitline column cell. Figure 4.4 shows how an SRAM cell can be combined with the multi-layer technique to form an SRAM-ROM cell. Again note that the ROM portion requires no extra transistors.

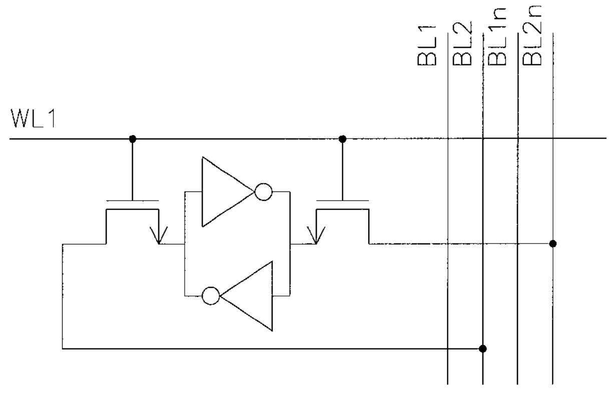

Figure 4.4: SRAM-ROM 1 Bitline Column Cell Schematic

Figure 4.5 shows the layout of the ROM portion of the SRAM-ROM cell. It is assumed that the SRAM cell fits under the ROM. In this cell the "true" (BL) and "complementary" (BLn) bitlines are created in a single column of bitlines. Placing the bitlines in a single column effectively reduces the storage and area of the ROM in relation to the SRAM. However, the single bitline column facilitates the use of a smaller SRAM cell.

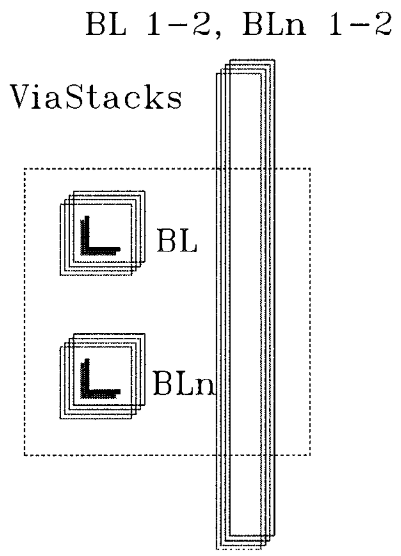

Figure 4.5: SRAM-ROM 1 Bitline Column Cell Layout, 0.50-bits per Λ

Wordlines are assumed to be strapped at multiples of the memory cell. The density, in bits per Λ, is given by:

$$BitsPer\Lambda = (log_2(n/2) + 1)/4\Lambda \tag{4.4}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.50$$

where n is the number of metal layer bitlines available to attach to in the column of bitlines. The number of metal bitlines is divided in half (half for the true bitline and half for the complementary bitline). Log base two of the total possible states (n/2) returns the number of bits stored in the ROM cell. One bit is added to account for the SRAM cell. Density is given by dividing the bits by the area of four Λ.

The SRAM-ROM in Figure 4.6 provides a contact location for wordline strapping in every SRAM-ROM cell. The density, in bits per Λ, is given by:

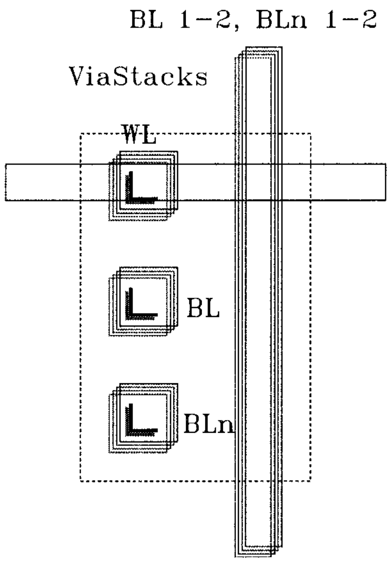

Figure 4.6: SRAM-ROM 1 Bitline Column Cell with Wordline Via Layout, 0.33-bits per Λ

$$BitsPer\Lambda = (log_2(n/2) + 1)/6\Lambda \tag{4.5}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.33$$

where *n* is the number of metal layer bitlines available to attach to in the column of bitlines. The number of metal bitlines is divided in half (half for the true bitline and half for the complementary bitline). Log base two of the total possible states (*n*/2) returns the number of bits stored in the ROM cell. One bit is added to account for the SRAM cell. Density is given by dividing the bits by the area of six Λ. Again, this cell with the wordline connection stores two-thirds of the information of the same cell without the wordline connection.

4.3.3 SRAM-ROM 1 Bitline Column with One Fixed Bitline

This cell is very similar to the one bitline column cell except that the "complementary" bitline is fixed to the first bitline layer. Fixing the "complementary" node of the SRAM to one bitline leaves the other "true" SRAM node free to choose any other bitline in the column as the "true" bitline. Note that this improves the ROM density but reduces the ROM speed. Figure 4.7 shows the schematic of this cell. In common with our other SRAM-ROMs, the ROM portion requires no extra transistors. As one bitline is fixed to the SRAM cell a specialized sense-amplifier would have to be designed that could compare the fixed bitline to all the other bitlines.

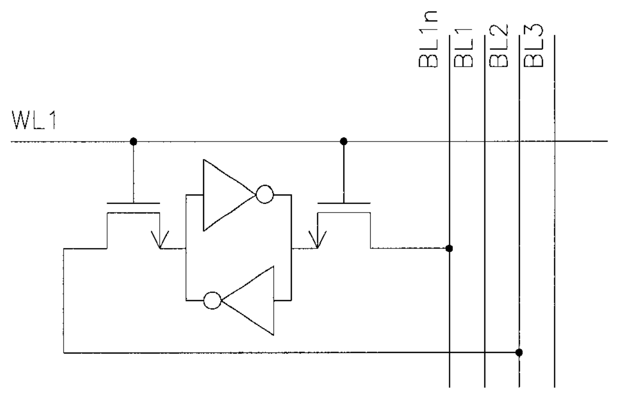

Figure 4.7: SRAM-ROM 1 Bitline Column with One Fixed Bitline Cell Schematic

Figure 4.8 shows the layout of the ROM portion of the SRAM-ROM cell. Here we have a very compact layout for the ROM, allowing the smallest of SRAM cells to fit under it. As can be seen the "complementary" bitline is attached to the first available bitline layer. This leaves more layers for the "true" bitline to attach to.

Wordlines are assumed to be strapped at multiples of the memory cell. The density, in bits per Λ, is given by:

$$BitsPer\Lambda = (log_2(n-1)+1)/2\Lambda \tag{4.6}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = 1.29$$

where n is the number of metal layer bitlines available to attach to in the column of bitlines. The "complementary" bitline uses one bitline layer, leaving the rest for the "true" bitline.

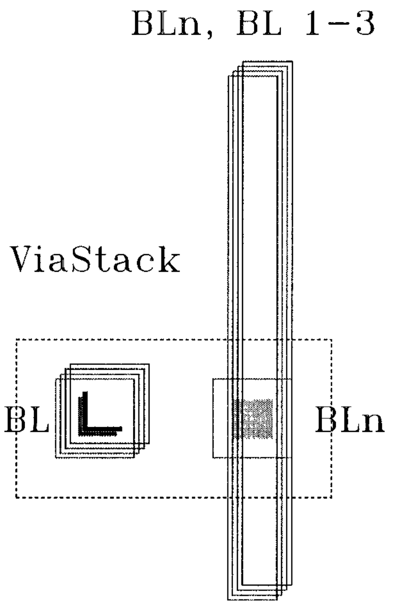

Figure 4.8: SRAM-ROM 1 Bitline Column with One Fixed Bitline Cell Layout, 1.29-bits per Λ

Log base two of the total possible states (n-1) returns the number of bits stored in the ROM cell. One bit is added to account for the SRAM cell. Density is given by dividing the number of bits by the area of two Λ.

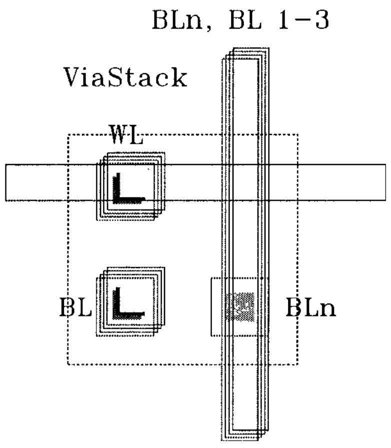

Figure 4.9: SRAM-ROM 1 Bitline Column Cell with One fixed Bitline and a Wordline Via Layout, 0.65-bits per Λ

Figure 4.9 provides a contact location for wordline strapping for every SRAM-ROM cell. The density, in bits per Λ , is given by:

$$BitsPer\Lambda = (log_2(n-1)+1)/4\Lambda \tag{4.7}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.65$$

where n is the number of metal layer bitlines available to attach to in the column of bitlines. One bitline is used for the "complementary" bitline. Log base two of the total possible states (n-1) returns the number of bits stored in the ROM cell. One bit is added to account for the SRAM cell. Density is given by dividing the bits by the area of four Λ.

4.3.4 SRAM-ROM Multiple Wordline

The multiple wordline SRAM-ROM is an interesting case of the SRAM-ROMs. Instead of using the extra process layers to implement bitlines, we use them to create more wordlines for each cell. By activating the wordlines within a stack of wordlines at different times, it is possible to determine which wordline is connected to an SRAM cell. This provides us with the ROM information from the cell. In this case, the SRAM can produce one of three states on the "true" and "complementary" bitlines: high-low, low-high and high impedance. Our ability to sense these states allows us to read both the ROM and SRAM. A drawback of multiple-wordline SRAM-ROM architecture is that additional wordlines drivers and decoders would be required. The capacitance of the wordlines due to the gate capacitance of the SRAM access transistors, however, would drop by a factor equal to the number of wordlines per row (but, on the other hand, we would introduce significant inter-wordline parasitic capacitance). To access the SRAM's data all the wordlines can be asserted simultaneously. To access the ROM data, the wordlines have to be asserted sequentially. Figure 4.10 shows the schematic representation of the the multiple wordline SRAM-ROM. Figures 4.11 and 4.12 show the layout variants of this cell.

The density of the layout in Figure 4.11 is given by:

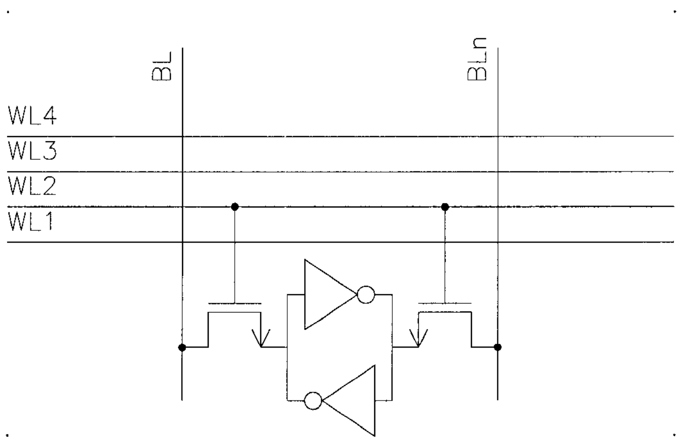

Figure 4.10: SRAM-ROM Multiple Wordline Cell Schematic

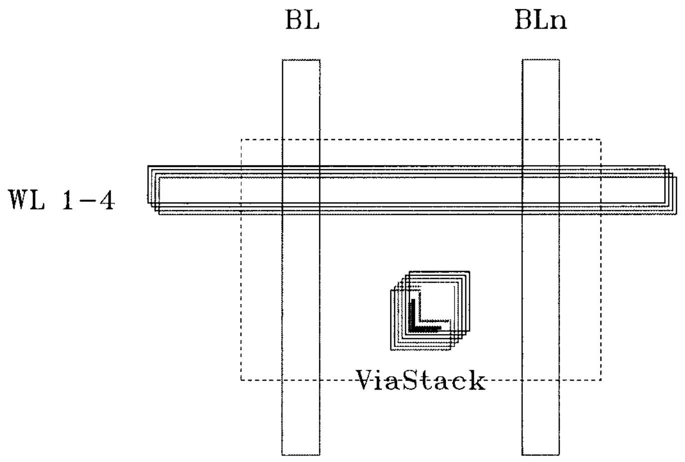

Figure 4.11: SRAM-ROM Multiple Wordline Cell Layout, 0.50-bits per Λ

$$BitsPer\Lambda = (log_2(n) + 1)/6\Lambda \tag{4.8}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.50$$

where n is the number of metal layers available for the bitlines and wordlines. Similar to the multiple bitline structure, the multiple wordline structure needs to connect to one of the wordlines. Log base two of the total possible states (n-1) returns the number of bits stored in the ROM cell. One bit is added to account for the SRAM cell. Density is given by dividing the bits by the area of six Λ.

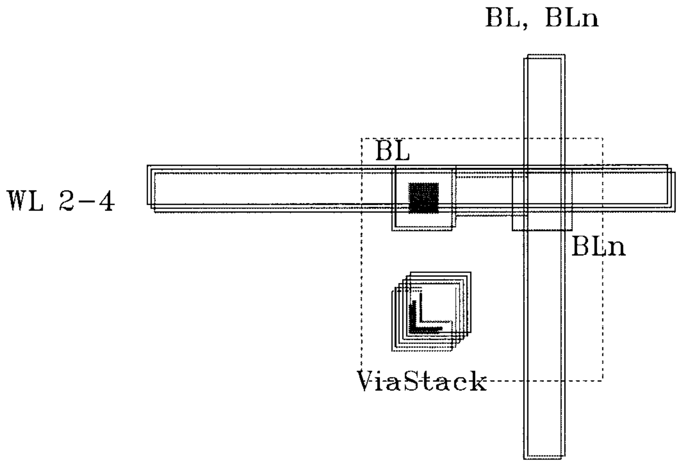

Figure 4.12: SRAM-ROM Multiple Wordline 1 BL Column Cell Layout, 0.64-bits per Λ

The density of the layout in Figure 4.11 is given by:

$$BitsPer\Lambda = (log_2(n-1) + 1)/4\Lambda \tag{4.9}$$

for $n = 4$ in a 6 metal layer process:

$$BitsPer\Lambda = 0.64$$

where n is the number of metal layers available for the bitlines and wordlines. Here the "true" (BL) and "complementary" (BLn) bitlines are placed overtop of each other in the first two layers available. This saves area but reduces the number of wordlines. Log base two of the total possible states (n-1) returns the number of bits stored in the ROM cell. One bit is added to account for the SRAM cell. Density is given by dividing the bits by the area of four Λ.

4.4 Comparison

Figure 4.13 and Table 4.1 show the cell densities of the SRAM-ROM cells in terms of bits per Λ. The cell layout and area measurement was done in use a mask layout editor (Cadence). These area measurements were entered into the density equations to produced Figure 4.13 and Table 4.1. The "1 BL Col" cells required multiples of two bitline layers (as the SRAM-ROM is connecting to a single stack of bitlines, which contains the bitline pairs). In Table 4.1 the first density (left-most side) for each cell type is the density of the SRAM without the ROM. An SRAM-ROM that uses two columns of bitlines requires at least two layers of bitlines. Alternatively, a design using a single column of bitlines requires two bitline layers just to implement the SRAM.

SRAM-ROM Cell	**1 M bPerΛ**	**2 M bPerΛ**	**3 M bPerΛ**	**4 M bPerΛ**	**6 M bPerΛ**	**8 M bPerΛ**	**12 M bPerΛ**	**16 M bPerΛ**
1BL-Col	na	0.25	na	0.50	0.65	0.75	0.90	1.00
1BL-Col Choose 2	na	0.25	na	0.75	1.04	1.25	1.54	1.75
2BL-Col	0.17	0.33	0.43	0.50	0.60	0.67	0.76	0.83
2BL-Col Choose 2	0.17	0.50	0.69	0.83	1.03	1.17	1.36	1.50
Fixed Bl 1BL-Col	na	0.50	1.00	1.29	1.66	1.90	2.23	2.45
Multi WL 1BL-Col	na	na	0.25	0.50	0.75	0.90	1.08	1.20

Table 4.1: SRAM-ROM Cell Density vs. Number of Extra Interconnect Layers.

We have shown that it is possible to have the SRAM connect to different "true" and "complement" bitlines (see "Choose 2" or "1 Bitline Column with One Fixed Bitline" cells in Figure 4.13 and Table 4.1). Connecting to different "true" and "complement" bitlines requires that you read the SRAM to determine which "true" or "complementary" bitline

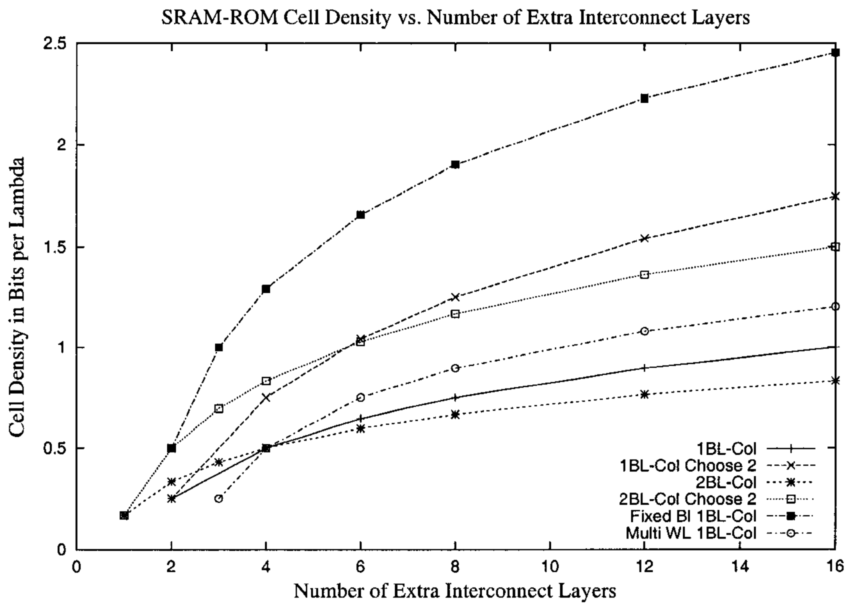

Figure 4.13: Theoretical Cell Density for SRAM-ROMs Versus Number of Bitline and Wordline Layers.

the information comes out on, then write-read the opposite value back to the SRAM to determine the other bitline that the information will come out on. Having to read twice and write once obviously decreases the ROM reading speed, but being able to connect to any “true” and “complement” bitline increases the density of the ROM. The effect on the SRAM read speed would be negligible. In regards to the actual increase in density of the ROM, the equation for the two column bitlines cells would be transformed from:

$$BitsPer\Lambda = (log_2(n) + 1)/const\Lambda \tag{4.10}$$

to

$$BitsPer\Lambda = (log_2(n) + log_2(n) + 1)/const\Lambda \tag{4.11}$$

where n is the number of bitline layers available. As we can see, the information stored in the ROM ($log_2(n)$) doubles when we allow the SRAM to connect to any “true” or “complement" bitline.

4.5 Summary

Given the size of a pre-built SRAM cell, an appropriate ROM layout can be chosen to fit above it, thus having little impact on the size of the SRAM cell, while adding ROM storage to it. For those designs that require both SRAM and ROM, the hybrid SRAM-ROM that we proposed can offer significant savings in terms of area as no extra wordline decoding, drivers or ROM cells (extra transistors) are required. However, the sense-amplifier design would have to be changed and extra column decoding circuitry would be needed for the ROM.

The two types of multiple bitlines SRAM-ROMs presented offer varying access times. The higher density SRAM-ROM which requires three operations, the initial read, a write back and then the final read. Density can be traded for speed, where the reading of the SRAM also reads the ROM. The multiple wordline SRAM-ROM is by far the slowest when it comes to reading the ROM, requiring the sequential assertion of every wordline within the column of wordlines. In general the SRAM access time in an SRAM-ROM design would be slower than an ordinary SRAM due to the increase in parasitic capacitance between the stacked layers.

Chapter 5

Putting the Array Together

In this chapter we will discuss the effect of the hybrid multi-layered cells on the density of the array and their impact on robustness. When computing the overall area of a ROM, a number of peripheral circuits need to be taken into account. Wordline decoders, wordline drivers, sense-amplifiers, bitline pre-charge and the data-bus interface are the primary peripheral circuits. In this chapter, we will also present conversion circuitry that could be used to convert a non-power of two symbols set (where a set of symbols is defined as an alphabet) to a binary representation.

5.1 Robustness

In DRAM, the most important factor is density. ROMs, on the other hand, are often used to look up information while performing arithmetic operations (as is the case in a lookup table). As such, many ROMs are required to have fast access times. While density remains important, bandwidth is also important (bandwidth being defined as the amount of information that can be transmitted over a given period of time). Cell architecture plays only a small role in the overall speed of the design. Specifically, cell architecture has the greatest impact on the bitline capacitance. Even the bitline capacitance, however, is not as critical as it once was in memory designs [16]. Other factors, such as the sense-amplifier architecture and column decoding also play a large role. Illustrating this point are current sense-amplifiers, where the bitline capacitance does not greatly affect the sensing speed [16] (assuming a relatively low bitline resistance). The multi-layer design of the hybrid cells, presented in the previous chapter, increases the parasitic capacitance between bitlines. While the added capacitance may not appreciably slow sensing, the bitline coupling due to parasitic capacitance may make it harder to achieve sufficient robustness, in terms of sensing the bitlines.

Designers tend to avoid routing two signals overtop of each other over large distances. The capacitive coupling between them can cause a false signal to appear on one of the lines when the other is switching. In the case of stacking bitlines we also must consider this effect.

Active BL	**BL M1 % VDD**	**BL M2 % VDD**	**BL M3 % VDD**	**BL M4 % VDD**	**BL M5 % VDD**
BL M1	0	20	40	60	80
BL M2	50	0	25	50	75
BL M3	67	33	0	33	67
BL M4	75	50	25	0	50
BL M5	80	60	40	20	0

Table 5.1: Worst-Case Bitline Capacitive Coupling.

The worst-case model for parasitic bitline coupling is where the only parasitic capacitance is from adjacent bitlines and the bitlines are electrically isolated (i.e. floating after pre-charge). After the bitlines are pre-charged to V_{DD}, the net effect of one bitline transitioning from V_{DD} to ground would cause adjacent bitlines to drop from their V_{DD} pre-charged state according to Table 5.1 (derived from Spice simulations). From this table the worst case arises when the bitline on metal2 is pulled-low; the bitline on metal1 would drop to 20% of V_{DD}, instead of remaining at V_{DD}. As a result, this parasitic capacitive coupling would cause severe problems for voltage sense-amplifiers. On the other hand, current sense-amplifiers would only see a momentary spike in current on the adjacent bitlines, while the activated bitline would provide a constant current. Additionally, current sense-amplifiers require the bitlines to be connected to active pull-up devices, which in turn diminishes the parasitic capacitive effect (as the active bitline falls).

5.2 Wordline Drivers and Decoders

Wordline decoders take in an address and activate a row of cells. The actual wordline is driven by scaled cascaded inverters. In reference to the hybrid multi-layered cells, the use of multiple bitlines has no effect on the size of the wordline decoders and drivers. As well, multiple bitlines do not significantly change the capacitance of the wordlines.

In the case of the multiple wordline architectures, the parasitic capacitance between stacked wordlines is large. Such parasitic capacitance can cause the access transistors of cells attached to other wordlines to momentarily turn on if the selected wordline rises too quickly, effectively generating a false signal on one or more bitlines for a short period of time. While multiple-wordline designs suffer from increased parasitic capacitance, they benefit from a reduced number of gate capacitances on each wordline (on average the number of gates attached to the wordline would be reduced by the factor of the number of wordlines). The net capacitive effect depends on the process technology (on the dielectric constants of the materials and the proximity of the wordline layers in both the vertical and horizontal directions).

In summary, most of the architectures presented in the previous chapters (with the exception of the multiple wordline cells) should not require significant changes in the area of the wordline drivers or decoders.

5.3 Column Decoders and Sense-Amplifiers

Column decoders multiplex the accessed data width from the number of cells in a row down to a reasonable word size. Column decoders can be placed before the sense-amplifiers, after the sense-amplifiers but before the data-bus, or in both locations. The tight pitch of the bitlines in these architectures results in the need to multiplex the bitlines before the sense-amplifiers. While multiplexing the bitlines can save sense-amplifier area, multiplexing through the use of pass-transistors before the sense-amplifiers can introduce resistance on a bitline, impacting the effectiveness of the current sense-amplifiers [17][18]. Normally very large pass transistors are used in an attempt to minimize the resistance. Wicht et al. [18] presents a method by which the effective resistance of the pass-transistor becomes negligible, thus removing this concern (for a relatively small increase in area).

In a ROM, a sense-amplifier converts an analog signal to a digital "one" or "zero" (or in our case, multiple "ones" and "zeros"). No modification to the sense-amplifier design is required (expect to possibly compensate for changes in the bitline capacitance). One might conclude that using multiple bitlines results in the use of more sense-amplifiers. By pre-decoding the columns by a greater factor than would normally be done, one can eliminate the need for those extra sense-amplifiers. This would minimize the required extra area. However, care must be take to minimize the amount of resistance introduced between the bitlines and sense-amplifiers, as the added resistance decreases the effectiveness of current sense-amplifiers [17][18].

The leakage from inactive cells is expected to increase with the decrease in transistor sizes [1]. Such an effect would limit the number of cells within a bitline, and in-turn reduce density. Agawa et al. [1], however, presents a technique whereby the negated bitline leakage current is injected onto the bitline before sensing to compensate for the leakage. The result is a bitline that appears to have no leakage in terms of sensing (i.e. leakage current will not upset the sense-amplifier).

5.4 Conversion from Multiple Values to Bits

In the case of the hybrid multi-layer cells, further decoding of the sense-amplifier data is required. The output of the sense-amplifiers can be viewed as a selection of symbols, from a set of symbols (alphabet), which needs to be converted to a binary code. This circuitry may reside within the sense-amplifiers, data-bus circuity, or may be external to the memory block.

This section covers the basics of converting a set of symbols to binary representation. First we will deal with converting one of n symbols to binary, then deal with converting x of n symbols to binary. By "symbols" we mean identifiable conditions. For example, on the bitlines we literally have an infinite number of voltages that could be present. But if we can only classify the voltages into two ranges of voltages, then we only have two symbols. A set of symbols is an alphabet.

The formula that we have been working with to determine the number of bits given n possible symbols (or states) is:

$$Bits = log_2(n) \tag{5.1}$$

where n is the number of of possible values that can be represented (including no value). Log base two of n is the maximum possible number of bits that can be recovered from n symbols. Implementing this generic formula in circuitry can be difficult (as a fractional number of bits is typically undesirable). On the other hand, converting an alphabet that totals a power of two symbols is a simple matter (as long as the total number of symbols in the alphabet does not get too large). For those alphabets that are not a power of two in size, we can either drop a number of symbols so that the alphabet size totals a power of two or we can combine alphabets to obtain a new alphabet that comes closer to a power of two, thus reducing the relative amount of symbols we have to drop. A third option is to convert an alphabet to binary, with the fore-knowledge that the binary code will not be able to represent all bits. In this section, we will focus on converting alphabets to full binary (where every binary combination of "ones" and "zeros" can be represented by the ROM cells).

For example, to convert an alphabet of three symbols to binary; one possibility is to use two of the symbols and convert that to one bit. By only using two of the symbols, this results in a loss of one third of the potential information. On the other hand, by combining two alphabets of three symbols we can reduce the potential information lost to one ninth (information lost is, 3*3-8). To accomplish this we combine the alphabets of symbols to form one large alphabet of symbols. Once done, this large alphabet can be used to generate the binary representation.

Table 5.2 shows how two alphabets of three symbols *A* and *B* are used to create a new alphabet *V* which will be used to generate a 3-bit value. Note, it is assumed that one and only one symbol will be selected from each alphabet.

Symbol Set A	Symbol Set B	Combined Symbol Set V	Bits
A0	B0	V0	000
A0	B1	V1	001
A0	B2	V2	010
A1	B0	V3	011
A1	B1	V4	100
A1	B2	V5	101
A2	B0	V6	110
A2	B1	V7	111
A2	B2	V8	000

Table 5.2: Converting Two Sets of Three Symbols to One Set of Nine Symbols.

In Figure 5.1 we are combining two alphabets (*A* and *B*) using AND gates to form a combined alphabet (*V*). Figure 5.2 uses OR gates to convert seven of 9 symbols to a 3-bit value.

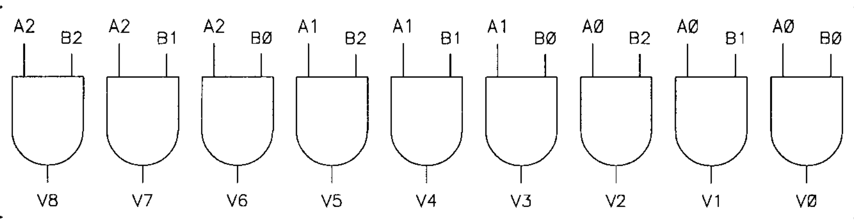

Figure 5.1: Combining Two Alphabets to Form a New Alphabet.

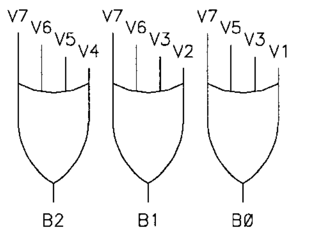

Figure 5.2: Converting an Alphabet to Binary.

In a previous chapter we introduced cells with a symbol set that includes a combination of two, one or no symbols in an alphabet. The formula for the number of bits is given by:

$$Bits = log_2(C(n,2) + C(n,1) + C(n,0)) \tag{5.2}$$

where n is the number of symbols (or states). Here we can choose one, none or two of the symbols. One method to convert to binary is to convert the possible combinations of symbols to a new alphabet, then drop any extra symbols such that the total number of symbols is equal to a power of two. For example, in Table 5.3 we show how a set of five symbols, where more than one can be selected at a time, is converted to a 4-bit binary representation. In this case, the total number of values works out to be a power of two. Figures 5.3 and 5.4 show the functional circuitry required to convert "five choose two", "five choose one" and "five choose none" to binary. Note, the case where only one or none of the alphabet is selected (V0-V5) requires us of verify that no other symbols are selected (five-input AND gates). Where two symbols in the alphabet are selected, the logic is much simpler (two-input AND gates). Once a single large alphabet is created (V0-V15) we can convert it to binary (Figure 5.4).

These circuit diagrams are for functional purposes and not indicative of actual circuitry that would be used. They are useful to review to get a sense of the amount of additional circuitry required to convert complex combinations of symbol sets to a binary value. The two methods presented above can be used in conjuction to form additional alphabets of

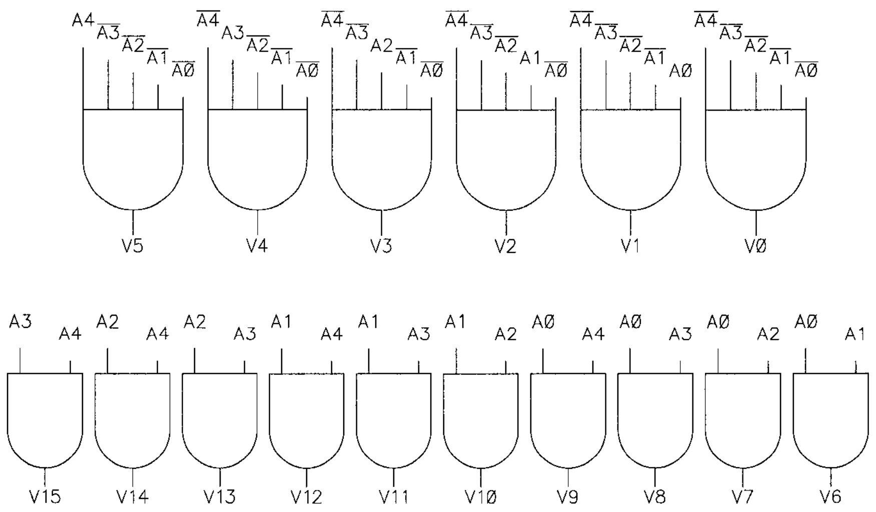

Figure 5.3: Converting Two, One or None of Five Symbols to a New Alphabet of Sixteen Symbols.

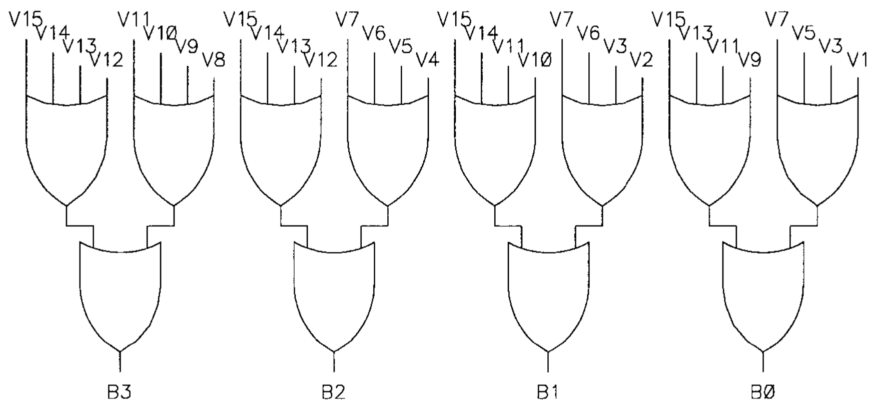

Figure 5.4: Converting an Alphabet of Sixteen Symbols to Binary.

Second Selection	First Selection	Combined Symbol Set V	Bits
		V0	0000
	A0	V1	0001
	A1	V2	0010
	A2	V3	0011
	A3	V4	0100
	A4	V5	0101
A0	A1	V6	0110
A0	A2	V7	0111
A0	A3	V8	1000
A0	A4	V9	1001
A1	A2	V10	1010
A1	A3	V11	1011
A1	A4	V12	1100
A2	A3	V13	1101
A2	A4	V14	1110
A3	A4	V15	1111

Table 5.3: Converting a Five-Symbol Set where Multiple Symbols can be Selected to Form One Set of Sixteen Symbols and their Binary Representation.

symbols. This allows us to transform the output of any hybrid multi-layered ROM cell or combination of cells into a binary value.

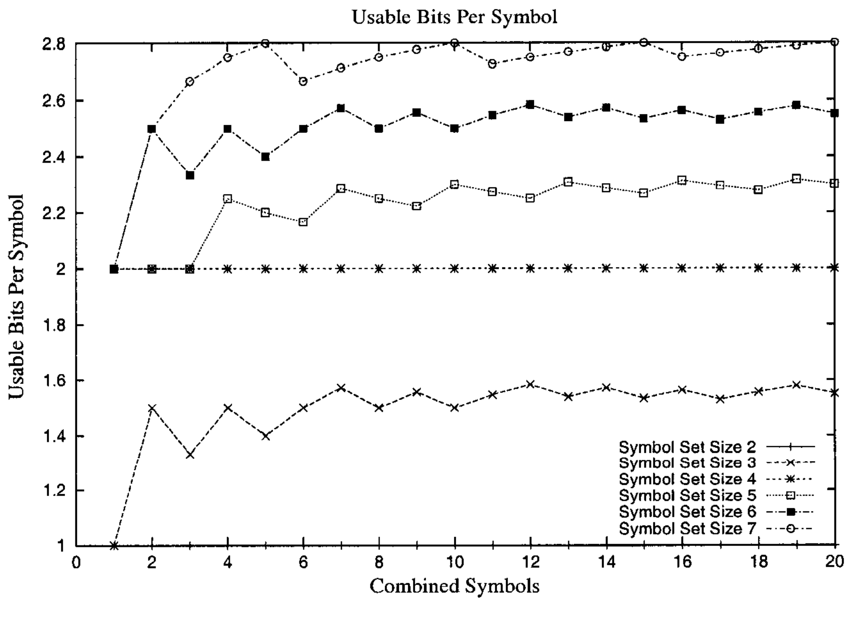

Figure 5.5: Usable Bits Per Symbol.

Figure 5.5 shows the usable number of bits that can be achieved by combining alphabets.

5.5 Summary

Overall, the areas effected the most by the hybrid multi-layered cells are the column decoders and sense-amplifiers. Converting non-powers of two alphabets to binary representation requires the combination of two or more alphabets to form a unified alphabet that equals or slightly exceeds a power of two. Additional circuitry, in the column decoders or at the databus interface, is required to convert the alphabets generated by these cells to binary values. Wordline decoders and drivers remain unaffected with the exception of the multiple wordline cells.

Chapter 6

Test Chip

We designed a test chip in the early stages of this thesis research. Since then many of the ideas have evolved and matured. The test chip, however, does include the basic multi-layer ROM idea. The chip consists of a more than 1.3 million transistors in the memory arrays and stores just under 2-Mbits, in a die size of 3-mm by 3-mm (Figure 6.1).

The hybrid cell architecture used in the test chip consists of one NOR cell and one multi-layer cell. The cell transistor's source connections are shared with the lower cell, however, the drain connections are not shared with the upper cell (a similar cell with a shared drain connection was presented as the Hybrid 3 NOR to 1 Multi-Layer ROM Cell in Chapter 3). The implemented cell architecture utilizes four metal layers for the bitlines and results in a density of of 1-bit per Λ. Each cell stores 3-bits within a 3Λ area.

Two errors in the layout were found after manufacturing. The first error prevents access to the lower half of the wordlines in the fourth bank of memory. The second and more serious error shorted the input of an inverter to its output in the third variant of the sense-amplifier. Schematic simulations have shown that the maximum amount of static current consumed due to this error to be 14 mA for the chip.

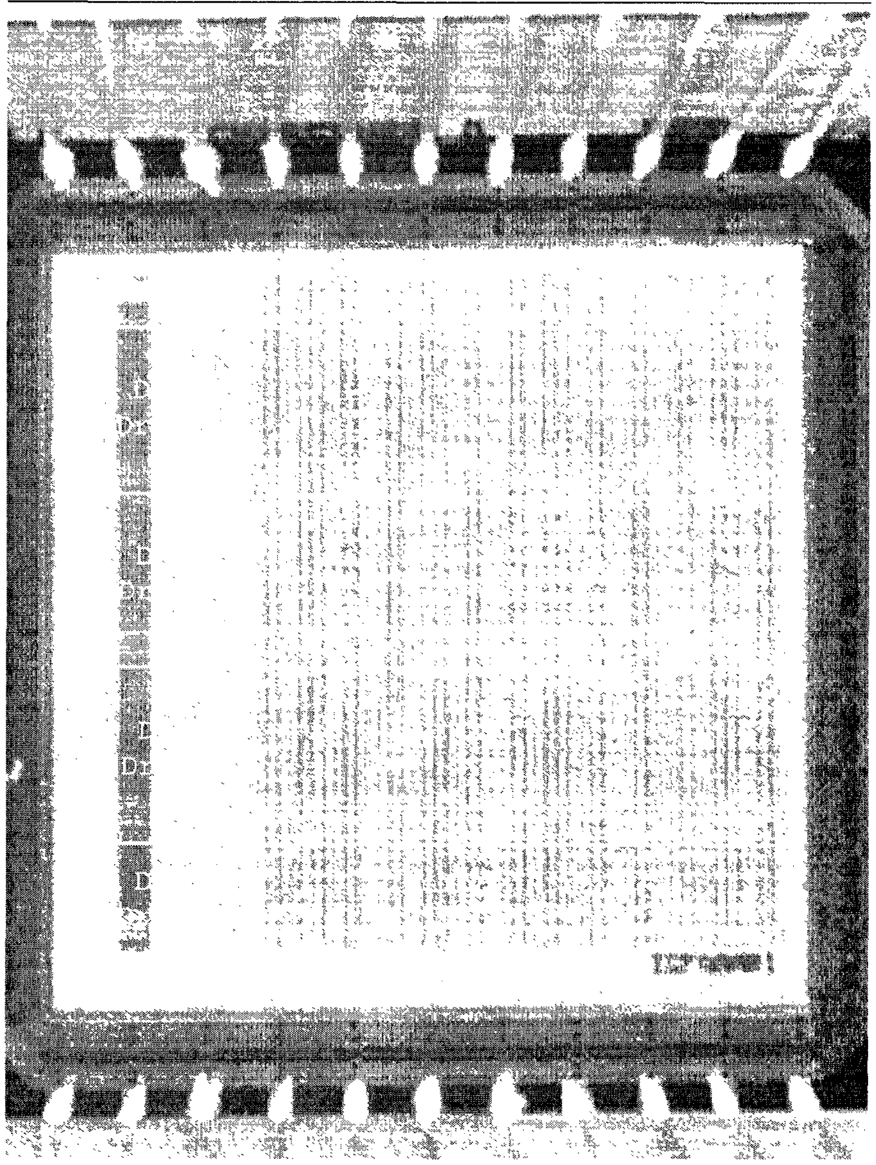

Figure 6.1: Die Photo.

6.1 ROM Cell

There are four banks of memory in the design. The first three contain 256 rows and the last bank contains 128 rows. Each bank has 768 columns, with four bitlines within each column.

The first three banks are broken down into six sections of 128 columns. Each section is further broken down into arrays of sixteen rows by eight columns. If one were to draw a diagonal line across one of the six sections, from the upper left-hand corner to the bottom right, those arrays below the line would contain ROM cells, but they would not be connected to any of the bitlines. The ROM cells above the virtual-line would be connected to the bitlines. The presence of few ROM cells connected to the bitlines on the left-hand side of the array reduces bitline loading (capacitance) compared to the right-hand side of the array. It was our intention to characterize the ROM speed relative to the amount of bitline loading (capacitance). Above the line the cells contain a connection pattern that tests all the possible cell configurations (as seen in Figure 6.2).

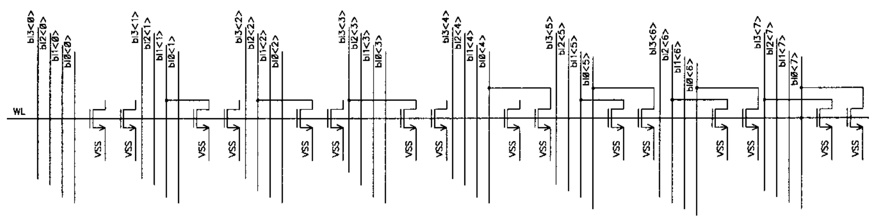

Figure 6.2: Eight Columns by One Row of ROM cells (Stores 24-bits of Information).

The fourth bank of memory is broken down into twelve sections of 64 columns. Each section is further broken down into arrays of sixteen rows by eight columns. If one were to draw a diagonal line across one of the twelve sections, from the upper left hand corner to the bottom right, those arrays below the line would contain ROM cells with no connections to the bitlines. Above the line the cells contain a connection pattern that tests all the possible cell configurations (as seen in Figure 6.2).

6.2 Sense-Amplifiers

Three sense-amplifier designs were chosen to provide a variety of choices of speed versus robustness. The first is a differential amplifier with no feedback. The second is a pseudo-NMOS gated inverter. Finally, the third design is a standard CMOS inverter (Figure 6.3). This design draws static current through the differential amplifier and the pseudo-NMOS inverter (when the *eval* signal is high).

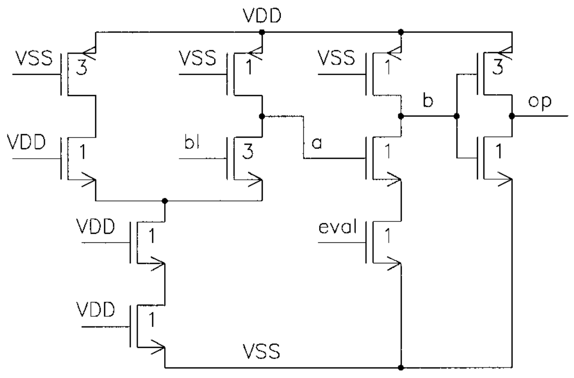

Figure 6.3: First Sense-Amplifier Design.

Figure 6.4 shows the switching point for the first sense-amplifier at a value of 1.48 V.

The second sense-amplifier is a basically an inverter with skewed sizes for the NMOS and PMOS transistors such that the switching point is moved closer to the power supply (V_{DD}). This inverter is followed by another inverter that can only output a zero when *eval* is high. This ensures that the output of the sense-amplifier is zero when *eval* is low (as the bitlines are pre-charged to V_{DD} when *eval* is low). Figure 6.6 shows the switching point for the second sense-amplifier at a value of 0.99 V.

Figure 6.7 shows the third sense-amplifier which contains a differential amplifier with an unbalanced current mirror. The output of the differential amplifier is followed by an inverter which can only output (node *b*) a low when *eval* is low. Node *b* is pre-charged when *eval* is low. Thus the output of this sense-amplifier is zero when pre-charged. Figure 6.8 shows the switching point for the third sense-amplifier at a value of 1.23 V.

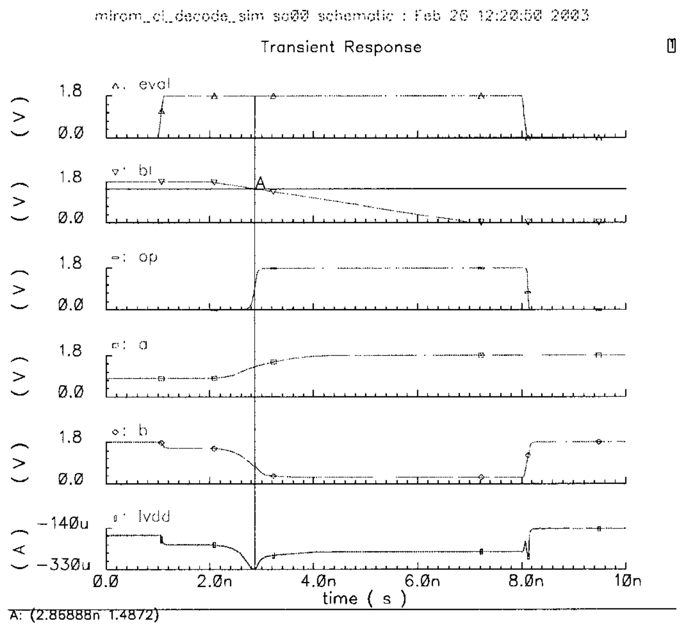

Figure 6.4: First Sense-Amplifier Switching Point Simulation.

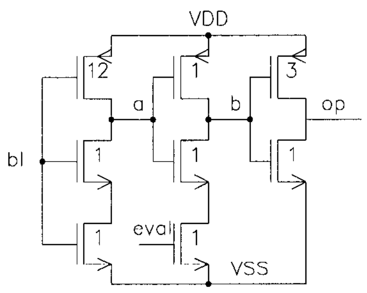

Figure 6.5: Second Sense-Amplifier Design.

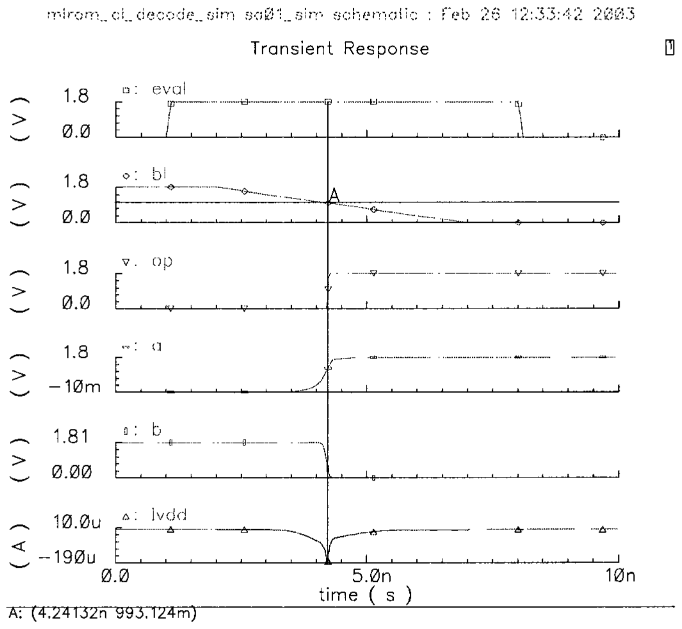

Figure 6.6: Second Sense-Amplifier Switching Point Simulation.

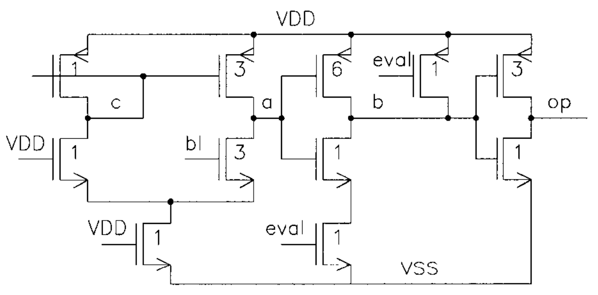

Figure 6.7: Third Sense-Amplifier Design.

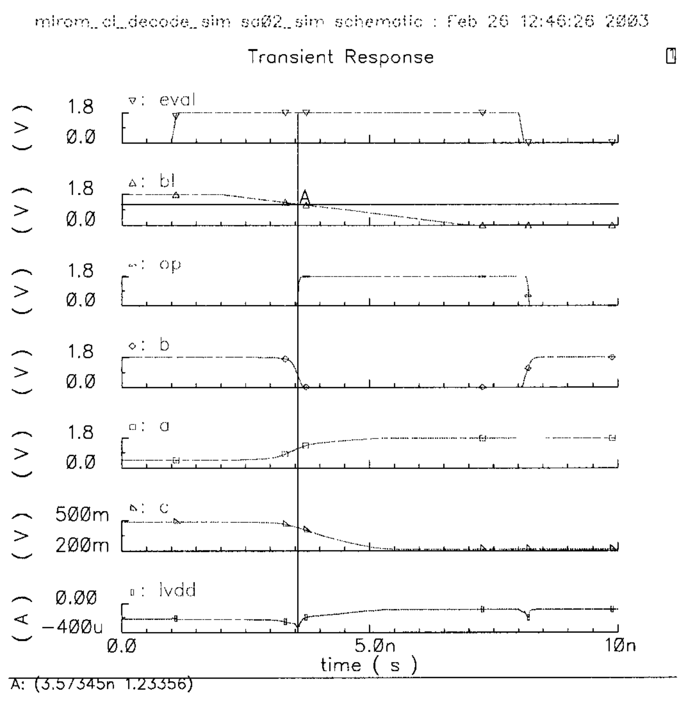

Figure 6.8: Third Sense-Amplifier Switching Point Simulation.

An error in the third sense-amplifier has *b* connected to *op*. In addition to corrupting the output, this error also causes the sense-amplifier power level to drop to 1.4 V and the ground potential to rise to 0.38 V. Sense-amplifier power and ground are shared for all sense-amplifiers in a bank of memory. Thus the error unfortunately affects all of the sense-amplifiers in the test chip.

6.3 Wordline Decode

For wordline decoding we used a dynamic AND gate with an additional enable circuit on its output. For the 256 wordline banks there were eight pull-down transistors in series. To avoid possible charge sharing, the decoder address then remains stable during pre-charge. There is no footer enable transistor on the dynamic AND gate, so one of the 256 dynamic gates will consume static power during pre-charge. Because of the large number of series transistors within these dynamic gates, the static current consumption remains small (64 μA for each of the four banks of memory). Figure 6.9 shows one row of the dynamic 8-bit decoder used in the test chip. The least significant bits of the wordline address access the transistors closest to the output.

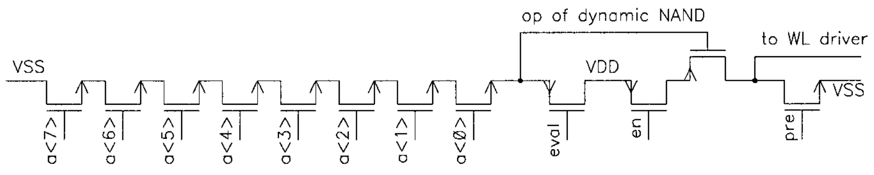

Figure 6.9: Dynamic 8-bit Decoder.

Note, *en* is generated by NANDing the memory bank enable with the clock and *pre* is the inverted clock.

6.4 Wordline Drivers

In order to pitch-match the wordline drivers to the ROM cells, the wordline drivers were staggered. Each driver consists of four cascaded inverters with a scaling factor close to four (shown in Figure 6.10).

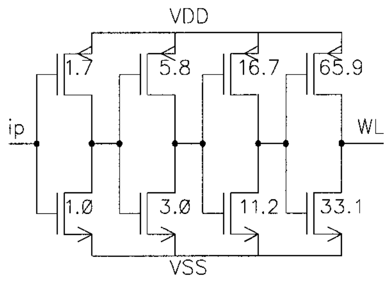

Figure 6.10: Wordline Driver Design.

From Figure 6.11, the propagation time for the wordline driver from zero to one is 230 ps and from one to zero is 280 ps. The rise time is 170 ps and the fall time is 130 ps. The output load is a conservative 2000 minimum-sized transistor gates. Peak current was recorded at 21 mA during the rise of the output. Signals *s1* *s2* and *s3* show the internal nodes of the wordline driver.

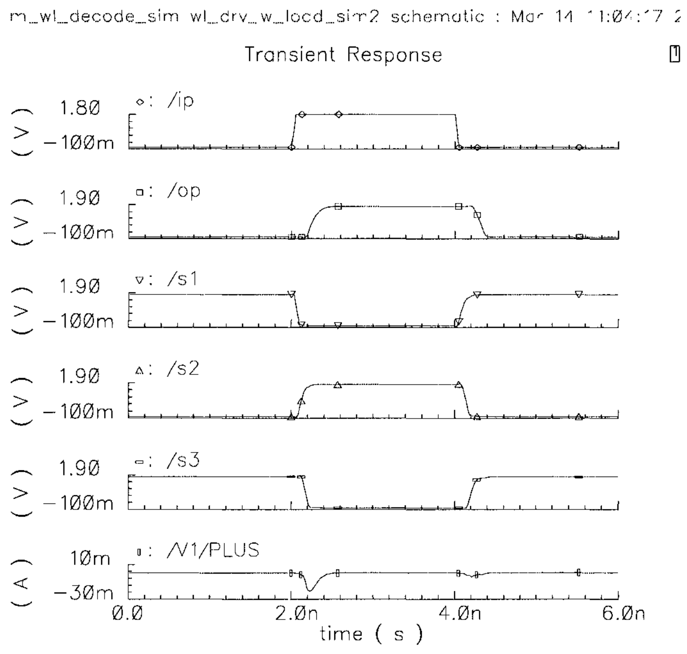

Figure 6.11: Wordline Driver Simulation.

6.5 Address Latch

Early in the design a decision was made to use a pipeline. Later many of the pipeline stages were removed from the design. The last remnants were the positive edge address flip-flops (Figure 6.12). They effectively store wordline address and bitline address (chooses between 64 columns in a subsection) on the falling edge of the chip clock. All other addresses in the design change with the input pins.

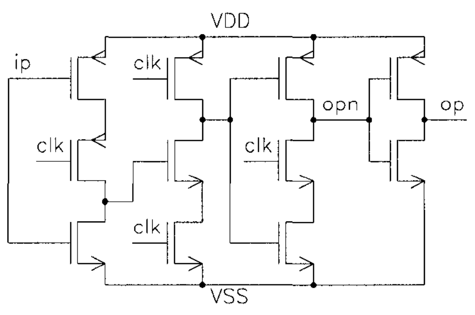

Figure 6.12: Rising Edge Flip-Flop.

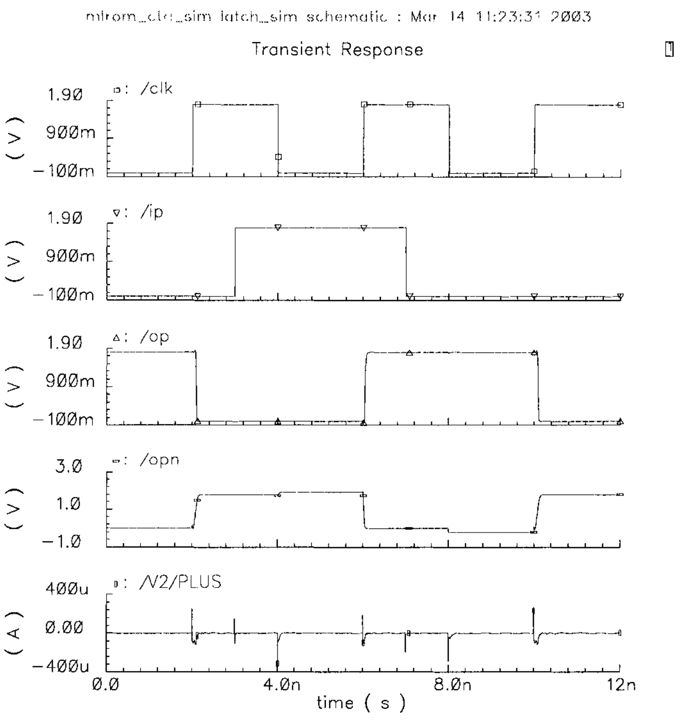

Figure 6.13: Rising Edge Flip-Flop Simulation.

The wordline address is stored in latches within the memory blocks.

6.6 Bitline Decoders

PMOS pass transistors in three stages were used to multiplex the bitlines from sixty-four down to one. Three stages were chosen to balance the control signal routing with the number of transistors, in order to maximize area efficiency. There exists a trade-off between pre-decoding, which creates more routing signals and decoding under the memory array, which creates more transistors in the area under the memory array. The PMOS device was chosen due to its ability to readily pass values near V_{DD}. Due to the extreme density of bitlines it was necessary to multiplex the bitlines before reaching the sense-amplifier.

6.7 Databus

Unlike in a RAM, a ROM's databus only runs one way, out of the memory. The databus is broken down into two levels: the sense-amplifier databus interface and the global databus.

6.7.1 Sense-Amplifier Databus Interface

The sense-amplifiers are connected to a databus that runs overtop the entire chip. There is one set of databus lines for each of the twelve subsections in each bank. Each set of databus lines contain four lines. As seen in Figure 6.14, data coming from the sense-amplifier *d* is ANDed with the data-line enable signal *dl_en*. The result is connected to a pull-down transistor tied to the global databus line.

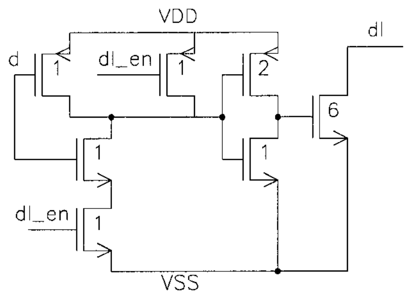

Figure 6.14: Sense-Amplifier Interface to the Global Databus Line.

6.7.2 Global Databus

The databus lines are shared across the memory banks and terminate in the periphery. A PMOS pre-charge transistor connects the databus lines to V_{DD} and is controlled by the clock (active low). The databus lines are connected to a 12:1 multiplexor that outputs four data signals from the chip (dl0, dl1, dl2 and dl3).

6.8 Timing Waveforms

The operation of the test chip is shown in Figure 6.15. When the clock (*clk*) is low the chip is in pre-charge mode; when the clock is high it is in evaluate mode. The address of the desired wordline and column is input while the clock is high. When the clock goes low these addresses are latched inside the memory banks. The databus (*addr_db*) and bank (*addr_bank*) address can be changed anytime before the clock goes high as these addresses are broadcast across the test chip without delay.

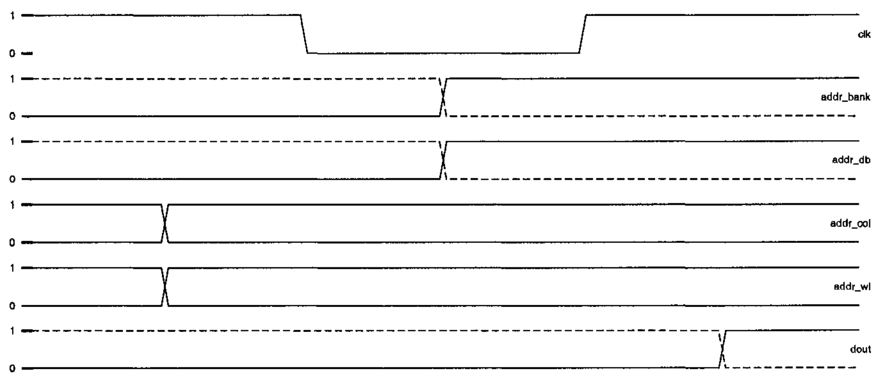

Figure 6.15: Test Chip Timing Diagram.

6.9 Summary

Due to the errors in the design of the test chip the test chip does not function as expected. The test chip was not completed until shortly before it was sent for manufacturing and the design errors were not caught in time. There was insufficient time to submit a repaired design and wait three to four months for a new set of prototypes. Given the opportunity to redesign a chip, these are the techniques that would benefit the design. Starting off, the sense-amplifiers should be replaced with current sense-amplifiers. Current sense-amplifiers can easily differentiate between a high impedance node and an actively pulled down node (regardless of the voltages on the nodes). The number of cells in a column should be increased to reduce the relative area overhead of the sensing and column decoding circuitry.

Finally, a more robust design for test scheme should be implemented with a pin-to-pin functional simulation being run before manufacturing resources are requested.

Chapter 7

Conclusion

With advancements in process technology, many more interconnect metal layers have been made available to the designer and the number is expected to continue to increase in the foreseeable future. Using these extra metal layers along with various hybrid base cell types, we have proposed a new group of higher-density ROM cell architectures. These new ROM cell architectures improve the data density by 30 to 70 percent in the cell array over a standard NOR ROM cell (assuming a six metal layer process), thus reducing silicon area and cost. In addition, these cell architectures require no modification to the manufacturing process. As well, we have shown how to integrate our design techniques with SRAM to construct SRAM-ROMs that have increased density characteristics over conventional SRAM and ROM arrays. The SRAM-ROMs offer designers the unique ability to use the same cell area to store both changing and permanent data. Nearly all personal computer microprocessors make use of on-chip SRAM caches and ROM-based lookup tables. With the variety of SRAM-ROM architectures that we have presented, we believe an SRAM-ROM can be chosen that improves density of the microprocessor design. A disadvantage of the hybrid multi-layer techniques is the extra circuitry required to decode non-powers of two symbol sets and the required multiplexing of the columns before reaching the sense-amplifiers. With the continuing push to add more metal layers in each new generation of manufacturing processes, the Hybrid Multi-layer ROM and SRAM-ROMs will continue to achieve higher densities than those offered by standard ROMs and SRAMs. The density advantages of the new architectures are supported by hand-compacted layouts and by the "Λ" process-independent analysis method that was also introduced in this thesis.

Future work can focus on the refinement of symbol-to-binary decoding of the multiple bitlines to reduce the amount of extra circuitry required. In addition, sense-amplifiers should be investigated that are geared towards current sensing where the bitlines are highly capacitively coupled. As well, techniques are required that preserve signal integrity while reducing the area of the pre-sense-amplifier column decoding circuitry. These components would then need to be combined in a test chip that could be used to characterize in silicon the speed and power consumption of the multi-layer hybrid cells.

Bibliography

[1] K. Agawa, H. Hiroyuki, T. Takayangi, and T. Kuroda. A Bitline Leakage Compensation Scheme for Low-Voltage SRAMs. *IEEE Journal of Solid-State Circuits*, 36(5):726–34, May 2001.

[2] George M. Ansel, Jeffery S. Hunt, Satish Saripella, Sudhaker Reddy Anumula, and Ajay Srikrishna. Read Only-Random Access Memory Architecture and Methods for Operating Same. Technical Report 5,880,999, US Patent Office, 1999.

[3] Semiconductor Industry Association. International Technology Roadmap for Semiconductors. Technical report, Semiconductor Industry Association, 2001.

[4] C. Bleiker and H. Melchior. A Four-State EEPROM Using Floating-Gate Memory Cells. *IEEE Journal of Solid-State Circuits*, 22(3):460–3, June 1987.

[5] William D. Brown and Joe E. Brewer. *Nonvolatile Semiconductor Memory Technology*. IEEE Press, 1st edition, 1998.

[6] Ran Dvir. Read-only-memory (ROM) having a Memory Cell that Stores a Plurality of Bits of Information. Technical Report 6,002,607, US Patent Office, 1999.

[7] Spencer M. Gold and Marc Lamere. Combining RAM and ROM into a Single Memory Array. Technical Report 6,438,024, US Patent Office, 2002.

[8] D.L. Kencke, R. Richart, Shyam Garg, and S.K. Banerjee. A multilevel approach toward quadrupling the density of flash memory. *IEEE Electron Device Letters*, 19(3):86–8, March 1998.

[9] IC Knowledge. History of the IC. http://www.icknowledge.com/history/history.html, 2001.

[10] Robert D. Lee. RAM-ROM Hybrid Memory Architecture. Technical Report 5,581,505, US Patent Office, 1996.

[11] Kenneth W. Marr. Method and apparatus for embedded read only memory in static random access memory. Technical Report 6,041,008, US Patent Office, 2000.

[12] Tetsuya Matsumura and Masahiko Yoshimoto. Semiconductor Memory Device Usable as Static Type Memory and Read-Only Memory and Operating Method Therefor. Technical Report 5,365,475, US Patent Office, 1994.

[13] Carver Mead and Lynn Conway. *Introduction to VLSI Systems*. Addison-Wesley, 1st edition, 1980.

[14] MOSIS Org. MOSIS Scalable CMOS Design Rules. http://www.mosis.org/Technical/Designrules/scmos/scmos-main.html, 2003.

[15] Min-Young You; Jong-Hoon Park. Hybrid Memory Device. Technical Report 6,128,218, US Patent Office, 2000.

[16] E. Seevinck, P.J. van Beers, and H. Ontrop. Current-Mode Techniques for High-Speed VLSI Circuits with Application to Current Sense Amplifier for CMOS SRAM's. *IEEE Journal of Solid-State Circuits*, 26(4):525–536, April 1991.

[17] N. Shibata, H. Inokawa, Tokunaga k., and S. Ohta. Megabit-Class Size-Configurable 250-MHz SRAM Macrocells with a Squashed-Memory-Cell Architecture. *IEICE Trans. Electron.*, E82-C(1):94–104, January 1999.

[18] B. Wicht, S. Paul, and D. Schmitt-Landsiedel. Analysis and Compensation of the Bitline Multiplexer in SRAM Current Sense Amplifiers. *IEEE Journal of Solid-State Circuits*, 36(11):1745–55, November 2001.