# ENERGY REDUCTION TECHNIQUES TO INCREASE BATTERY LIFE FOR ELECTRONIC SENSOR NODES

by

**John K. Lynch**

**A Dissertation**

*Submitted to the Faculty of Purdue University*

*In Partial Fulfillment of the Requirements for the degree of*

**Doctor of Philosophy**


School of Electrical & Computer Engineering

West Lafayette, Indiana

December 2017

# THE PURDUE UNIVERSITY GRADUATE SCHOOL
# STATEMENT OF COMMITTEE APPROVAL

Dr. Saeed Mohammadi, Chair
School of Electrical and Computer Engineering
Dr. Kaushik Roy
School of Electrical and Computer Engineering
Dr. Byunghoo Jung
School of Electrical and Computer Engineering
Dr. Eugenio Culurciello
School of Electrical and Computer Engineering

**Approved by:**
Dr. Venkataramanan Balakrishnan
Head of the Graduate Program

*This thesis is dedicated to my wife, my son and daughter-in-law, and my four grandchildren.*

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABSTRACT

Author: Lynch, John, K. PhD
Institution: Purdue University
Degree Received: December 2017
Title: Energy Reduction Techniques to Increase Battery Life for Electronic Sensor Nodes
Committee Chair: Saeed Mohammadi

Preserving battery life in duty-cycled sensor nodes requires minimizing energy use in the active region. Lowering the power supply of CMOS gates down into sub-threshold mode is a good way to decrease energy. In this work, a unique technique to control the current in CMOS gates to reliably operate them in sub-threshold mode is described. Compared to the current state-of-the-art for running digital gates in the sub-threshold regime, this work is often superior in its lack of complexity and in reduced variance in delay caused by process variations. In addition to presenting the design considerations, a demonstration of a complete digital design flow is given using the custom gates. An AES128 encryption/decryption engine is designed using the aforementioned digital flow in a commercial 180nm process. The resulting design has a ratio of maximum to minimum frequency variation over corners of only 50% with a 0.3V power supply where the same ratio with standard CMOS gates biased under the same supply voltage is 5600%. In addition, the custom gates are used to design a Wallace tree multiplier in an SOI 45nm process that is fully functional with an optimum energy power supply level of 0.34V with a typical operating frequency of 8 MHz having a variation over corners of 80%.

For a proof of concept memory chip designed in this work, the architecture uses a logic-compatible CMOS process particularly suitable for embedded applications. The differential pair construct causes the read and refresh power to be independent of any process parameter including the within-die threshold voltage. The current stop feature keeps the read voltage transition low to further minimize read power. The bit cell operates in both single bit BASE2 and multi-bit BASE4 modes. An expression for the read signal was verified with bit cell simulations. These simulations also compare the performance impact of threshold voltage variance in the architecture with a standard gain cell. A DRAM bit cell array was fabricated in the XFab 180nm CMOS process. Measured waveforms closely match theoretical results obtained from a system

simulation. The silicon retention time was measured at room temperature and is greater than 150 ms in BASE2 mode and greater than 75 ms in BASE4 mode. 180nm, 25C analysis predicts 0.8uW/Mbit refresh power at 630 MHz, the lowest in the literature. Further: the memory bit cell architecture presented here has a refresh power delay product several times lower than any other published architecture.

The current controlled memory architecture in this work improves or overcomes the drawbacks of the 1T1C and gain cell memory architectures. A current controlled memory design was fabricated as a 131K bit array in an 180nm process to provide silicon proof. The bit cell configuration with shared read and write bit cells gives effectively two memory banks. The grouping of rows together into common source domains allows only two opamps to control the current in all the bit cells across the whole chip. The sense amplifiers have a globally controlled switching threshold point and keep their static power in the nano-amp range. The bit cells can operate either in BASE2 or BASE4 mode and the read bit line transitions are reduced with a current stop construct. Parts were received from the foundry in an 84-pin PLCC and were tested at a number of locations on the die. They proved to be fully functional in BASE4. The silicon retention time was measured at room temperature and was greater than four seconds.

# INTRODUCTION

The goal that is trying to be achieved in this work is to increase battery life in electronic sensor nodes. The sensor is often part of a duty cycled system. An example of such a system is one where the system wakes up, takes some sort of a measurement, performs a calculation, stores the results, and then goes back to sleep. To increase battery life the average power needs to be reduced. For duty cycled systems, this means the energy must be reduced in the active region. For blocks that are always on, the power needs to be reduced. Therefore, duty cycled systems care about saving energy, while for always on blocks it is reducing power that is important.

Figure Intro.1 is a block diagram for a typical duty cycled system. The sensor in the figure is a MEMS pressure sensor. The analog components might be a power management block and a switched capacitor to voltage CDS circuit to interface with the sensor. The sensor interface takes pressure readings throughout the day. The digital signal processing section performs calculations and stores the results in the memory, either SRAM or DRAM. The timer block is likely to be an always on block.

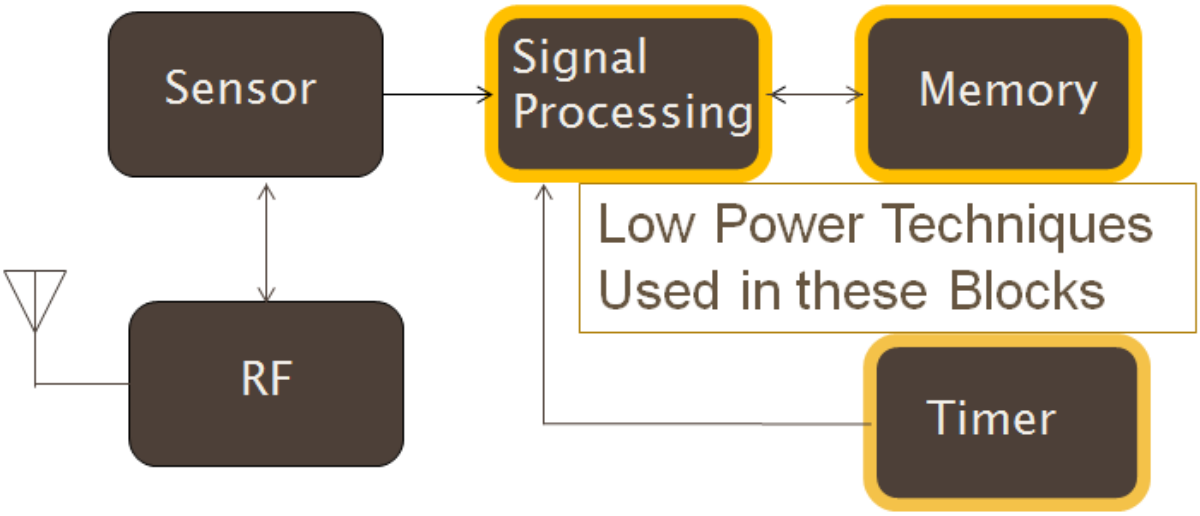


Figure Intro.1 – Typical duty cycled system.

The sensor node might be part of the so-called Internet of Things (IoT). The IoT is a network of interconnected devices. Anything with an RFID can be given an IP address and be connected to the internet. The necessary infrastructure already exists through smart phone networks. The system depicted in Figure Intro.1 is an IOT example. A cardio or glaucoma patient wears the sensing device taking pressure data throughout the day. The device sends data to the doctor's office once per day through the RF communication block through its internet connection. In this way the doctor gets constant feedback from the patient.

One of the biggest dangers to prevent the Internet of Things from being adopted in an even bigger way is for there to be a breakdown in security. For the pressure sensing system in Figure Intro.1, an AES128 system is added for encryption and decryption. AES stands for Advanced Encryption Standard. AES was chosen in 2001 to replace the original standard.

The low power techniques that are described in this work are for the digital sections: the signal processing and the memory. For CMOS logic, the supply is reduced to run reliably in sub-threshold or near-threshold mode, thereby reducing energy. For memory, the energy is also greatly reduced and the memory capacity is increased.

# CHAPTER 1. OBTAINING OPTIMAL ENERGY CONSUMPTION IN SUB-THRESHOLD MODE USING CURRENT CONTROLLED CMOS LOGIC

## 1.1 Introduction

There is increasing interest in improving the battery life for electronic sensor nodes in duty cycled systems and as such, there is a need for creative thinking that will result in large energy reductions across a wide range of chip designs including the digital interface to the sensor.

To preserve battery life in duty cycled systems it is important to minimize the energy consumption during the active mode as is depicted in Figure 1.1.

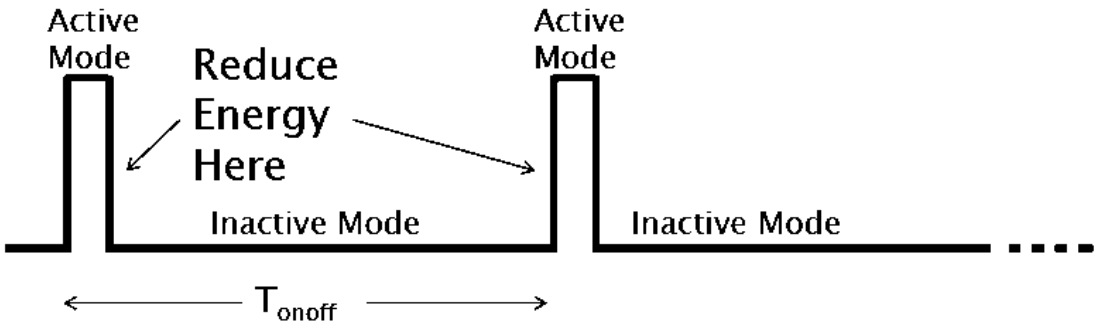


Figure 1.1 To preserve battery life, reduce energy in the active mode.

It is well known [1] that lowering the power supply, *vdd*, is an excellent way to reduce energy in CMOS logic gates as the energy varies proportionally to the square of *vdd*.

Once *vdd* is reduced below the absolute value of the threshold voltages of pmos and nmos transistors, digital gates start operating in the sub-threshold region. In this region, the sensitivity of important design parameters relative to variations in the process greatly increases. A lot of research has been published in the area of running logic in the sub-threshold region. These techniques are mostly focused on either adapting the clock frequency of the sub-threshold circuit or suppressing the variations in the circuit performance and are compared to the current controlled CMOS (3C) gates in this work.

In [2] the authors dynamically adjust for the operating frequency of standard CMOS gates with a circuit that needs a delay monitor, a comparator, a clock, and a shift register. The implementation in [2] has much more complexity than the equivalent functionality accomplished in this work, thereby increasing the power and area.

In [3] the authors try to remove variation in the gate operating frequency by adjusting the sizes of the transistors in the standard gates. They also make the intrinsic load equal to at least 16fF and specify the input rise/fall time to be 2% of the input switching period. While sizing the gates in this way does reduce the variance in delay for a given operating point, the frequency of the gates over the model corners with *vdd* at 0.4V DC still varies by over 500% [3]. Because of the large 16fF intrinsic load, the area is large, the power is increased and the speed is decreased compared to the 3C logic in this work.

Approaches discussed in [4] [5] use an adaptive body biasing feedback scheme using both the n and p transistor tubs to control the statistical variations in the imbalance factor of the p-strengths versus the n-strengths in the logic gates. For this work, the main difference from their technique for the tub servos is that in this work only the pmos transistor tubs are used leaving the nmos tubs free as a hook to vary the operating frequency, if necessary.

Some researchers have used source coupled current mode logic techniques to run in sub-threshold mode (STSCL) [6] [7]. Logic design using this technique works in the current domain using differential inputs and differential outputs. Each gate has a current source supplying a tail current. Although STSCL implementations break the tradeoff between power and robustness, it comes with a penalty of needing at least double the number of transistors increasing the complexity and power dissipation compared to the gates in this work. Also the minimum achievable *vdd* for STSCL is higher than the 3C logic proposed here.

This paper describes a new logic implementation that very effectively reduces sub-threshold energy use. The implementation is based on current control of CMOS gates. It uses a differential construct to control the logic gate current causing a very large reduction in the standard deviation of the delay and a corresponding reduction in energy use. The reduction in the delay standard

deviation also enables an improved operating frequency, a tighter design specification, and a digital design methodology that includes RTL coding, synthesis and place and route.

It is generally considered best practice to size the p-transistors two or three times larger than the n-transistors in the gates to maintain good symmetry in the output rise and fall times. With the architecture used in this work such a sizing requirement is not needed. The energy is reduced by allowing for the use of p-transistors that are the same size as the n-transistors while maintaining the desired symmetry. Reduced p-transistor size means reduced capacitance and reduced power. Smaller p-transistors also decrease the area and enable the low energy system to operate at higher frequencies.

The implementation also further reduces the energy by adaptive body biasing, but only with the p-channel transistor tubs.

In Section 1.2, design considerations are presented for the current controlled CMOS logic and Section 1.3 explains the advantages of this approach. Section 1.4 discusses a chip implementation for a Wallace Tree multiplier. Section 1.5 describes a digital design flow used with the 3C logic gates. Section 1.6 highlights the performance of a 3C logic implementation of an AES128 encryption/decryption engine. Section 1.7 gives some practical ideas for future research. The conclusion follows in Section 1.8.

## 1.2 Current Controlled Logic

### 1.2.1 Theory of Operation

A modified differential pair (MDP) construct, as depicted in Figure 1.2, is the major building block for controlling the current in CMOS digital gates. The MDP removes [8] the effect of within-die threshold voltage by adding a transistor M0 and an opamp that serve as a current controller for the transistor Mn. Since the within-die threshold voltage of the two neighboring transistors M0 and Mn are effectively equal within a specified tolerance, the voltages cancel each other out and the current in Mn is not a function of within-die threshold voltage.

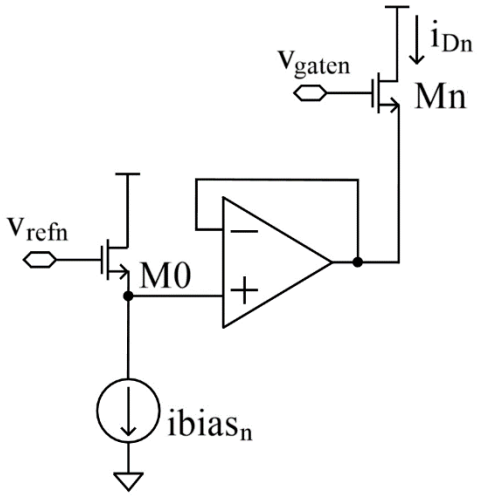


Figure 1.2 Modified Differential Pair Construct

1.2.1.1 Threshold Voltage Cancellation in Sub-Threshold

To understand why the current in Mn is not a function of threshold voltage, consider Kirchhoff's Voltage Law (KVL) around the loop containing the $v_{GS}$*'s* of the two transistors. From [9], if the drain induced barrier lowering (DIBL) effect of *vdd* on the current is not included, in general in sub-threshold mode $v_{GS}$ is defined as

$$v_{GS} = n\varphi_t \ln(i_D) - n\varphi_t \ln(I_{STR}) \quad (1)$$

where *n* is the slope factor, $\varphi_t = kT/q$ is the thermal voltage, $i_D$ is the drain current, and $I_{STR}$ is the sub-threshold current strength defined as

$$I_{STR} = I_0 \frac{W}{L} e^{\frac{-Vth_{eff}}{n\varphi_t}} \quad (2)$$

and $I_0$ is the technology-dependent parameter that equals the current when $v_{GS}$ equals $Vth_{eff}$, the effective threshold voltage including body biasing. *W* is the width and *L* is the length.

By substituting (1) into the KVL equation for the two transistors, the terms containing $I_{STR}$ cancel out and the result is

$$i_{Dn} = ibias_n * e^{\left[\frac{v_{gaten} - v_{refn}}{n\varphi_t}\right]}. \quad (3)$$

The above equation shows that the drain current of Mn is not a function of any threshold voltage. Therefore, the premise is proven that in sub-threshold operation the on-current in the controlled transistor is only a function of the difference between its gate and the reference voltage.

#### 1.2.1.2 Threshold Voltage Cancellation in Super-Threshold

The same analysis can be performed for the MDP operating in super-threshold. Assuming an ideal nmos transistor, the results of such an analysis is given by

$$i_{Dn} = \frac{\mu_0 C_{OX}}{2}\frac{W}{L}\left(v_{gaten} - v_{refn} + \sqrt{ibias_n \frac{2}{\mu_0 C_{OX}}\frac{L}{W}}\right)^2 \quad (4)$$

where $\mu_0$ is the mobility and $C_{OX}$ is the gate oxide capacitance density.

The drain current is a function of the difference between the gate voltage and the reference voltage but not the threshold voltages. All of the other terms in the equation are constant and in no way depend on threshold voltage.

### 1.2.2 Digital Logic Application

#### 1.2.2.1 n-Channel On-Current not a Function of Threshold Voltage

The current, $I_{ONn}$, is defined to be the n-transistor current in the CMOS digital gates when the gate input is set to the logic HI voltage for *vsp*. In Figure 1.3, an MDP current controller as described above sets the on-current in the n-channel transistors in the digital gates.

The opamp buffers the source of M0 and sets the negative supply, *vsn*, of all of the digital gates. Since an MDP construct is used, the on-current in the n-channel gate transistors, such as M1, is a function of the difference between the logic HI voltage level of the gate, *vsp*, and the voltage value of $v_{refn}$. The n-channel current $I_{ONn}$ is not a function of the value of the threshold voltage of the transistors.

Equation (3) derived above from the MDP theory of operation applies directly to $I_{ONn}$ with the logic HI voltage, *vsp*, used in place of $v_{gaten}$

$$I_{ONn} = ibias_n * e^{\left[\frac{vsp - v_{refn}}{n\varphi_t}\right]}. \quad (5)$$

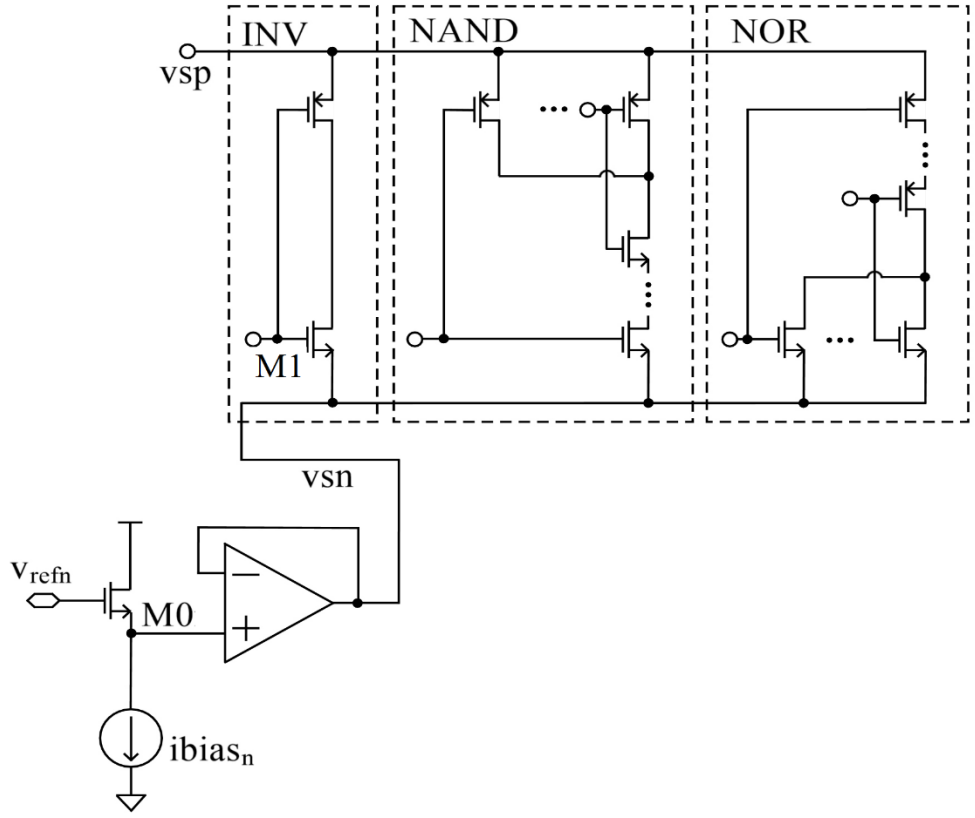


Figure 1.3 MDP current controller to set the currents in the n-channel transistors in the digital gates.

1.2.2.2 Wide Range of Control in Current

A definition is made as $\Delta_n = vsp - v_{refn}$. While remaining in sub-threshold mode, as $\Delta_n$ is varied, it can control the n-channel on-current, $I_{ONn,}$ in a well-defined way over a very large range, as in an order of magnitude change in current for every one tenth of a volt change in $\Delta_n$.

As the operation moves into super-threshold, the control moves into a square-law formulation as in (4).

## 1.3 Advantages of Current Controlled CMOS Logic

### 1.3.1 Minimizes Variance in Gate Operating Frequency

#### 1.3.1.1 Background

The gate operating frequency is the reciprocal of the delay and a single gate delay is the time it takes for the nmos or pmos transistors in that gate to charge the load capacitance to half *vdd*.

The operating frequency for a digital system is generally defined by the logic depth in the critical path and the single gate delay. The operating frequency required by a digital block is usually constrained by the overall system it resides in. As an example, a transceiver may specify the serial data should be clocked into the digital block at 10 MHz or a sensor interface needs to operate at a frequency around 100 KHz.

#### 1.3.1.2 Frequency Variation Figure of Merit

As the supply for a digital logic system is lowered down into the sub-threshold region, the frequency of operation will reduce rapidly. To make matters worse, the reduced operating frequency varies by a large amount over the process corners.

The system has a minimum, typical, and maximum operating frequency. Due to the variations, there is usually a need to design for the minimum frequency instead of the typical frequency.

In addition, during static timing analysis, the setup times are checked with the minimum frequency clock but the hold times are checked using the maximum frequency. The larger the frequency variance the harder it will be to close timing, especially for large digital designs.

In order to measure and compare the frequency variance, we define a figure of merit, *Df*, and specify it as

$$Df = [(f_{max} - f_{min}) / f_{min}] * 100\% \quad (6)$$

where $f_{min}$ is the minimum frequency of operation and $f_{max}$ is the maximum frequency of operation due to the variations in process corners.

#### 1.3.1.3 Standard CMOS Gate Operating Frequency Variance is Exponential with Respect to Threshold Voltage

The operating frequency analysis can be done considering either the n-channel charging current or the p-channel charging current. If the nmos is used, in standard CMOS gates the operating frequency in sub-threshold mode is proportional to $I_{ONn}$ divided by *vdd* [1].

The off-current, $I_{off}$, is defined when $v_{GS} = 0$ and $v_{DS} = vdd$. If the DIBL effect is ignored [9] in sub-threshold mode, following (2)

$$I_{off} \approx I_{STR} = I_0 \frac{W}{L} e^{\frac{-Vth_{eff}}{n\varphi_t}}. \tag{7}$$

and if (1) and (2) are combined

$$f_{oper} \propto \frac{I_{off*} e^{\frac{vdd}{n\varphi_t}}}{vdd}. \tag{8}$$

and it is evident that for standard CMOS gates the operating frequency is an exponential function of the threshold voltage.

#### 1.3.1.4 Current Controlled CMOS Logic Operating Frequency Variance is Linear with Respect to Threshold Voltage

The MDP construct controls the gate on-currents diminishing their sensitivity to the die-to-die parameters. In this way, there is less sensitivity in the gate operating frequency to the process variations, even while operating in the sub-threshold mode.

In this work, the current controller block is designed to use an n-type MDP construct to control the n-transistor gate on-currents with the positive supply, *vsp*, according to (5). The logic HI is

$$vsp\ (logic\ HI) = v_{refn} + \Delta_n. \tag{9}$$

where $\Delta_n$ is the difference between the positive supply and the reference voltage.

It is desired to have the negative supply of the gates to be zero; therefore, assume a change in Figure 1.3 such that *vsn* is set to zero, M0 is diode connected with a current of $ibias_n$, $v_{refn\_ub}$ is equal to the $v_{GS}$ of M0, and the opamp buffers the signal $v_{refn\_ub}$ and its output is summed with a signal $\Delta_n$ to create *vsp*.

Similar to (8), in sub-threshold mode the operating frequency of the current-controlled CMOS gates is proportional to the ratio of the nmos charging current and the positive power supply

$$f_{oper} \propto \frac{I_{ONn}}{vsp}. \quad (10)$$

Using $\Delta_n = vsp - v_{refn}$ in (5) and substituting from (1) into (9)

$$f_{oper} \propto \frac{ibias_n * e^{\frac{\Delta_n}{n\varphi_t}}}{n\varphi_t \ln\left(\frac{ibias_n}{I_{off}}\right) + \Delta_n}. \quad (11)$$

The current $ibias_n$ and the signal $\Delta_n$ in (11) are not a function of threshold voltage. The only term in (11) that has any dependence on the threshold voltage is $I_{off}$. Since this dependence is a natural logarithmic function of an exponential, the operating frequency of the current controlled gates in this work varies linearly with the threshold voltage, not exponentially as with the standard gates.

#### 1.3.1.5 Comparison

Figure 1.4 is a plot of the frequency vs. *vdd* for three different values of the change in the threshold voltage, $V_{th}$: -50mV, 0, and 50mV for standard CMOS gates biased in sub-threshold found from (8). At a sample point of *vdd = 0.*205, when the change in the threshold voltage is zero, the frequency is approximately $f_{oper}$ = 17MHz. The variation in sub-threshold performance due to the changing threshold voltage is from $f_{min}$ = 5MHz to $f_{max}$ = 60MHz. The figure of merit defined in (6), *Df,* for this variation is 1100%, more than an order of magnitude change in the operating frequency for a +/-50mV change in threshold voltage.

The plot in Figure 1.5 is for the current-controlled CMOS logic in this work found from (11). The variance due to changing the threshold voltage is much smaller. For the same positive supply value of 0.205V, the total min. to max. variation, *Df,* is 50% compared to 1100% for the standard gates.

The logic gate operating frequency variance is minimized, even in sub-threshold mode. The system design can have a much tighter timing specification by not having to account for such a high variance. In the above analysis, the $f_{min}$ is 5MHz for standard gates and is 14MHz for 3C

gates. The current controlled logic system can be designed to guarantee a throughput nearly three times higher. Having a smaller variation over process corners enables a digital design flow to be used even in sub-threshold mode and it will be easy to close timing with static timing analysis thereby enabling mass production.

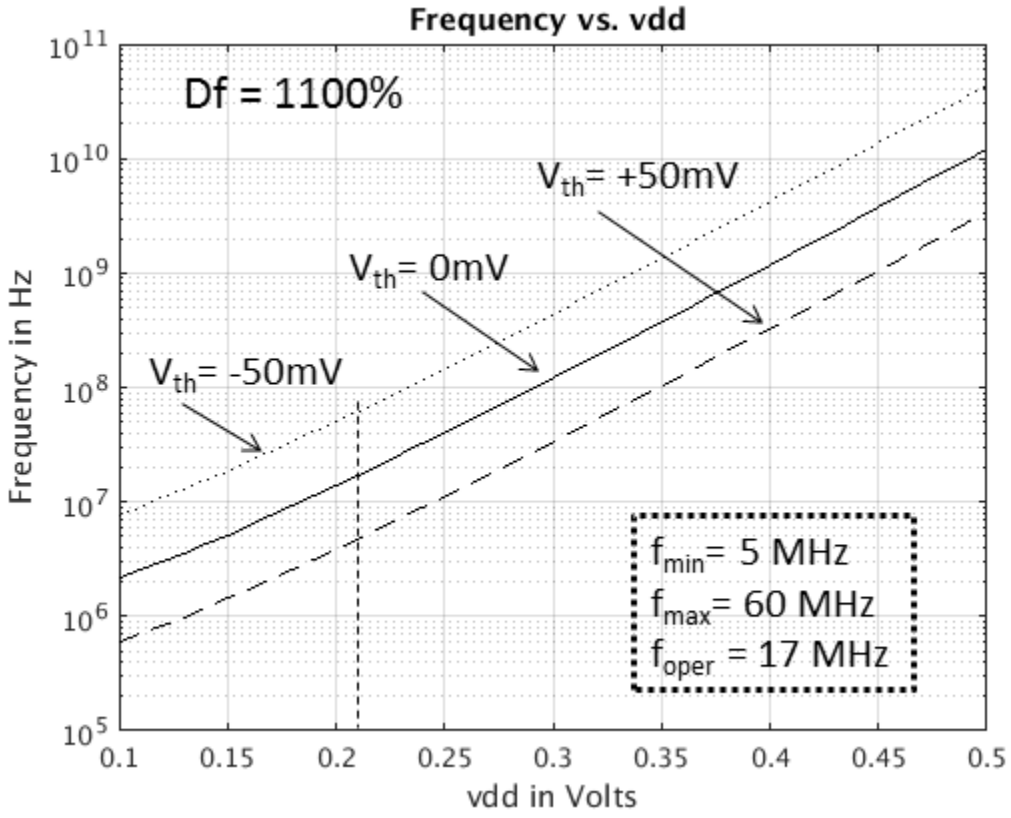


Figure 1.4 Plot illustrating wide variance in standard CMOS gate frequency.

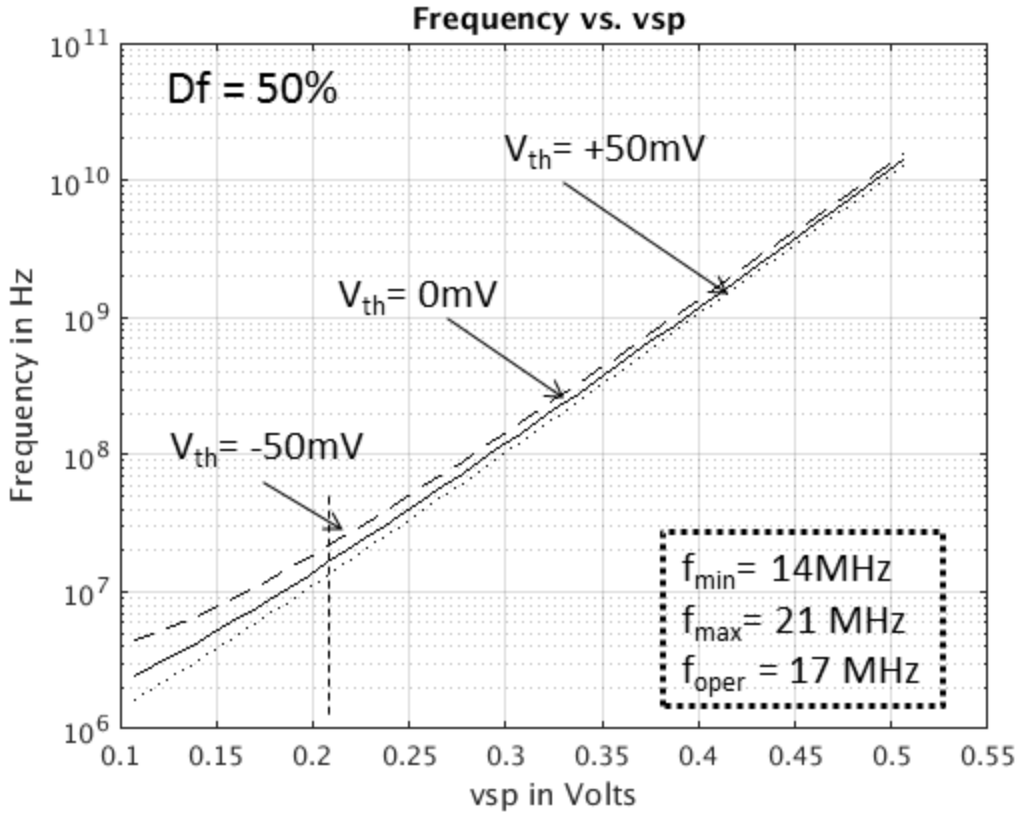


Figure 1.5 Plot illustrating narrow variance in current controlled CMOS logic frequency.

### 1.3.2 Maintaining Symmetry with Reduced Area and Increased Frequency

Because of their reduced strength, it is customary to make the size of the p-channel transistors larger than the n-channel transistors.

With the current controlled logic in this work it is not necessary to make the p-channel transistors larger.

In current controlled logic, both the n-transistors and p-transistors have independent control of their on-current through their respective MDP based current controller. By adjusting the ratios of the bias currents in the reference transistors, the nmos and pmos sizes in the gates can be made equal while still maintaining the symmetry.

Figure 1.6 is a representation of a simulation of a minimum inverter with a 0.3V *vdd* driving a fairly large 50fF capacitive load having a current controller. For the inverter in this simulation, the pmos transistors have the same dimensions as the nmos transistors. The top curve is the current in the gate, the middle curve is the inverter input and the bottom curve is the inverter output. Note the symmetry in the rise and fall time of the output voltage waveform. The symmetry is achieved even with the p-channel W/L ratio equal to the n-channel W/L ratio.

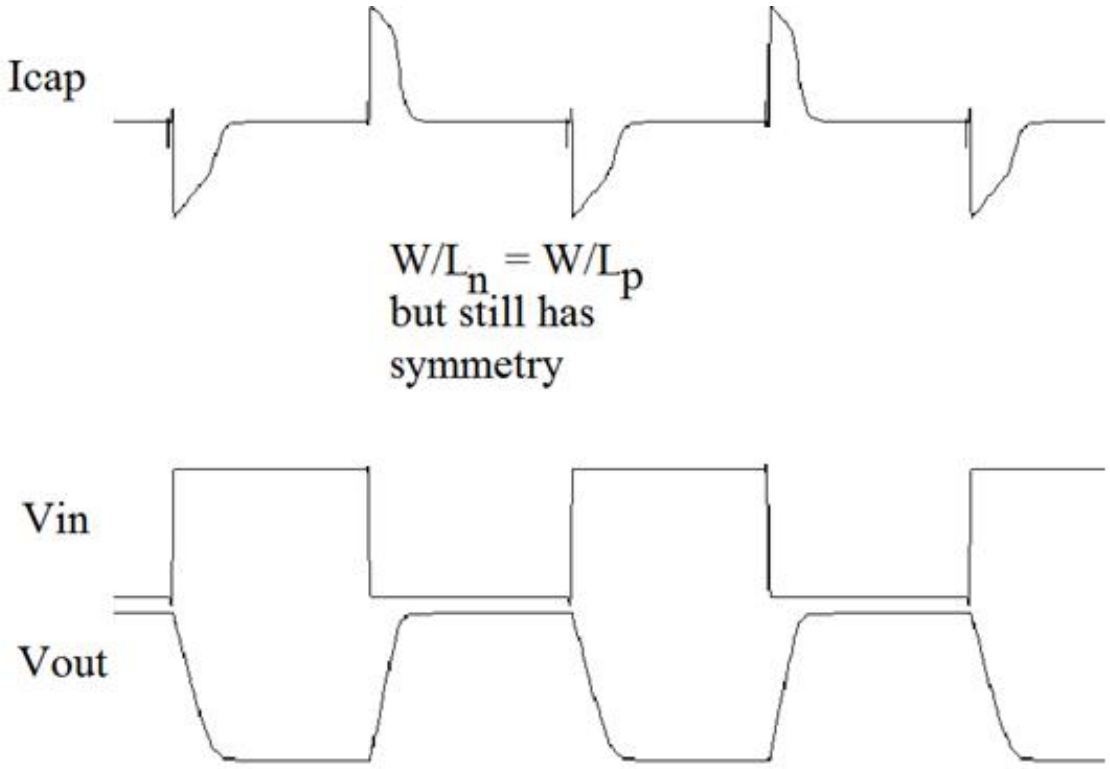


Figure 1.6 Plot illustrating symmetry of a 3C minimum inverter having 0.3V *vdd* with equal pmos and nmos geometry driving a 50pF load.

The result of sizing all of the gates in this way is a reduction in the area and an increase in the speed of the gates. Current controlled logic pmos transistors are half the size, or less, of standard gate pmos transistors and operate at least 30% less capacitance. The lower capacitance enables current controlled gates to operate at higher frequencies.

The comparison plot in Figure 1.4 and Figure 1.5 assumed equal capacitance; actually, the 3C gates have less capacitance. So if the standard gates operate at 17MHz, all other things being

equal, the same current controlled gates operate at 22MHz, or more, while maintaining the improved variance ratio.

### 1.3.3 Operate at Optimal Energy

#### 1.3.3.1 Ideal Minimum Supply Voltage

At some point as the supply is lowered, the dynamic energy becomes optimally balanced with respect to the static leakage energy thereby creating a minimum in the energy vs. *vdd* function. The dynamic energy decreases at a rate of *vdd* squared as the supply is lowered. As the supply is lowered so as to enter the sub-threshold region, the leakage energy increases exponentially due to the increasing gate delay.

The equation for the total energy is which the minimum as described above is manifested is

$$E_{tot} = C_{eff} * vdd^2 + vdd * I_{off} * T_{CK} = E_{dyn} + E_{lkg} \quad (12)$$

where $E_{dyn}$ is the dynamic energy, $E_{lkg}$ is the leakage energy, and $E_{tot}$ is the total energy. $C_{eff}$ is the effective capacitance, including the activity factor, *vdd* is the supply voltage, $I_{off}$ is the drain current measured with $v_{GS} = 0$ and $v_{DS} = vdd$, and $T_{CK}$ is the clock cycle, which is a function of the gate delay within the critical path, the logic depth, and the average stacking factor within the gates.

#### 1.3.3.2 Lower Bound for Supply Voltage

The imbalance factor (IF) sets a practical lower bound for the supply voltage. The IF is a statistical mismatch of the p-channel transistor strengths compared to the n-channel strength. The IF parameter is much greater in sub-threshold mode [9]. If the imbalance factor increases enough, the dc characteristics are degraded and the noise margin decreases.

There is a lower bound in *vdd* value, called $VDD_{break}$, which causes the gates to stop functioning correctly due to lack of positive noise margin. $VDD_{break}$ needs to be lower than the $VDD_{ideal\_min}$ point, in order to be able to operate at the optimum energy.

Equation (13) defines the imbalance factor as the ratio of the strength in (2) for the pmos and nmos transistors regardless of whether the stronger one is n or p.

$$IF = max\left(\frac{I_{STRp}}{I_{STRn}}, \frac{I_{STRn}}{I_{STRp}}\right) \geq 1 \quad (13)$$

For the commercial 180nm process used in the work, the IF is between 80 and 100 when considering the 'sf' or the 'fs' model corner files in sub-threshold mode. Of course, the intra-die variations might add to that number, so a given gate might have an IF in the hundreds, even thousands. Whereas at a *vdd* of 1V in super-threshold, the IF for the same extreme corner models would be about 2 or 3.

A rule of thumb in calculating the $VDD_{break}$ point for a given gate as a function of IF [9] is

$$VDD_{break} = VDD_{min-theory} + n\varphi_t \ln(IF) = 200mV \quad (14)$$

*(for IF = 50)*

where $VDD_{min\text{-}theory}$ is the theoretical limit for an inverter's *vdd* and is on the order of 50mV. With an IF value of 50 the value of $VDD_{break}$ will increase to about 200mV.

1.3.3.3 Use Adaptive Body Bias with p-Channel Tubs

The variation in IF is greatly improved with a servo for the tubs for all of the p-channel gate transistors as is depicted in Figure 1.7. Only the p-channel tubs are used. The tubs of the n-channel gate transistors are not needed in the IF control process and instead of tying the n-channel tubs to ground they can be used to tune the frequency. In sub-threshold mode the gate operating frequency is a strong function of threshold voltage.

The input *delta* represents both $\Delta_n$ and $\Delta_p$ and is an analog voltage representing the desired ratio between the respective on-currents in the gates to the bias currents in the reference transistors. The input *delta* is on the gate of the reference transistor for a p-type MDP and sets the on-currents in the p-channel transistors in the gates. The negative supply, *vsn,* is at ground and the positive supply, *vsp,* is whatever the MDP current controller makes it to be to establish the desired ratio of p-channel gate on-currents with respect to $ibias_p$. The value of *vsp_ub* on the source of the MDP reference transistor is $v_{SGp}$ higher than *delta.*

The output of the opamp in the tub servo is tied to the tub of the reference p-transistor as well as all of the tubs of the gate p-transistors. The body effect adjusts the threshold voltage of the p-transistors. At the input of the servo opamp, the output of the summation block, *vsp_ref,* is

compared with an opamp to *vsp_ub*, an un-buffered version of *vsp*. The threshold voltages of the p-transistors are adjusted to meet the condition of (9), $vsp\ (logic\ HI) = v_{refn} + \Delta_n$ thereby setting the n-channel on-currents to have the correct ratio with respect to $ibias_n$.

To summarize, first the p-type current controller sets the p-channel currents with respect to $ibias_p$. Then the tub servo sets the p-channel threshold voltage to make the n-channel currents equal to the p-channel currents. The imbalance factor is adaptively controlled thereby maintaining the desired symmetry in the VTCs of the gates, even in the presence of the extreme corners of the process, such as weak p's and strong n's, and vice versa.

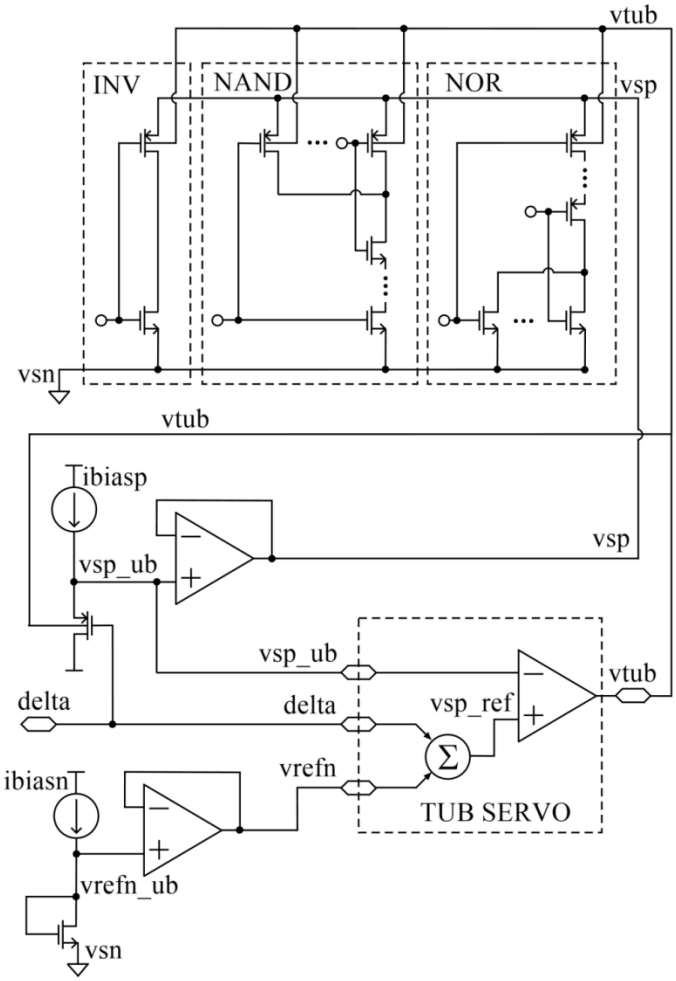


Figure 1.7 Circuit to servo only the n-well tubs of the p-transistors in the logic gates to control the imbalance factor.

## 1.4 Chip Implementation for a Wallace Tree Multiplier

We create an 8-bit Wallace Tree multiplier using both standard gates and custom 3C gates. The process is a 45nm SOI process from Global Foundries. For the 3C gates, it is necessary to use the provided p-channel transistors that have an n-well contact in the gates. These p-channel transistors have more capacitance and are larger than their counterparts without the n-well connection; however, it is an avoidable penalty when using this process.

Figure 1.8 is a schematic of the 8x8 input Wallace Tree multiplier that is implemented. The three stages on the left progressively reduce all of the partial products of the inputs generated in the first stage until there are two words left to be added\. There is one stage of pipelining inserted right before the final addition.

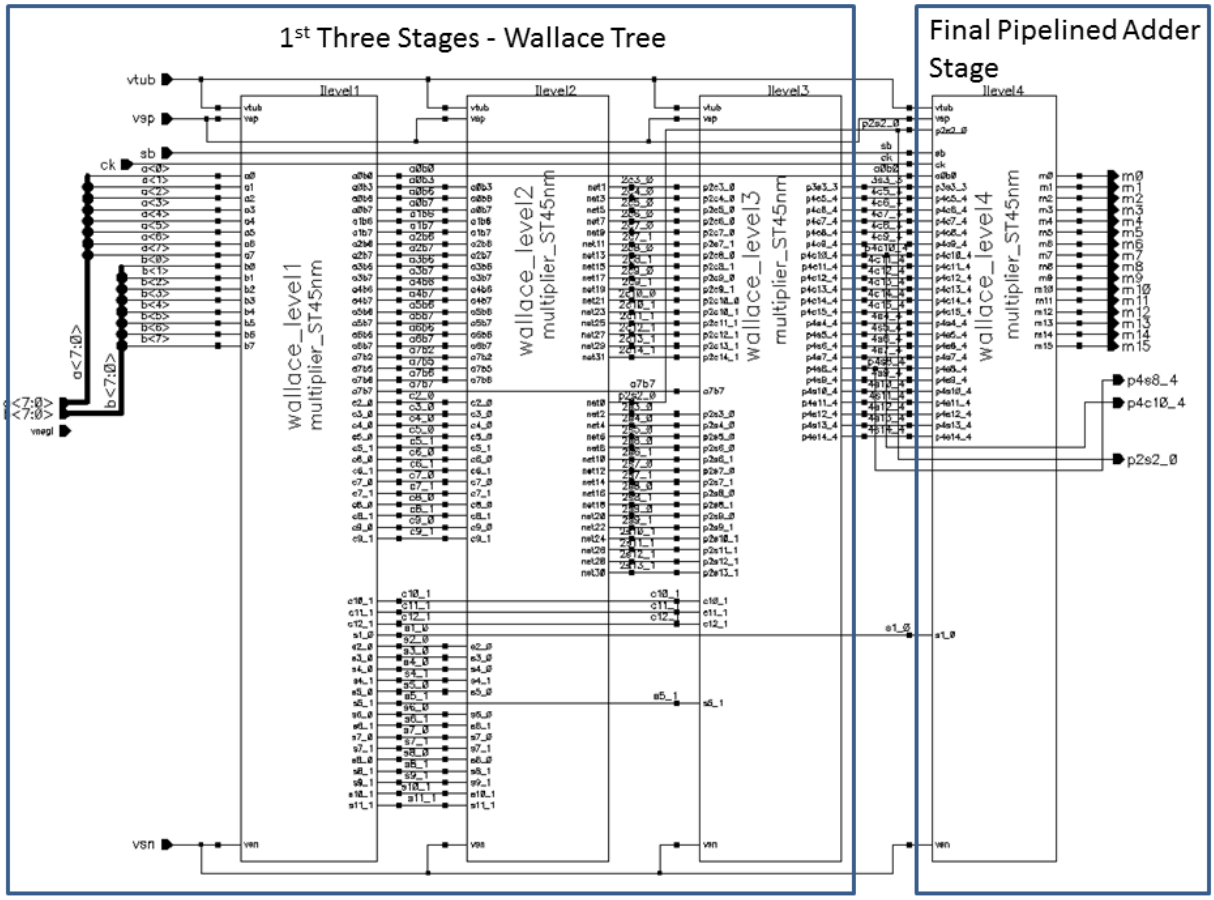


Figure 1.8 Schematic for the 8x8 Wallace Tree multiplier that was implemented.

A full analog transistor simulation applying random 8 bit words to each of the inputs is generated. The energy and operating frequency versus *vdd* was measured with the simulation.

Figure 1.9 shows the results with a plot for the multiplier in the standard gates. Likewise, Figure 1.10 gives the results for the 3C logic. The solid line in both plots represents the multiplier operating frequency determined by measuring the delay up to the input to the pipeline stage. Then that result is multiplied by ten for a cushion that is consistent with the static timing analysis for the digital gates. The dotted line is for the total energy and was determined by measuring the power and multiplying by the delay.

The optimal energy point for the standard gates is for a *vdd* equals 3.7V DC and for the 3C logic case it is equal to 3.4V DC. In both cases, operation at the *vdd* for the minimum energy point is in the relatively near threshold regime. Both designs run at a typical frequency of 8MHz with the optimal *vdd.*

With *vdd* at the optimal point, two more simulations are run with the fast and slow corners and the delay for each case is noted. The figure of merit, *Df,* is defined as

$$Df = [(f_{max} - f_{min}) / f_{min}] * 100\%. \quad (15)$$

This FOM for the standard gates is 372% while for the 3C gates it is 80%. The conclusion to be drawn from this data about the performance of the Wallace Tree multiplier designed with 3C logic is that it is fully function from 0.2 to 0.7 and at the optimal energy point the frequency variance is improved by nearly 5X compared to the standard gates.

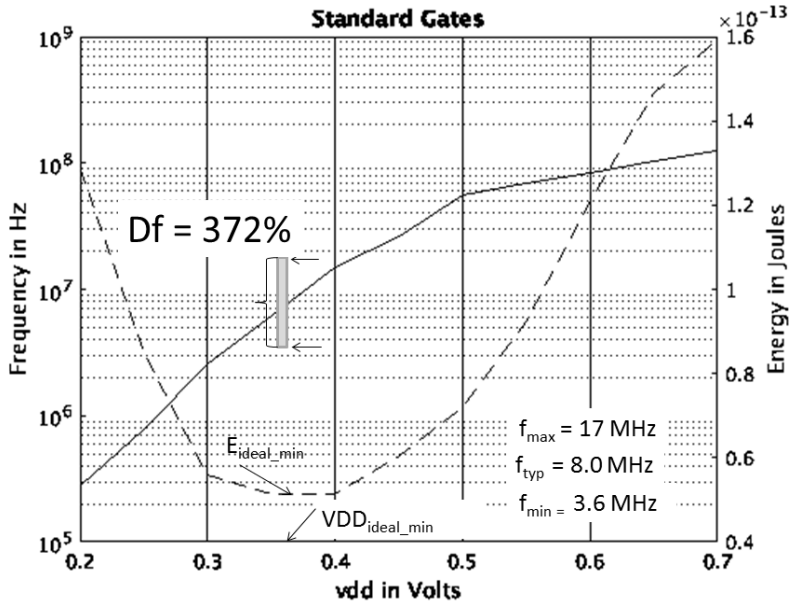


Figure 1.9 Frequency and energy for the Wallace Tree multiplier design plotted as a function of *vdd* for standard 45nm gates.

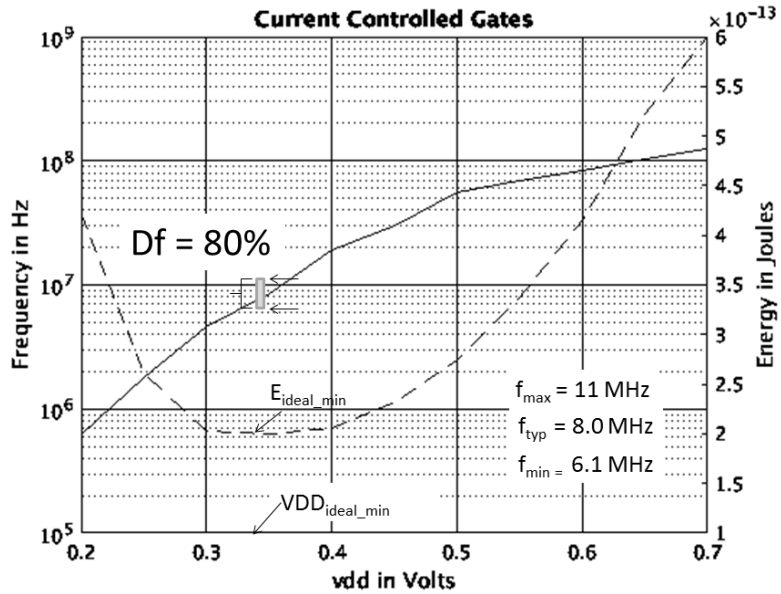


Figure 1.10 Frequency and energy for the Wallace Tree multiplier design plotted as a function of *vdd* for 3C logic 45nm gates.

## 1.5 Digital Design Flow for Current Controlled CMOS Logic

For very large circuits it is not possible to simulate in transistor mode, including Monte Carlo or corners simulations. To design a very large integrated system for high volume manufacturing, it is necessary to use a digital flow, one that involves RTL coding, synthesis, and place-and-route.

### 1.5.1 Standard Cell Library

Standard cells for the 3C gates are needed so place and route tools can be used. When creating a standard cell library of 3C digital gates, the GDSii layout of the library cells needs to be LVS/DRC clean both standalone and also when the cells abut one another. It is also necessary that the pins be on a grid with the correct pitch suitable for routing. The height and width of the cells need to be multiples of the pitch.

We create a standard cell library for the 3C digital gates. There are 18 cells total. The process is an 180nm commercial process. The cells all have a standard height. Some are 2X taller and the width is a multiple of an appropriate minimum site size. The terminals are all on the routing grids. The list of cells in the 3C standard cell library is as follows: invx1, invx2, invx4, bufx1, bufx2, bufx4, nand2x1, nand3x1, nand4x1, nor2x1, nor3x1, nor4x1, dffr, dffs, xor2, latchx3, latch_nores, tsinvx1.

### 1.5.2 Creating LEF Files

To get the cells into the Place and Route tool, a LEF file is needed. Library Exchange Format (LEF) represents the GDSii layout in an ASCII format; however, the LEF file has only the minimum amount of layer information needed for the place and route tool. To create the LEF file from the GDSii layouts, we use an abstract generator tool that does most of the work to make a text file with a ‘.lef’ extension.

So, to enable place and route we create LEF views for all eighteen of the standard cells.

## 1.6 Characterization for Synthesis .lib File

We build a synthesis library including minimum and maximum corners with *vdd* of 0.3V DC for the custom standard cell library in a commercial 180nm process.

We simulate all of the cells over typical, minimum, and maximum corners at 0.3V DC with back-annotated parasitics and keep data for setup time, clk to q, gate intrinsic delay, and so on. Then we convert this data into a .lib file for the synthesis tool. We also measure the capacitance for each pin with simulation. The delay is characterized as

$$Cell|_{delay} = Intrinsic|_{delay} + Transition|_{delay} + Slope|_{delay}. \quad (15)$$

Intrinsic delay is the cell's delay with no load while being driven by the same cell also with no load. See Figure 1.11. Transition delay is the additional delay of a cell driving a capacitive load. Slope delay is extra delay associated with a driving cell having a different slope than the un-loaded case.

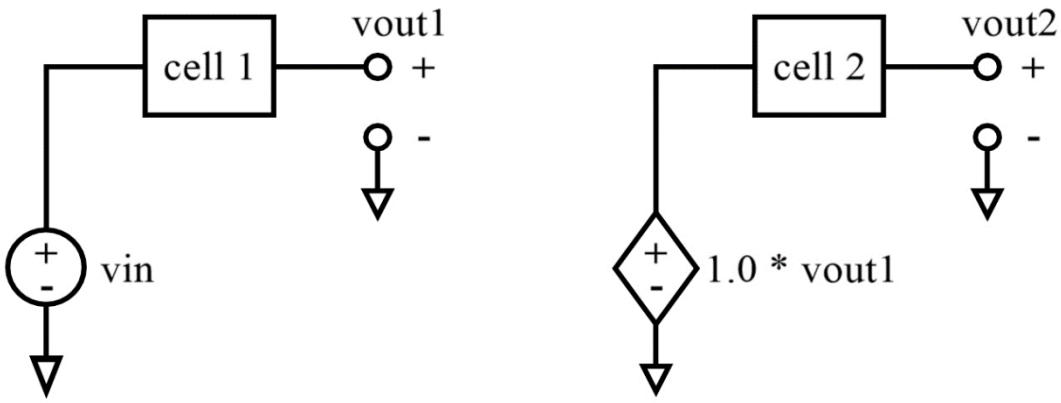


Figure 1.11 Schematic defining measurements for intrinsic delay.

The transition delay is proportional to the load capacitance.

$$Transition|_{delay} = k1 * C_L \quad (16)$$

where $C_L$ is the load capacitance and *k1* is proportionality constant sometimes referred to as the output resistance, although it is not a resistance per se.

The slope delay is described as

$$Slope|_{delay} = k2 * Transition|_{delay} \quad (17)$$

In (17), *k2* is a proportionality constant, sometimes referred to as the slope sensitivity.

We characterize the delay for each cell from the simulation data to determine the constants in the above equations. Along with the pin capacitance measurements, we compute the I/O path timing arcs for each of the standard cells and place the resulting data into a synthesis .lib liberty file. See Appendix A for the .lib file created for the standard cell library.

## 1.7 AES128 Encryption and Decryption Engines

### 1.7.1 AES128 Description

In 2001, an Advanced Encryption Standard (AES) was chosen to replace the standard that had been in place previously for the past twenty years, or so.

AES works on blocks of 128 bits to process data along with a 128, 192, or 256 bit key. Figure 1.12 is a block diagram illustrating an AES128 Encryption Engine. The inputs are a 128-bit block of plain text and a 128-bit key. The output is the cipher text. The processing goes through 12 total rounds – an initial round, 10 repeated intermediate rounds and a final round. There is a key expansion routine which generates the appropriate key for each round as it is being processed.

Figure 1.13 is for the Decryption Engine. The signal processing is the same; that is, 12 rounds. A key storage buffer is needed, though, because the expanded keys need to be presented to the processed rounds in reverse order compared to the encryption engine.

### 1.7.2 RTL Implementation of an AES128 Engine

The AES128 engine as has been described is implemented in RTL for both encryption and decryption. The design works with 128 bit wide words going in and coming out.

The AES128 RTL block is shown in Figure 1.14. There are 128-bit serial to parallel blocks for the input text and for the key. The serial inputs are converted to 128-bit words. The serial clock is synchronous with respect to the serial inputs and is divided down by sixteen and also is divided further for the conversion rate clock. The conversion rate is 512 times slower than the serial clock. The decryption engine has a 128-bit output word and is converted to a serial stream with a parallel to serial converter.

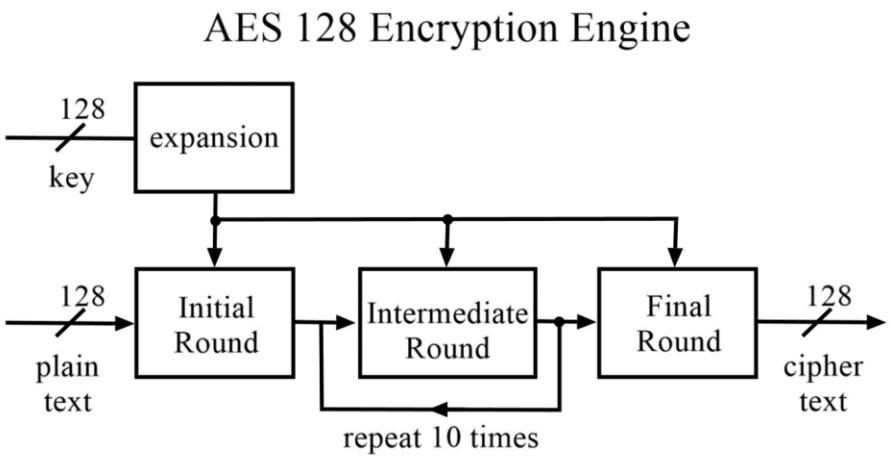


Figure 1.12 Block diagram for an AES128 Encryption Engine.

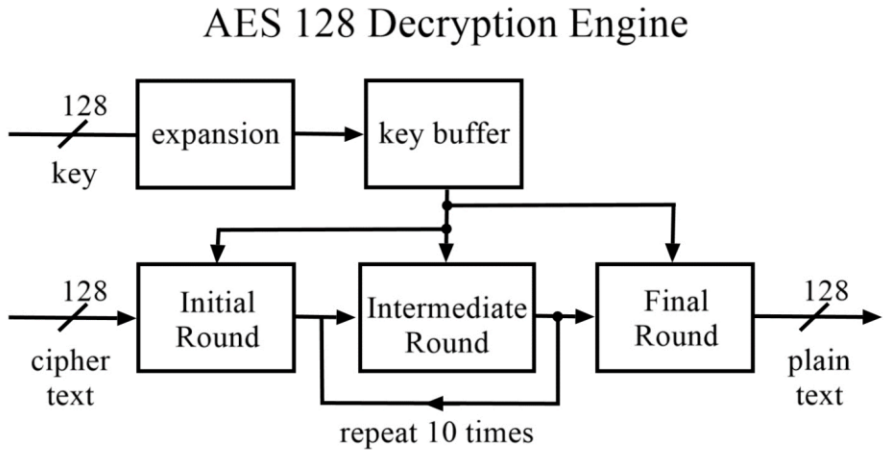


Figure 1.13 Block diagram for an AES128 Decryption Engine.

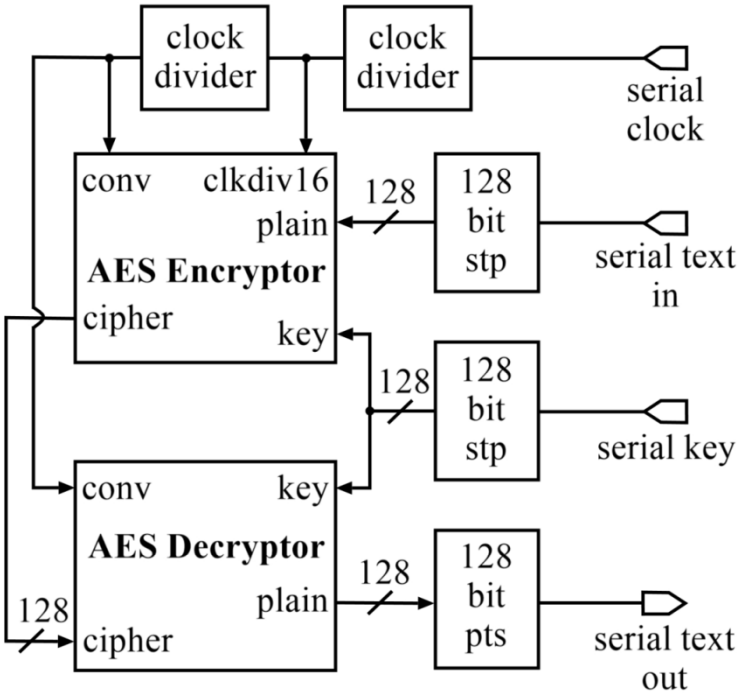


Figure 1.14 Block diagram depicting AES system that was implemented in RTL.

For the AES128 RTL implementation of the design, only two 2K look-up tables (LUT) are used and are shared in series. Only one look-up table (LUT) was chosen for each the encryption and decryption. A word from the respective state is presented one after another to a single LUT instead of multiple words in parallel to multiple LUTs. With this method, there is a penalty in elapsed time and throughput with the benefit of reduced area.

### 1.7.3 Synthesis and Place-and-Route

We synthesize the RTL design for the AES128 encryption and decryption engines using two different libraries. The first library is a 180nm commercial standard cell library characterized for 1.0V. The second is the custom 3C logic standard cell library in the same process characterized for 0.3V. The threshold voltage is typically 0.42V DC in this 180nm technology. The gate count is approximately 100,000 gates for both.

For the commercial standard cell case a serial clock with a frequency of 2.56MHz is chosen. The conversion clock frequency with this serial clock corresponds to a throughput of 640,000 bits/sec. This frequency selection is much lower than what could have been achieved from the static timing analysis for the standalone AES128 block. For the 3C logic sub-threshold case at 0.3V the serial clock is chosen at 3.3 KHz with a conversion clock frequency having a throughput of 833 bit/sec. This frequency selection is limited by the static timing analysis of the 0.3V design for the AES128 block.

The synthesis tool reports the power for the commercial standard cell library as 8.8uW being dominated by the dynamic power. The power for the 3C logic version is around 3nW, dominated by the leakage current.

After synthesis, we place and route the two versions of the gate level designs. The biggest difference for the 3C logic libraries compared to normal standard cells is the need to bring out the tub voltages and route the tub voltages along with the other supplies. Figure 1.15 is an image of the place-and-route of both versions of the AES128 design. The custom layout is larger because the library is simpler, 18 cells total, and the layout area for the standard cells is not optimized compared to the commercialized standard library.

### 1.7.4 Analog Simulation of Clock Divider

We extensively simulate the synthesized gates for the clock dividers at the transistor level. We compare the performance of the clock divider blocks between the standard cell and 3C logic versions of the design.

It is observed from the synthesis report for both the standard cell and 3C gate cases that a large portion of the total AES128 power, about half, comes from the clock divider cells.

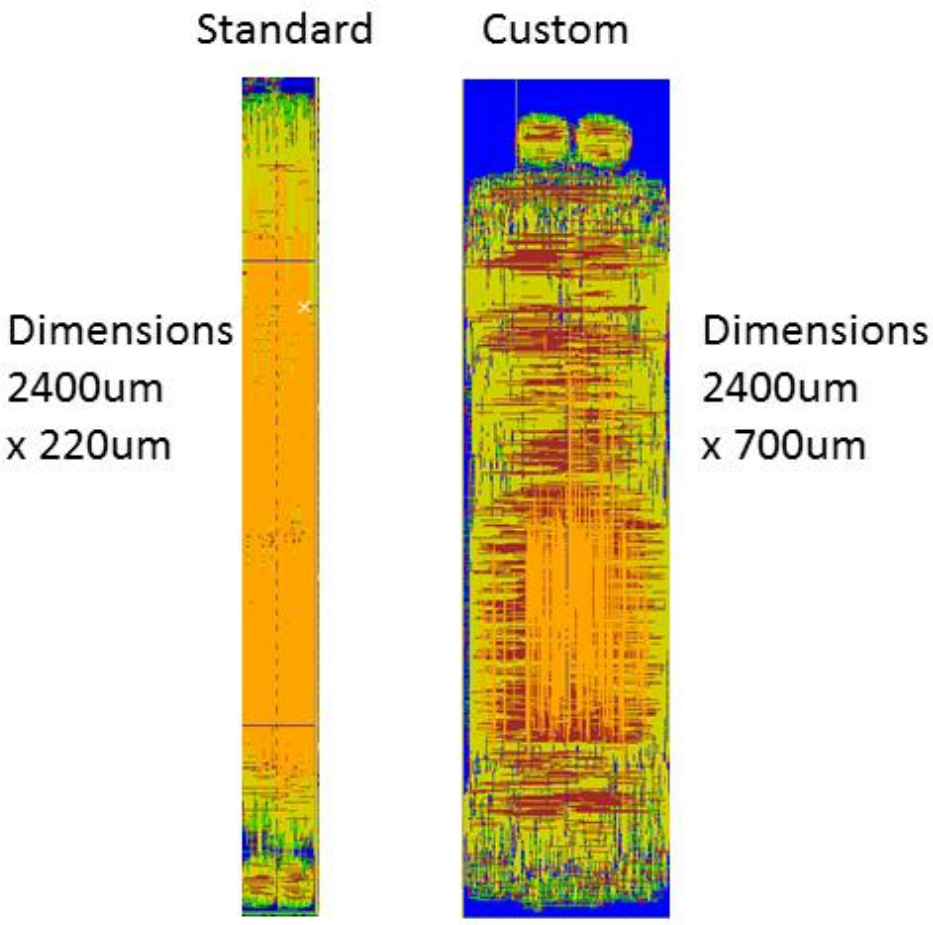


Figure 1.15 Image of Place and Route of AES128 engine with 3C Logic Custom Library vs. Standard Library

We run the complete clock divider chain for both gate architectures with *vdd* from 0.2 V to 1.0 V over the five different model corners and measure the delay of the divider chain. The data from these simulations are plotted in Figure 1.16 for the 3C logic and in Figure 1.17 for the standard gates. The improved variance in the delay data with respect to corners for the current controlled gates is evident from the graphs. The min. to max. difference, *Df*, equals 50% for the 3C logic and 5600% for the standard cell gates when *vdd* equals 0.3V. At 0.3V the minimum frequency for the 3C logic case is 40 KHz whereas for the standard gates it is 5 KHz, more than 8X more guaranteed improvement in the delay for the clock divider.

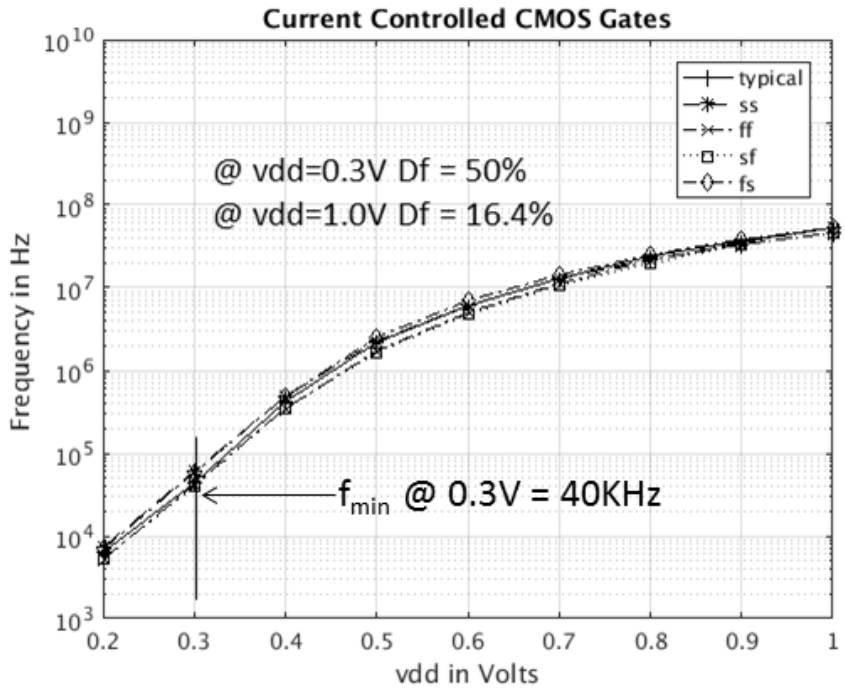


Figure 1.16 Simulation Results for clock divider block over corners with 3C logic illustrating variance in operating frequency.

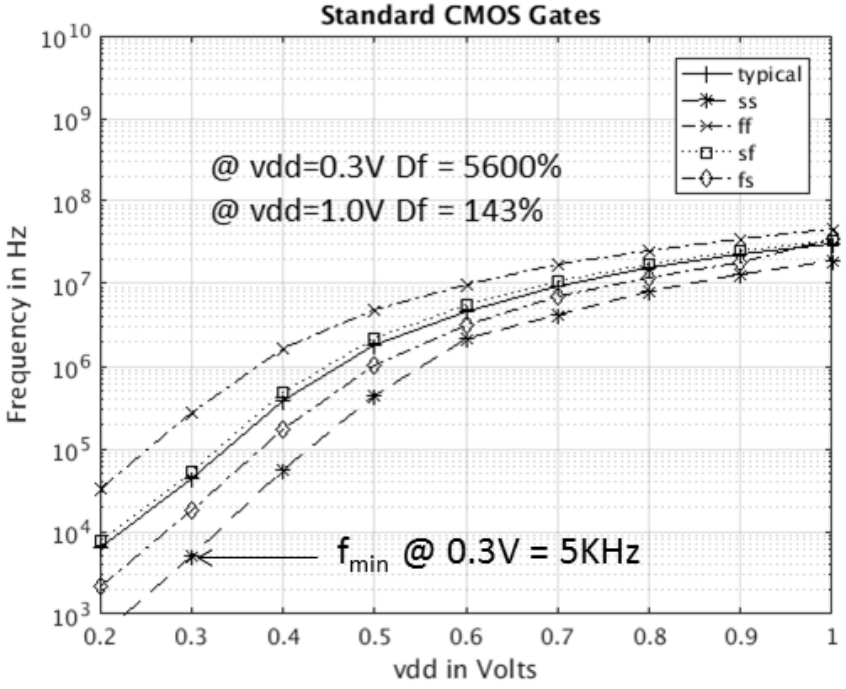


Figure 1.17 Simulation Results for clock divider block over corners with standard gates illustrating variance in operating frequency.

### 1.7.5 Interpretation

According to the synthesis report, the total power for the AES128 block with 3C logic at 0.3V is dominated by the leakage power. For the standard gates at 1.0V, the dynamic power is dominant. Neither situation is operating at the optimal energy.

Between the synthesis reports and the data from the transistor simulations, there is enough information to estimate the *vdd* of the optimal energy from (12). According to the synthesis report for the 1.0V standard design for the entire AES128 block the effective capacitance, including the switching activity factor, equals 3.4pF. For the 0.3V 3C logic design, it equals 1.6pF. The leakage power is 65nW for the 1.0V standard design and is 3nW for the 0.3V logic design. The transistor simulation data that is plotted in Figure 1.16 and Figure 1.17 is divided by ten to represent an appropriate ratio between the static timing analysis and the delay of the clock divider block. From all of these values Figure 1.18 and Figure 1.19 plot the total energy, leakage energy, and dynamic energy from (12) for the AES128 block in 3C logic and the standard case, respectively. These figures show the *vdd* for optimal energy for both designs is 0.4V DC.

Table 1.1 summarizes important design parameters for the AES128 blocks for both the 3C logic and the standard gate cases. For example, the resulting guaranteed throughput for the 3C logic AES128 system is nearly 6X compared to the standard gate version of the design at the optimal *vdd* of 0.4V.

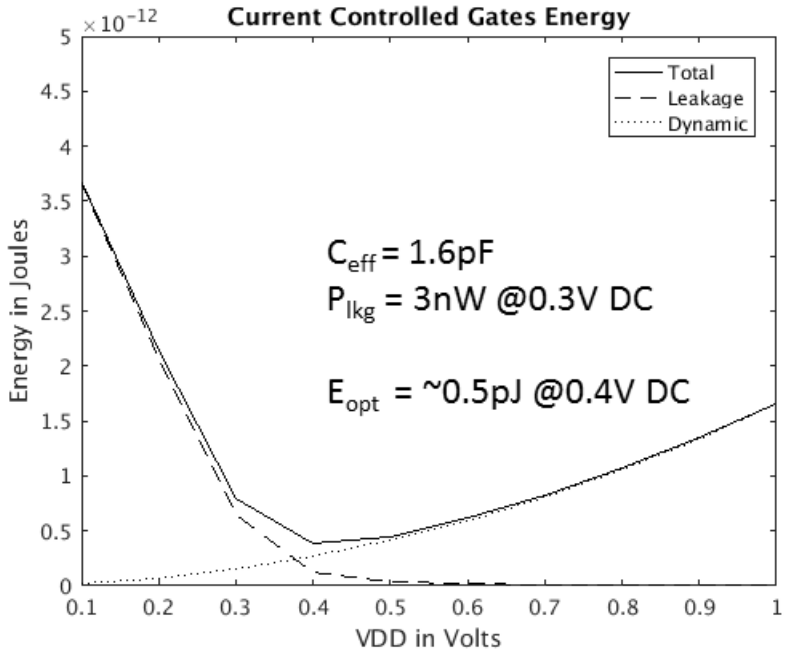


Figure 1.18 Prediction for optimum energy point in AES128 design with current controlled logic.

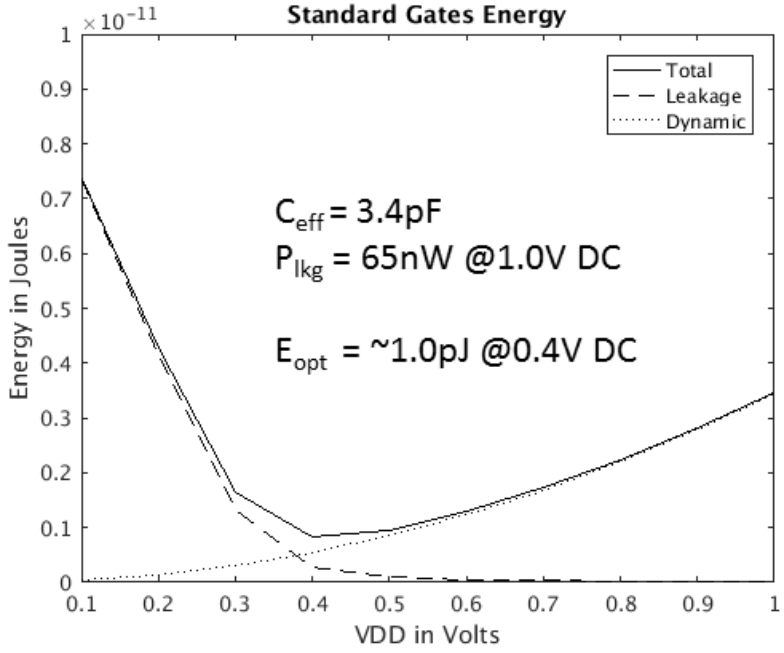


Figure 1.19 Prediction for optimum energy point in AES128 design with standard gates.

TABLE 1.1 AES128 comparison 3C logic vs. standard gates

| | **Symbol** | **3C logic** | **Standard** |
|---|---|---|---|
| Effective capacitance | $C_{eff}$ | 1.6 pF | 3.4 pF |
| Leakage power | $P_{lkg}$ | 3nW @0.3V | 65nW @ 1.0V |
| Ideal min. energy | $E_{ideal\_min}$ | 0.5 pJ | 1.0 pJ |
| Optimal *vdd* | $vdd_{ideal\_min}$ | 0.4 V | 0.4 V |
| AES serial clock freq. (0.4 V) | $f_{ser}$ | 35 KHz | 6 KHz |
| AES guaranteed throughput (@ 0.4 V) | | 8750 bits/sec. | 1500 bits/sec. |

## 1.8 Future Research

### 1.8.1 Current Controlled Temperature Compensation

In addition to the operating frequency varying exponentially with the threshold voltage in (8) for standard CMOS gates, it also varies with temperature. The architecture for the current controlled logic gates in this work can compensate for this variation in temperature.

We create the temperature compensation by generating a voltage proportional to absolute temperature (VPTAT) and summing it with the input voltage *delta* while at the same time also making the $ibias_n$ current *proportional* to temperature. In this way, the variance in the delay is compensated. Generally speaking, the proportionality constant associated with the bias current is used for the more coarse adjustment to cancel the exponential term in (11). The VPTAT summed into the *delta* input is a finer adjustment to cancel the first term having $I_{off}$ in the denominator in (11).

### 1.8.2 Current Controlled SRAM

The standard 8T SRAM bit cell added two read transistors and a read bit line to a 6T SRAM bit cell. By adding these two transistors, the read is de-coupled from the write operation by sensing data through a separate read path. This de-coupling of the read process from the write process improves the robustness, increasing the read and write margins of the 8T SRAM [10]. The 8T bit cells show marked improvement in read stability compared to 6T bit cells.

But for standard 8T SRAM there is still a major drawback in that the predictability of the bit cell's control of the read bit line is dependent on the threshold voltage of the read transistor. This is similar to what is seen with normal CMOS gate performance.

Figure 1.20 depicts a current controlled 8T SRAM architecture. The same technique of executing a servo on the tubs and controlling the current with an MDP construct applies equally well to the 8T SRAM bit cell. If the *vsp* comes from the current controller along with *vtub*, the read bit line current does not vary at all with the die-to-die threshold voltage variations and therefore the delay does not vary either. In addition, the tub control will enhance the read stability.

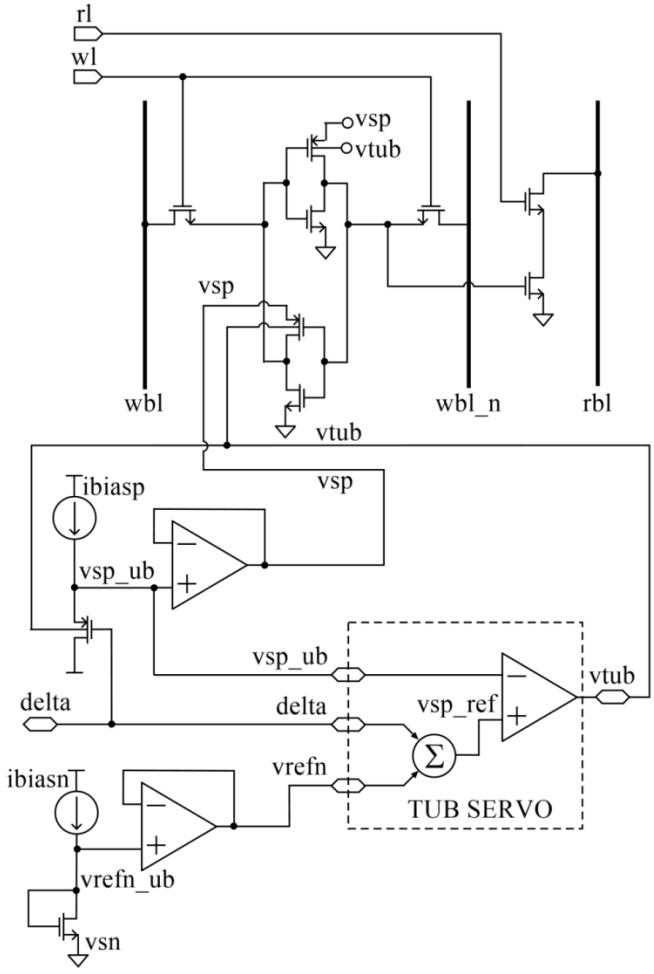


Figure 1.20 Schematic for current controlled 8T SRAM.

The only constraint on the value of *vsp* is the ratio of *ION* to *IOFF*. This ratio is important in determining the number of memory bit cells that can be used on a common read bit line.

Eliminating the unpredictable aspect of the threshold voltage on the current controlling the read bit line allows increased predictability and finer control of this current. With the increased stability, smaller logic voltage levels can be used in the latch, causing considerably less power dissipation.

## 1.9 Conclusion

This chapter introduces the concept of controlling the current in CMOS logic. The major building block required is a modified differential pair which removes the effect of threshold voltage on the controlled currents. The three major advantages for the logic due to the current controller is a large reduction in the standard deviation of the delay, a smaller area due to reduced p-transistor size, and adaptive body biasing to improve the noise margin at low supply voltages.

A standard cell library in a commercial 180nm process has been created including LEF files for place and route and synthesis .lib liberty files. An AES128 engine is designed using the digital flow described in this work. The optimal *vdd* for the AES128 design is 0.4V DC. The delta frequency figure of merit for the AES128 block running at 0.3V is 50%, whereas for standard CMOS gates running at the same supply it would be 5600%.

A Wallace Tree multiplier in a commercial SOI 45nm process is implemented. The *Df* FOM is 80% and the positive supply for the optimal energy point is 0.34V. The multiplier is fully functional at this value of *vdd* with an operating frequency of 8 MHz.

## 1.10 References

[1] Rahul Sarpeshkar, Ultra Low Power Bioelectronics – Fundamentals, Biomedical Applications, and Bio-Inspired Systems, Cambridge University Press, Chap. 21.

[2] Goichi Ono, “Threshold-Voltage balance for minimum supply operation”, *IEEE J. Solid-State Circuits*, vol. 38, no. 5, pp. 830-833 May 2003

[3] S. K. Samal, Y. Peng, M. Pathak, and S. K. Lim, “Ultralow power circuit design with subthreshold/near-threshold 3-D IC technologies”, *IEEE Trans. on Packaging and Manufacturing Tech.*, vol. 5, no. 7, July 2015.

[4] Y. Pu, J. P. de Gyvez, H. Corporaal, and Y. Ha, “An Ultra-low-energy/frame multi-standard JPEG co-processor in 65nm CMOS with sub/near-threshold supply”, *ISSCC 2009*, Session 8.

[5] C. H. Kim, H. Soeleman, and K. Roy, “Ultra-low-power DLMS adaptive filter for hearing aid applications”, *IEEE Trans. on VLSI Systems*, vol. 11, no. 6, Dec 2003.

[6] A. Tajalli, E. J. Brauer, Y. Leblebici, and E. Vittoz, “Subthreshold source-coupled logic circuits for ultra-low-power applications”, *IEEE J. Solid-State Circuits*, vol. 43, no. 7, pp. 1699-1710, Jul. 2008

[7] K. A. Jyotsna, P. Satish Kumar, and B. K. Madhavi3, “Subthreshold circuit design techniques for ultra low-power applications”, *International Jour. Of Adv. Research in Electrical, Electronicsand Instrumentation Eng.*, vol. 5, issue 12, Dec. 2016.

[8] J. Lynch, P. P. Irazoqui, “A low power logic-compatible multi-bit memory bit cell architecture with differential pair and current stop constructs”, *IEEE Trans. on Circuits and Systems – I*, vol. 61, no. 12, pp. 3367- 3375, Dec. 2014.

[9] Massimo Alioto, “Ultra-low power VLSI circuit design demystified and explained: a tutorial”,*IEEE Trans. on Circuits and Systems – I*, vol.59, no. 1, Jan. 2012

[10] Y. Lee, Y. Kim, D. Yoon, D. Blaauw, D. Sylvester, “Circuit and system design guidelines for ultra-low power sensor nodes,” University of Michigan

# CHAPTER 2. A LOG POWER LOGIC-COMPATIBLE MULTI-BIT MEMORY BIT CELL ARCHITECTURE WITH DIFFERENTIAL PAIR AND CURRENT STOP CONSTRUCTS

## 2.1 Introduction

The memory industry is continually seeking to improve the attributes of power consumption, read access time, and memory capacity. The relationship within each attribute and between attributes is complex and each attribute has multiple contributing factors. Power consumption consists of read, write, restore and refresh power. These in turn are affected by noise sensitivity, retention time, leakage, and threshold voltage. Read access time is affected by rate and amplitude of bit line change, delay, and required clock cycles. Capacity is affected by technology node, architecture (1T, 2T, 3T, 6T) and bits per cell. In addition, when one attribute is improved a tradeoff is often needed with one or more of the other attributes. For example, when power consumption is decreased read access time increases, capacity decreases, or both.

Recent 1T1C approaches have reduced read access time at the expense of capacity [1] and increased memory capacity at the expense of power consumption [2]. Both of these, along with all other 1T1C architectures, suffer from full-scale signal swings on high capacitance bit lines, a read implementation based on charge sharing, and a destructive read process. The first two ultimately cause higher power consumption and the latter lengthens read access time.

After falling away in the 1970's, research on logic-compatible gain cells is once again becoming prevalent. Although smaller in capacity, gain cells overcome the shortcomings related to the 1T1C architecture. In the literature some recent gain cell improvements include reduced refresh power and read access time [3], reduced leakage power [4], increased capacity and reduced refresh power [5] and increased capacity [6]. In general, the literature for gain cell research describes operation over a wide range of frequencies with a variety of technology nodes. Yet each of these designs is unsuccessful in completely overcoming two drawbacks of gain cell architecture: first, the read signal, defined as the change in read bit line voltage due to the bit cell current, is a strong function of the threshold voltage of the read transistor; second, the read access time increases rapidly as the stored voltage is lowered.

In this chapter we present analysis, simulation and silicon test results for a logic-compatible memory architecture [7] that improves or overcomes the 1T1C and gain cell drawbacks and effectively reduces read power requirements, sometimes by an order of magnitude. The new architecture also increases memory capacity, in some cases doubling it, while maintaining comparable read access times to both 1T1C and gain cell architectures.

## 2.2 Design

Figure 2.1(a) depicts a standard gain cell architecture. Read power is caused by voltage transitions on the read bit lines. Refresh power is caused by voltage transitions on both the read and write bit lines. The rate at which refresh occurs is the retention time and controls the refresh power. The retention time is determined from the rate the storage capacitor is discharged by the various leakage currents in the bit cell.

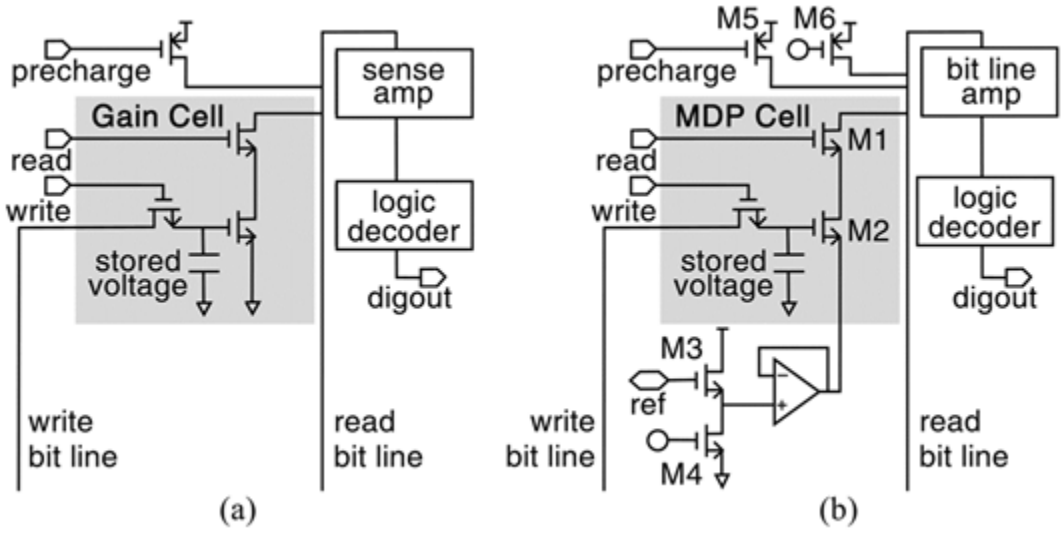


Figure 2.1 (a) Standard 3T gain cell. (b) Modified differential pair bit cell.

Compared to the 1T1C, gain cells have reduced active power due to smaller voltage transitions. Gain cells have shorter read access times due to nondestructive reads. Gain cells also have reduced noise sensitivity because the read signal is not derived from a charge sharing process. These advantages for the gain cell with respect to the 1T1C come at the expense of reduced capacity. In addition, the gain cell read signal is a function of the threshold voltage of the read

transistor in the bit cell. The threshold voltage variance necessitates increased voltage swings on the write bit line to accommodate the largest variance thereby increasing write and refresh power.

The first goal then is to eliminate the unpredictable effect of threshold voltage variance. The second is to significantly reduce the voltage swing on the high capacitance read bit lines.

### 2.2.1 Modified Differential Pair Construct

In standard gain cell architecture die-to-die read bit line voltage variance is caused by within-die threshold voltages and negatively affects the behavior of bit line current. The proposed architecture [7] removes the effect of within-die threshold voltage by adding a transistor, M3. M3 forms a differential pair construct with M2 in the gain cell, as shown in Figure 2.1(b). Since the within-die threshold voltage of the two transistors M2 and M3, are effectively equal within a specified tolerance (see Appendix C for a method to determine a tolerable within-die mismatch), they cancel each other out. Thus the effect of within-die threshold voltage on bit line current is removed. Consequently the problem of die-to-die bit line voltage variance is also removed and the predictability of bit line current is greatly improved.

An op amp supplies the necessary current allowing multiple bit cells to share the M3 reference transistor. The shared reference transistor, op amp and M2 read transistor form a modified differential pair. Subsequently, we refer to this bit cell architecture as an MDP bit cell or MDP memory. To understand the relationship between the two transistors M2 and M3 in Figure 2.1(b), consider Kirchhoff's Voltage Law. Applying the law we have,

$$0 = v_{GS3} - v_{GS2} + v_{hold} - v_{ref} \quad (1)$$

where $v_{GS2}$ and $v_{GS3}$ are the gate to source voltages of M2 and M3 respectively, $v_{hold}$ is the voltage on the storage node and $v_{ref}$ is the reference voltage. Using the equation for the saturation current for a transistor in weak inversion mode [8], $v_{GS}$ is defined as

$$v_{GS} = n\varphi_t \, ln(i_D) - n\varphi_t \, ln(I_0) \quad (2)$$

where $n$ = slope factor, $\varphi_t$ = thermal voltage, $i_D$ = drain current and $I_0$ = saturation current and is defined as

$$I_0 = f(\mu, C_{OX}, \varphi_t, W/L\,, n, V_{th}) \quad (3)$$

where $\mu$ = mobility of electrons in the channel, $C_{OX}$ = oxide capacitance per unit area, $\varphi_t$ = thermal voltage, $W/L$ = ratio of the transistor, $n$ = slope factor, and $V_{th}$ = threshold voltage. It is generally held that the within-die process parameters, the factors in (2) and (3) are considered equal within a specified tolerance for all transistors on any one die. Making the appropriate substitutions it is clear that $v_{GS2} = v_{GS3}$ and that the value of the threshold voltage of M3 cancels the value of the threshold voltage of M2.

To define the read signal, or change in bit line voltage, for the MDP memory we start with

$$i = C_{BL} * dv/dt. \tag{4}$$

Making the appropriate substitutions based on Figure 2.1(b) and solving for $\Delta V_{BL}$ we have

$$\Delta V_{BL} = (1/C_{BL}) \int [(i_{D2} - i_{D6})dt] \tag{5}$$

where $\Delta V_{BL}$ is the change in bit line voltage, $C_{BL}$ is the bit line capacitance, $i_{D2}$ is the drain current for M2, and $i_{D6}$ is the drain current for M6. We intentionally use a low ibias to operate M6 as a current source in weak inversion mode so that its saturation voltage, 4 * $\varphi_t$, is approximately 100mV [8]. During the read operation the read bit line is driven to an equilibrium condition where $i_{D2}$ equals $i_{D6}$ and the current discharging the bit line capacitance is zero. Hence, the read bit line voltage does not fall any further. And specifically, for the logic 0 condition, M6 functions to strictly define and limit the change in bit line voltage to its saturation voltage, approximately 100mV. In the case of non-logic 0 on the storage node, (5) simplifies to

$$\Delta V_{BL} = \left[(i_{D2} - ibias) * t_p\right]/C_{BL} \tag{6}$$

where $\Delta V_{BL} = vddr - V_{BLtsample}$, $V_{BLtsample}$ is the read bit line voltage at time $t_{sample}$, $i_{D2}$ is the storage node transistor M2 current, *ibias* is the value of the p-channel current source transistor M6, $t_p$ is a portion of the time the read transistor M1 is held closed and $t_p = t_{sample} - t_{read}$, $t_{sample}$ is the point in time the read bit line is measured, $t_{read}$ is the point in time the read transistor M1 goes closed, $t_{sample}$ occurs in time after $t_{read}$ and $C_{BL}$ is the value of the read bit line parasitic capacitance.

By making appropriate substitutions of (1) and (2) into (6), the equation defining the MDP change in read bit line voltage, or read signal, for non-logic 0 values is

$$\Delta V_{BL} = \left\{exp\left(\frac{v_{hold} - v_{ref}}{n\varphi_t} + ln(ibias)\right) - ibias\right\} * t_p/C_{BL} \tag{7}$$

where $v_{hold}$ is the voltage on the storage node, $v_{ref}$ is the voltage on the reference node, $n$ = slope factor, $\varphi_t$ = thermal voltage, *ibias* is the value of the p-channel current source transistor, $t_p$ is the time the read transistor is held closed, and $C_{BL}$ is the value of the read bit line parasitic capacitance. Threshold voltage is not a factor in this equation.

### 2.2.2 MDP Bit Cell Operation

At this point it is useful to review the MDP single bit per cell, or BASE2, read operation [7]. In BASE2 operation the stored voltage is one of two values, typically 0.5V or 0.8V. The timing diagrams are depicted in Figure 2.2. Before the read process starts the transistor M5 pre-charges the read bit line. At the start of the read process the precharge input is de-asserted and shortly after the read input is asserted. The voltage on the read bit line is then controlled with a current that is a function of the difference between the stored voltage and the reference voltage as seen by the term *vhold-vref* in (7). If the stored voltage is less than the reference voltage the read bit line voltage will change no more than the value of the saturation voltage of M6. But if the stored voltage is greater than the reference voltage the read bit line will be pulled down by the current in the storage transistor M2 until the topological limit is reached. The bit line amplifier acts as a comparator and uses an appropriately low switching voltage to detect change, and so discerns the value represented by the voltage on the storage node.

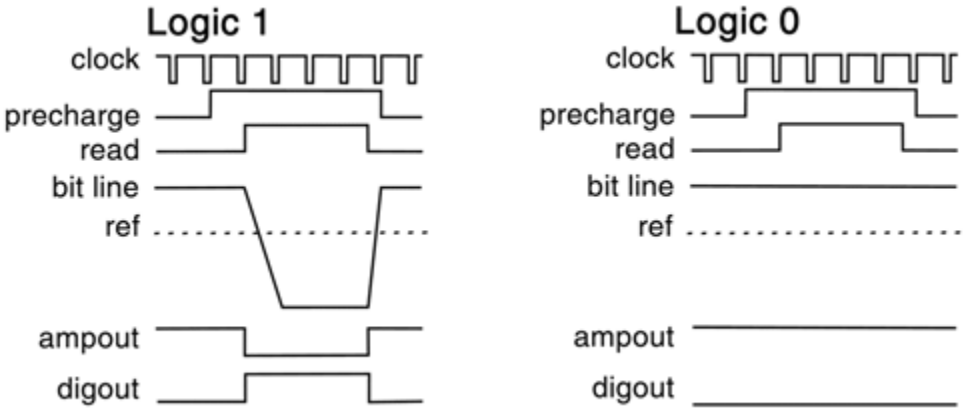


Figure 2.2 Timing diagrams for MDP BASE2 read operation.

Compared to a standard gain cell, the MDP bit cell, with its three transistors and shared reference transistor, has approximately the same storage capacity. However, the capacity of an MDP bit

cell is nearly doubled when it is used in multi-bit mode having multiple logical bits in one bit cell. Subsequently, we refer to an MDP implementation having one of four logical values in the bit cell as two bits per cell or MDP BASE4. The modified differential pair eliminates the impact of the unpredictable threshold voltage variance on the required bit cell voltage, and subsequently on the current controlling the read bit line [7]. The insensitivity of the design to threshold voltage variance enables smaller voltage intervals between logic values and allows the MDP bit cell to reliably accommodate four logical values. The read operation of MDP BASE4 is similar to the MDP BASE2 operation. In BASE4 mode, instead of comparing the storage node to a single reference, it is compared to three references one at a time and one after another in a sequential order causing the read bit line to respond accordingly. The point in the sequence of comparisons that the read bit line first drops beyond the switching voltage of the comparator indicates the value stored on the storage node. The bit line amplifier acts as the comparator and outputs a digital indicator to the logic decoder. The logic decoder uses the indicator, specifically the point in the sequence of comparisons the indicator is asserted, to discern the digital value represented by the voltage on the storage node. Figure 2.3 illustrates MDP BASE4 timing diagrams.

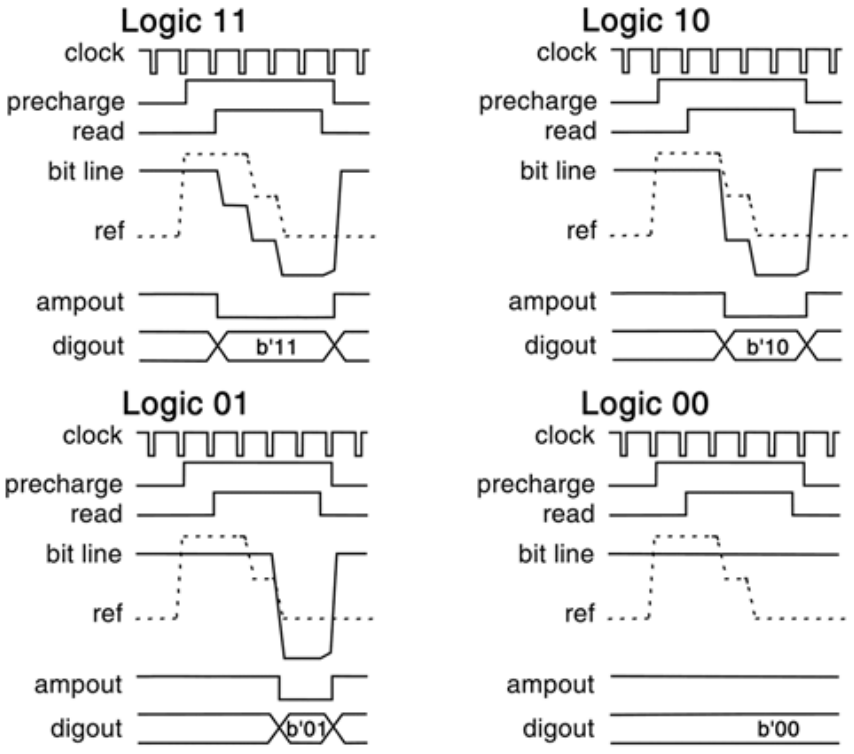


Figure 2.3 Timing diagrams for MDP BASE4 read operation.

### 2.2.3 MDP Logic and Reference Voltage Generation

The logic voltage levels for MDP memory are generated from a stable source, such as a band gap, and are spaced as a function of the desired frequency of operation. MDP architecture facilitates the accurate reference voltages required for multi-bit operation. The reference voltage sources see high impedance at the gate of M3 in Figure 2.1(b) and are tapped from matched components relative to the logic level voltages.

An analysis of MDP BASE4 voltage reference levels and threshold voltage mismatch is in Appendix B.

### 2.2.4 Current Stop Construct

The important information content, the determination of the digital value represented by the voltage on the storage node, is contained in the movement of the read bit line away from its clamped value to a value greater than the switching voltage level of the bit line amplifier. Any movement in the read bit line beyond the switching level of the bit line amplifier causes wasted power. Therefore, it is advantageous to stop the current at a point after the voltage change is deemed significant and before the inherent limit due to the circuit topology is reached [7].

Figure 2.4 illustrates an MDP memory with current stop. The istop transistor acts as a switch to disconnect the opamp from M2 thereby terminating M3's control of the read bit line. In this way, the voltage transitions on the high capacitance read bit lines are drastically reduced because the signal on the read bit line changes only enough to be sensed reliably. Power is correspondingly reduced for all reads of the memory system [7].

Current stop control is shared between all rows in one column. Thus there need be only one current stop circuitry per column.

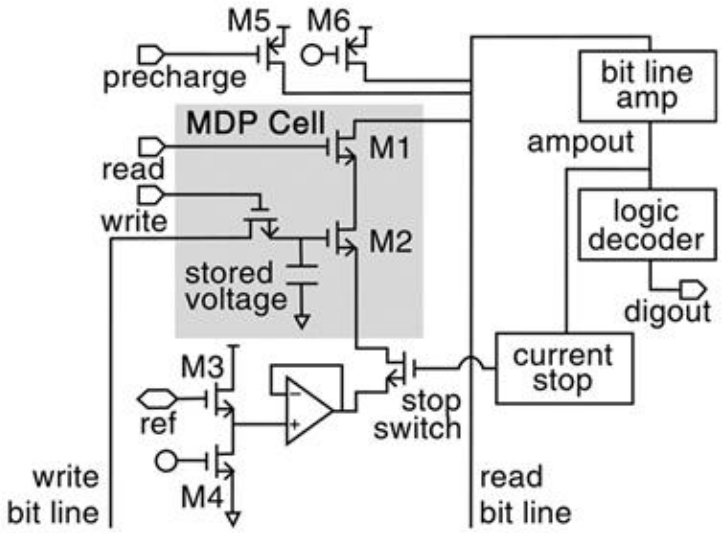


Figure 2.4 MDP bit cell architecture with current stop.

Figure 2.5 displays timing diagrams in BASE4 mode using the current stop feature. It is clear from these timing diagrams that the magnitude of the voltage transition on the read bit line is substantially smaller than the timing diagrams in Figure 2.3 without current stop. The current stop feature not only reduces power it also reduces read access time.

### 2.2.5 Read Power

Electrical power is defined as

$$P = I * v \tag{8}$$

where $I=C*v_{BL}*f$, $v=vdd$, $C$ is the bit line capacitance, $f$ is the frequency, and the bit line voltage swing, $v_{BL}$, is assumed to equal $vdd$ yielding the common equation

$$P = Cv^2 f. \tag{9}$$

In an MDP memory using current stop as previously mentioned, a typical $v_{BL}$=0.15V and $vdd$=0.9V, and so $vdd$ does not equal $v_{BL}$ and we use the more precise

$$P = C * v_{BL} * f * vdd \tag{10}$$

The power saved is determined by the ratio of read bit line voltage to $vdd$. Therein lies major read power savings.

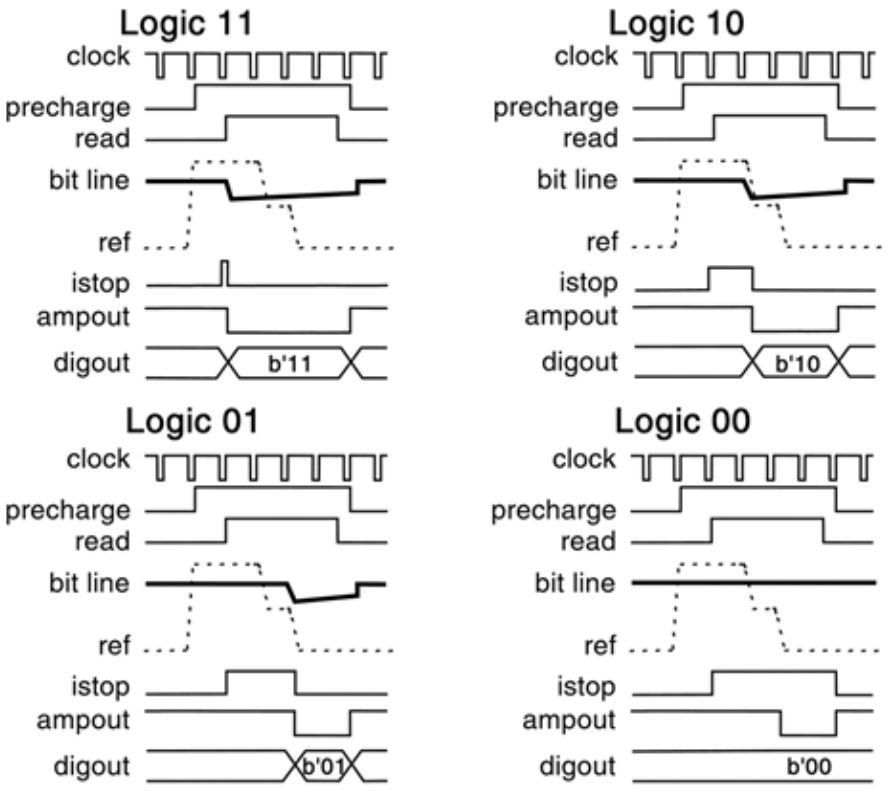


Figure 2.5 MDP bit cell architecture with current stop.

Assuming total power includes read, refresh and DC bias power we use

$$P_{rd\_tot} = i_{rd} vdd_r + i_{ref} vdd_w + \beta\alpha i_{DC} vdd_h + P_{per} \tag{11}$$

where $i_{rd}$ is the read current, $vdd_r$ is the read bit line clamp voltage, $i_{ref}$ is the refresh current, $vdd_w$ is the write bit line highest level, $\beta$ is a constant representing the read duty cycle, $\alpha$ is the corresponding refresh duty cycle, $i_{DC}$ is the DC bias current for the MDP opamps and bit line amplifiers, $vdd_h$ is an analog voltage supply and $P_{per}$ is the power in the row decoders, logic decoders, read channel and write channel peripherals. The read current is further defined as

$$i_{rd} = C_{BL} * v_{BL} * f_{rd} * k * m \tag{12}$$

and the refresh current is further defined as

$$i_{ref} = C_{BL} * (v_{BL} + v_{WR}) * f_{ref} * k * m \tag{13}$$

where $C_{BL}$ is the read bit line capacitance, $v_{BL}$ is the bit line voltage swing for read, $v_{WR}$ is the bit line voltage swing for write, $f_{rd}$ is the read frequency, $f_{ref}$ is the refresh frequency, $k$ is the number of columns of bit cells, and $m$ is the number of memory modules.

The refresh frequency is a fixed value based on the retention time and the number of rows being refreshed. Read power at lower frequencies is dominated by the refresh current and conversely read power at higher frequencies is dominated by the read current.

### 2.2.6 System Benefits of MDP Architecture

*Bit Line Amplifier:* The MDP principle is also applicable to bit line amplifiers and greatly reduces static power. It is therefore reasonable to have the amplifiers active while they are waiting for a read bit line transition.

The MDP principle in bit line amplifiers also increases noise performance by allowing global adjustment of the bit line amplifier switching voltage.

*Bit Cell:* MDP bit cell architecture is more robust with respect to noise. It has much larger read signal than 1T1C architectures because it is not subject to the attenuation caused by charge sharing between the bit cell and the bit line. MDP architecture also has much larger read signal than some gain cell designs [3] [6]. Instead of a short and finite read signal with a small maximum, MDP memory has an ever increasing read signal.

*Opamp:* The MDP op amp in Figure 2.1(b) and Figure 2.4 is multiplexed between groups of rows with the hold node transistor sources in each column tied together. We refer to one group of rows as a Common Source Domain, Figure 2.6. Figure 2.7 illustrates an opamp shared among multiple Common Source Domains.

The multiplexer limits the active number of bit cells driven by an opamp at any one time by enabling a reasonable capacitive load for the opamp. For example, assuming 0.6fF of parasitic capacitance for each bit cell and Common Source Domains made up of 32 rows and 512 columns of bit cells, the opamp drives a 10pF capacitive load for the 16,384 bit cells in the active Common Source Domain. A reasonable number of active and inactive bit cells multiplexed with a single opamp is 250,000.

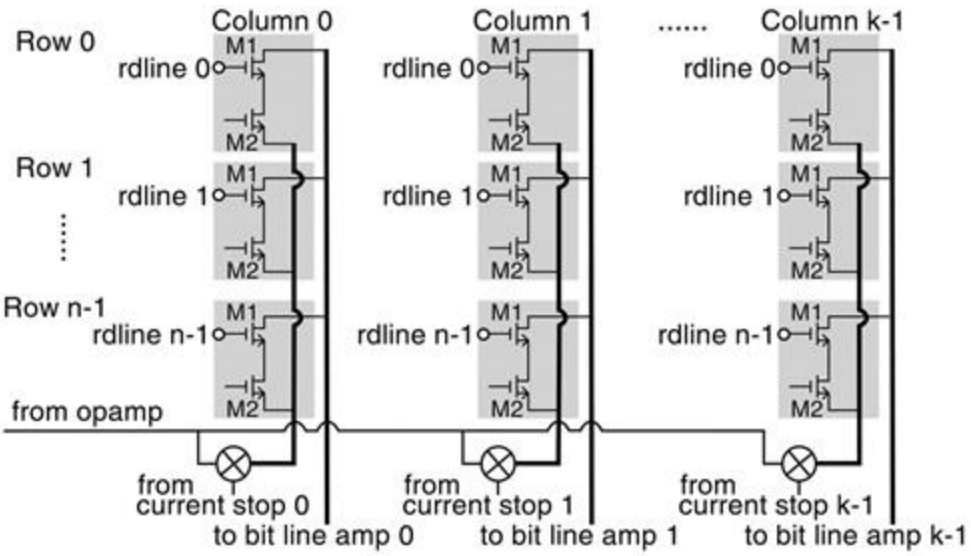


Figure 2.6 One Common Source Domain with n rows and k columns.

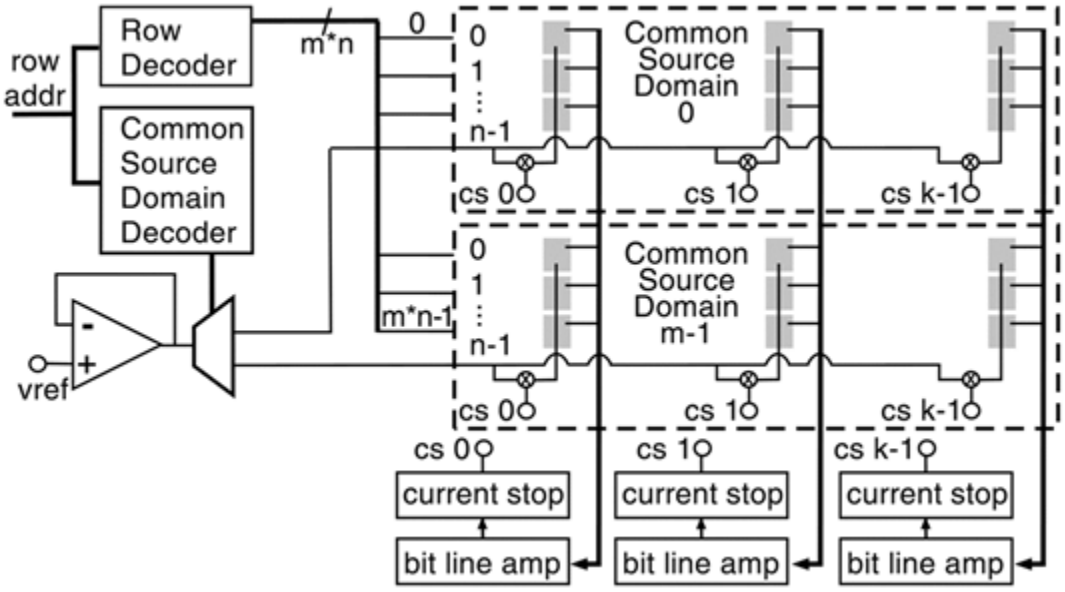


Figure 2.7 One opamp shared between m*n(k) bit cells.

In BASE2 for any frequency the output of the opamp is constant. In BASE4 at low frequencies the output of the opamp is stepping between three levels. In BASE4 at higher frequencies an opamp stepping quickly between three voltage levels needs a higher bias current than the power budget allows. In this case, three opamps are used with constant outputs instead of one opamp with a stepping output. Designing in this way allows for much more speed and there is no longer

a limit imposed by the finite step response of the opamp. Each of the three opamps working in BASE4 mode can work with the same quiescent current as opamps in the BASE2 mode.

## 2.3 Results

We simulated the MDP bit cell with Spectre to confirm (7). We compared the simulated performance of the MDP bit cell with simulated performance of a traditional gain cell. We fabricated an MDP test structure, simulated the test structure, and compared the system simulation results to the measured silicon results. Thus we established that our equations are verified by the MDP bit cell simulation, and our system simulation accurately models the MDP test structure. Finally, we used our analysis as the basis for calculating power for larger 1T, 3T gain cell and MDP BASE2 and BASE4 memory systems. Our results are divided into four sections: a comparison of analysis and bit cell simulation, a comparison of system simulation and silicon, estimated power usage for the four types of larger memory systems, and comparison to other state of the art eDRAM.

### 2.3.1 Analysis and Simulation

Our bit cell simulations were divided into two groups using the input parameters in Table 2.1. In the first group of simulations we measured the read signal. We applied a read pulse of 100 ns which equates to a frequency of 10 MHz, set the threshold voltage variance to 0mV, varied the storage node voltage and measured read signal for the MDP bit cell of Figure 2.1(b) in both BASE2 and BASE4 modes across the stated range of storage node voltages. Graphs in Figure 2.8 plot the read signal of (7) compared to the MDP BASE2 and MDP BASE4 bit cell simulation read signal, and illustrate the bit cell simulations closely resemble the equation.

Read signal results from the first group of bit cell simulations are also graphed in Figure 2.9 for a standard gain cell and a MDP BASE2 bit cell across the range of threshold variances in Table 2.1 and illustrate the effect that threshold variance has on design and write power requirements.

TABLE 2.1 Bit-cell simulation parameters

| | Gain Cell<br>Fig. 4.1(a) | MDP<br>Fig. 4.1(b) |
|---|---|---|
| process | 0.18µm | 0.18µm |
| threshold voltage | 0.6V | 0.6V |
| bit line capacitance | 0.1pF | 0.1pF |
| threshold variances | -100mV<br>0mV<br>+100V | -100mV<br>0mV<br>+100mV |
| required read signal | 0.3V | 0.15V |
| logic 0 max BASE2 | 0V | 0.55V |
| Ref. voltage BASE2 | 0V | 0.60V |
| Ref. voltages BASE4 | n/a | 0.55V<br>0.85V<br>1.15V |
| storage node - read signal tests | 0.15V to 0.60V | 0.65V to 0.80V |
| storage node - frequency tests | 0.3V to 1.05V | 0.75V to 1.05V |
| read pulse - read signal tests | 100 ns | 100 ns |
| read pulse - frequency tests | 0 ns to ∞ | 0 ns to ∞ |
| write transition defined as | storage node – logic 0 max | storage node – logic 0 max |

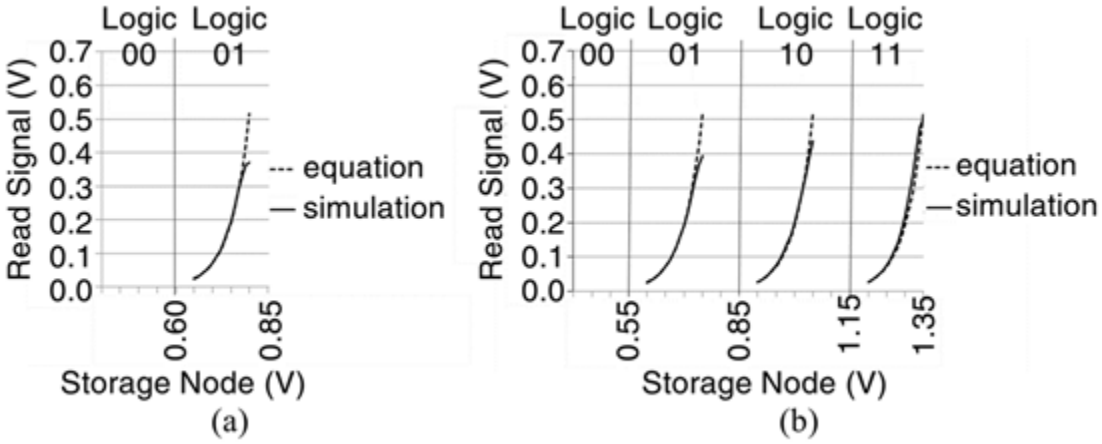


Figure 2.8 Simulation of read signal verifies equation of (a) MDP BASE2 and (b) MDP BASE4 at 100ns read pulse.

According to the values in Table 2.1, we graphed the read signal with respect to the write bit line transition, or swing, necessary to change from logic 0 to logic 1. Figure 2.9(a) illustrates the impact threshold variance has on the logic 1 minimum voltage in a standard gain cell. With 0V threshold variance the gain cell logic 1 minimum needs to transition the write bit line about 0.44V to attain the desired read signal. But by accommodating a typical +100mV die-to-die

variance, the write bit line transition must be approximately 0.11V greater, or about 0.55V, to attain the required read signal. Thus in the gain cell the die-to-die threshold variance requires the write bit line transition an additional 0.11V in order to accommodate variance. In this way the threshold variance has a direct impact on the gain cell write and refresh power consumption.

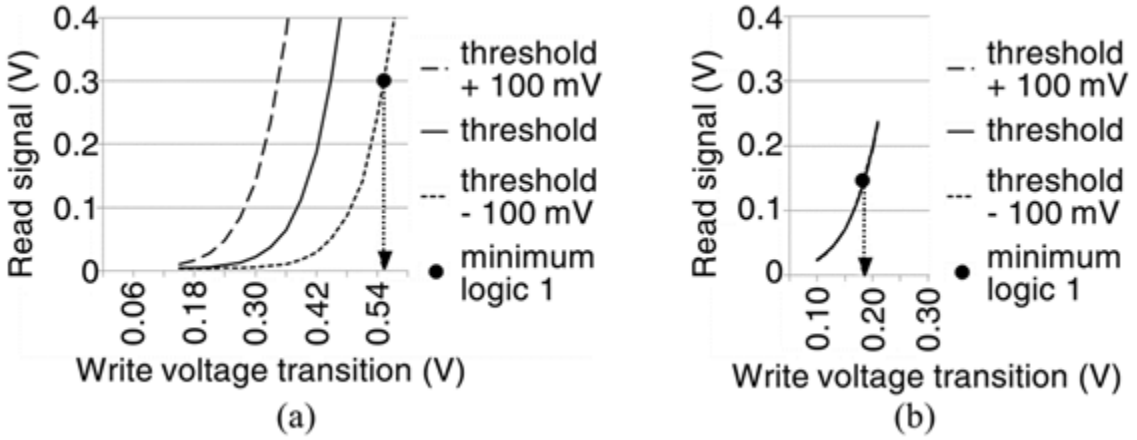


Figure 2.9 Simulated impact of threshold voltage variance on minimum logic_1 and write voltage transition at 100ns for (a) the gain cell and (b) the MDP BASE2 bit cell.

In contrast to Figure 2.9(a), Figure 2.9(b) shows the lack of effect the threshold variance has in an MDP bit cell. As expected from (1), the MDP graph shows no difference in the read signals over the range -100mV to +100mV of threshold variance. The three simulation curves are directly on top of each other. The logic 1 minimum in the MDP simulation requires only about a 0.18V write transition, regardless of the threshold variance. The write bit line transition necessary to change logic states is less than half compared to the gain cell. In this way the MDP bit cell uses approximately 50% less average power to write the bit cell.

In the second group of bit cell simulations we measured delay and compared the gain cell to the MDP BASE2 bit cell. We set values according to Table 2.1, including the threshold voltage variance and storage node, turned on the read pulse and measured the elapsed time it took to reach the desired read signal and so determined the operating frequency for any particular logic 1 minimum voltage. The simulation data graphed in Figure 2.10 for a standard gain cell and for a MDP BASE2 bit cell illustrate the effect threshold variance has on frequency and write power

requirements. The change in bit line voltage necessary to write a logic 1 was graphed on the x-axis to depict power requirements. Figure 2.10 illustrates the tradeoff between frequency and write power; as frequency increases the power necessary to write or refresh the bit cell increases. Figure 2.10(a) for the gain cell shows the read frequency falls off rapidly as write voltage transition lowers and varies sometimes as much as two orders of magnitude due to die-to-die threshold variance. Figure 2.10(b) shows the frequency falls off only modestly for the MDP bit cell as write transition voltage lowers and the threshold variation has virtually no effect. For example, operating at 100 MHz the gain cell needs to transition about 0.72V on the write bit line while the MDP bit cell need only transition about 0.34V and saves power accordingly.

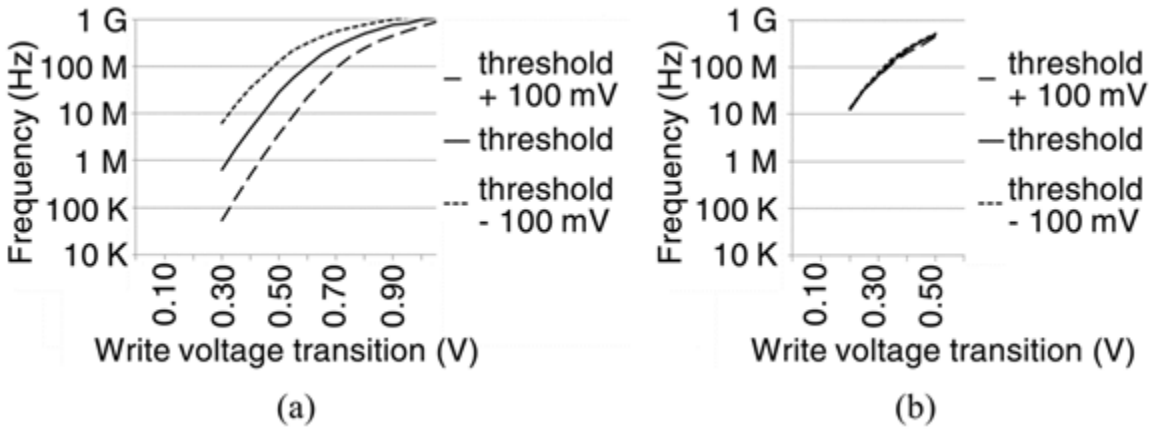


Figure 2.10 Simulated impact of threshold voltage variance on write voltage transition at various frequencies for (a) the gain cell and (b) the MDP bit cell.

### 2.3.2 Simulation and Silicon

We fabricated an MDP memory test structure die in a 180 nm CMOS process to prove the modified differential pair construct with and without current stop. We verify silicon waveforms closely resemble system simulation waveforms. Layout and bit cell dimensions are in Fig. 2.11. The test structure die consists of one MDP module with two columns and eight rows for a total of 16 bit cells. There is a row decoder on the die, a reference transistor, and a transistor for each column to implement the current stop switch. We wire-bond the die to a board that also has three discrete opamps. One opamp acts as a source follower interacting with the bit cells to make up the modified differential pair construct and the other two opamps buffer the bit lines. There are

also transistors on the board to supply the bias current for the bit lines and clamps to clamp the bit lines to the voltage level *vddr* when the precharge input is asserted LO. There are DACs to generate the write bit line and read reference voltages. The comparators and logic to control the current stop feature are also on the board.

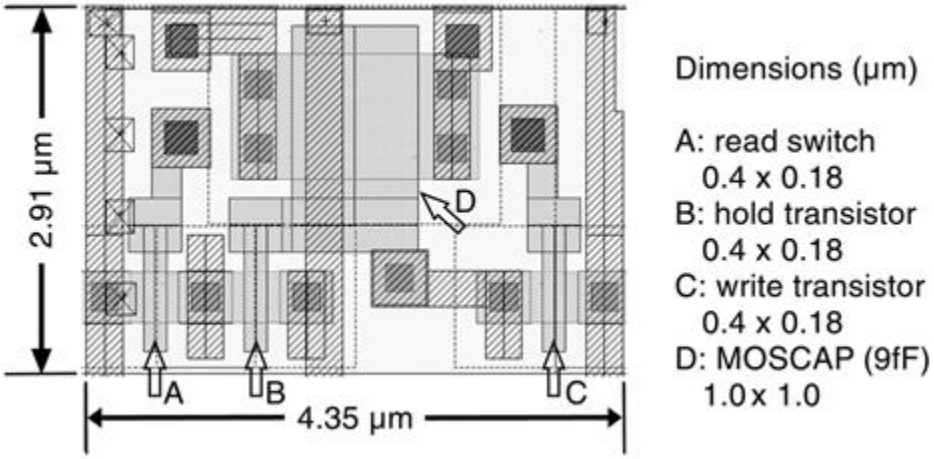


Figure 2.11 Layout dimensions of the bit cell.

We create a system simulation schematic with appropriate models to emulate both the test structure die and the PCB functionality. We generate vectors and import them into the pattern generator on a logic analyzer. We use the vectors from the pattern generator to stimulate the circuit under test on the bench and the same vector values to stimulate the system simulation.

Fig. 2.12 contains both simulation waveforms and oscilloscope waveforms from the silicon tests grouped by logic value. Waveforms with and without current stop are in the figure. The logic 00 case is trivial and is not included. The images without current stop clearly show the bit line voltage stepping down into the ohmic region and the images with current stop clearly show the impact of terminating the voltage change. All images show the silicon behavior matches the system simulation.

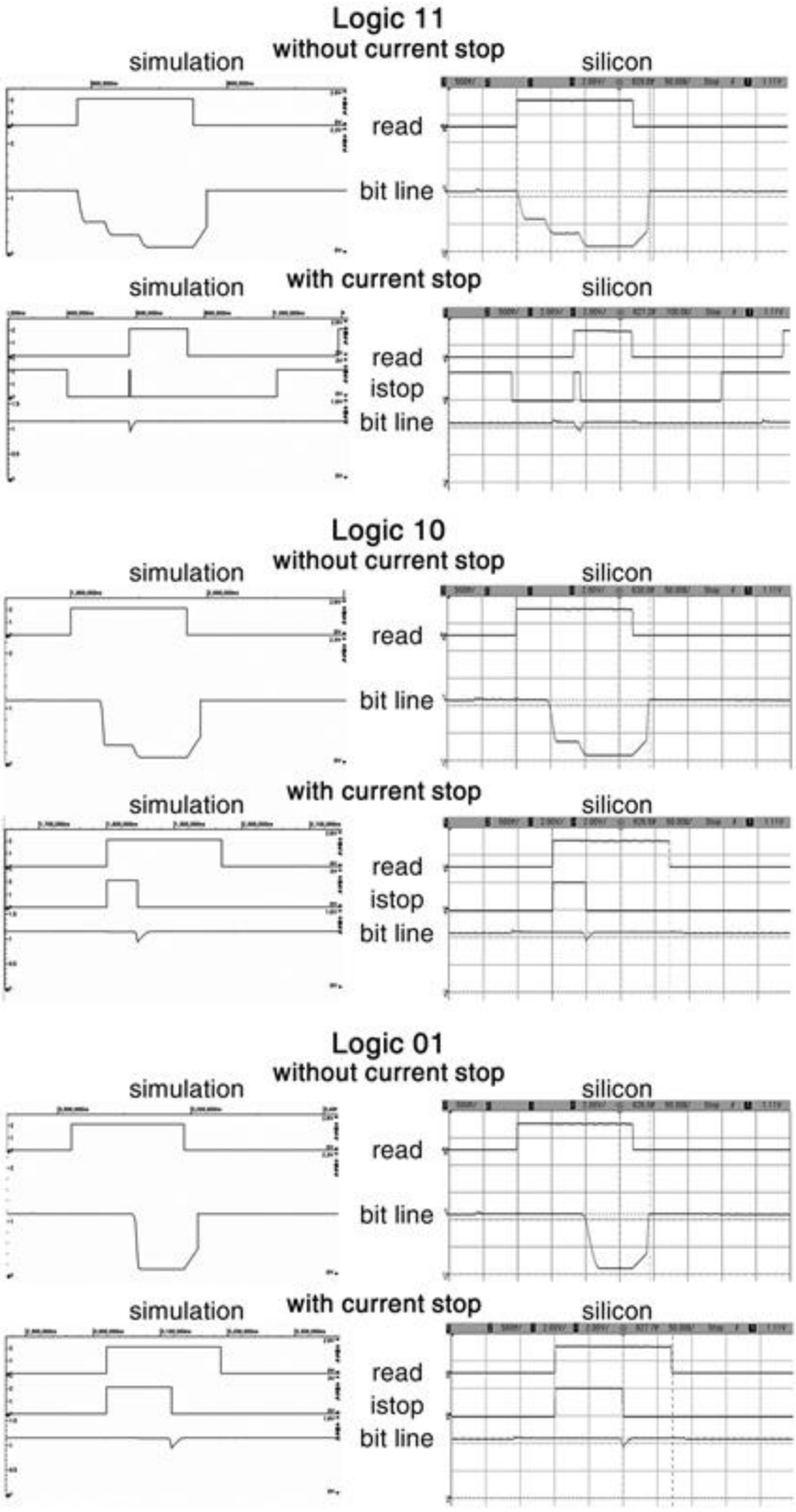


Figure 2.12 Silicon behavior matches system simulation.

We measure the worst case retention time at room temperature in both BASE2 and BASE4 modes. A common voltage for each bit cell is placed on the hold node by writing to each of the bit cells. The time between the write and the reading of each cell is gradually increased until the first read failure in any one of the cells occurs. In this way, the worst case retention time is observed for the test structure. The worst case retention time was measured to be 150ms for BASE2 and 75ms for BASE4 at room temperature.

### 2.3.3 Read Power Comparison

Using (11) and values for the equation variables given in Table 2.2 we calculate the total power to read 16-bit words in both random access and pipelined reads for the 1T1C, 3T gain cell, MDP BASE2 and MDP BASE4 memory architectures. Fig. 2.13 and Fig. 2.14 summarize the results.

In Table 2.2 a conservative retention time, three times less than the value measured, is used. The value of the power due to the peripherals is verified small, much less than 5%, compared to the bit line power in the system; consequently, the value of $P_{per}$ in Table 2.2 is assumed zero. The power in the opamps and the bit line amplifiers are factored in with the term $i_{DC}$ in Table II.

The insensitivity to the threshold voltage and reduced voltage swing on high capacitance bit lines greatly reduce the total power for the MDP memory compared to the 1T1C and the 3T gain cell. For example, an MDP BASE4 pipelined read of 16-bit words at 630MHz uses 90µW and the 1T1C pipelined read uses 2276µW. At 630MHz the MDP BASE4 power is 96% less than 1T1C. An MDP BASE2 random access read of 16 bit words at 630 MHz uses 419µW and the 3T gain cell uses 2458µW. At 630 MHz the MDP BASE2 power is 82% less than the 3T gain cell.

### 2.3.4 eDRAM Comparison

Table 2.3 compares bit cell size normalized to technology node, bit cell storage capacitance, our measured results, simulation and analysis to eDRAM in the literature.

Our MDP BASE2 analysis in 180 nm at 630 MHz predicts 0.8µW/Mbit refresh power at 25C. The work of [3] in 65 nm at 667 MHz reports 109 µW/Mbit refresh power at 85C and the work of [5] in 180 nm at 330 KHz reports 5.4 µW/Mbit refresh power at 25C. We standardized the

comparison in Fig. 2.15 using the power-delay product. This graph shows the MDP relative energy per bit is more than three times lower than the work in [3] and more than 10000 times lower that the other cited works.

TABLE 2.2 Read power parameters used in (11)

| $P_{rd\_tot}$ (11) | 1T | 3T | MDP BASE2 | MDP BASE4 |
|---|---|---|---|---|
| Process | 0.18µm | 0.18µm | 0.18µm | 0.18µm |
| $C_{BL}$ | 0.1pF | 0.1pF | 0.1pF | 0.1pF |
| threshold voltage | 0.6V | 0.6V | 0.6V | 0.6V |
| $v_{BL}$ | 1.5V | 0.9V | 0.15V | 0.15V |
| $v_{WR}$ | 1.5V | $f_1(freq)^a$ | $f_2(freq)^b$ | 0.85V |
| $i_{DC}$ | 0 | 0 | 4 µA | 2 µA |
| $vdd_h$ | n/a | n/a | 1.8V | 2.0V |
| $vdd_r$ | 1.5V | 0.9V | 0.9V | 1.1V |
| $vdd_w$ | 1.5V | $f_3(freq)^a$ | $f_4(freq)^b$ | 1.45V |
| $f_{rd}$ pipelined | 1Hz to 1GHz | 1Hz to 1GHz | 1Hz to 1GHz | 1Hz to 1GHz |
| $f_{rd}$ random access | 1Hz to 20MHz | 1Hz to 1GHz | 1Hz to 1GHz | 1Hz to 17MHz |
| $f_{ref}$ (1440 rows) | 28.8K | 28.8K | 28.8K | 57.6K |
| retention time | 50 ms | 50ms | 50ms | 25ms |
| k | 48 | 48 | 48 | 48 |
| m pipelined | 16 | 16 | 16 | 8 |
| m random access | 1 | 1 | 1 | 1 |
| β | n/a | n/a | 100ns*f | 100ns*f |
| α | n/a | n/a | 150ns*f | 150ns*f |
| $P_{per}$ | 0 µW | 0 µW | 0 µW | 0 µW |

[a] $f_1$ and $f_3$ are in Figure 2.9(a); [b] $f_2$ and $f_4$ are in Figure2.9(b)

| MHz | 1T (µW) | 3T (µW) | MDP BASE2 (µW) | MDP BASE4 (µW) |
|---|---|---|---|---|
| 0.01 | 5.0 | 1.2 | 0.8 | 2.6 |
| 25 | 95 | 34 | 7 | 6 |
| 250 | 909 | 328 | 59 | 38 |
| 400 | 1438 | 518 | 94 | 58 |
| 630 | 2276 | 820 | 144 | 90 |

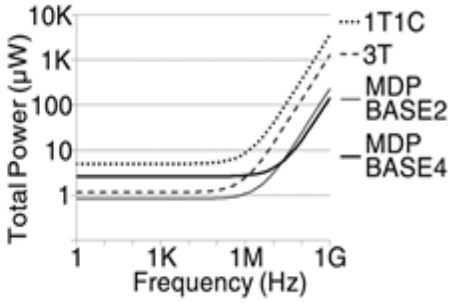


Figure 2.13 Pipelined total read power.

| MHz | 3T (µW) | MDP BASE2 (µW) |
|---|---|---|
| 0.01 | 1.2 | 0.8 |
| 25 | 100 | 25 |
| 250 | 980 | 172 |
| 400 | 1552 | 268 |
| 630 | 2458 | 419 |

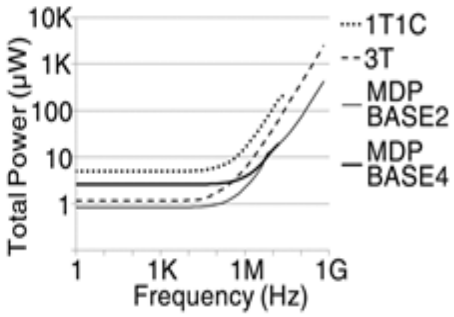


Figure 2.14 Random access total read power.

## 2.4 Conclusion

The memory system in this work successfully reads and writes BASE2 and BASE4 values to the bit cells with no observable errors proving good noise margin even with large capacitive loads from the pads on the bit lines. The measured silicon and the bit cell simulations prove the MDP memory concept and show that the assumptions made for the large memory system power calculations are reasonable.

The modified differential pair architecture eliminates the effect of die-to-die threshold voltage variance in the memory system. This has two major effects on system operation. It lowers write bit line voltage transition by more than 0.3V and enables four logic values per cell. The current stop feature has the additional effects of reducing read bit line voltage transition down to 150mV and reducing read access time accordingly. These reductions to the write and read bit line voltage transitions result in 90µW active power for 16 bit pipelined reads in a BASE4 MDP at 630 MHz compared to 2276µW in 1T1C, a power savings of 96%. The multi-bit capability results in increased capacity by up to 50% over standard 3T gain cells. We verified our equation describing read signal with our bit cell simulation and verified our system simulation with our test silicon. The retention time measurements of the test structure quantify the worst case bit cell leakage. This work, with a projected refresh power of 0.8µW/Mbit and a random access frequency of 630 MHz, compares favorably with reported results from the literature. Implementing this eDRAM memory in a larger memory system and smaller technology node and applying the architecture to SRAM are topics for further research.

TABLE 2.3 EDRAM topology comparison

| | MDP | [1] 1T | [2] 1T | [3] 2T | [4] 3T | [5] 2T | [6] 3T |
|---|---|---|---|---|---|---|---|
| Process (nm) | 180 | 150 | 180 | 65 | 90 | 180 | 90 |
| VDD (V) | 0.9 | 1.5 | 1.8 | 1.1 | 0.5 | 0.75 | 1.2 |
| Random Access Freq. (MHz) | 630[a] | 105 | 10 | 667 | 5 | 0.33 | 50[a] |
| Retention Time (ms@25C) | 150 | 0.1[c] | x | 0.11[d] | 0.001 | 300 | x |
| Refresh Power (uW/MBit @25C) | 0.8[b] | 4200 | x | 109[d] | 90 | 5.4 | x |
| Active Read Power @5MHz (uW) | 4[b] | x | x | x | 750 | x | x |
| Active Read Power @105MHz (uW) | 100[b] | 2460 | x | x | x | x | x |
| Refresh Power Delay Product (1e-21 J/bit) | 1.27 | 40K | x | 4.5[e] | 18K | 16K | x |
| Size with MOSCAP ($F^2$/bit) | BASE 2: 390 BASE 4: 195 | 48 | x | x | x | x | x |
| Size no MOSCAP ($F^2$/bit) | BASE 2: 140[f] BASE 4: 70[f] | x | x | 65 | 120 | 103 | x |
| Bit Cell Storage Capacitance with MOSCAP (fF) | 9 | 5 | x | < 1 | < 1 | < 1 | x |
| Bit Cell Storage Capacitance no MOSCAP (fF) | < 1 | x | x | < 1 | < 1 | < 1 | x |

**(a)** result is from simulation; (b) result is from analysis; (c)@ 110C; (d) @85C;
(e) Adjusted from 85C to 25C by 0.5/11C factor;
(f) estimated using implemented layout design rules

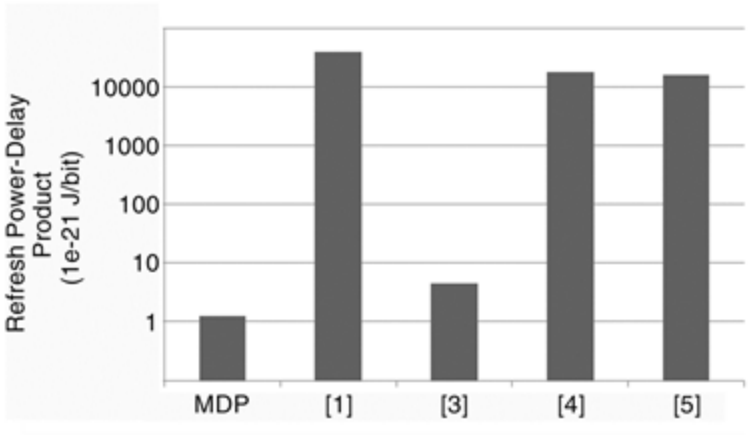


Figure 2.15 eDRAM standardized energy comparison.

## 2.5 References

[1] D. Somasekhar, S. Lu, B. Bloechel, G. Dermer, K. Lai, S. Borkar and V. De, "A 10Mbit, 15GBytes/sec Bandwidth 1T DRAM Chip with Planar MOS Storage Capacitor in an Unmodified 150nm Logic Process for High-Density On-Chip Memory Applications", in Proc. of ESSCIRC, Grenoble, France, 2005, pp. 355-358.

[2] J. C. Koob, S.A. Ung, B F. Cockburn, and D. G. Elliott, "Design and Characterization of a Multilevel DRAM", IEEE Trans. VLSI Syst., vol. 19, no. 9, pp. 1583-1596, Sep. 2011.

[3] K. C. Chun, P. Jain, T. Kim, and C. H. Kim , "A 667 MHz Logic-Compatible Embedded DRAM Featuring an Asymmetric 2T Gain Cell for High Speed On-Die Caches", IEEE J. Solid-State Circuits, vol. 47, no. 2, pp. 547-559, Feb. 2012.

[4] M. Ichihashi, H. Toda, Y. Itoh, and K. Ishibashi, "0.5V Asymmetric Three-Tr. Cell (ATC) DRAM Using 90nm Generic CMOS Logic Process", in Proc. IEEE Symp. VLSI Circuits, 2005, pp. 366-369.

[5] Y. Lee, M. Chen, J. Park, D. Sylvester, D. Blaauw, “A 5.4nW/kB Retention Power Logic-Compatible Embedded DRAM with 2T Dual-Vt Gain Cell for Low Power Sensing Applications”, Proc. IEEE A-SSCC, 2010.

[6] M. Khalid, P. Meinerzhagen, and A. Burg, “Replica Bit-Line Technique for Embedded Multilevel Gain-Cell DRAM”, in Proc. of IEEE NEWCAS, Jun. 2012, pp. 77-80.

[7] J. Lynch, “Memory architecture with a current controller and reduced power requirements”, U.S. Patent 8 169 812, May 1, 2012.

[8] E. Sanchez-Sinencio and A. Andreou, “A Current-Based MOSFET Model for Integrated Circuit Design”, Low-Voltage/Low-Power Integrated Circuits and Systems, 1st ed., Piscataway, NY, IEEE Press, 1999, ch.2, sec. 2, pp. 9-21.

[9] P. Kinget, “Device Mismatch and Tradeoffs in the Design of Analog Circuits”, IEEE Journal of Solid-State Circuits, Vol. 40, No. 6, pp. 1212-1224, Jun 2005.

# CHAPTER 3. 131K-BIT LOGIC-COMPATIBLE LOW POWER CURRENT CONTROLLED DRAM ARRAY

## 3.1 Introduction

It is well known that the 1T1C DRAM bit cell architecture has the following drawbacks [1]:
(i) The read operation is destructive. The bit cell must be re-written after the read operation that requires extra cycles and a full signal swing on high capacitance bit lines.
(ii) The 1T1C arrays require additional processing technology to implement high-density capacitance; hence, they are not easily embeddable.
(iii) Due to the need for charge sharing to create the read signal, the read signal amplitude is reduced and the noise sensitivity is high.
(iv) The bit line capacitance, which is heavily layout dependent, requires complicated analysis and layout crossovers to reduce the impact of inter-wire cross coupling.

Gain cells solve the problems associated with the 1T1C architecture but increase the cell size and reduce the capacity. Although the industry left the gain cell architecture more than thirty years ago, new research on logic-compatible gain cells is becoming prevalent [2][3]. Gain cells, however, suffer from dependence of their currents on the threshold voltages of transistors, leading to their large performance variations at process corners.

In [4] analysis, simulation, and silicon test results of a logic-compatible, current controlled memory architecture were presented that improves or overcomes the drawbacks of 1T1C and gain cell architectures. The new architecture effectively reduces read power requirements and increases memory capacity while maintaining comparable read access times.

The proof of concept die in [4] successfully read from and wrote to the bit cells with no observable errors proving good noise margin even with large capacitive loads from the pads on the bit lines. The bit cell simulations and the measured silicon proved the modified differential pair construct. The retention time measurements of the test structure quantified the bit cell leakage; however, since the proof of concept die only contained 2x8 DRAM bit cells, the size was too small to provide the silicon proof for real applications. Additionally, peripheral blocks

such as the row decoders, the sense amplifiers, and the current control opamps were not part of the chip design.

To provide a design large enough to give the necessary silicon proof, the current controlled memory architecture was implemented and taped-out as a 131K bit array in an 180nm CMOS process.

Section 3.2 provides some background about the original proof of concept design in [4]. In Section 3.3, we present the implementation of the array and highlight some of the major problems that were solved: keep the peripheral logic power small with respect to the power in the bit lines, and design the opamps within the power budget. In addition, the implementation also incorporates a current controlled sense amplifier, and a dynamic refresh algorithm. In Section 3.4, results from simulation and from measuring a packaged part with the memory array are presented. The test board schematic, the test setup, simulated and measured results are discussed. Finally, Section 3.5 gives the conclusion.

## 3.2 Background

Figure 3.1 is a schematic illustrating the current controlled memory architecture in [4] that uses a modified differential pair (MDP) construct to eliminate the unpredictable effect of threshold voltage variance. The MDP construct works as a current controller made up of the reference transistor, M0, with its source buffered by a source follower opamp that controls the current of the read transistor, M2. The current controller removes the effect of within-die threshold voltage in the currents of the read transistors in the bit cells because the within-die threshold voltage of the transistors M0 and M2 are equal to each other within a specified tolerance and cancel each other out. The opamp supplies sufficient current to drive multiple bit cells from the reference transistor.

Tighter control of the read currents in the MDP architecture decreases refresh power because the highly capacitive write bit line voltage transitions are lowered by more than 0.3V [4].

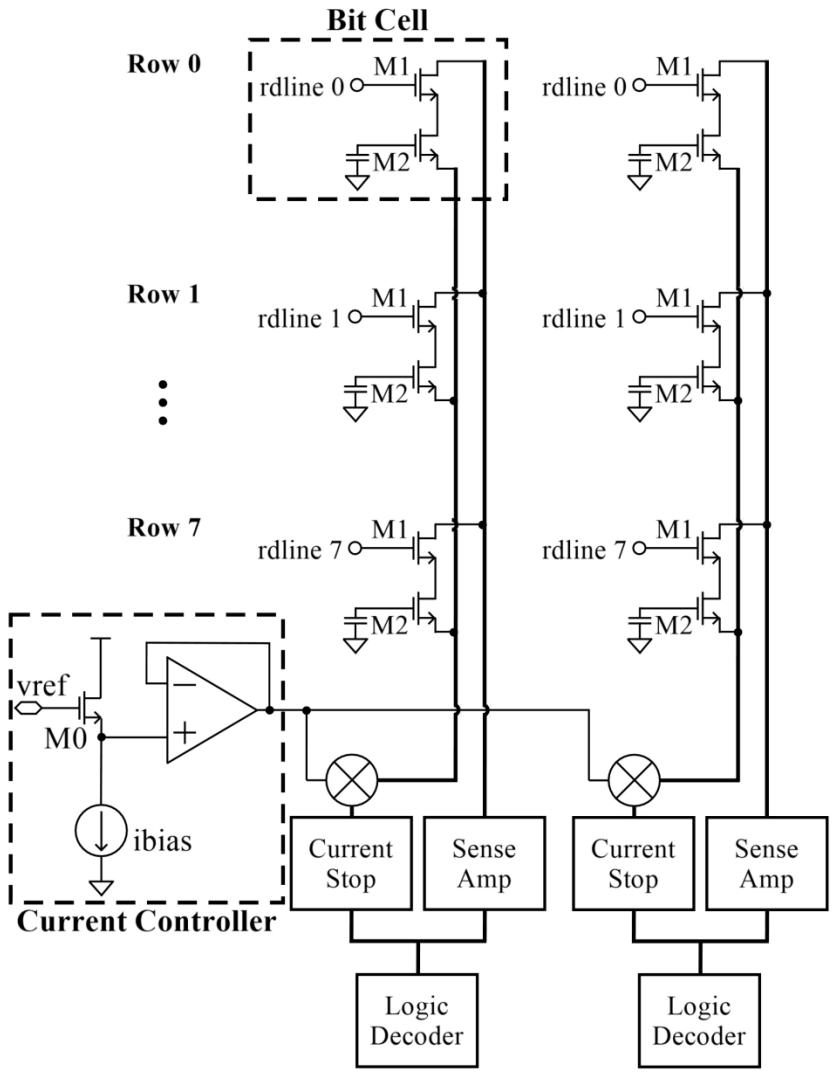


Figure 3.1 Schematic illustrating the current controlled memory architecture.

The MDP architecture also enables reading multi-bit logic levels per cell. The bit cell can operate in either BASE2 mode, two logic levels stored on the bit cell, or BASE4 mode, four logic levels stored on the bit cell.

In BASE4 mode, the stored voltage is sequentially compared to stepping reference voltages. The sense amplifier in Figure 3.1 acts as a comparator and outputs a digital indicator to the logic decoder. The logic decoder uses the indicator, specifically the point in the sequence of comparisons where the indicator is asserted, to discern the digital value represented by the voltage on the storage node.

The current controlled memory architecture also includes a current stop construct to significantly reduce the voltage swing on the high capacitance read bit lines [5]. The current stop construct limits the read bit line transitions on the highly capacitive bit lines to around 0.2V. The current stop block senses a significant change on the read bit line as indicated from the output of the sense amplifier, Figure 3.1, and generates a current stop pulse that stops the current in the bit cell. Hence, power on the high capacitance bit lines is proportional to VDD * Vbl instead of VDD squared.

The calculations in [4] concluded that the reductions in the write and read bit line voltage transitions result in power savings of 96% compared to a comparable 1T1C architecture [6].

## 3.3 131K-Bit DRAM Chip Implementation

This section discusses some of the issues and features that are relevant to the design of the memory array.

Each bit cell on the memory chip has a read switch, a hold transistor, a write transistor and a MOSCAP. The memory chip has 128 columns and 512 rows for a total of 65,536 BASE4 bit cells. Logic for a read channel and a write channel are also on the chip. The chip is packaged into an 84 pin PLCC.

### 3.3.1 Driving a Reasonable Capacitive Load

As seen in Fig. 1 and described in [4], the modified differential pair (MDP) opamp in the current controller is connected to a number of bit cell hold node transistor sources through the current stop switch in multiple columns, each column having multiple rows. It is advantageous to limit the number of bit cells driven by the opamp at any one time thereby enabling a reasonable capacitive load for the opamp. To do so, the opamp in the current controller is shared among a group of rows called a common source domain (CSD).

Figure 3.2 shows a block diagram for the top half of the chip. There are four CSDs driven by one current controller on the top half. The addressing scheme activates a single row in a single CSD using a multiplexor to select the CSD and a row decoder to select the row. The multiplexor limits

the number of bit cells being driven by the opamp in the current controller at any one time thereby enabling a reasonable capacitive load seen by the opamp. The bottom half of the chip is a duplicate of the top half.

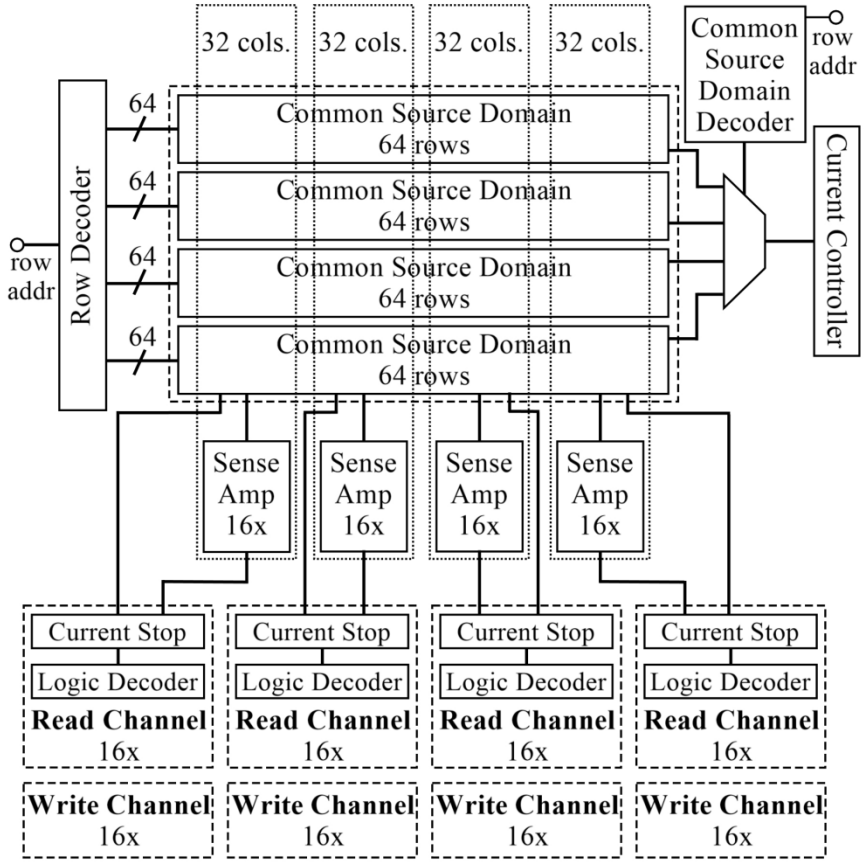


Figure 3.2 Top level chip implementation for 131K DRAM with top array shown and bottom array assumed to be mirror image with respect to top.

### 3.3.2 Reducing Area of Bit Lines and Reducing Number of Sense Amplifiers

In order to save area and also be able to reduce the total number of sense amplifiers by a factor of two, the bit cells in the memory array share the read and write bit lines, effectively creating two independent memory banks. Figure 3.3 depicts the configuration of the bit cell array that illustrates the sharing of the read bit lines creating a so-called side-a and a side-b. Each column has two bit cells, only one of which is activated at a time. Either all of the a-cells are active or all of the b-cells are active. The current controller output is connected to the sources of the bit cells

through the current stop switches to whichever side is active, either side-a or side-b. There is a single sense amp, current stop block, and logic decoder for each column.

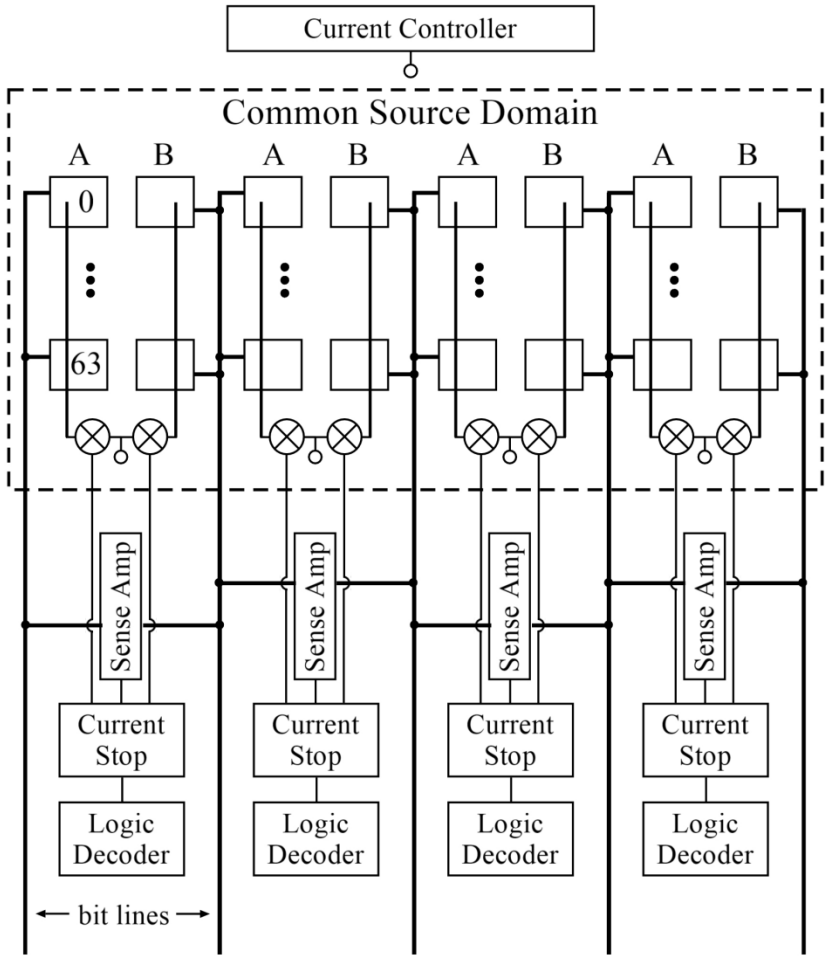


Figure 3.3 – Diagram illustrating the sharing of bit lines.

### 3.3.3 BASE4 Operation at Higher Frequencies

Figures 3.1, 3.2, and 3.3 show a single current controller opamp with a single output driving through the current stop switches into the bit cell sources. The current controller generates the necessary reference levels. For BASE2 the output of the opamp is constant, as there is only one reference level, and has acceptable performance. In BASE4 however, the output of a single opamp steps between the three reference levels and performance is only acceptable at low frequencies. At higher frequencies the opamp stepping quickly between three voltages needs a

higher bias current than fits within the power budget. Therefore an alternate design is used replacing the single op amp with three op amps each driving one of the three reference levels. Figure 3.4a illustrates the three opamps. Such a methodology gives good performance, but at the expense of a more complicated layout. Three wires are piped around to the common source domains instead of just one.

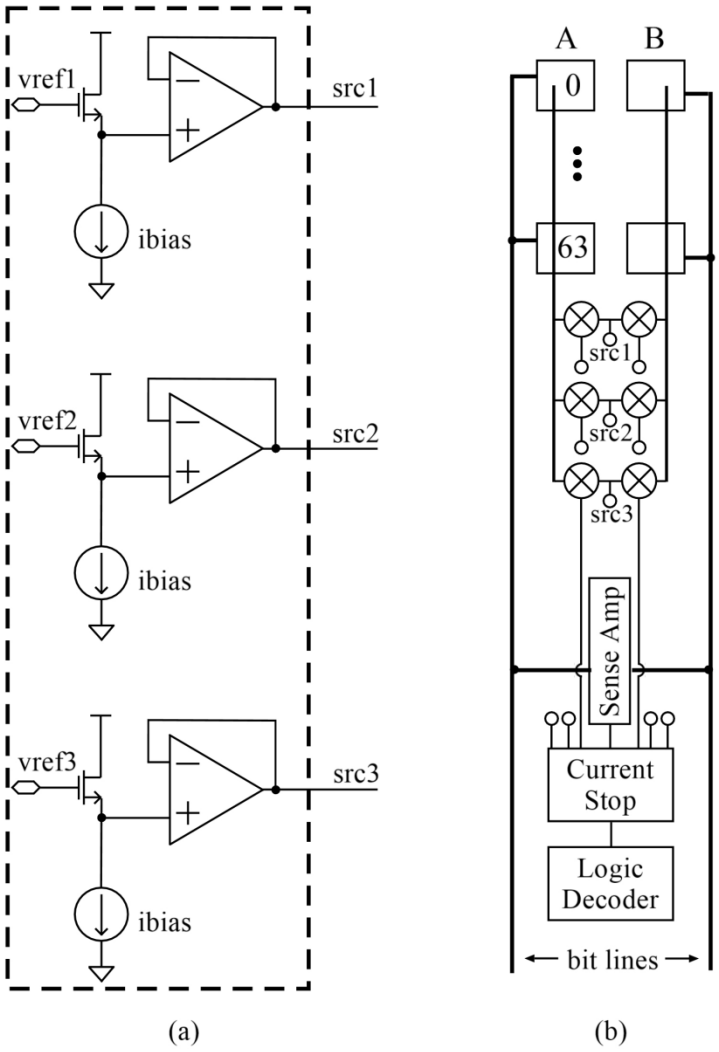


Figure 3.4 (a) Three opamps have three distinct DC references instead of a single opamp stepping between the three levels. (b) Three current stop switches are necessary to multiplex into each bit cell source being driven.

In addition to a current controller with three opamps, there also must be a current stop switch for each reference level as is depicted in Fig. 4b. The current stop multiplexes the three opamp reference signals so that the bit cell source sees a fast stepping of the reference levels.

### 3.3.4 Current Controlled Sense Amplifiers

The modified differential pair design principle used in the memory bit cell is also applied to the sense amplifier to reduce static power. The sense amplifier is enabled when a current stop signal is de-asserted and stays on until a transition on the read bit line is sensed; or in the case of a low value on the hold node in the active bit cell, a read bit line transition is not sensed.

Advantages of the sense amplifier design in this work are:
*(i)* A pre-determined digital threshold for the sense amplifiers is adjusted globally allowing for increased noise performance.
*(ii)* The static power for each sense amplifier is kept low, on the order of nAs while waiting to make a decision about the state of the read bit line.
*(iii)* Since the sense amplifier is kept active while waiting for the state of the bit line to change, the timing performance is superior to other types of memory sense amplifiers. An actual event triggers the resulting action. A separate event from a dummy timing generator is not set up to trigger the sense amp, the way some SRAMs and DRAMs do [7].

A simplified schematic of a current controlled sense amplifier is given in Figure 3.5. M0, M2 and the op amp form the modified differential pair. A global reference current controller is depicted driving multiple sense amplifiers. Assume current with value *ibias* is flowing in M0. A reasonable value for this current is 50nA. There are four different states in the operating conditions for the circuit.

Figure 3.6 shows a timing diagram and the state definition. The precharge condition is defined as STATE0. The *pc_n* input signal is LO and the read bit line is clamped to *vddr* with the transistor M1. The static current in the sense amplifier is zero because the current stop switch is off. The current stop switch is activated with the *istop* logic signal. When the *istop* logic signal is HI, the transistor M2 and the current compare block are shut off and have zero static current. While still

in STATE0, the output of the current compare block, *ampout_b*, is initialized internally to force the state of the output, *ampout_b*, to LO.

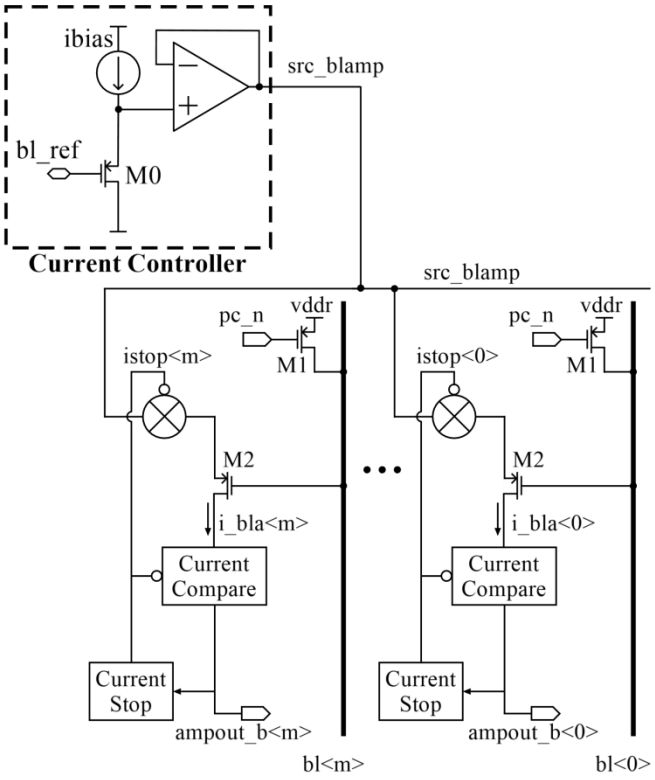


Figure 3.5 A simplified schematic of current controlled sense amplifiers.

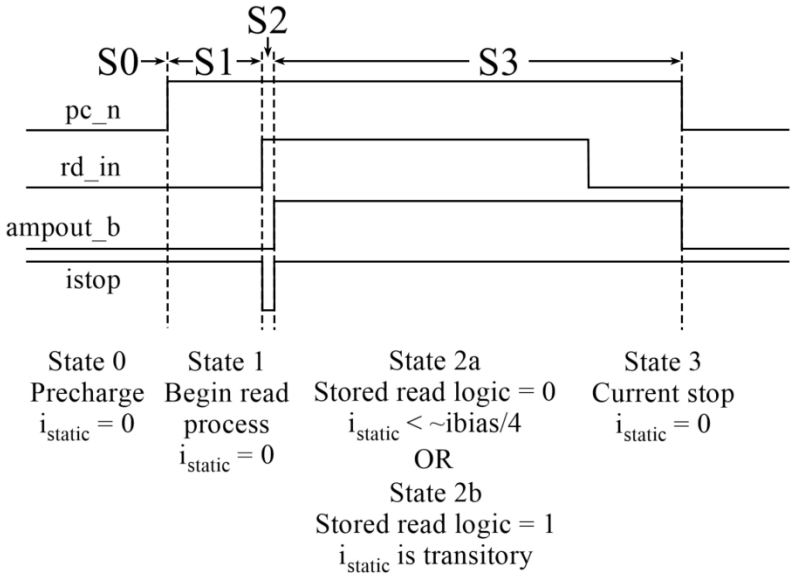


Figure 3.6 Timing diagram and state definition for current controlled sense amplifier.

The read process begins with STATE1 and occurs when the *pc_n* signal is de-asserted HI. As a result of the *pc_n* being de-asserted, the bit line is floating. In this time period, the static current of the amplifier is still zero because the current stop switch is kept off with the signal *istop*.

The next defined condition is STATE2 beginning when the *istop* signal is de-asserted LO. The *istop* logic signal is designed to go LO coincident with the read line signal, *rd_in*, and the point in time in the active bit cell when the storage node is examined. There are two possible options. If the active bit cell that is tied to the read bit line has its storage node lower than the reference in its current controller, STATE2A exists, and the read bit line will stay floating at *vddr*. The transistor M2 in the sense amplifier forms a modified differential pair with M0 in the current controller reference transistor. For example, if the voltage tied to the gate of M0 is set approximately 150mV lower than *vddr*, the total current in the sense amplifier is the sum of:
(i) the current flowing in M2, a value much less than that of *ibias* and
(ii) the current in the compare block, which is also designed to be less than *ibias*.
The sense amplifier output, ampout_b, will stay LO for this case.

The STATE2B is defined when the case of the storage node in the active bit cell is higher than the reference voltage. As a result for this case, the read bit line voltage will be falling with respect to *vddr*. The switching threshold for the bit line amplifier is set with the reference *bl_ref*. When the read bit line has a lower value than *bl_ref*, the current will exponentially increase in M2 which will in turn cause the output of the current compare block, *ampout_b*, to switch from LO to HI.

The next state is STATE3, the current stop state. The signal *istop* is the output from the current stop logic being triggered from the *ampout_b* signal transitioning from LO to HI. The *istop* signal being asserted in this way shuts down all current in the sense amplifier.

To summarize, for the active bit cell storage node high case, the current in the sense amplifier has a very brief transitional current to switch the current compare block. Otherwise, for the active bit cell storage node low case, the static current value is either zero or a value less than *ibias*. It is reasonable to have the amplifiers active while they are waiting for a read bit line

transition, since the idle current is in the nano-Amp range. The read bit line processed by the sense amplifier can be selected between side-a and side-b. Only one sense amplifier is needed to process two columns of bit cells.

### 3.3.5 Dynamic Refresh to Improve Retention Time

As described in Section 3.2, the current controlled memory architecture compares the bit cell hold node voltage with a reference voltage. As the bit cell voltage dissipates over time it must be refreshed to retain its logic value. The retention time of the bit cells in the 131K Bit DRAM is improved with a dynamic refresh method.

Dummy refresh bit cells are added to the layout and as the refresh of the active bit cells is carried out row by row, the dummy bit cells are also written. These dummy bit cells are identical in physical dimensions to the actual bit cells. As the value stored on the actual bit cell begins to droop, the dummy bit cells likewise droop. The dummy bit cells are monitored and the information is used to actually adjust the references in the current controller. In this way, the retention time is increased.

Figure 3.7 depicts a simplified schematic for a set of three dummy bit cells, one for each level in BASE4 mode. These levels are written with an appropriate voltage to represent the respective logic level. For BASE2 operation only one is used.

### 3.3.6 Layout

The sense amplifier, read channels, write channels, and row decoders are laid out to a pitch that matches that of the bit cells.

In addition to the 131K bit chip, a 1Meg bit chip was also manufactured. The building blocks for both die are identical. Fig. 8 is an image showing both layouts. Table 3.1 compares the sizes. The total area for the 131K and 1Meg die is 1.4 mm. sq. and 7.9 mm. sq. respectively, not including the pad ring. The ratio between the area for the bit cells compared to the total area is 29% for the 131K chip and is 41% for the 1Meg chip.

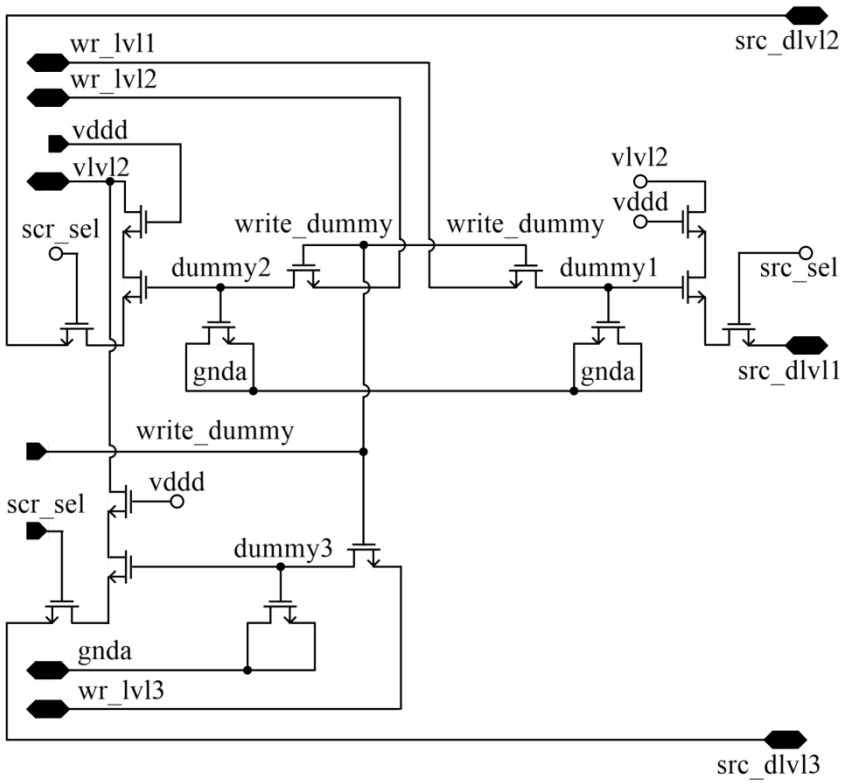


Figure 3.7 Schematic for dummy bit cells to increase retention time.

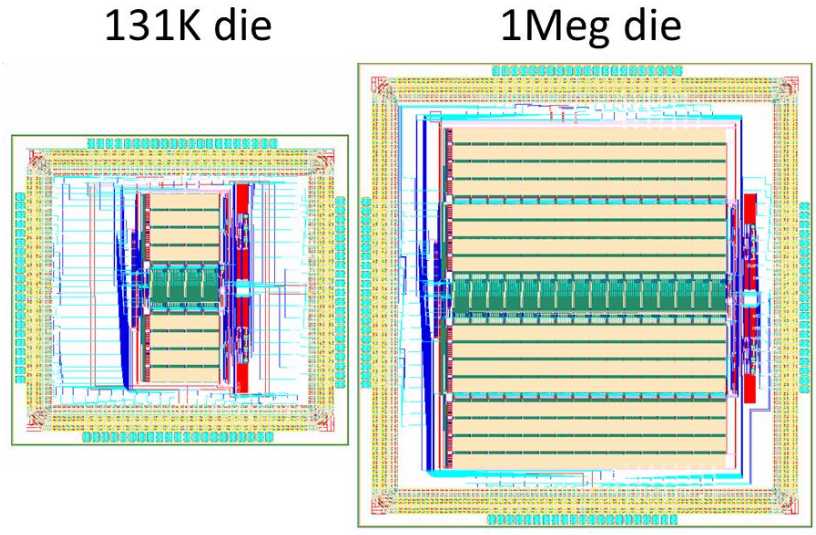


Figure 3.8 Images of layouts for the 131K bit die and a 1Meg bit die.

TABLE 3.1 Bit-cell utilization

| | **131K Bit** | **1Meg Bit** |
|---|---|---|
| **Single Bit Cell Area** | 110 $F^2$ | 110 $F^2$ |
| **Total Bit Cell Area** | 0.4 $mm^2$ | 3.2 $mm^2$ |
| **Total Area** | 1.4 $mm^2$ | 7.9 $mm^2$ |
| **Bit Cell Area of Total** | 29% | 41% |

## 3.4 Results

### 3.4.1 Block Simulation

#### 3.4.1.1 Bit Cell Source Driver Simulation

Figure 3.4b shows three current stop switches per column are needed to multiplex the DC levels from the three current controllers into the common source signal of the column. The switches are controlled by three distinct logic signals from the current stop block.

Figure 3.9 illustrates the resulting waveforms from a simulation for the multiplexing process of three DC levels from the current control opamps into a common source line of a column as is necessary for BASE4 operation. The three signals *sw1, sw2, and sw3* are the control lines for the switches coming from the current stop logic block in Figure 3.4b. Observe that the source line, *Srca,* presents three stepping voltage levels to the bit cell as is desired but it is doing so by switching among three DC levels from the opamps that are labeled *src1, src2, and src3.*

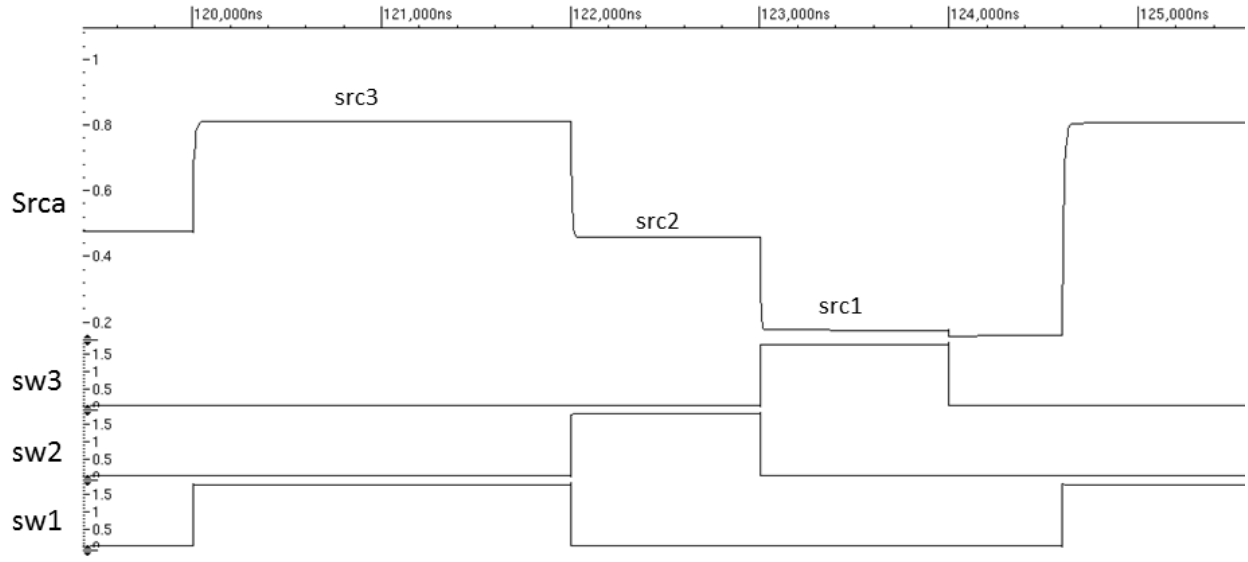


Figure 3.9 Simulation waveforms for the source line, *Srca,* driven by the three switched reference levels.

### 3.4.2 Sense Amplifier

Given below in Figure 3.10 are waveforms from a simulation of a single column for a current controlled sense amplifier working in BASE4 mode.

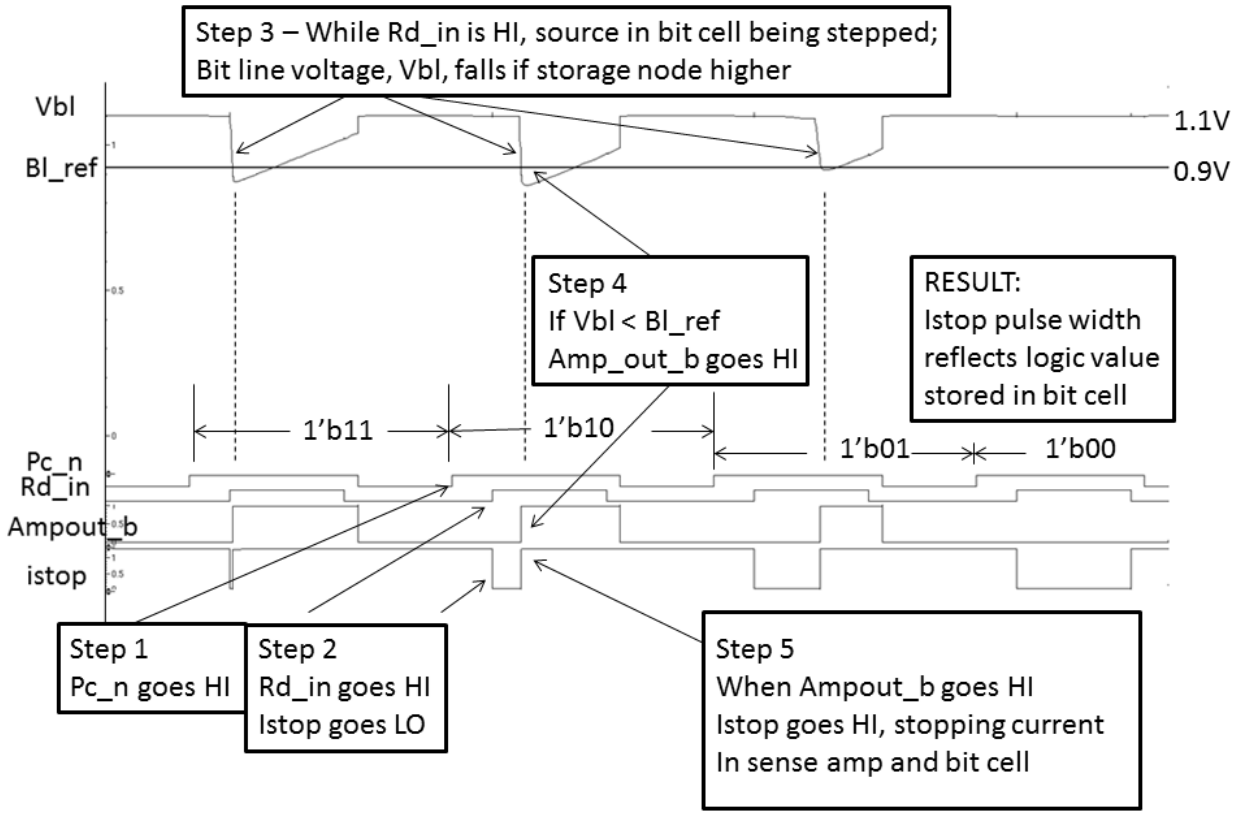


Figure 3.10 Simulation waveforms of the current controlled sense amplifier.

The waveforms reflect reading four values from the memory bit cell. The values stored on the hold node in the bit cell are 2'b11, 2'b10, 2'b01, and 2'b00 respectively. In each case:

Step 1. *Pc_n* goes HI and starts the read process.

Step 2. *Rd_in* goes HI and *istop* goes LO.

Step 3. During the time the read signal, *rd_in,* is asserted, as is depicted as *Srca* in Figure 3.9, the source in the active bit cell is stepped through the three levels representing appropriate references to compare to the hold node in the bit cell. If the storage node is higher than the respective reference node, the bit line starts to fall.

Step 4. In the sense amplifier, the bit line is monitored and is being compared to the MDP global reference *bl_ref* as in Figure 3.5. When the sense amp senses that the read bit line has started to fall and is less than the threshold set by *bl_ref*, the *ampout_b* output goes high.
Step 5. This event causes *istop* to go HI thereby killing the current in the bit cell, which causes the bit line to stop falling and also kills the current in the sense amp.
Result: As can be seen in Figure 3.10, the process causes the width of the *istop* pulse to reflect the value of the digital word stored on the bit cell and indeed the logic decoder uses that fact in the decode process.

### 3.4.3 Test Methodology – Vector Generation

The die is packaged with an 84-pin PLCC. We create a test PCB to provide stimulus to the chip. The test methodology is to run a simulation and capture input stimulus vectors in the form of waveform VCD files during serial DAC initialization of the reference voltages, writing, refresh, and reading. An accurate digital model written in Verilog code is created to model the chip that aids in the vector generation process by allowing the simulations to run much faster. The resulting VCD files are converted to the proper format for the pattern generators that are used to drive the chip. The read data from the chip as a result of the stimulus from the pattern generators are compared to the expected values.

#### 3.4.3.1 Serial DAC Initialization Stimulus Vectors

For flexibility, the voltage references for the chip are generated with serial DACs on the test PCB. There are five dual DACs on the board used to create nine different voltage references.

The stimulus input signals needed for the DAC initialization are five DAC chip select signals, a serial data input, and a serial clock. These signals going into the DACs determine the point in time the particular reference for the chip is changed to the desired value.

#### 3.4.3.2 Write Stimulus Vectors

There is also a set of input vectors for the write process to the chip. Among the signals needed for writing are a write enable signal, a write clock, a signal to select either the a-side or the b-side, an eight bits data bus, and an eight bit address bus.

A factor limiting the stimulus used for the pattern generator is that the data must fit into its memory. Given this limitation, write vectors are chosen to write 16 consecutive rows with eight bit words, thereby using four bit cells in BASE4.

### 3.4.3.3 Read Stimulus Vectors

Read vectors are generated causing the reading of the same 16 rows that were described for the write vectors above. Some of the signals required for the read vector set are a read clock, a precharge signal, a read enable signal, and a latch signal. There is also an 8 bit read address.

### 3.4.3.4 Expected Read Results

Figure 3.11 depicts simulation read waveforms. The outputs are shown for the two bit lines, *Bl_0* and *Bl_1*, having read data that represents four bits. The respective *istop* signals are also shown on the plot. Some of the signals from the read stimulus vector are given at the top of the plot: the *rd_clk, pc_n, rd_in,* and *latch* signals.

\

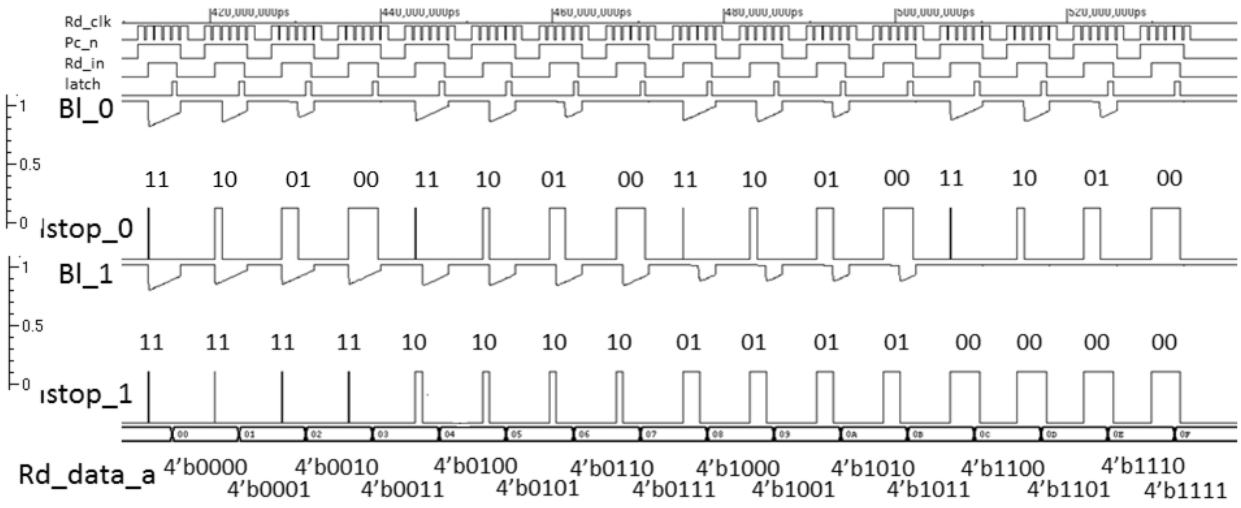


Figure 3.11 Response to read input stimulus vectors.

For the bit line outputs, observe first of all the amplitude of the transitions on the read bit lines are small, around 0.2V due to the operation of the current stop function. Next, it is seen that the *istop* pulse width indicates the value stored on the bit cell in keeping with the discussion of Figure 3.10. For each read, the value that was stored and read from the respective bit cell is listed.

The *rd_data_a* bus in the figure is depicted at the bottom. It is a bus that is eight bits wide and its polarity is inverted with respect to what is stored on the bit cell. The output data bus changes every time the *latch* signal is asserted. From the *rd_data_a* bus, the data of the least significant four bits decoded from the given two bit lines is listed on the figure.

### 3.4.4 Silicon Results

#### 3.4.4.1 Silicon Matches Expected Simulation Results

The vectors, as discussed, are created and output from a pattern generator to a PCB instantiating the packaged parts. An image of the test setup is shown in Figure 3.12. Cables from the pattern generators are going into the board to carry the stimulus.

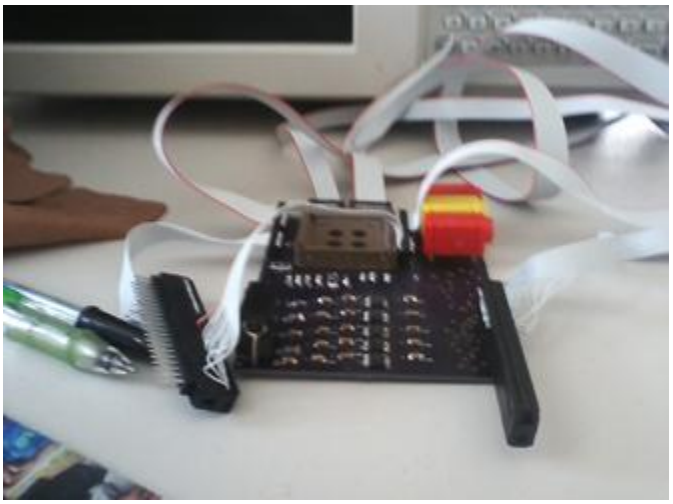

Figure 3.12 Actual test board driving a PLCC packaged part for the 131K memory chip.

Vectors as described above are generated for DAC initialization, writing, and reading. The vectors are sent to the pattern generator to write and read repeatedly 8 bit words from 16 consecutive rows. Multiple locations in both a top quadrant and bottom quadrant of the chip are chosen.

The packaged parts are fully functional in BASE4 mode with a sampling clock below 1 Hz and up to 135 MHz. The tests are limited by the maximum speed at which the pattern generator generates the stimulus waveforms. Fully functional is defined as the 8 bit read data matching exactly with the expected output based on the simulation used to create the vectors.

#### 3.4.4.2 Retention Time

We measure the retention time at room temperature in BASE4 mode. After each bit cell is written, the time between the write and the reading of each cell is gradually increased until there is a read failure. The time is measured using the time base on an oscilloscope. The retention time is quite high, between four and five seconds for BASE4.

The threshold voltage for the 180nm process technology used for the memory arrays in this work is 0.42VDC for the typical case. By way of comparison, the retention time in [3] was reported to be 300ms with threshold devices of 0.8V.

The high retention time is due to the nature of the architecture in that: (*i*) the write transistor is always in super cutoff condition when it is off, as defined by having its gate voltage much lower than the source voltage. (*ii*) There is a MOSCAP that is part of each bit cell with the drain and source both connected to ground. Since the gate always has a high enough potential with respect to the source, there is always a channel so its capacitance is much higher than it would be if it was turned off. *(iii)* The refresh algorithm is understood to add a factor of 2-3X to the retention time in most cases as can be seen when properly analyzing waveforms from the refresh algorithm.

The fully functional silicon proves we achieved DRAM with a standard 180nm process node with no extra layers or processing steps.

By way of summary, the reasons for ultra-low energy performance of the current controlled DRAM in this work are: *(i)* Power on the high capacitance bit lines is proportional to VDD * Vbl and not VDD squared; (*ii*) Lower delay variance reduces write bit line transitions for lower power during refresh; (*iii*) The current stop function working in conjunction with the sense amplifiers reduces read access time and limits read bit line transitions; and (*iv*) There is very impressive retention time performance.

## 3.5 Conclusion

It was explained how the read and write bit lines are shared between two different banks of bit cells, the a-side and the b-side allowing for a reduced area and allowing for half as many sense amplifiers The method was described for multiplexing the outputs of the opamp between common source domains allowing only two sets of opamps to drive the entire chip. The sense amplifiers were described and it was shown how their static power is kept low. The sense amplifiers were functional with 256 rows per column. A dynamic refresh algorithm improved the retention time.

Extensive simulations were accomplished proving the functionality of the chip. In addition, packaged parts were received from the foundry and tested at a number of locations on the dies. The parts proved to be fully functional in BASE4 at moderate down to very low speeds. The retention time was very high, between 4 and 5 seconds.

In the proof of concept chip in [4], the bias current mirror for the bit lines, the clamps for the pre-charge of the bit line, the sense amplifiers, the current control opamp, and the current stop functionality were all implemented on the board in the test PCB. For the packaged parts of the 131K DRAM array in this work, all of these blocks were designed on the chips.

The parts being fully functional with retention time even higher than expected proves that the power estimation and comparisons in [4] was germane.

## 3.6 References

[1] Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic, Digital Integrated Circuits – A Design Perspective, 2nd Edition, Prentice Hall Electronics and VLSI Series, Chap. 12.

[2] K. C. Chun, P. Jain, T. Kim, and C. H. Kim , “A 667 MHz Logic-Compatible Embedded DRAM Featuring an Asymmetric 2T Gain Cell for High Speed On-Die Caches”, IEEE J. Solid-State Circuits, vol. 47, no. 2, pp. 547-559, Feb. 2012.

[3] Y. Lee, M. Chen, J. Park, D. Sylvester, D. Blaauw, “A 5.4nW/kB Retention Power Logic-Compatible Embedded DRAM with 2T Dual-Vt Gain Cell for Low Power Sensing Applications”, Proc. IEEE A-SSCC, 2010.

[4] J. Lynch, P. P. Irazoqui, “A Low Power Logic-Compatible Multi-Bit Memory Bit Cell Architecture With Differential Pair and Current Stop Constructs”, IEEE Transactions on Circuits and Systems – I, vol. 61, no. 12, pp. 3367- 3375, Dec. 2014.

[5] J.C. Koob, S.A. Ung, B.F. Cockburn, and D.G. Elliott, “Design and characterization of a multilevel DRAM,” IEEE Trans. VSLI Syst., vol. 19, no. 9 pp. 1583-1586, Sep. 2011.

[6] D. Somasekhar, S. Lu, B. Bloechel, G. Dermer, K. Lai, S. Borkar, and V. De, “A 10Mvit, 15GBytes/sec bandwidth 1T DRAM chip with planar MOS storage capacitor in an unmodified 150 nm logic process for high-density on-chip memory applications,” in Proc. ESSCIRC, Grenoble, France, 2005, pp. 355-358.

[7] M. Khalid, P. Meinerzhagen, and A. Burg, “Replica bit-line technique for embedded multilevel gain-cell DRAM,” in Proc. IEEE NEWCAS, Jun. 2012, pp. 77-80.

# APPENDIX A

Synthesis Liberty .lib file for 3C standard cell library

```
library (logicST) {

   delay_model                  : table_lookup;
   simulation                   : false;
   time_unit                    : "1ns";
   voltage_unit                 : "1V";
   current_unit                 : "1mA";
   pulling_resistance_unit      : "1kohm";
   capacitive_load_unit           (1, ff);

   default_max_transition       : 10000;
   default_input_pin_cap        : 0.000;
   default_output_pin_cap       : 0.000;
   default_inout_pin_cap        : 3.0;


   operating_conditions (typical_case) {
      process     :  1.0;
      temperature : 25.0;
      voltage     :  0.3;
      tree_type   : "balanced_tree";}

   operating_conditions (worst_case) {
      process     :  1.5;
      temperature : 50.0;
      voltage     :  0.3;
      tree_type   : "balanced_tree";}

   default_operating_conditions : typical_case ;

   wire_load("AL_SMALL") {
       resistance : 0 ;
       capacitance : 1.0 ;
       area : 0 ;
       slope : 0.18 ;
       fanout_length(1,0.4) ;
   }
   wire_load("AL_MEDIUM") {
       resistance : 0 ;
       capacitance : 1.0 ;
       area : 0 ;
       slope : 0.20 ;
       fanout_length(1,0.5) ;
   }
   wire_load("AL_LARGE") {
```

```
    resistance : 0 ;
    capacitance : 1.0 ;
    area : 0 ;
    slope : 0.30 ;
    fanout_length(1,0.7) ;
}

wire_load_selection(ALUMINUM) {
    wire_load_from_area(0,100,"AL_SMALL") ;
    wire_load_from_area(100,1000,"AL_MEDIUM") ;
    wire_load_from_area(1000,10000,"AL_LARGE") ;
}

wire_load("CU_SMALL") {
    resistance : 0 ;
    capacitance : 0.1 ;
    area : 0 ;
    slope : 0.018 ;
    fanout_length(1,0.04) ;
}
wire_load("CU_MEDIUM") {
    resistance : 0 ;
    capacitance : 0.1 ;
    area : 0 ;
    slope : 0.020 ;
    fanout_length(1,0.05) ;
}
wire_load("CU_LARGE") {
    resistance : 0 ;
    capacitance : 0.1 ;
    area : 0 ;
    slope : 0.030 ;
    fanout_length(1,0.07) ;
}
wire_load_selection(COPPER) {
    wire_load_from_area(0,100,"CU_SMALL") ;
    wire_load_from_area(100,1000,"CU_MEDIUM") ;
    wire_load_from_area(1000,10000,"CU_LARGE") ;
}

default_wire_load_selection : ALUMINUM;

nom_process     :  1.0;
nom_temperature : 25.0;
nom_voltage     :  0.3;


lu_table_template(template1) {
    variable_1 : input_net_transition;
    variable_2 : total_output_net_capacitance;
    index_1 ("25, 75, 225, 500");
    index_2 ("1, 4, 15, 50");
```

```
}

cell (nandon2x1) {
   area : 1.0;
   pin(a) {
      direction   : input;
      capacitance : 3.0;
   }
   pin(b) {
      direction   : input;
      capacitance : 3.2;
   }
   pin(y_b) {
      direction : output;
      function  : "(a b)'";
      timing() {
         related_pin : "a";
         timing_type  : combinational;
         timing_sense : negative_unate;
         cell_fall(template1) {
            values("296, 349,  545,  1166",  \
                   "345, 398,  593,  1214", \
                   "490, 544,  739,  1360",  \
                   "758, 811,  1006, 1627"   );
         }
         cell_rise(template1) {
            values("257, 323, 565, 1335", \
                   "297, 363, 605, 1374", \
                   "416, 482, 724, 1494", \
                   "635, 701, 943, 1713"  );
         }
         fall_transition(template1) {
            values("76,  172,  523,  1641",  \
                   "163, 259,  610,  1729", \
                   "426, 521,  873,  1991", \
                   "907, 1002, 1354, 2472"  );
         }
         rise_transition(template1) {
            values("75,  194, 629,  2016",  \
                   "147, 266, 701,  2087", \
                   "362, 481, 916,  2302", \
                   "756, 874, 1310, 2696"  );
         }
      }
      timing() {
         related_pin : "b";
         timing_type  : combinational;
         timing_sense : negative_unate;
         cell_fall(template1) {
            values("208, 262, 457, 1078",  \
                   "232, 285, 480, 1101", \
                   "301, 355, 550, 1171",  \
```

```
                "429, 482, 678, 1299"   );
         }
         cell_rise(template1) {
            values("221, 281, 504, 1213", \
                   "257, 318, 540, 1249", \
                   "366, 427, 650, 1358", \
                   "567, 627, 850, 1559"  );
         }
         fall_transition(template1) {
            values("53,  149, 500, 1618",  \
                   "95,  191, 542, 1660", \
                   "220, 316, 668, 1785", \
                   "450, 546, 898, 2016"  );
         }
         rise_transition(template1) {
            values("69,  179, 580,  1855",  \
                   "135, 244, 645,  1921", \
                   "332, 441, 842,  2118", \
                   "692, 801, 1202, 2478"  );
         }
      }
   }
}
cell (nandon3x1) {
   area : 1.90909090;
   pin(a) {
      direction   : input;
      capacitance : 3.2;
   }
   pin(b) {
      direction   : input;
      capacitance : 3.9;
   }
   pin(c) {
      direction   : input;
      capacitance : 4.4;
   }
   pin(y_b) {
      direction : output;
      function  : "(a b c)'";
      timing() {
         related_pin : "a";
         timing_type  : combinational;
         timing_sense : negative_unate;
         cell_fall(template1) {
            values("432, 487,  688,  1328",  \
                   "517, 572,  773,  1413", \
                   "773, 827,  1029,  1669",  \
                   "1241, 1296,  1497, 2137"   );
         }
         cell_rise(template1) {
            values("430, 496, 737, 1502", \
```

```
                "472, 537, 778, 1542", \
                "595, 661, 901, 1666", \
                "822, 887, 1127, 1893"  );
      }
      fall_transition(template1) {
         values("110,  208,  571,  1723",  \
                "263, 367,  724,  1876", \
                "723, 821,  1184,  2336", \
                "1566, 1665, 2027, 3179"  );
      }
      rise_transition(template1) {
         values("76,  194, 627,  2004",  \
                "151, 269, 701,  2079", \
                "373, 491, 924,  2301", \
                "781, 899, 1331, 2709"  );
      }
   }
   timing() {
      related_pin : "b";
      timing_type  : combinational;
      timing_sense : negative_unate;
      cell_fall(template1) {
         values("483, 541, 754, 1429",  \
                "500, 558, 770, 1445", \
                "549, 607, 819, 1494",  \
                "638, 696, 908, 1584"   );
      }
      cell_rise(template1) {
         values("401, 462, 687, 1402", \
                "443, 505, 729, 1445", \
                "571, 633, 857, 1572", \
                "806, 867, 1092, 1807"  );
      }
      fall_transition(template1) {
         values("49,  154, 536, 1751",  \
                "79,  183, 565, 1780", \
                "167, 271, 653, 1868", \
                "328, 432, 814, 2029"  );
      }
      rise_transition(template1) {
         values("75,  186, 590,  1877",  \
                "152, 262, 667,  1954", \
                "382, 493, 897,  2184", \
                "805, 915, 1319, 2607"  );
      }
   }
   timing() {
      related_pin : "c";
      timing_type  : combinational;
      timing_sense : negative_unate;
      cell_fall(template1) {
         values("247, 300, 497, 1122",  \
```

```
                    "302, 355, 552, 1177", \
                    "467, 521, 717, 1342",  \
                    "771, 824, 1021, 1646"   );
            }
            cell_rise(template1) {
               values("326, 384, 594, 1264", \
                    "371, 428, 639, 1309", \
                    "504, 561, 772, 1442", \
                    "748, 806, 1016, 1686"  );
            }
            fall_transition(template1) {
               values("82,  178, 532, 1657",  \
                    "181,  278, 631, 1756", \
                    "497, 576, 929, 2055", \
                    "1026, 1122, 1476, 2601"  );
            }
            rise_transition(template1) {
               values("74,  178, 557,  1763",  \
                    "154, 258, 637,  1843", \
                    "394, 498, 877,  2082", \
                    "834, 937, 1316, 2522"  );
            }
         }
      }
   }
   cell (nandon4x1) {
      area : 2.54545454;
      pin(a) {
         direction   : input;
         capacitance : 3.8;
      }
      pin(b) {
         direction   : input;
         capacitance : 3.9;
      }
      pin(c) {
         direction   : input;
         capacitance : 4.8;
      }
      pin(d) {
         direction   : input;
         capacitance : 3.1;
      }
      pin(y_b) {
         direction : output;
         function  : "(a b c d)'";
         timing() {
            related_pin : "a";
            timing_type  : combinational;
            timing_sense : negative_unate;
            cell_fall(template1) {
               values("584, 645,  872,  1593",  \
```

```
            "652, 713,  940,  1661", \
            "855, 917,  1144,  1865",  \
            "1229, 1291,  1518, 2239"   );
      }
      cell_rise(template1) {
         values("613, 677, 914, 1666", \
            "655, 719, 957, 1708", \
            "781, 845, 1082, 1834", \
            "1013, 1077, 1314, 2066"  );
      }
      fall_transition(template1) {
         values("98,  210,  617,  1915",  \
            "221, 332,  740,  2038", \
            "588, 699,  1107,  2405", \
            "1260, 1372, 1780, 3077"  );
      }
      rise_transition(template1) {
         values("77,  193, 618,  1973",  \
            "152, 269, 694,  2049", \
            "380, 496, 922, 2276", \
            "796, 913, 1338, 2693"  );
      }
   }
   timing() {
      related_pin : "b";
      timing_type  : combinational;
      timing_sense : negative_unate;
      cell_fall(template1) {
         values("606,  667,  891,  1605",  \
            "654,  715,  940,  1654",  \
            "800,  861,  1085, 1799",  \
            "1067, 1128, 1352, 2066"   );
      }
      cell_rise(template1) {
         values("578,  638,  859,  1563", \
            "626,  686,  907,  1611", \
            "768,  829,  1050, 1754", \
            "1032, 1092, 1313, 2017"  );
      }
      fall_transition(template1) {
         values("80,  191,  595,  1880",  \
            "168, 278,  682,  1967", \
            "430, 540,  944,  2229", \
            "910, 1020, 1424, 2710"  );
      }
      rise_transition(template1) {
         values("79,  188,  586,  1852",  \
            "165, 274,  672,  1938", \
            "423, 532,  930,  2196", \
            "896, 1004, 1402, 2669"  );
      }
   }
```

```
timing() {
   related_pin : "c";
   timing_type  : combinational;
   timing_sense : negative_unate;
   cell_fall(template1) {
      values("480, 539, 755, 1449",  \
             "537, 596, 813, 1502", \
             "710, 769, 986, 1675",  \
             "1027, 1086, 1303, 1992"   );
   }
   cell_rise(template1) {
      values("533, 593, 809, 1499", \
             "579, 638, 855, 1544", \
             "715, 774, 991, 1680", \
             "965, 1024, 1241, 1930"  );
   }
   fall_transition(template1) {
      values("87,  194, 584, 1825",  \
             "191,  297, 688, 1929", \
             "502, 609, 999, 2240", \
             "1073, 1179, 1569, 2810"  );
   }
   rise_transition(template1) {
      values("76,  183, 573,  1814",  \
             "158, 265, 655,  1896", \
             "403, 510, 900,  2141", \
             "853, 960, 1350, 2591"  );
   }
}
timing() {
   related_pin : "d";
   timing_type  : combinational;
   timing_sense : negative_unate;
   cell_fall(template1) {
      values("286, 347, 569, 1276",  \
             "336, 396, 618, 1325", \
             "484, 547, 767, 1474",  \
             "756, 817, 1039, 1746"   );
   }
   cell_rise(template1) {
      values("453, 512, 733, 1433", \
             "496, 556, 776, 1476", \
             "625, 685, 905, 1605", \
             "861, 921, 1141, 1841"  );
   }
   fall_transition(template1) {
      values("81,  190, 590, 1863",  \
             "170, 279, 679, 1952", \
             "437, 546, 946, 2219", \
             "928, 1037, 1437, 2709"  );
   }
   rise_transition(template1) {
```

```
            values("75,  183, 579,  1839",  \
                   "152, 260, 656,  1916", \
                   "384, 492, 888,  2148", \
                   "810, 918, 1314, 2574"  );
         }
      }
   }
}

cell (noron2x1) {
   area : 0.90909090;
   pin(a) {
      direction : input;
      capacitance : 2.1;
   }
   pin(b) {
      direction : input;
      capacitance : 2.8;
   }
   pin(y) {
      direction : output;
      function : "(a + b)'";
      timing() {
         related_pin : "a";
         timing_type  : combinational;
         timing_sense : negative_unate;
         cell_fall(template1) {
            values("250, 319, 572,  1377",  \
                   "300, 369, 622,  1427", \
                   "450, 519, 772,  1577",  \
                   "725, 794, 1047, 1852"   );
         }
         cell_rise(template1) {
            values("240, 312, 576,  1416",  \
                   "305, 377, 641,  1481", \
                   "500, 572, 836,  1676",  \
                   "858, 930, 1194, 2034"   );
         }
         fall_transition(template1) {
            values("86,  211,  666,  2115", \
                   "176, 301,  756,  2205", \
                   "446, 571,  1026, 2475", \
                   "941, 1066, 1521, 2970"  );
         }
         rise_transition(template1) {
            values("102,  231,  707,  2219", \
                   "219,  348,  824,  2336", \
                   "571,  700,  1175, 2687", \
                   "1215, 1345, 1819, 3332"  );
         }
      }
      timing() {
```

```
            related_pin : "b";
            timing_type  : combinational;
            timing_sense : negative_unate;
            cell_fall(template1) {
               values("222, 289, 533,  1312",  \
                      "273, 340, 584,  1363",  \
                      "426, 493, 738,  1517",  \
                      "708, 774, 1019, 1798"   );
            }
            cell_rise(template1) {
               values("217, 297, 588, 1516",  \
                      "241, 320, 612, 1539",  \
                      "317, 391, 683, 1610",  \
                      "441, 521, 812, 1740"   );
            }
            fall_transition(template1) {
               values("86,  206,  647,  2049", \
                      "178, 298,  739,  2141", \
                      "454, 574,  1015, 2417", \
                      "960, 1081, 1521, 2923"  );
            }
            rise_transition(template1) {
               values("69,  212, 737,  2406", \
                      "111, 255, 779,  2449", \
                      "239, 382, 907,  2576", \
                      "472, 615, 1140, 2810"  );
            }
         }
      }
   }

   cell (noron3x1) {
      area : 1.45454545;
      pin(a) {
         direction : input;
         capacitance : 2.7;
      }
      pin(b) {
         direction : input;
         capacitance : 2.8;
      }
      pin(c) {
         direction : input;
         capacitance : 2.5;
      }
      pin(y) {
         direction : output;
         function : "(a + b + c)'";
         timing() {
            related_pin : "a";
            timing_type  : combinational;
            timing_sense : negative_unate;
```

```
      cell_fall(template1) {
         values("484, 554,  812,  1632",  \
                "531, 601,  859,  1679", \
                "672, 742,  1000, 1829",  \
                "930, 1000, 1258, 2078"   );
      }
      cell_rise(template1) {
         values("486, 551,  787,  1537",  \
                "536, 601,  837,  1587", \
                "686, 751,  987,  1737",  \
                "961, 1026, 1262, 2012"   );
      }
      fall_transition(template1) {
         values("84,  211,  675,  2151", \
                "169, 296,  759,  2236", \
                "423, 549,  1013, 2489", \
                "887, 1014, 1478, 2954"  );
      }
      rise_transition(template1) {
         values("84,  199,  624,  1974", \
                "174, 289,  714,  2064", \
                "444, 559,  984,  2334", \
                "939, 1054, 1479, 2829"  );
      }
   }
   timing() {
      related_pin : "b";
      timing_type  : combinational;
      timing_sense : negative_unate;
      cell_fall(template1) {
         values("454, 524, 782,  1602",  \
                "501, 572, 829,  1649",  \
                "643, 713, 971,  1791",  \
                "903, 973, 1231, 2051"   );
      }
      cell_rise(template1) {
         values("461, 531, 791,  1616",  \
                "495, 566, 825,  1650",  \
                "598, 668, 928,  1753",  \
                "786, 857, 1116, 1941"   );
      }
      fall_transition(template1) {
         values("85,  211,  675,  2151", \
                "170, 296,  760,  2236", \
                "425, 552,  1015, 2492", \
                "893, 1019, 1483, 2959"  );
      }
      rise_transition(template1) {
         values("73,  201, 667,  2152", \
                "135, 262, 729,  2214", \
                "320, 447, 914,  2399", \
                "659, 786, 1253, 2738"  );
```

```
         }
      }
      timing() {
         related_pin : "c";
         timing_type  : combinational;
         timing_sense : negative_unate;
         cell_fall(template1) {
            values("401, 468, 711,  1486",  \
                   "444, 510, 754,  1529",  \
                   "572, 638, 882,  1657",  \
                   "806, 872, 1116, 1891"   );
         }
         cell_rise(template1) {
            values("359, 425, 667,  1437",  \
                   "390, 456, 698,  1468",  \
                   "481, 547, 789,  1559",  \
                   "649, 715, 957,  1727"   );
         }
         fall_transition(template1) {
            values("78,  198,  636,  2031", \
                   "155, 274,  713,  2108", \
                   "385, 504,  943,  2338", \
                   "806, 926,  1364, 2959"  );
         }
         rise_transition(template1) {
            values("67,  186, 622,  2008", \
                   "122, 241, 676,  2062", \
                   "287, 406, 941,  2227", \
                   "589, 708, 1143, 2529"  );
         }
      }
   }
}
cell (noron4x1) {
   area : 2.72727272;
   pin(a) {
      direction : input;
      capacitance : 3.1;
   }
   pin(b) {
      direction : input;
      capacitance : 3.5;
   }
   pin(c) {
      direction : input;
      capacitance : 3.4;
   }
   pin(d) {
      direction : input;
      capacitance : 4.4;
   }
   pin(y) {
```

```
direction : output;
function : "(a + b + c + d)'";
timing() {
   related_pin : "a";
   timing_type  : combinational;
   timing_sense : negative_unate;
   cell_fall(template1) {
      values("699,  774,  1049, 1924",  \
             "750,  825,  1100, 1975",  \
             "905,  980,  1255, 2130",  \
             "1189, 1264, 1539, 2414"   );
   }
   cell_rise(template1) {
      values("622,  692,  950,  1769",  \
             "688,  758,  1015, 1834",  \
             "884,  954,  1211, 2030",  \
             "1243, 1314, 1571, 2389"   );
   }
   fall_transition(template1) {
      values("91,  226,  721,  2296", \
             "184, 319,  814,  2389", \
             "463, 598,  1093, 2668", \
             "974, 1109, 1604, 3179"  );
   }
   rise_transition(template1) {
      values("101,  227,  691,  2165", \
             "219,  345,  808,  2283", \
             "572,  698,  1161, 2636", \
             "1219, 1345, 1809, 3283"  );
   }
}
timing() {
   related_pin : "b";
   timing_type  : combinational;
   timing_sense : negative_unate;
   cell_fall(template1) {
      values("681,  753,  1017, 1857",  \
             "732,  804,  1068, 1908",  \
             "885,  957,  1221, 2061",  \
             "1164, 1236, 1500, 2340"   );
   }
   cell_rise(template1) {
      values("707,  780,  1045, 1888", \
             "748,  820,  1086, 1929", \
             "870,  942,  1208, 2051", \
             "1094, 1166, 1431, 2275"  );
   }
   fall_transition(template1) {
      values("89,  219,  694,  2206",  \
             "180, 310,  785,  2297", \
             "455, 584,  1060, 2572", \
             "958, 1087, 1562, 3074"  );
```

```
      }
      rise_transition(template1) {
         values("80,  210, 687,  2206",  \
                "153, 283, 761,  2279", \
                "373, 503, 980,  2498", \
                "775, 906, 1383, 2901"  );
      }
   }
   timing() {
      related_pin : "c";
      timing_type  : combinational;
      timing_sense : negative_unate;
      cell_fall(template1) {
         values("644,  710,  953,  1727",  \
                "697,  763,  1006, 1779",  \
                "855,  922,  1165, 1938",  \
                "1147, 1213, 1456, 2229"   );
      }
      cell_rise(template1) {
         values("621, 693, 958,  1802",  \
                "652, 725, 990,  1833",  \
                "747, 819, 1084, 1928",  \
                "921, 993, 1258, 2101"   );
      }
      fall_transition(template1) {
         values("87,  207,  644,  2037", \
                "183, 302,  740,  2132", \
                "469, 588,  1026, 2418", \
                "993, 1112, 1550, 2942"  );
      }
      rise_transition(template1) {
         values("72,  202, 679,  2197", \
                "129, 259, 736,  2254", \
                "299, 429, 906,  2424", \
                "611, 741, 1218, 2737"  );
      }
   }
   timing() {
      related_pin : "d";
      timing_type  : combinational;
      timing_sense : negative_unate;
      cell_fall(template1) {
         values("574,  641,  884,  1657",  \
                "622,  689,  932,  1707",  \
                "767,  833,  1076, 1850",  \
                "1032, 1098, 1341, 2115"   );
      }
      cell_rise(template1) {
         values("466, 544, 829,  1735",  \
                "492, 570, 855,  1761",  \
                "570, 648, 933,  1839",  \
                "714, 791, 1076, 1983"   );
```

```
         }
         fall_transition(template1) {
            values("83,  203,  640,  2032", \
                   "170, 289,  727,  2119", \
                   "430, 550,  987,  2379", \
                   "907, 1027, 1464, 2856"  );
         }
         rise_transition(template1) {
            values("70,  210, 723,  2355", \
                   "117, 257, 770,  2401", \
                   "258, 398, 910,  2542", \
                   "516, 656, 1168, 2800"  );
         }
      }
   }
}
cell (xoron2x1) {
   area : 4.0;
   pin(a) {
      direction : input;
      capacitance : 6.8;
   }
   pin(b) {
      direction : input;
      capacitance : 7.9;
   }
   pin(y) {
      direction : output;
      function : "a' b + a b'";
      timing() {
         related_pin : "a";
         timing_type  : combinational;
         timing_sense : non_unate;
         cell_fall(template1) {
            values("684,  738,  936,  1566",  \
                   "728,  782,  980,  1610", \
                   "860,  914,  1112, 1742",  \
                   "1102, 1156, 1354, 1984"   );
         }
         cell_rise(template1) {
            values("665,  725,  945,   1645",  \
                   "719,  779,  999,   1699", \
                   "880,  940,  1160,  1860",  \
                   "1176, 1236, 1456,  2156"   );
         }
         fall_transition(template1) {
            values("72,  169, 526,  1278",  \
                   "151, 248, 605,  1660", \
                   "389, 486, 842,  1976", \
                   "824, 921, 1278, 2412"  );
         }
         rise_transition(template1) {
```

```
            values("84,   192,  588,  1848",  \
                   "181,  289,  685,  1945", \
                   "471,  579,  975,  2234", \
                   "1004, 1112, 1508, 2767"  );
         }
      }
      timing() {
         related_pin : "b";
         timing_type  : combinational;
         timing_sense : non_unate;
         cell_fall(template1) {
            values("529, 582, 776,  1395",  \
                   "570, 623, 817,  1435", \
                   "691, 744, 939,  1557",  \
                   "914, 967, 1162, 1780"   );
         }
         cell_rise(template1) {
            values("549, 608, 825,  1513",  \
                   "576, 635, 852,  1540", \
                   "658, 717, 933,  1622",  \
                   "807, 866, 1082, 1771"   );
         }
         fall_transition(template1) {
            values("68,  164, 514,  1627",  \
                   "141, 237, 587,  1700", \
                   "360, 456, 806,  1919", \
                   "762, 857, 1207, 2321"  );
         }
         rise_transition(template1) {
            values("60,  166, 556,  1785",  \
                   "109, 215, 604,  1844", \
                   "255, 361, 751,  1990", \
                   "524, 630, 1019, 2259"  );
         }
      }
   }
}

cell (invonx1) {
   area : 0.36363636;
   pin(a) {
      direction : input;
      capacitance : 2.0;
   }
   pin(y) {
      direction : output;
      function : !a;
      timing() {
         related_pin : "a";
         timing_type  : combinational;
         timing_sense : negative_unate;
         cell_fall(template1) {
```

```
            values("172, 234, 463, 1189 ",  \
                   "215, 277, 505, 1231", \
                   "342, 405, 633, 1359",  \
                   "577, 639, 867, 1593"   );
         }
         cell_rise(template1) {
            values("165, 221, 424, 1072",  \
                   "206, 261, 465, 1112", \
                   "328, 384, 588, 1235",  \
                   "554, 610, 813, 1461"   );
         }
         fall_transition(template1) {
            values("76, 188, 599, 1906",  \
                   "152, 264, 675, 1982", \
                   "382, 494, 905, 2212", \
                   "804, 916, 1327, 2634"  );
         }
         rise_transition(template1) {
            values("70,  170, 536, 1702",  \
                   "144, 244, 610, 1776", \
                   "365, 465, 831, 1997", \
                   "771, 871, 1237, 2402"  );
         }
      }
   }
}
cell (invonx2) {
   area : 0.45454545;
   pin(a) {
      direction : input;
      capacitance : 3.8;
   }
   pin(y) {
      direction : output;
      function : !a;
      timing() {
         related_pin : "a";
         timing_type  : combinational;
         timing_sense : negative_unate;
         cell_fall(template1) {
            values("118, 151, 272, 657 ",  \
                   "163, 195, 316, 702", \
                   "296, 328, 449, 834",  \
                   "539, 572, 693, 1078"   );
         }
         cell_rise(template1) {
            values("110, 137, 234, 543",  \
                   "149, 175, 272, 582", \
                   "265, 291, 388, 697",  \
                   "477, 504, 601, 910"   );
         }
         fall_transition(template1) {
```

```
            values("60, 119, 337, 1030",  \
                   "140, 199, 417, 1110", \
                   "379, 438, 656, 1349", \
                   "818, 877, 1095, 1788"  );
         }
         rise_transition(template1) {
            values("51, 98, 273, 830",  \
                   "120, 170, 343, 899", \
                   "329, 377, 551, 1108", \
                   "711, 759, 934, 1490"  );
         }
      }
   }
}
cell (invonx4) {
   area : 0.72727272;
   pin(a) {
      direction : input;
      capacitance : 7.4;
   }
   pin(y) {
      direction : output;
      function : !a;
      timing() {
         related_pin : "a";
         timing_type  : combinational;
         timing_sense : negative_unate;
         cell_fall(template1) {
            values("108, 125, 185, 378 ",  \
                   "146, 163, 223, 416", \
                   "260, 276, 337, 529",  \
                   "468, 484, 545, 737"   );
         }
         cell_rise(template1) {
            values("103, 117, 168, 330",  \
                   "142, 156, 206, 369", \
                   "258, 271, 322, 484",  \
                   "470, 484, 535, 697"   );
         }
         fall_transition(template1) {
            values("44, 74, 183, 529",  \
                   "112, 142, 251, 597", \
                   "316, 346, 455, 802", \
                   "691, 721, 830, 1176"  );
         }
         rise_transition(template1) {
            values("43, 68, 160, 452",  \
                   "113, 137, 229, 521", \
                   "321, 346, 438, 730", \
                   "704, 729, 820, 1112"  );
         }
      }
```

```
   }
}

cell (bufonx1) {
   area : 0.72727272;
   pin(a) {
      direction : input;
      capacitance : 2.0;
   }
   pin(y) {
      direction : output;
      function : "a";
      timing() {
         related_pin : "a";
         timing_type  : combinational;
         timing_sense : positive_unate;
         cell_fall(template1) {
            values("360, 425, 664,  1425",  \
                   "413, 478, 717,  1479", \
                   "573, 639, 878,  1639",  \
                   "867, 933, 1172, 1933"   );
         }
         cell_rise(template1) {
            values("365, 417, 610,  1222",  \
                   "417, 469, 662,  1274", \
                   "573, 626, 818,  1431",  \
                   "860, 913, 1105, 1718"   );
         }
         fall_transition(template1) {
            values("87,   205,  635,  2006",  \
                   "184,  301,  732,  2102", \
                   "472,  590,  1020, 2390", \
                   "1001, 1119, 1549, 2920"  );
         }
         rise_transition(template1) {
            values("78,  173,  519,  1622",  \
                   "172, 267,  613,  1716", \
                   "454, 548,  895,  1997", \
                   "970, 1065, 1411, 2514"  );
         }
      }
   }
}

cell (bufonx2) {
   area : 0.81818181;
   pin(a) {
      direction : input;
      capacitance : 1.9;
   }
   pin(y) {
      direction : output;
```

```
      function : "a";
      timing() {
         related_pin : "a";
         timing_type  : combinational;
         timing_sense : positive_unate;
         cell_fall(template1) {
            values("357, 393, 524,  945",   \
                   "404, 440, 572,  992",   \
                   "545, 581, 713,  1133",  \
                   "803, 839, 971,  1391"   );
         }
         cell_rise(template1) {
            values("375, 409, 535, 935",   \
                   "415, 450, 575, 975",   \
                   "537, 572, 697, 1097",  \
                   "761, 795, 921, 1321"   );
         }
         fall_transition(template1) {
            values("64,  129, 366,  1122", \
                   "149, 213, 451,  1207", \
                   "403, 467, 705,  1461", \
                   "868, 933, 1171, 1927"  );
         }
         rise_transition(template1) {
            values("57,  119, 345,  1065",  \
                   "130, 192, 418,  1138", \
                   "350, 411, 638,  1358", \
                   "752, 814, 1040, 1760"  );
         }
      }
   }
}
cell (bufonx4) {
   area : 1.09090909;
   pin(a) {
      direction : input;
      capacitance : 1.9;
   }
   pin(y) {
      direction : output;
      function : "a";
      timing() {
         related_pin : "a";
         timing_type  : combinational;
         timing_sense : positive_unate;
         cell_fall(template1) {
            values("461, 482, 560, 808",   \
                   "501, 523, 601, 949",   \
                   "624, 645, 723, 972",   \
                   "849, 870, 948, 1197"   );
         }
         cell_rise(template1) {
```

```
            values("501, 522, 599, 844",    \
                   "528, 549, 626, 871",    \
                   "607, 628, 705, 950",   \
                   "752, 773, 850, 1095"    );
         }
         fall_transition(template1) {
            values("50,  88,  229, 676", \
                   "123, 161, 302, 749", \
                   "344, 382, 523, 970", \
                   "748, 786, 927, 1374"  );
         }
         rise_transition(template1) {
            values("36,  74,  213, 654",  \
                   "84,  122, 260, 701", \
                   "227, 265, 403, 844", \
                   "488, 526, 665, 1106"  );
         }
      }
   }
}

cell (dffron) {
   area : 6.18181818;
   ff("IQ", "IQN") {
      clocked_on : "ck";
      next_state : "d";
      clear      : "rb";
   }
   pin(d) {
      direction : input;
      capacitance : 1.7;
      timing() {
         related_pin : "ck";
         timing_type  : hold_rising;
         timing_sense : non_unate;
         rise_constraint(scalar) { values("0");}
         fall_constraint(scalar) { values("0");}
      }
      timing() {
         related_pin : "ck";
         timing_type  : setup_rising;
         timing_sense : non_unate;
         rise_constraint(scalar) { values("960");}
         fall_constraint(scalar) { values("880");}
      }
   }
   pin(ck) {
      clock : true;
      direction : input;
      capacitance : 2.0;
   }
   pin(rb) {
```

```
      direction : input;
      capacitance : 2.0;
      timing() {
         related_pin : "ck";
         timing_type  : recovery_rising;
         timing_sense : non_unate;
         rise_constraint(scalar) { values("1000");}
      }
   }
   pin(q_b) {
      direction : output;
      function : "IQN";
      timing() {
         related_pin : "ck";
         timing_type  : rising_edge;
         timing_sense : non_unate;
         cell_fall(template1) {
            values("2579, 2646, 2891, 3671",  \
                   "2647, 2714, 2960, 3740", \
                   "2853, 2920, 3166, 3946", \
                   "3231, 3298, 3543, 4324"   );
         }
         cell_rise(template1) {
            values("2035, 2099, 2336, 3088",  \
                   "2099, 2164, 2400, 3153", \
                   "2292, 2357, 2593, 3346", \
                   "2647, 2711, 2948, 3700"   );
         }
         fall_transition(template1) {
            values("102,  222,  664,  2069",  \
                   "226,  346,  788,  2192", \
                   "597,  717,  1158, 2563", \
                   "1277, 1397, 1839, 3243"  );
         }
         rise_transition(template1) {
            values("97,   213,  638,  1993",  \
                   "213,  329,  754,  2109", \
                   "561,  677,  1102, 2457", \
                   "1199, 1315, 1740, 3095"  );
         }
      }
      timing() {
         related_pin : "rb";
         timing_type  : preset;
         timing_sense : negative_unate;
         cell_fall(template1) {
            values("2500, 2600, 2800, 3600",  \
                   "2600, 2700, 2900, 3700", \
                   "2800, 2900, 3100, 3900", \
                   "3200, 3200, 3500, 4300"   );
         }
         cell_rise(template1) {
```

```
         values("2000, 2100, 2300, 3000",  \
                "2100, 2200, 2400, 3100", \
                "2200, 2300, 2500, 3300", \
                "2600, 2700, 2900, 3700"   );
      }
      fall_transition(template1) {
         values("100,  220,  660,  2060",  \
                "220,  340,  788,  2190", \
                "590,  710,  1150, 2560", \
                "1270, 1390, 1830, 3240"  );
      }
      rise_transition(template1) {
         values("90,   210,  630,  1990",  \
                "210,  320,  750,  2100", \
                "560,  670,  1100, 2450", \
                "1190, 1310, 1740, 3090"  );
      }
   }
}
pin(q) {
   direction : output;
   function : "IQ";
   timing() {
      related_pin : "ck";
      timing_type  : rising_edge;
      timing_sense : non_unate;
      cell_fall(template1) {
         values("2751, 2880, 3354, 4860 ",  \
                "2862, 2991, 3465, 4971", \
                "3195, 3325, 3799, 5305",  \
                "3808, 3937, 4410, 5917"   );
      }
      cell_rise(template1) {
         values("2200, 2320, 2760, 4160",  \
                "2305, 2425, 2865, 4265", \
                "2620, 2741, 3181, 4581",  \
                "3201, 3321, 3761, 5161"   );
      }
      fall_transition(template1) {
         values("76, 188, 599, 1906",  \
                "152, 264, 675, 1982", \
                "382, 494, 905, 2212", \
                "804, 916, 1327, 2634"  );
      }
      rise_transition(template1) {
         values("70,  170, 536, 1702",  \
                "144, 244, 610, 1776", \
                "365, 465, 831, 1997", \
                "771, 871, 1237, 2402"  );
      }
   }
   timing() {
```

```
         related_pin : "rb";
         timing_type  : clear;
         timing_sense : positive_unate;
         cell_fall(template1) {
            values("2500, 2600, 2800, 3600",  \
                   "2600, 2700, 2900, 3700", \
                   "2800, 2900, 3100, 3900", \
                   "3200, 3200, 3500, 4300"   );
         }
         cell_rise(template1) {
            values("2000, 2100, 2300, 3000",  \
                   "2100, 2200, 2400, 3100", \
                   "2200, 2300, 2500, 3300", \
                   "2600, 2700, 2900, 3700"   );
         }
         fall_transition(template1) {
            values("100,  220,  660,  2060",  \
                   "220,  340,  788,  2190", \
                   "590,  710,  1150, 2560", \
                   "1270, 1390, 1830, 3240"  );
         }
         rise_transition(template1) {
            values("90,   210,  630,  1990",  \
                   "210,  320,  750,  2100", \
                   "560,  670,  1100, 2450", \
                   "1190, 1310, 1740, 3090"  );
         }
      }
   }
}

cell (dffson) {
   area : 5.63636363;
   ff("IQ", "IQN") {
      clocked_on : "ck";
      next_state : "d";
      preset     : "sb";
   }
   pin(d) {
      direction : input;
      capacitance : 1.7;
      timing() {
         related_pin : "ck";
         timing_type  : hold_rising;
         timing_sense : non_unate;
         rise_constraint(scalar) { values("0");}
         fall_constraint(scalar) { values("0");}
      }
      timing() {
         related_pin : "ck";
         timing_type  : setup_rising;
         timing_sense : non_unate;
```

```
      rise_constraint(scalar) { values("880");}
      fall_constraint(scalar) { values("770");}
   }
}
pin(ck) {
   clock : true;
   direction : input;
   capacitance : 2.0;
}
pin(sb) {
   direction : input;
   capacitance : 2.0;
   timing() {
      related_pin : "ck";
      timing_type  : recovery_rising;
      timing_sense : non_unate;
      rise_constraint(scalar) { values("1000");}
   }
}
pin(q) {
   direction : output;
   function : "IQ";
   timing() {
      related_pin : "ck";
      timing_type  : rising_edge;
      timing_sense : non_unate;
      cell_fall(template1) {
         values("2202, 2271, 2526, 3338",  \
                "2270, 2340, 2595, 3407",  \
                "2476, 2546, 2801, 3613",  \
                "2854, 2924, 3179, 3991"   );
      }
      cell_rise(template1) {
         values("2991, 3055, 3292, 4044",  \
                "3033, 3097, 3334, 4086",  \
                "3159, 3223, 3460, 4212",  \
                "3391, 3455, 3692, 4444"   );
      }
      fall_transition(template1) {
         values("104,  229,  688,  2150",  \
                "227,  353,  812,  2273",  \
                "598,  723,  1183, 2644",  \
                "1278, 1404, 1863, 3325"   );
      }
      rise_transition(template1) {
         values("77,  193, 618,  1973",  \
                "152, 268, 694,  2049",  \
                "380, 496, 922,  2276",  \
                "797, 913, 1338, 2693"   );
      }
   }
   timing() {
```

```
         related_pin : "sb";
         timing_type  : preset;
         timing_sense : negative_unate;
         cell_fall(template1) {
            values("2500, 2600, 2800, 3600",  \
                   "2600, 2700, 2900, 3700", \
                   "2800, 2900, 3100, 3900", \
                   "3200, 3200, 3500, 4300"   );
         }
         cell_rise(template1) {
            values("2000, 2100, 2300, 3000",  \
                   "2100, 2200, 2400, 3100", \
                   "2200, 2300, 2500, 3300", \
                   "2600, 2700, 2900, 3700"   );
         }
         fall_transition(template1) {
            values("100,  220,  660,  2060",  \
                   "220,  340,  788,  2190", \
                   "590,  710,  1150, 2560", \
                   "1270, 1390, 1830, 3240"  );
         }
         rise_transition(template1) {
            values("90,   210,  630,  1990",  \
                   "210,  320,  750,  2100", \
                   "560,  670,  1100, 2450", \
                   "1190, 1310, 1740, 3090"  );
         }
      }
   }
}

cell (latchonx3) {
   area : 4.0;
   latch("IQ", "IQN") {
      enable     : "en";
      data_in    : "lat_in";
      clear      : "r_b";
   }
   pin(lat_in) {
      direction : input;
      capacitance : 1.5;
      timing() {
         related_pin : "en";
         timing_type  : hold_falling;
         timing_sense : non_unate;
         rise_constraint(scalar) { values("0");}
         fall_constraint(scalar) { values("0");}
      }
      timing() {
         related_pin : "en";
         timing_type  : setup_falling;
         timing_sense : non_unate;
```

```
      rise_constraint(scalar) { values("500");}
      fall_constraint(scalar) { values("500");}
   }
}
pin(r_b) {
   direction : input;
   capacitance : 2.0;
   timing() {
      related_pin : "en";
      timing_type  : recovery_falling;
      timing_sense : non_unate;
      rise_constraint(scalar) { values("1000");}
   }
}
pin(en) {
   clock : true;
   direction : input;
   capacitance : 6.4;
}
pin(q) {
   direction : output;
   function : "IQ";
   timing() {
      related_pin : "en";
      timing_type  : rising_edge;
      timing_sense : non_unate;
      cell_fall(template1) {
         values("2015, 2036, 2114, 2363",  \
                "2050, 2071, 2149, 2398",  \
                "2155, 2176, 2254, 2503",  \
                "2348, 2369, 2447, 2696"   );
      }
      cell_rise(template1) {
         values("1838, 1860, 1938, 2186",  \
                "1889, 1910, 1988, 2237", \
                "2040, 2061, 2139, 2388", \
                "2317, 2339, 2417, 2665"   );
      }
      fall_transition(template1) {
         values("44,  83,  223, 671",  \
                "107, 146, 286, 734", \
                "297, 335, 476, 923", \
                "644, 682, 823, 1170"  );
      }
      rise_transition(template1) {
         values("58,  97,  237,  684",  \
                "149, 187, 328,  775", \
                "421, 460, 600,  1048", \
                "921, 959, 1100, 1547"  );
      }
   }
   timing() {
```

```
      related_pin : "lat_in";
      cell_fall(template1) {
         values("1795, 1816, 1893, 2138",  \
                "1842, 1863, 1940, 2185",  \
                "1983, 2004, 2081, 2326",  \
                "2241, 2262, 2339, 2584"   );
      }
      cell_rise(template1) {
         values("1902, 1923, 1999, 2240",  \
                "1956, 1977, 2053, 2295",  \
                "2119, 2140, 2216, 2458",  \
                "2418, 2439, 2515, 2757"   );
      }
      fall_transition(template1) {
         values("55,  93,  231,  672",  \
                "139, 177, 316,  757", \
                "393, 430, 569,  1010", \
                "857, 895, 1033, 1474"  );
      }
      rise_transition(template1) {
         values("61,  99,   235,  670",  \
                "159, 196,  333,  768", \
                "453, 490,  627,  1061", \
                "991, 1028, 1165, 1599"  );
      }
   }
   timing() {
      related_pin : "r_b";
      timing_type  : clear;
      timing_sense : positive_unate;
      cell_fall(template1) {
         values("2500, 2600, 2800, 3600",  \
                "2600, 2700, 2900, 3700", \
                "2800, 2900, 3100, 3900", \
                "3200, 3200, 3500, 4300"   );
      }
      cell_rise(template1) {
         values("2000, 2100, 2300, 3000",  \
                "2100, 2200, 2400, 3100", \
                "2200, 2300, 2500, 3300", \
                "2600, 2700, 2900, 3700"   );
      }
      fall_transition(template1) {
         values("100,  220,  660,  2060",  \
                "220,  340,  788,  2190", \
                "590,  710,  1150, 2560", \
                "1270, 1390, 1830, 3240"  );
      }
      rise_transition(template1) {
         values("90,   210,  630,  1990",  \
                "210,  320,  750,  2100", \
                "560,  670,  1100, 2450", \
```

```
                    "1190, 1310, 1740, 3090"  );
          }
       }
    }
}

cell (latch_nores) {
   area : 2.54545454;
   latch("IQ", "IQN") {
      enable     : "en_n";
      data_in    : "lat_in";
   }
   pin(lat_in) {
      direction : input;
      capacitance : 1.0;
      timing() {
         related_pin : "en_n";
         timing_type  : hold_rising;
         timing_sense : non_unate;
         rise_constraint(scalar) { values("0");}
         fall_constraint(scalar) { values("0");}
      }
      timing() {
         related_pin : "en_n";
         timing_type  : setup_rising;
         timing_sense : non_unate;
         rise_constraint(scalar) { values("500");}
         fall_constraint(scalar) { values("500");}
      }
   }
   pin(en_n) {
      clock : true;
      direction : input;
      capacitance : 2.8;
   }
   pin(q) {
      direction : output;
      function : "IQ";
      timing() {
         related_pin : "en_n";
         timing_type  : falling_edge;
         timing_sense : non_unate;
         cell_fall(template1) {
            values("992,  1066, 1335, 2193",  \
                   "1052, 1126, 1395, 2253", \
                   "1232, 1305, 1575, 2432", \
                   "1561, 1634, 1904, 2761"   );
         }
         cell_rise(template1) {
            values("1396, 1466, 1726, 2552",  \
                   "1431, 1502, 1762, 2610", \
                   "1539, 1610, 1869, 2695", \
```

```
                "1736, 1807, 2066, 2892"   );
      }
      fall_transition(template1) {
         values("97,   230,  715,  2259",  \
                "205,  338,  823,  2367", \
                "529,  661,  1147, 2689", \
                "1121, 1253, 1739, 3281"  );
      }
      rise_transition(template1) {
         values("76,  230, 670,  2156",  \
                "139, 266, 734,  2261", \
                "333, 461, 927,  2414", \
                "688, 815, 1739, 2768"  );
       }
   }
   timing() {
      related_pin : "lat_in";
      cell_rise(template1) {
         values("1112, 1185, 1452, 2303",  \
                "1176, 1248, 1516, 2367", \
                "1367, 1440, 1707, 2558", \
                "1717, 1790, 2057, 2908"   );
      }
      cell_fall(template1) {
         values("1004, 1085, 1382, 2327",  \
                "1066, 1147, 1444, 2389", \
                "1251, 1332, 1628, 2574", \
                "1590, 1671, 1968, 2913"   );
      }
      fall_transition(template1) {
         values("104,  250,  785,  2486",  \
                "216,  362,  896,  2597", \
                "549,  695,  1228, 2930", \
                "1159, 1305, 1840, 3541"  );
      }
      rise_transition(template1) {
         values("101,  232,  713,  2245",  \
                "216,  346,  828,  2360", \
                "560,  691,  1172, 2704", \
                "1190, 1321, 1802, 3334"  );
      }
   }
}
pin(q_b) {
   direction : output;
   function : "IQN";
   timing() {
      related_pin : "en_n";
      timing_type  : falling_edge;
      timing_sense : non_unate;
      cell_fall(template1) {
         values("1176, 1319, 1845, 3516 ",  \
```

```
          "1281, 1424, 1949, 3620", \
          "1593, 1737, 2261, 3933",  \
          "2167, 2310, 2835, 4506"   );
}
cell_rise(template1) {
   values("1277, 1406, 1876, 3375",  \
          "1382, 1509, 1981, 3479", \
          "1695, 1824, 2295, 3793",  \
          "2271, 2400, 2870, 4369"   );
}
fall_transition(template1) {
   values("76, 188, 599, 1906",  \
          "152, 264, 675, 1982", \
          "382, 494, 905, 2212", \
          "804, 916, 1327, 2634"  );
}
rise_transition(template1) {
   values("70,  170, 536, 1702",  \
          "144, 244, 610, 1776", \
          "365, 465, 831, 1997", \
          "771, 871, 1237, 2402"  );
}
```

# APPENDIX B

## MANAGING THE WITHIN-DIE VARIATION MISMATCH

The preceding results compared the performance of MDP memory to gain cells in the presence of die-to-die threshold voltage variance. For both MDP memory and traditional gain cells, there are also within-die variations.

Within-die threshold voltage variation mismatch is inversely proportional to the area of the transistors and is given by

$$\sigma^2(\Delta V_T) = A_{VT}^2/W * L \tag{B1}$$

where $W$ is the gate width, $L$ is the gate length, and the proportionality constant $A_{VT}$ is technology dependent [9]. To determine the effect of the mismatch in (24) on the read signal in (17), substitute the expression

$$vref_{eff} = vref + \delta V_T \tag{B2}$$

in place of *vref* where

$$\delta V_T = 4 * \sqrt{\sigma^2(\Delta V_{T2}) + \sigma^2(\Delta V_{T3})} \tag{B3}$$

where $vref_{eff}$ is the effective reference voltage due to the within-die mismatch of the threshold voltage, *vref* is the reference voltage, $\delta V_T$ is the 4-sigma value of the within-die mismatch of the threshold voltage of the modified differential pair, $\Delta V_{T2}$ is the M2 threshold voltage mismatch and $\Delta V_{T3}$ is the M3 threshold voltage mismatch.

Figure C.1 illustrates the relationship used to set the voltage reference levels in a BASE4 MDP memory bit cell as a function of the within-die threshold voltage mismatch. A Gaussian curve for each of the three reference voltages represents the within-die mismatch of the threshold voltage. The mismatch determines the $vref_{eff}$ value which in turn defines the minimum logic value using (C2).

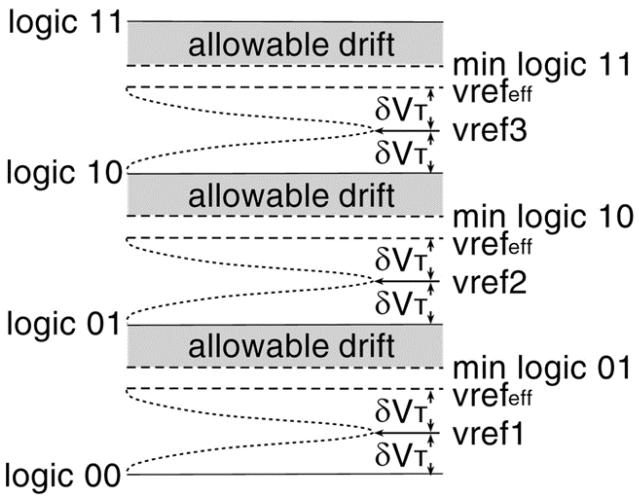


Figure B.1 Relationship between reference voltage levels, logic voltage levels and mismatch in MDP BASE4 bit cell.

The test structure in this work has modified differential pair transistor sizes of W=0.4µ, L=0.18µ for M2 and W=4.0µ, L=0.18µ for M3. The proportionality constant $A_{VT}$ equals 5.5mVµm for the XFAB 180 nm process. Therefore, from (C3) the theoretical within-die mismatch for the bit cell implemented in this work is 86mV.

In MDP memory, since M3 is shared with many bit cells, it can be arbitrarily large. The M2 hold node transistor can also be made much larger than minimum size for the technology node because its gate capacitance can make up part or all of the storage capacitance. If the larger sizes of W=1.0µ, L=0.5µ were chosen for M2 and W=10µ, L=0.5µ for M3, the theoretical within die mismatch is reduced to 33mV.

Figures C.2, C.3, and C.4 show the results of Monte Carlo mismatch simulations each with 500 runs for the MDP bit cell of Figure 2.1(b) with the physical values implemented in the test structure given in Figure 2.11. The three simulations illustrate the effect on read access time of the drifting in the hold node voltage from 1.1 to 1.0V for logic b'10. The reference voltage for logic b'10 of 0.84V is $\delta V_T$ greater than the b'01 logic level of 0.75V. The simulations are run with 0.1pF on the read bit line, the equivalent of approximately 128 bit cells per bit line. For all cases, the specified target of 40ns read access time was achieved in the presence of mismatch.

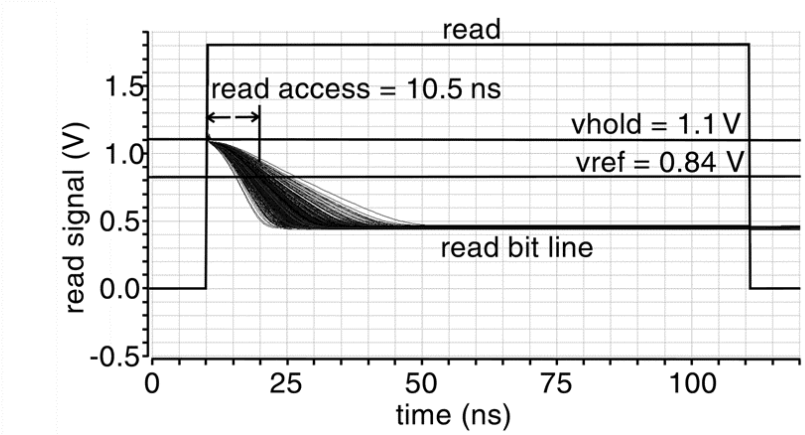


Figure B.2 Monte Carlo mismatch simulation for MDP memory read access time with vhold = 1.1V.

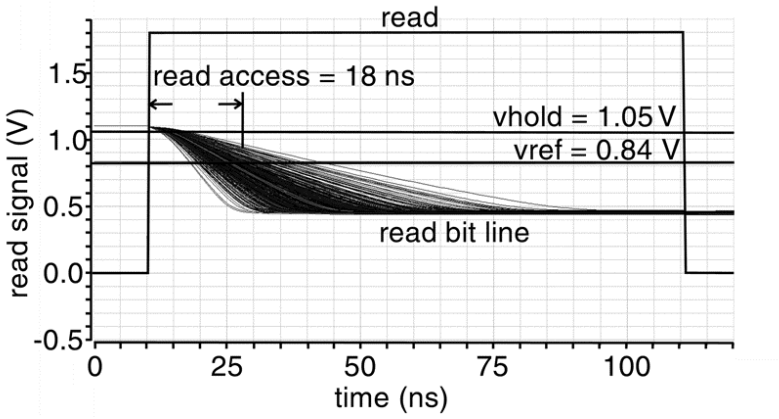


Figure B.3 Monte Carlo mismatch simulation for MDP memory read access time with vhold = 1.05V.

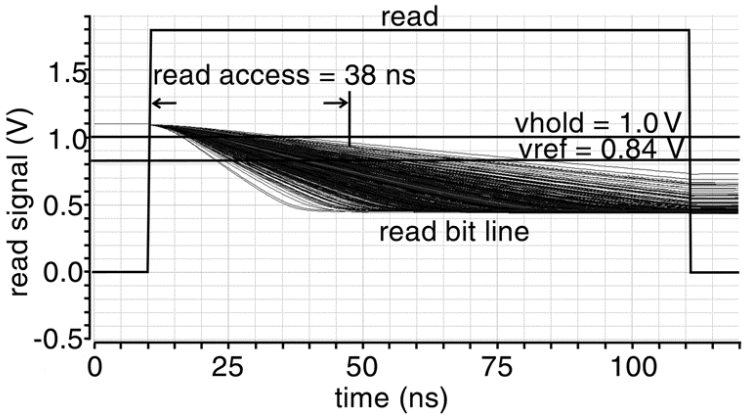


Figure B.4 Monte Carlo mismatch simulation for MDP memory read access time with vhold = 1.0V.

# PUBLICATIONS

Published

H. Bhamra, Y. Kim, J. Joseph, J. Lynch, O. Gall, Henry Mei, Chuizhou Meng, J. Tsai, P. Irazoqui "*A Batteryless, Crystal-free, Multinode, Synchronized SoC Bionode for Wireless Prosthesis Control"* IEEE JSSC, Vol. 50 No. 11, Sept. 2015

J. Lynch, P. Irazoqui "*A Low Power Logic-Compatible Multi-Bit Memory Bit Cell Architecture With Differential Pair and Current Stop Constructs"* IEEE TCAS-I, Vol. 61, No. 12, Dec. 2014

H. Bhamra, J. Lynch, M. Ward, P. Irazoqui, "*A Noise-Power-Area Optimized Biosensing Front End for Wireless Body Sensor Nodes and Medical Implantable Devices"* IEEE Trans. on VLSI Systems, Vol. 25, No. 10 May 2017

Under Preparation

J. Lynch, S. Mohammadi, "*Optimal Energy Use in Sub-Threshold Mode Using Current Controlled CMOS Logic"*

J. Lynch, S. Mohammadi, "*131K Bit Logic-Compatible Low Power Current Controlled DRAM Array"*