ABSTRACT

ILES, PHILIP MICHAEL. A Testbed for Technology Characterization. (Under the direction of Dr. W. Rhett Davis).

As feature sizes continue to decrease, fundamental properties of MOSFET devices begin to hinder the performance gains from one generation to another. The advent of the Tunneling Field Effect Transistor (TFET) provides hope for continued reduction in feature size whilst solving some of the scaling issues such as leakage current. The purpose of this work is to discuss key metrics that help to quantify the improvements among technology nodes, specifically a comparison between TFETs and traditional MOSFETs. Test structures that allow for the measurement of on and off current, device speed, variation as it relates to on current and threshold voltage, as well as SRAM yield and bitcell read and write noise margins are discussed. In addition, a slight modification to a rapid characterization test structure used to measure threshold variation is proven to help reduce leakage seen within the test structure. Lastly, the structures are fabricated in a 90nm bulk and a 45nm SOI process and measurements from the 90nm bulk process are presented.

A Testbed for Technology Characterization

by
Philip Michael Iles

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Electrical Engineering

Raleigh, North Carolina

2009

APPROVED BY:

Dr. Xun Liu

Dr. Paul Franzon

Dr. W. Rhett Davis
Chair of Advisory Committee

DEDICATION

To my wife, mom, dad, and the rest of my family for all their support over these years.

BIOGRAPHY

Philip Iles, born in Downer's Grove, Illinois, a suburb of Chicago, moved to North Carolina at an early age. He attended North Carolina State University from 2003 to 2007 to obtain his two Bachelor of Science degrees in Electrical and Computer Engineering. Since January of 2008 he has pursued his Master of Science degree under the direction of Dr. Rhett Davis. After graduating Philip plans to join IBM in Research Triangle Park, NC to continue his Co-op work with high performance register file design as a full-time employee.

ACKNOWLEDGEMENTS

The opportunity afforded for this research project was provided by funding from DARPA with special thanks to Dr. Michael Fritze and Matt Casto.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

The physical limitations of traditional fabrication techniques are not many more generations away. Photolithography engineers have bought some time by developing lithography techniques that manipulate light in such a way that devices smaller than the wavelength of the light used to expose the silicon can be created. However, it has become apparent that a new type of device must be designed if engineers are to continue to push the limits of transistors. Tunneling transistors, referred to as TFETs by some, seem to be proving themselves worthy of the becoming the future de facto device. With the advancement of TFET device technology, a set of metrics must be established to quantify the improvement over current state-of-the-art technologies. This work discusses key metrics that help to define the advancement of TFET technology and provides discussion and implementation of test structures that can be used to measure these metrics.

Section 1.1 Defining Characteristics of Technologies

This work will focus on four key metrics that can be used to help quantify improvements observed between traditional devices and the new TFET structures. The four metrics include: current consumption, device variation, device speed, and yield. These four characteristics of technology provide insight into the overall usefulness of a device.

Current consumption, both on- and off-state, or leakage, of a device is critical to fully characterize. As technologies have shrunk, the leakage of a device has become increasingly problematic. Methods, such as body biasing, to reduce the leakage through a device in a sleep state are under constant development in existing technologies. However, the methods used to reduce the leakage of a circuit generally have negative impacts on the overall design of the circuit. As with the example of body biasing, a large area increase is observed when an extra body contact is necessary for each device. TFETs provide hope that the on and off state current consumption of a device can be dramatically improved without this necessary overhead in current technologies.

The next characteristic, device variation, has become an increasingly interesting topic as feature sizes have reduced to less than the wavelength of the light used in traditional lithography. Because of this, among other reasons, the device to device process variation has become more and more significant. Since the variation has become greater as technology shrinks, it is important to understand how much variation should be expected in order to accurately simulate the behavior of a circuit. For the purposes of this work, the variations in threshold voltage and on-state current are the primary focus of device to device variation.

While simultaneously reducing both current consumption and device variation, the speed of the device cannot be left as an afterthought. This is one area that TFETs may not be able to outperform current technology, at least as of yet. TFETs operate at a lower supply voltage than traditional transistors and thus cannot necessarily be expected to perform to the same standards. This leads to one of the issues in comparing two strikingly different devices. Though the speed of a new TFET needs to be fast enough to "compete" with its traditional FET counterpart, a designer must take into consideration the potentially enormous savings in power consumption as it relates to the reduction in performance. However the comparison is quantified, the importance of characterizing the speed of a device is unquestionable.

Lastly, almost no complex digital system is complete without some sort of storage mechanism. In the majority of cases, this temporary storage exists in the form of an SRAM. Not only must the SRAM be fast, but it must also be reliable. Measuring the yield of an SRAM is vital considering their ubiquitous use in so many applications today. Due to the extensive use of SRAMs in many complex designs, the power consumption is also an important metric to measure of an SRAM. Arguably, the most important power metric related to an SRAM is its leakage current. Considering the sheer number of devices in a bit-cell array itself, the impact of reducing the overall leakage of each device becomes more and more significant.

Section 1.2 Premise of the STEEP program and the XChips

All of these attributes lead up to the fundamental characteristics of interest of the STEEP program. The STEEP program [1] is a DARPA initiative for investigating **S**teep-Subthreshold-Slope **T**ransistors for **E**lectronics with **E**xtremely Low-**P**ower. In order to determine the validity of the development of a new type of transistor, the specifications of current technologies should be investigated as a comparison for these new devices. This work focuses on the design and development of two chips to provide a baseline for these comparisons, referred to hereafter as XChip and X2Chip, and collectively, XChips. XChip, the first of two, was fabricated in a 90nm bulk process. The X2Chip is an improvement on the first and was developed in a 45nm SOI technology. As of the writing of this document, the X2Chip is not available for testing.

The test structures implemented in these chips provide insight into the intrinsic characteristics of the technologies in which they are implemented. Establishing a baseline of the performance of current state-of-the-art devices provides metrics to measure the success of newly developed TFET devices. Each of the test structures mentioned in the next chapter will establish a data point in at least one of the areas of interest. If the test structures are similarly implemented in any technology, a reasonable comparison can be made through the same testing methodologies.

Section 1.3 Tunneling Device Introduction

As devices have continued to shrink in feature size, one branch of semiconductor development has begun investigating tunneling devices. TFETs, as a general statement, operate by electrons moving from the valence band to the conduction band by passing through the semiconductor bandgap [2]. Referred to as HETTs (Heterojunction Tunneling Transistors) in [3], the authors present their findings related to one manifestation of a tunneling device that has been developed in conjunction with the STEEP program.

HETTs function in a way similar to traditional MOSFETs where by a voltage induced at the gate allows current to flow. However for HETTs, and more generally TFETs, it is not that a channel has formed allowing for the electrons to move from one node to another, rather, the gate bias has reduced the semiconductor bandgap allowing electrons to more easily move from the valence band to the conduction band. The below figure from [3] illustrates the on and off state of a HETT device.

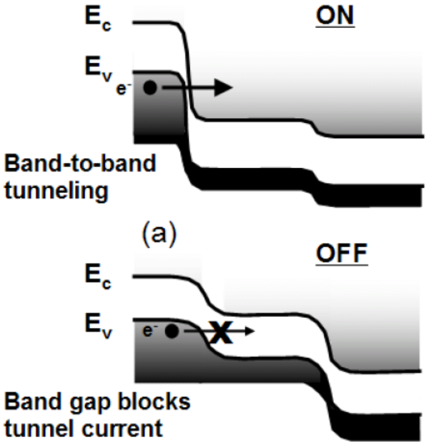


Figure 1.3.1 - Functionality of a HETT [3]

One of the main methods in substantially reducing power consumption of traditional FETs is by lowering the supply voltage. However, reducing the supply voltage below the threshold voltage of a device causes a large increase in off state current. This limitation correlates to the theoretical sub-threshold slope limit of 60 mV/decade. However, HETTs, and more generally TFETs, do not suffer from the thermionic limitations that create the inversion layer in a MOSFET and thus can push the limit of subthreshold slopes below 60 mV/decade all while reducing supply voltage and leakage.

Chapter 2 Test Structures and Methods Explored

Section 2.1 Individual Transistors

This basic intention of the XChips was to provide very the most fundamental metrics of individual devices. Thus, to ensure that this goal was met, both N- and P- type transistors were individually padded out. In order to maximize the number of devices that could be tested with the pad set and area limitations, nodes of the devices were tied together. The two illustrations below depict the schematic arrangement of the individual transistor probe sites.

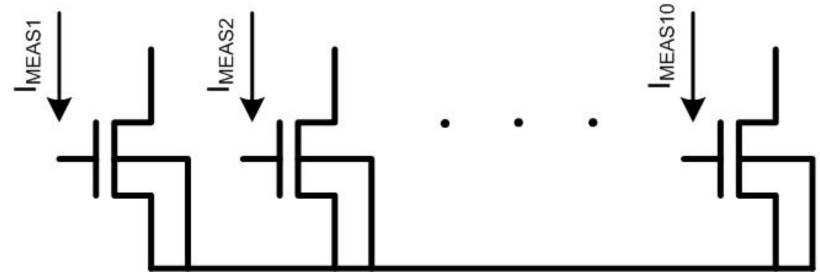


Figure 2.1.1 - NMOS Transistor Structure

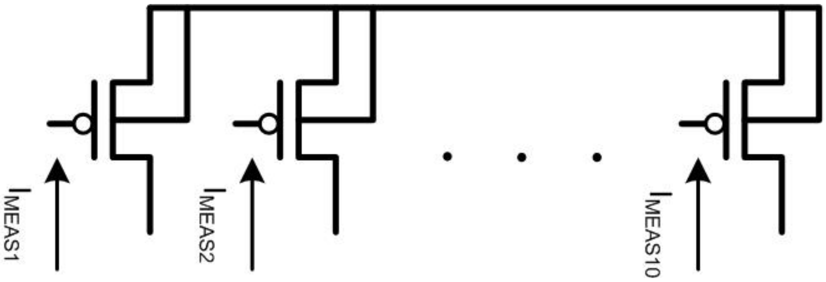


Figure 2.1.2 - PMOS Transistor Structure

Section 2.2 Ring Oscillator

The most fundamental logic building block used in digital design today is the inverter. Not only do so many logic operations require both the positive logic as well as its complement (for example: muxing) but this simple device is also the basis of distributing clocks throughout an entire chip. The inverter, in odd numbers, also serves as the basis for a ring oscillator. Because the inverter is used so widely it has become a unit of comparison for a number of defining characteristics of technologies. For this reason, we can use the current and power consumption of an inverter to understand the power consumption of a particular technology. By creating a simple ring oscillator in any technology, the current consumed by each inverter in the chain can be determined and thus a known value of power consumption for this device can be observed. With a common implementation of a ring oscillator being enabled by a NAND gate, when the ring oscillator is disabled, the static power consumption can also be measured since the ring oscillator is no longer oscillating.

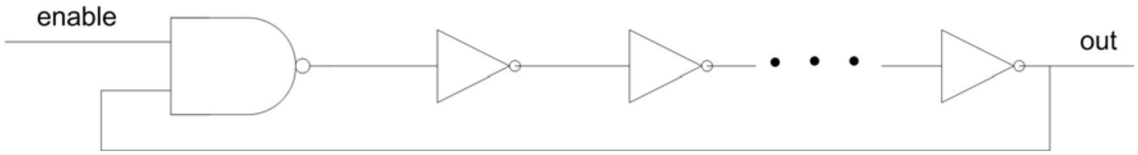


Figure 2.2.1 - Basic Ring Oscillator

When analyzing a ring oscillator in a technology, if the number of inverting stages is known and the output frequency can be measured, then the switching speed of the devices can be extrapolated. Assuming the number of inverters is much, much larger than the single NAND used for the enabling of the ring oscillator, the difference between the average delays seen through the inverters compared to the single NAND can be ignored thus providing an accurate measure of delay through the inverter. Furthermore, if the ratio of N to P devices in the inverter is known, the general "strength" of each of these devices can be measured, providing even more insight into the behavior of the individual device.

Section 2.3 Rapid Characterization of Threshold Variation

Another characteristic of a technology that warrants thorough exploration is the process variations that impact circuit design, particularly threshold voltage variation. Especially when a technology is in its infancy, it is crucial for the foundry to be able to provide the most accurate models of their devices to allow the design engineers to accurately simulate the behavior of their circuits. The ability to create statistical models of device parameters allows the foundry to improve the simulation capabilities and thus improving yield of the designer's circuits due to increased understanding of the behavior of a circuit.

The test structure from [4] describes a circuit that allows the variation of threshold voltages to be measured in a process that is many times faster than traditional methods. The fundamental operation of the circuit is to force a constant current through a device and measure VGS which will be shown to directly relate to the threshold voltage. This operation relies upon the drain voltage to vary in such a way to keep the current through the device constant regardless of the change in threshold variation.

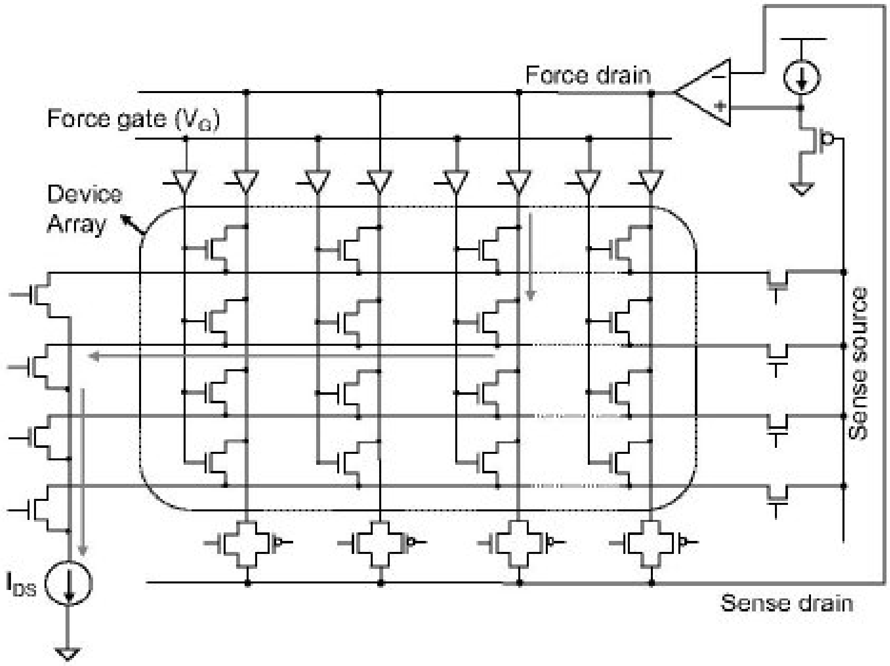


Figure 2.3.1 - Rapid Vt Characterization Structure [4]

The device array depicted in the circuit diagram above can be arbitrarily large; for the purposes of [4] the size of the array was approximately 8000 devices. A device is selected by enable signals driven by peripheral logic, such as flip-flops or a decoder. The row select signals control the devices along the left and right side of the structure, the source current path and the sense source path. The column enable signals control the transmission gates at the top and bottom of the circuit providing a path for the drain current to flow. It's worthwhile to mention that the use of a traditional tri-state buffer symbol at the top of the circuit as suggested in the author's original circuit diagram is a bit misleading. In order observe the expected behavior of the circuit, these gates should actually be considered transmission gates, the same as seen at the bottom of the circuit. By enabling one row and one column, one particular device in the array becomes the device-under-test (DUT). The current is forced to flow in one direction from the opamp through the DUT and along the common source node for the row then out of the circuit to the current source. This path is illustrated by arrows in the above figure.

Since the current is forced in one direction only, both the drain and source voltages can be sensed without the worry of parasitic IR drop. The sensed voltage of the drain and the inverted sensed voltage of the source (through the source follower at the input of the opamp) provide the necessary input to the opamp to modify the output voltage in order to maintain a constant current through a selected device.

As each device of the array is selected, VGS can be measured. As proven in [4], VGS varies directly with Vth so long as the current is kept constant. This is due to the fact that the current is dependent on the quantity of (VGS – Vth) and not VGS or Vth separately. Therefore the variation in the threshold voltage directly correlates to the variation in VGS and thus the method of measuring Vth variation for this circuit.

To further accelerate the process of measuring the threshold variation, a method to use an oscilloscope or multi-meter to gather first-order statistical information is presented in [5]. The premise of this approach is to use the DC average as the statistical mean and the

RMS value as the first standard deviation of the device mismatch. This approach has been used with the Vt variation structure as mentioned in [5].

Section 2.4 SRAM Bitcell

Any technology that is used to implement complex digital systems must provide reliable, and preferably fast, SRAMs. But before an SRAM can be designed, the bitcell itself must be fully understood in order to properly design portions of the SRAM such as the pre-charge circuitry and the sense amplifiers. Two key metrics that can be quantified to help in this design are the read and write noise margins of the bitcell as discussed in [6]. For the purposes of this work, a typical 6-T SRAM bitcell, below, is used as a test device.

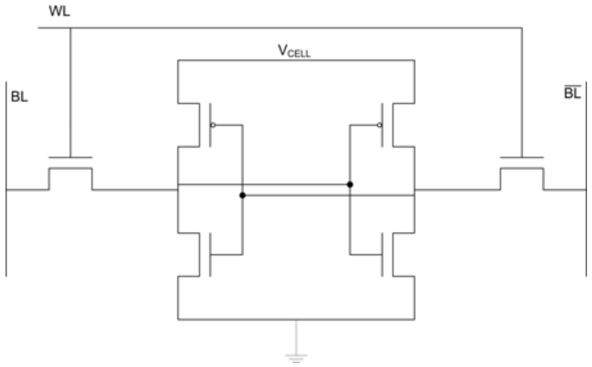


Figure 2.4.1 - 6T SRAM Bitcell

The simplest way of being able to perform the noise margin measurements and to be able to fully characterize the bitcell is to simply pad out each of the nodes depicted in the figure above. With a padded out SRAM bitcell,, the following method can be used to measure the read margin: set WL, BL, and BL_b (denoted as BL with an overbar in the figure) at Vdd; sweep VCELL from Vss to Vdd and the point at which there is an abrupt increase in current through the BL node indicates the read noise margin of the cell. Similarly, the bit line write margin can be identified by setting BL, VCELL, and WL to VDD. Then, BL_b is swept from Vss to Vdd and the point at which the current through the BL node abruptly changes indicates the write noise margin [6]. Another important measure of the performance of the bitcell is its leakage current. In order to measure this, WL is set to Vss while BL and BL_b are set to Vdd, somewhat simulating a pre-charge, and VCELL is swept from Vss to Vdd. These three measurements of the bitcell help to provide necessary information for designers to develop robust SRAMs.

Section 2.5 SRAM March Tests

Beyond understanding the behavior of the bitcell itself, it is crucial to also be able to characterize the effective yield of a particular technology. SRAMs lend themselves very well to yield tests considering the extreme density normally observed within the bitcell array. In addition, the uniformity of the array itself helps to mitigate systematic variation observed between devices. For the purposes of this work, a march test will be used to measure the effective yield of an SRAM.

The premise of a march test is to "march" through each word in the SRAM and read and write certain transitions to and from 1 and 0 for each bit. There are several algorithms discussed in [7] and the trade-offs among each is discussed. For the purposes of this work, two algorithms will be discussed: the first for its pure simplicity to exemplify the concept, and the second to present a straight-forward algorithm with extremely high fault coverage. It should be noted that the algorithm chosen is independent of the hardware unless the march test algorithm is actually implemented on-chip. In this work, the control logic is located off-chip and thus any march test pattern can be used to measure the yield of the SRAM.

A march test consists of several "march elements" that are chained together in a specified order. Obviously there are two operations permitted on a memory: read (r) and write (w). Secondly, in stride with the binary operation of traditional digital electronics, there are only two values that are permitted to be stored in a memory cell: 0 or 1. The last part of the march element is the order in which the operation should be performed in relation to other words in the structure: u (up - ↑), d(down - ↓), ud(up or down - ↕). In [7] the arrow notation is used while in this work u, d, and ud are used as they are the syntax used for the pattern generation script written for this project.

The first, and simplest, algorithm discussed is the MATS+ algorithm. The algorithm, written in the original format looks like:

$$\updownarrow\{w0\};\ \uparrow\{r0,w1\};\ \downarrow\{r1,w0\}$$

This algorithm, in plain English, means to write 0's to all words of the SRAM in any order (ascending or descending). Next, beginning at the bottom of the address range, attempt to read the 0 from each word and then write a 1 to the word, then proceed to the next word until the end of the address range has been reached. Next, start from the top of the address range and attempt to read the 1 from each word followed by writing a 0. This simple notation can very easy describe much more complex algorithms, such as the March C algorithm (which should easily be deciphered based on the previous example):

ud{w0}; u{r0,w1}; u{r1,w0}; d{r0,w1}; d{r1,w0}; ud{r0}

Van de Goor discusses the advantages and disadvantages of each algorithm presented in his work and based on the simplicity of the March C algorithm coupled with its high fault coverage percentage, the March C algorithm is the algorithm of choice for this project.

Chapter 3 Novel Modifications and Improvements

Section 3.1 Rapid Threshold Characterization Improvement

With the continued reduction of transistor sizes accompanied by the increase in leakage current, improvements may be necessary upon the rapid characterization test structure for future use. In order to continue to use this structure as these problems become more significant, this work proposes an approach for reducing the amount of leakage seen in the rapid characterization structure.

In order to quantify the improvement over the original design, the following test circuit was created and simulations with the a few variants were performed and compared to the same measurements of the original design. In this setup, a column of devices to be tested has been extracted from the full circuit. Because the primary focus of the leakage in this structure is the leakage through other devices in the same column, this approach provides a simple yet effective mechanism to compare the currents though the devices. Since tracking the variation in threshold voltage is not in question for this portion of the work, the opamp can also be eliminated thus leaving a simple voltage supply to provide a constant voltage source. To best emulate the behavior of the original circuit, a current source still forces a constant current through the structure just as it would in the original design.

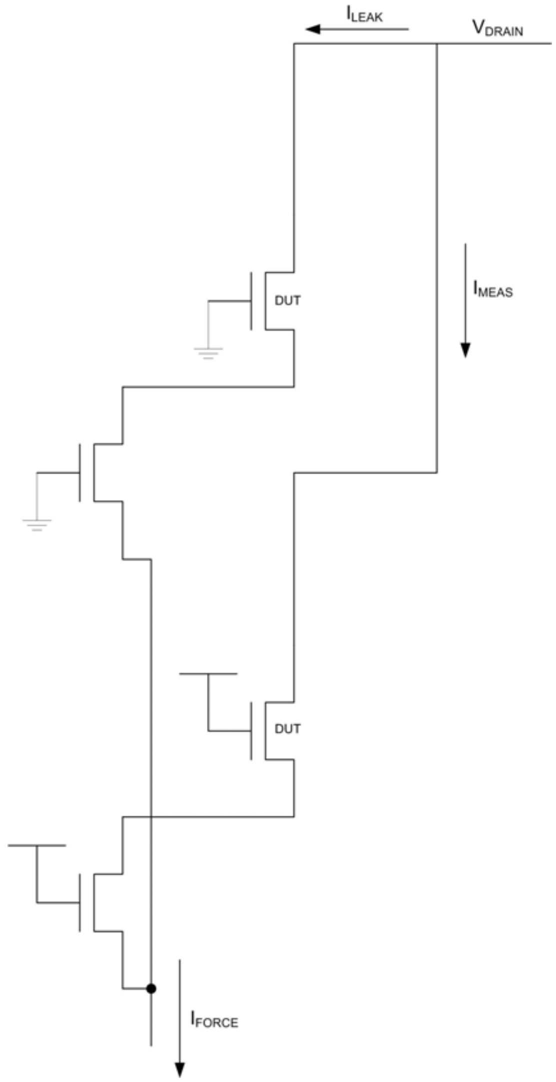


Figure 3.1.1 - Schematic of Leakage Test for Rapid Vt Structure

By placing transmission gates at the drain nodes of the transistors in the DUT array, the leakage though the other inactive devices in the array can be reduced. The below figure illustrates one implementation of this improvement. This implementation uses the same row enable signal for the sense source and sense drain transistors in the structure to enable the drain isolation transmission gate. Since the drain voltage is applied per column, enabling the transmission gate per row effectively reduces the amount of leakage observed in a column of devices.

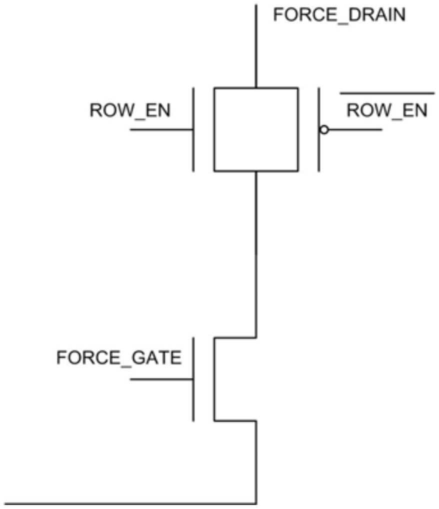


Figure 3.1.2 - DUT Cell with Drain T-gate for Vt Characterization

As an example, using 5um wide device as the force source device (the devices along the left side of the structure), the leakage through an inactive device in the column can be reduced by as much as nearly 18% if the sizes of the transmission gates are selected correctly. The following figure shows the correlation between device size and leakage observed.

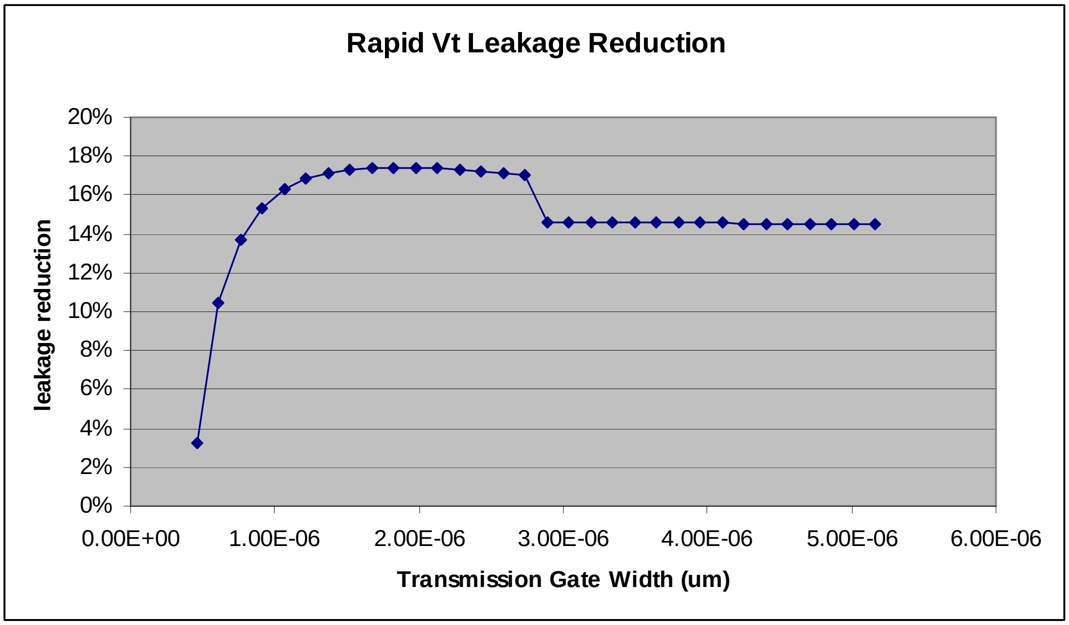


Figure 3.1.3 – Simulated Leakage Reduction with Transmission Gates at Gate and Drain

Two different approaches to an enabling scheme for the transmission gates used to isolate each device were explored. The first, as depicted in the above figure, is to enable the drain isolation transmission gates by row. Another enabling scheme is to force the enable explicitly by both column and row by introducing a simple NAND gate followed by an inverter whose input is the column and the row enable signals of the array. The device under test cell would then look like the following:

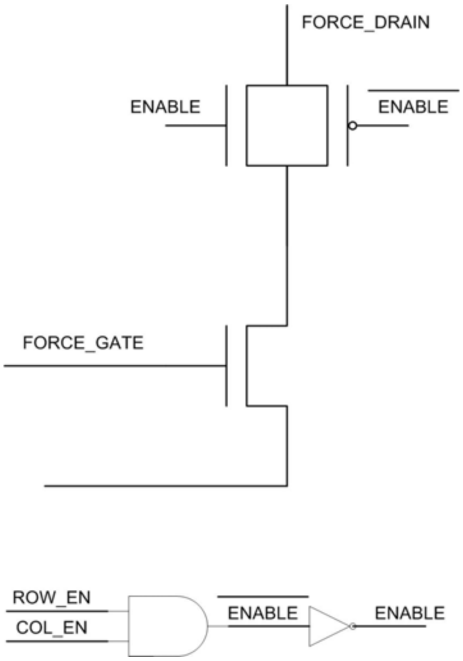


Figure 3.1.4 - NAND Enable Scheme for DUT Cell for Rapid Vt Characterization

The trade off between enabling schemes is obviously area. As logic is added to each of the DUT cells, the size will increase. However, it should be noted the transistor sizes needed for the NAND and inverter can be minimum size as high-performance is unnecessary for this test structure due to the relatively slow clocking speed.

Section 3.2 Usage of Rapid Characterization for On Current

With only slight modifications to the previous discussed test structure, it can also be used to measure the on state current variation of devices in an array. In order accomplish this, the circuitry used to sense the drain and source voltages and the operational amplifier are simply replaced with a voltage source or, as implemented for this work, an off-chip pin. By providing a constant voltage for the drain and simply attaching the force-source node to ground instead of a current source, the same procedure used to measure threshold variation can be used to measure the on-state current variation. The following figure shows the topology of this test structure.

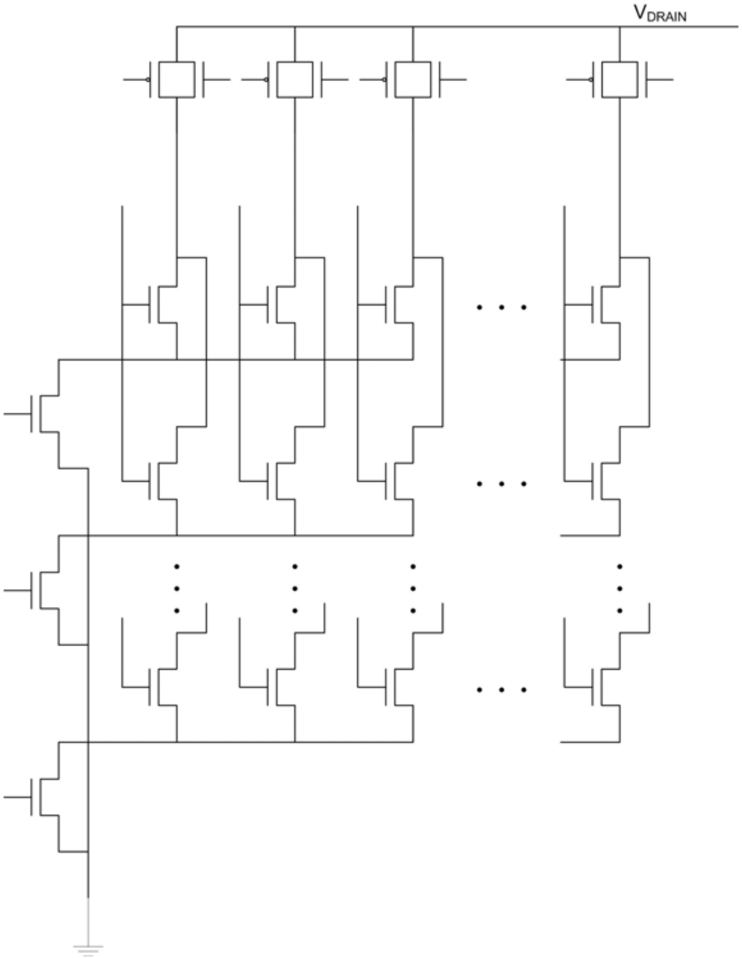


Figure 3.2.1 - Rapid Ion Variation Test Structure

In this implementation of the circuit, the stand-by (or leakage) current can be measured by disabling all select signals and measuring current draw. Then, just as performed with the Vt variation structure, each device in the array is individually selected by row and column enables. The same measurement approach denoted in [6] can be used as well to expedite the measurement process.

Chapter 4 Implementation

As with any project, design trade-offs must be weighed to determine the most effective solution to the problem at hand. The development of the XChips was no different. In this section the design decisions and actual implementations in each chip are discussed.

Section 4.1 Pad Restrictions

The first challenge of the project was pad restrictions. The testing goal of the STEEP program was to design test structures and methodologies that can be applied by each development team with minimal changes. Therefore it was required that each team use the same testing equipment, namely a common probe card, thus limiting the number of pads to 25. In addition the pad size was also required to be the same among all test groups to ensure the probe card could be used to measure results in each implementation. Because the pads were so large and the limit of 25 pads per test structure was imposed, the size of the layouts was actually dominated by the sheer number of pads rather than the circuits themselves.

Section 4.2 SRAM Test Pattern Generator

Generating test patterns by hand is not only extremely time consuming for large tests but also very prone to errors. For this reason, code was developed that would generate test patterns for the SRAM march test. The March Test Pattern Generator (MTPG) takes the size of the SRAM (data and address sizes) and the algorithm of the march test pattern desired to be performed and creates a test pattern to perform the desired algorithm. This code can be seen in the appendix. The MTPG generates a generic output that indicates the operation, address, and data instruction to be exercised upon the SRAM. This is not enough to be able to run the test. Each team must take the generated pattern and convert it into a format that the available test equipment can understand. The MTPG output can be in decimal, hexadecimal, or binary, providing ample flexibility for an easy conversion between the generic output and the input for the appropriate equipment.

Section 4.3 90nm bulk

The first XChip was manufactured in a 90nm bulk technology. The size for the first chip was rather large, nearly a 5mm x 5mm die. This large die allowed for plenty of room for many padsets. Because of this there were 12 pad sets for individual transistor probing, 6 for N- and 6 for P-type devices. Between each device set, the dimensions of the transistors were varied.

In order to properly assess the yield of a given technology, instantiating a large SRAM was crucial to gather enough data to be statistically significant. However, because the padset was limited to 25 pins and the pins were intended to go off chip, the size of the SRAM was restricted to 5kb. But, with the available area on this die, 20 instances of the 5k SRAM were instantiated bringing the total SRAM bits capable of being measured to 100kb. Though this fell short of the Phase III goal of 128kb, this alternative was satisfactory considering the restrictions and that XChip was only required to meet Phase I goals.

The SRAM bitcell, as mentioned earlier, was instantiated on its own allowing for the noise margin tests to be conducted on a single bitcell. Due to time constrains and license agreement issues, an official dense SRAM bitcell layout was not available in time for tapeout. However a custom layout of the same dimension devices was created to serve as a proof of concept for the measurement methodology.

Section 4.4 45nm SOI

The 45nm X2Chip had an even tighter area restriction. With a little less than 3mm by 3mm of area to utilize, the same test structures were required to be implemented. Again, with the size of the pads dominating the total area, removing as many probe pads as possible led to the greatest reduction in overall area. In this case, modifying the SRAM test structure would prove to be the most effective solution. For the X2Chip the data out pins were muxed and the data in pins were tied together to allow for a 16kb SRAM to be instantiated. VHDL simulations were run to ensure that this muxing approach would not affect the functionality of the SRAM. The downfall to this approach is that not every single bit can be individually written to; instead blocks of bits are written to at any given point in time. However, since march tests write the same value to all bits in a word, this functionality restriction doesn't hinder the yield measurements performed via a march test. The following figure illustrates the muxing technique used for the X2Chip.

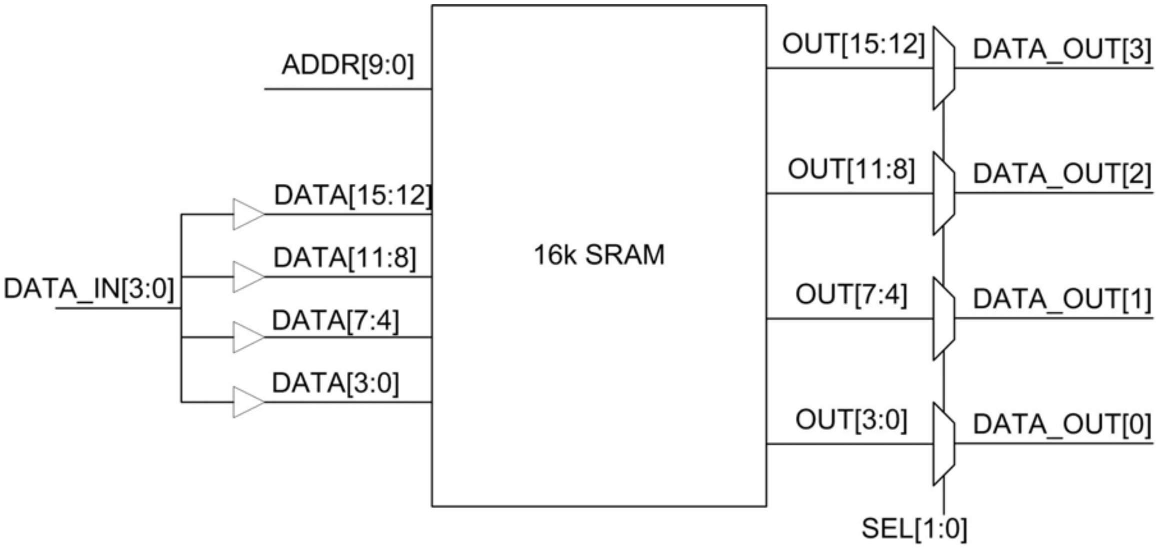


Figure 4.4.1 - SRAM Muxing Approach

Fortunately, an official dense SRAM bitcell was acquired for the tapeout of X2Chip. The key improvements over the XChip are that this SRAM bitcell is both state-of-the-art industry standard and is surrounded by other bitcells just as it would be in a real SRAM bitcell array. The significance of both of these changes is to hopefully reproduce measurement values that are more similar to the behavior of a bitcell as it would operate in an SRAM. Since bitcells are designed to be tiled into a very large, dense array, the wiring for the bitlines and wordlines is built into the cell itself and the overlapping of the bitcells creates the wiring among them. For the purposes of this test, the connectivity had to be removed from surrounding the bitcells in order to isolate just one bitcell for test.

Another modification made between the generations of XChips dealt with the layout of the individual transistor sizes. Because transistor sizing has become less granular as the technology has shrunk, varying the size of the transistors ever so slightly was not a reasonable task to accomplish. As mentioned throughout this work, the concern of variation plays an important role in the characterization of a technology; therefore, individual transistor sizes with varying layouts were created to measure how this will affect the basic characteristics of the transistors. Below are images of the six different layouts of individual transistor sites for the X2Chip with a briefly descriptive caption indicating the physical variation. The layouts chosen for this are based on the study performed in [8].

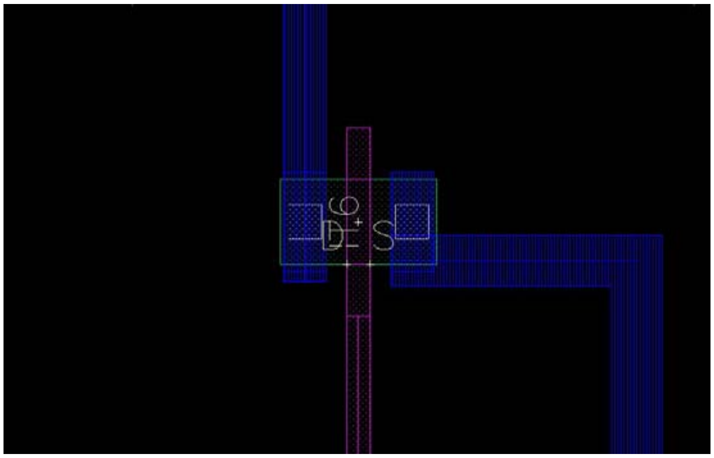


Figure 4.4.2 - Single Transistor with no ACLV Gate

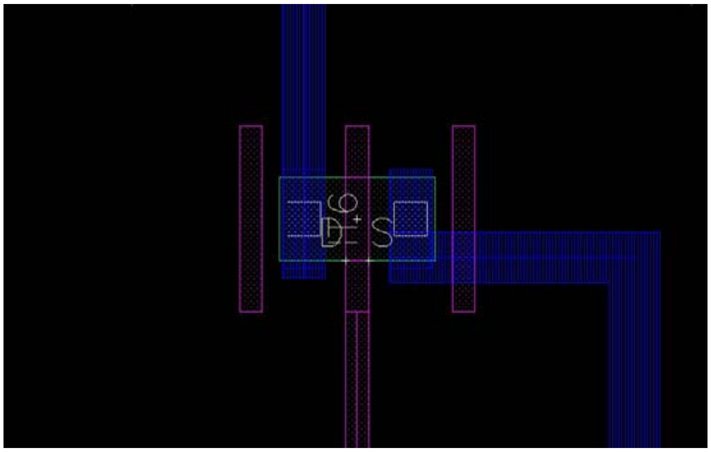


Figure 4.4.3 - Single Transistor with ACLV Gate on both sides

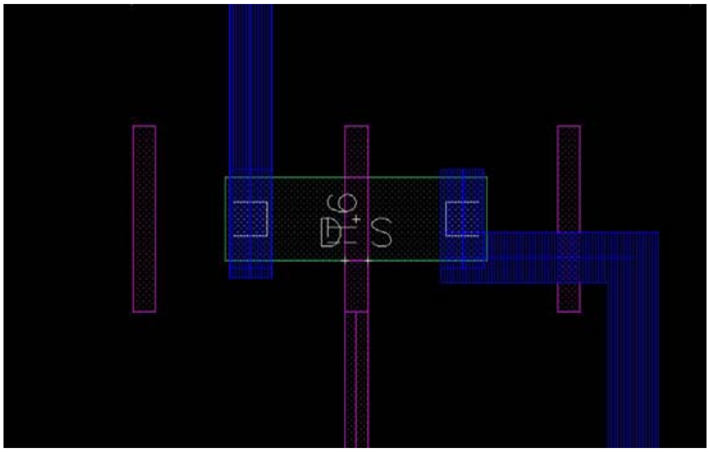


Figure 4.4.4 - Single Transistor with ACLV at 2x Poly Pitch

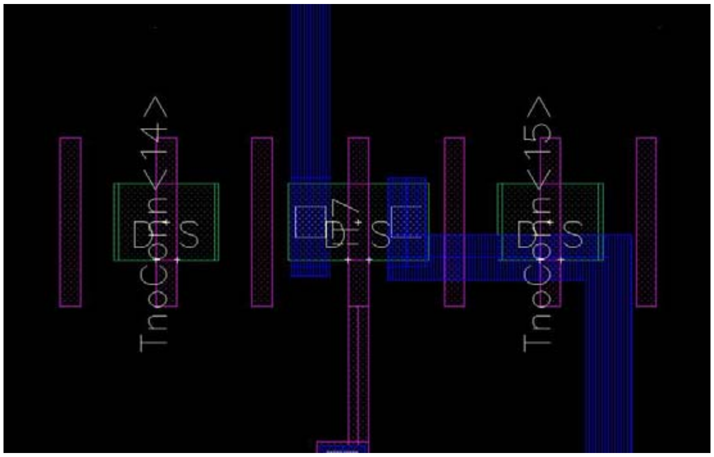


Figure 4.4.5 - Single Transistor Abutted to Transistors with ACLV

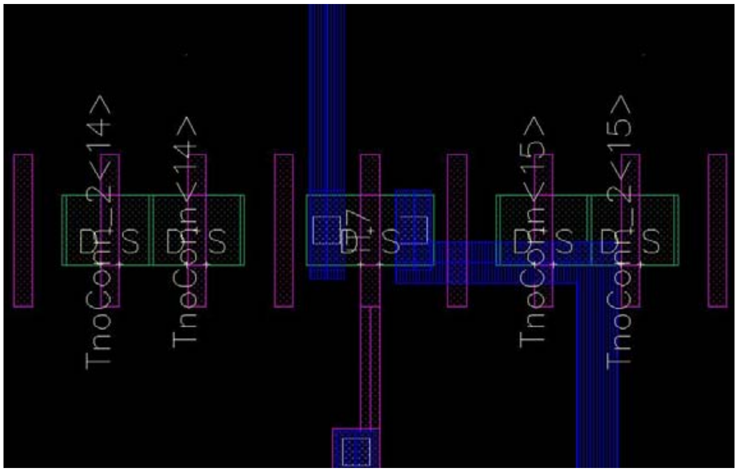


Figure 4.4.6 - Single Transistor with Multi-finger Devices on each side

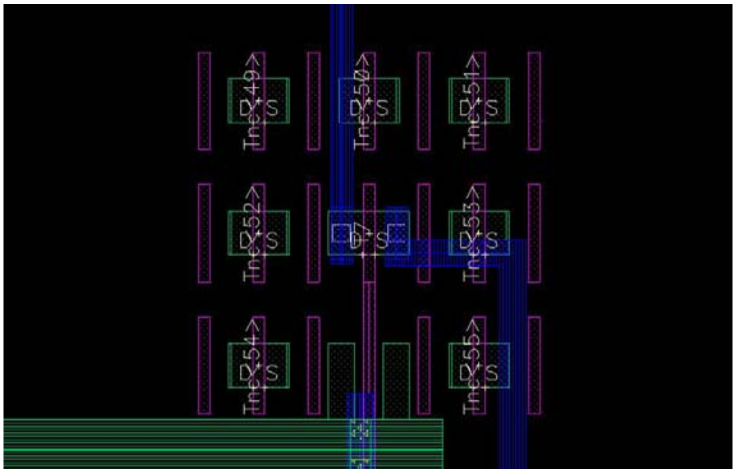


Figure 4.4.7 - Single Transistor surrounded by Single-finger Devices

Chapter 5 Lessons Learned

As with any design process, not all aspects of the design can be foreseen and planned for from the first phase of the design process. This section recounts several lessons learned at later stages in the development of the XChips and provides commentary of their impacts on the two designs.

Section 5.1 Antenna Diodes

Antenna diodes are required because of the manufacturing process through which modern semiconductors are fabricated. During the fabrication process, as metal layers are created and vias are created between these layers, electrical charge actually begins to build up on these metal wires. The wires, acting essentially as capacitors, hold this energy until enough accumulates that the barrier preventing its discharge is broken. During the manufacturing process, the wafer itself is grounded by the equipment used. Therefore, the path to ground must pass through the wafer, whatever path that happens to be. For the case in which the path is directly connected to a drain or source region of a transistor, this is not a problem because these regions can withstand the current flow. However, this becomes a problem when the only connection on a particular net is polysilicon. The polysilicon is not designed to be able to pass such currents and hence the development of the antenna design rule checks.

The antenna design rule checks calculate the total area of metal connected to each polysilicon shape. When this ratio breaches a particular threshold a violation is flagged. In the general design of circuits, this ratio is rarely reached. However, once the metal of the pad and corresponding vias are taken into consideration at the chip level, this becomes a prevalent issue. The following depicts an antenna rule violation that was commonly observed.

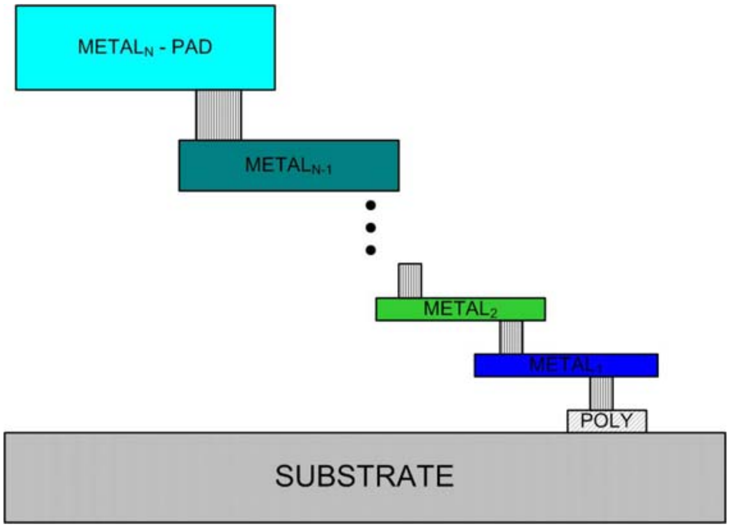


Figure 5.1.1 - Antenna Diode Failure

This example illustrates the common configuration in which a pad directly drives to an inverter as its first sink. Since the size of the polysilicon is quite small relative to the total area of the multiple metal layers, it is no surprise that this violation occurs for this situation. The next figure shows how this problem can be solved. By adding a small diode in parallel to the gate in question the antenna rules are satisfied thus avoiding the potential destruction of a gate.

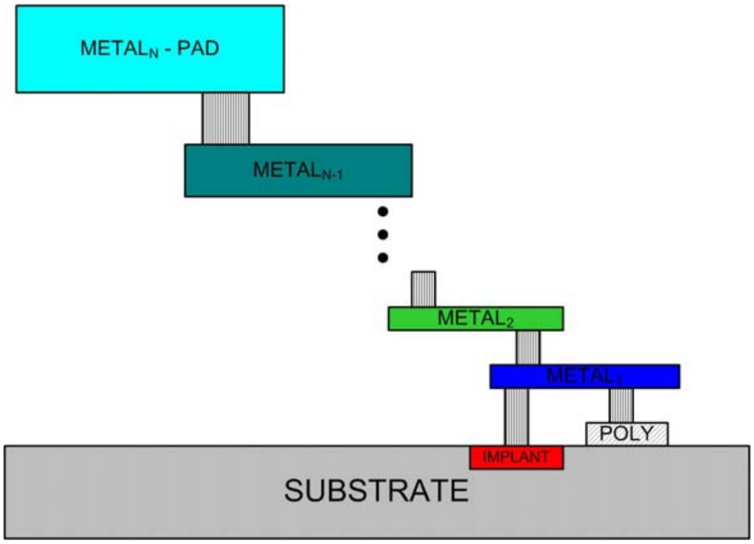


Figure 5.1.2 - Antenna Rule Fix

Antenna diodes are essentially reverse biased diodes that are used in parallel with any net that exhibits this violation. However, they have no schematic representation; i.e., LVS does not recognize an antenna diode as a device which needs a schematic counterpart. To some degree an antenna diode can be thought of in the same manner as ESD protection on a chip, with the difference being that it is intended to safeguard the chip during fabrication not packaging or use.

Since these antenna diodes are required in order to pass DRC rules, the concern of these diodes affecting the accuracy of the measurements performed on the test structures became a concern. In order to measure the impact that these antenna diodes had on the design, all unused pads had an antenna diode attached so that the I-V characteristics of the diode could be measured and its leakage taken into consideration should it prove to be significant enough.

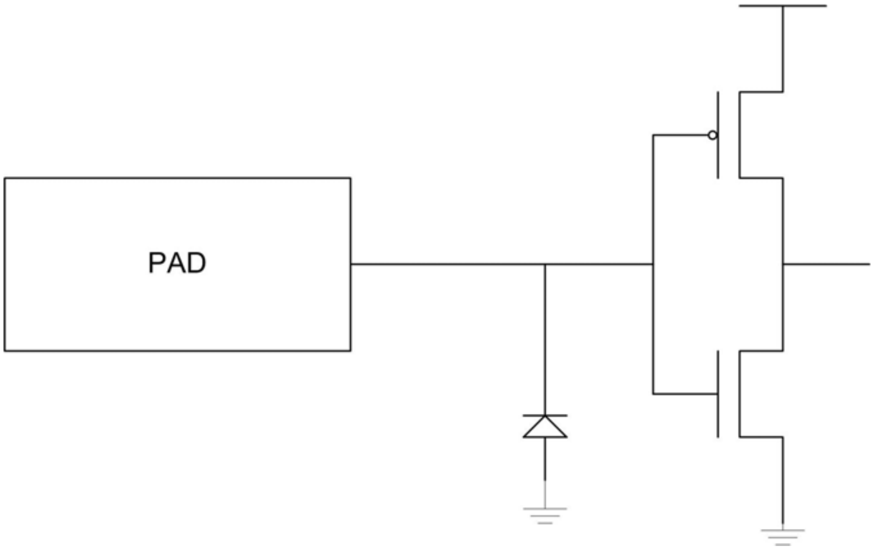


Figure 5.1.3 - Antenna Diode Equivalent Schematic

Section 5.2 Frequency response of Celadon probe card

As mentioned in previous chapters, part of the requirement of the STEEP program was for all groups to utilize the same probe card. An unforeseen limitation of the probe card was its frequency response. The probe card was chosen to satisfy the Phase I goals of the STEEP program which only sought to measure the DC characteristics of transistors. However, since the XChips provided test structures for metrics beyond Phase I of the program, the frequency response of the probe card became a concern.

As an example, testing the frequency output of the ring oscillators required using manual probes landed on the probe pads instead of utilizing the probe card which lands all pads at once. This proved to be a rather complex task as five probes were required to monitor one ring oscillator. The process of setting up the probes and ensuring proper connectivity can take upwards of an hour. Since the goal was to measure the power consumption and frequency of small transistors, larger output buffers had to be created to give enough power to the drivers to get off-chip. However, this required that the buffers be on a separate power rail than the ring oscillator core, thus introducing the fifth pin. The total pin out was: vdd_core, vdd_buf, enable, out, and gnd as seen in the below figure.

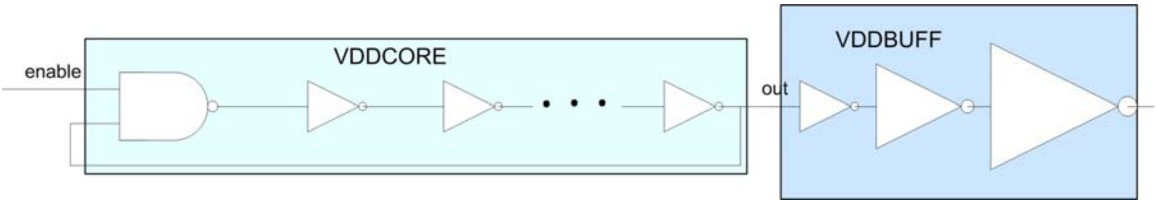


Figure 5.2.1 - XChip Ring Oscillator Structure

The problem realized by this situation is that another way of measuring the power consumption and frequency of a ring oscillator was necessary while still enabling the use of the probe card for more efficient measurements. To solve this problem for the X2Chip, two options were explored. The first was to add enough inverters to the chain to slow the oscillation down to a frequency within the frequency response range of the probe card. However, this option was quickly discounted as it required hundreds of thousands of inverters to obtain a low enough frequency. The second option was to create a core ring oscillator and use flip-flops to perform frequency division. Though some accuracy could be lost by this method due to intrinsic behaviors of the flip-flop, this approach provides enough accuracy when considering it greatly improves the testing procedures since all ring oscillators on a pad set can be tested by landing the probe card only once on the chip.

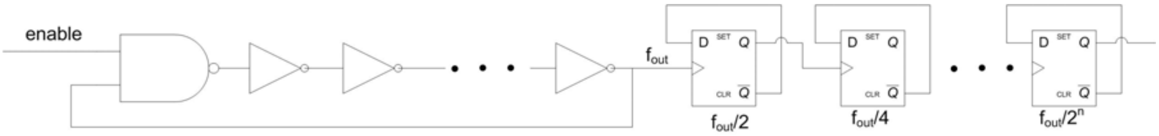


Figure 5.2.2 - X2Chip Ring Oscillator Structure

Section 5.3 Layout Considerations of the Rapid DUT Cell

The last lesson learned discussed in this work involves the development of the DUT cell used in the rapid characterization test structures. In the XChip this cell was laid out in a manner similar to a generic standard cell. The majority of the pins were located on the lowest metal layer with the expectation that all connections would be made after the cell was instantiated. The following figure shows the layout of the DUT cell for the XChip.

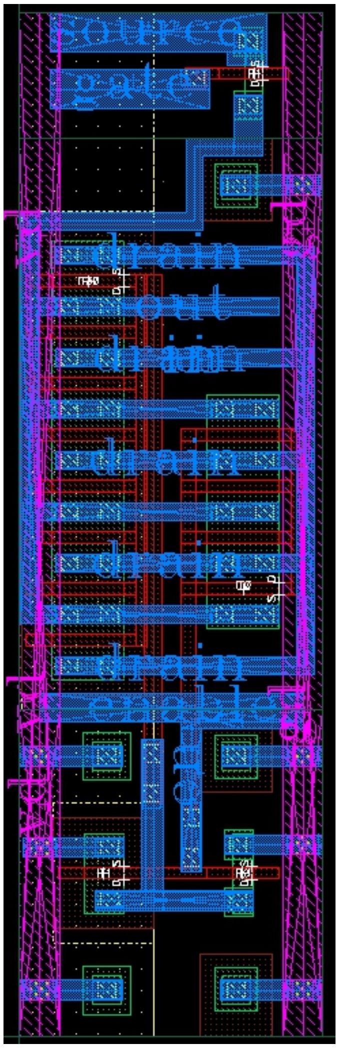


Figure 5.3.1 - DUT Cell from XChip

Unfortunately, this design requires significant effort to wire up as individual vias must be placed at each pin. To improve upon this, the DUT cell was completely redesigned to take advantage of connectivity made through abutment. As seen in the next figure, by creating pins and metal shapes that extended to the edges of the DUT cell, tiling the cells in an array automatically creates all the connections necessary without a single metal wire. This improved approach of the DUT cell allowed an order of magnitude more DUT cells to be instantiated in the X2Chip (1024) in comparison to the XChip (100).

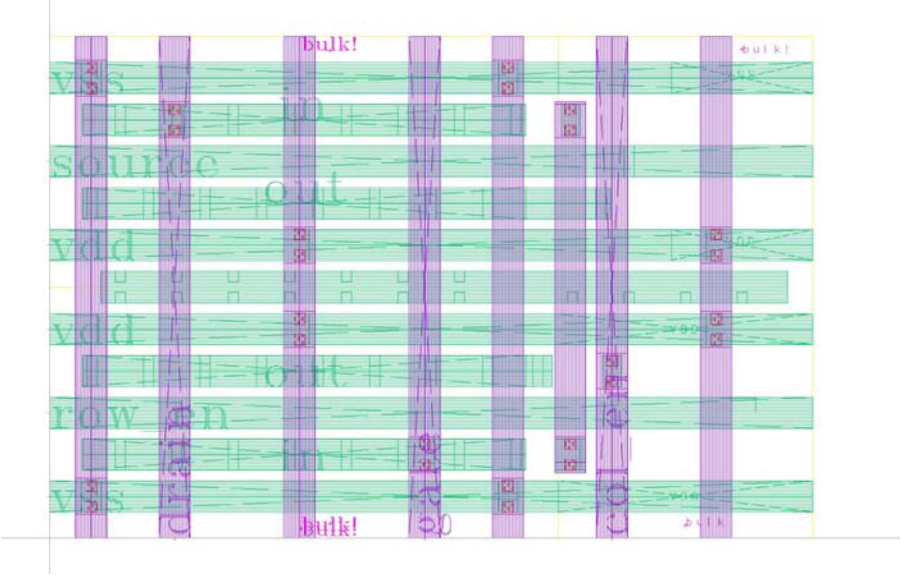


Figure 5.3.2 – Wiring from DUT Cell of X2Chip

Though tiling does solve the problem of wiring the array, LVS will still not pass if pin shapes are not correct in the layout. Because these arrays grew hierarchically the number of pins at the top level approached 256, resulting in another problem. The mere time required to properly place full rectangular pins across the metal in order to successfully pass LVS at all levels became an increasing problem. To solve this problem, basic skill code was developed to automatically recognize and create these pins based on the lower level subcell. Attached in the appendix is the code used to assist the layout of the X2Chip rapid characterization structures. This code, written in less than an hour, proved to save hours of time since the only time spent at each level of hierarchy of the DUT array was directed toward the instantiation of the lower level cell, the function call, and then appropriate DRC and LVS checks.

Chapter 6 Results

As with any design process, the verification of the final product and the comparisons between the simulated results and the actual measured results are of the utmost importance. In the following sections, both simulations of each design and measured results of the XChip are presented.

Section 6.1 Simulations

Section 6.1.1 Ring Oscillator

The ring oscillator designed for the XChip operated at a frequency of ~88 MHz with an average current draw of 34.38uA at 1.0V (nominal for the technology). The following figure shows the waveforms of the output of the ring oscillator at Vdd ranging from 0.5 (bottom) to 1.0V (top). As expected the operating frequency of the ring oscillator reduces as the supply voltage reduces.

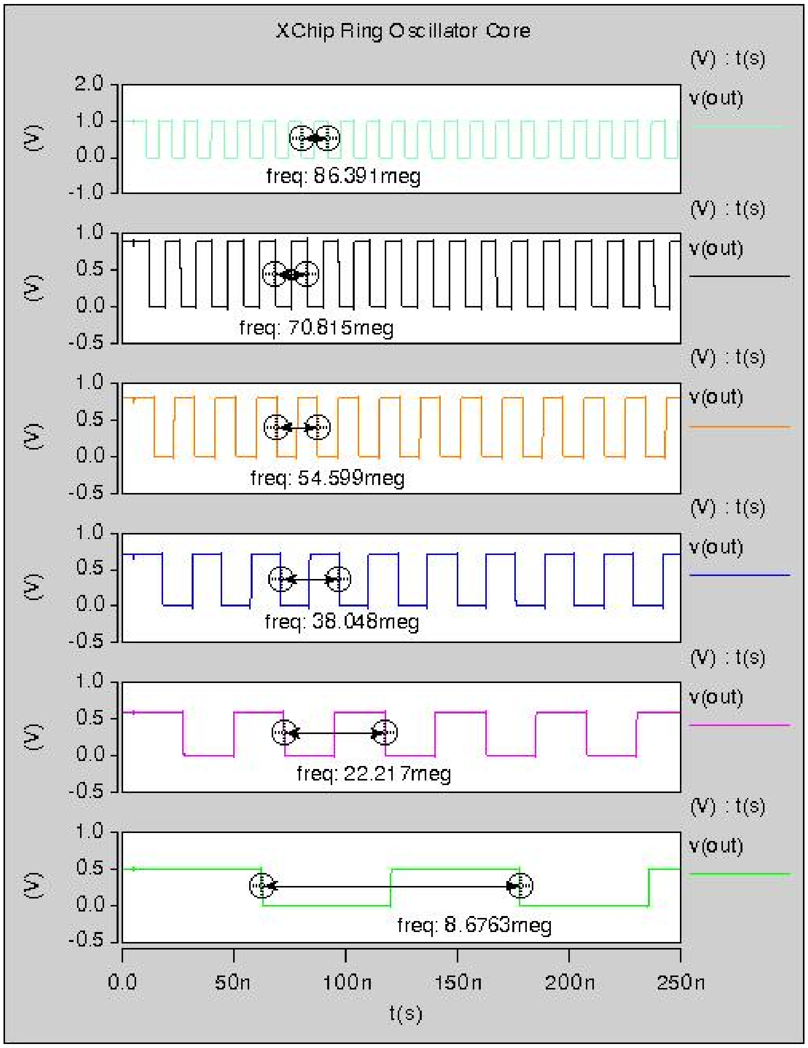


Figure 6.1.1 - XChip Ring Oscillator Simulations – Vdd from 0.5 to 1.0V

The core ring oscillator for the X2Chip performs at a frequency of ~3 GHz with a current draw of 226uA at 1.0V (nominal for the technology) as seen in the next figure. A verification waveform of the ring oscillator operating with frequency dividers is left out of this portion of the work. However, for completeness, it has been attached as an appendix.

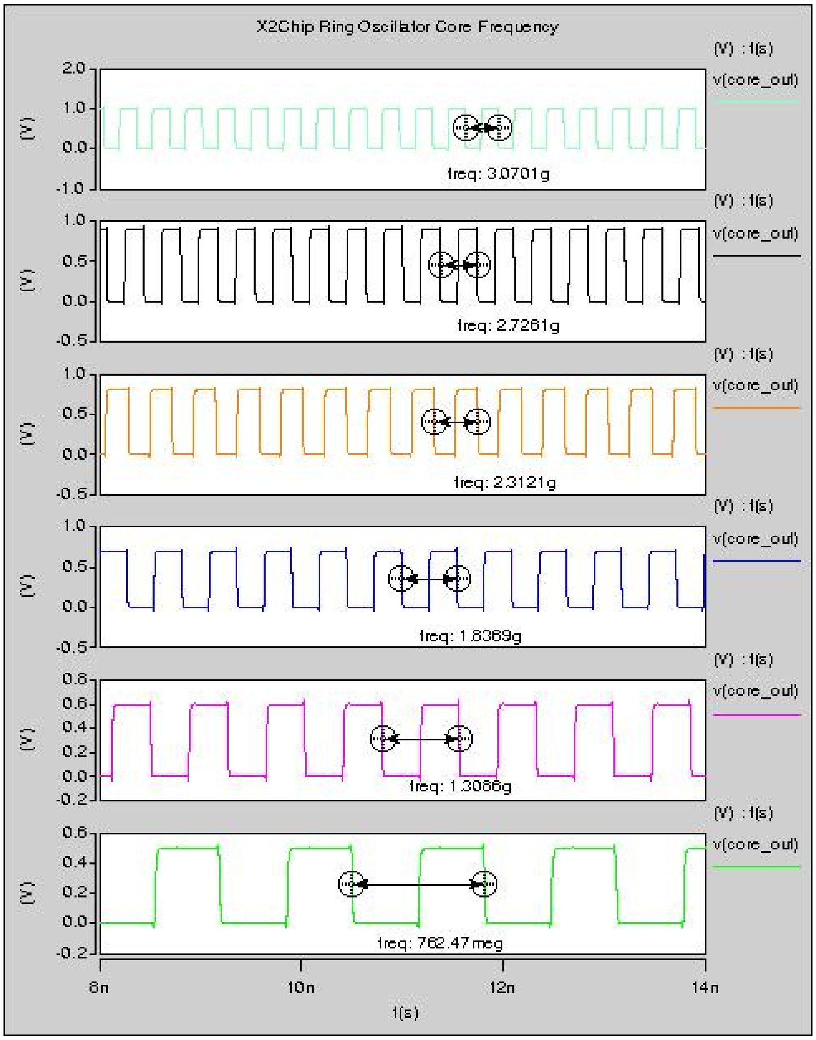


Figure 6.1.2 - X2Chip Ring Oscillator Core Simulations – Vdd from 0.5 to 1.0V

Section 6.1.2 SRAM

Since SRAMs are so extraordinarily complex, simulating these with spice is not a practical method of validation. Because of this, VHDL simulations are used to validate the design. Since no external modifications were performed on the SRAMs for the XChip, their simulations are omitted from this work. However, validation is shown for the SRAM implemented for the X2Chip since the data in was tied together and the data out was muxed. The test bench of this simulation can be found in the appendix.

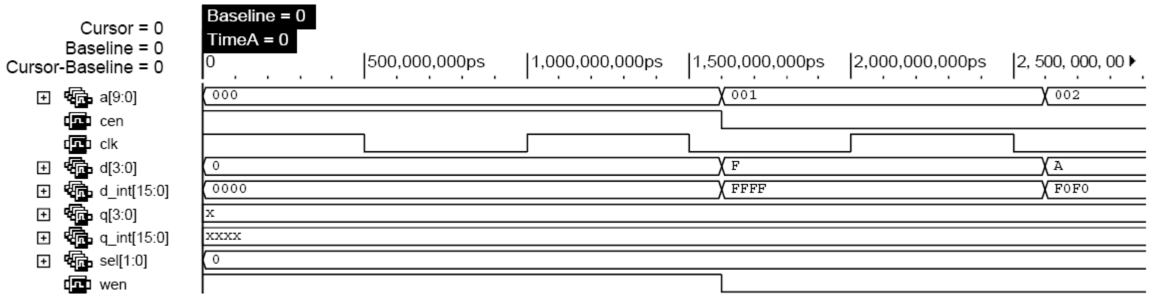


Figure 6.1.3 - X2Chip Muxed-SRAM Validation Simulation - Write

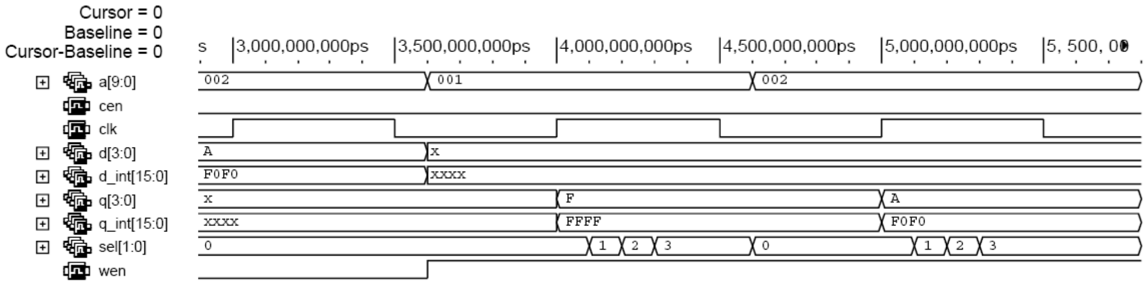


Figure 6.1.4 - X2Chip Muxed-SRAM Validation Simulation – Read

The above waveforms show the write and read operations of the SRAM. As seen, a value of FFFF is written to address 1 and a value of F0F0 is written to address 2. If the function of the muxing and data-in redrives behave as intended, the output should read all 1's, or F since it is four bits, followed by 1010, or A in hexadecimal. As seen above, this is the case. As the select signal (sel[1:0]) selects each sub-word the output value does not change as expect based on the muxing implementation.

Section 6.1.3 Threshold Voltage Variation Test Structure

The rapid characterization test structure has been rigorously tested in [4] and thus the proof of the overall circuit functionality is omitted from this work. However, with the modifications to reduce leakage, confirmation of a high degree of correlation between expected and simulated values is necessary. Below is a plot showing the how variations in Vt track against the perceived Vt as measured through the variation structure for the the XChip and X2Chip. In order to perform this simulated variation, a threshold adder parameter in the implemented technology kits was utilized to specify a particular threshold value.

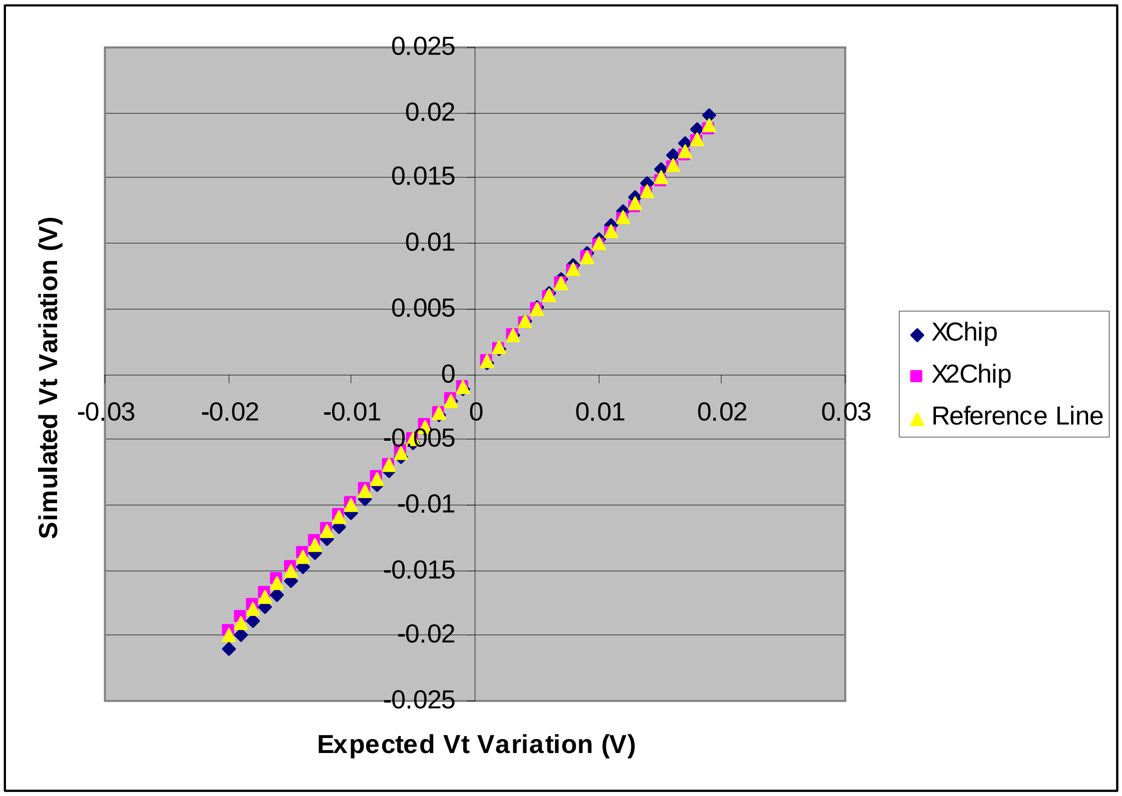


Figure 6.1.5 – Simulated Accuracy of Leakage-Reduced Vt Variation Structure

The following table shows a brief statistical analysis of the error observed in the improved version of the rapid Vt variation structure. As can be seen by the table, the X2Chip implementation was a substantial improvement over the initial implementation.

	XChip	X2Chip
average	4.598%	1.524%
max	8.550%	1.850%
min	0.333%	1.250%
std dev	1.590%	0.192%

Table 6-1 – Simulated Error Margin of Vt Test Structures

Section 6.1.4 On current Variation Test Structure

The on state current variation structure, though based on the threshold variation structure, does require some proof of concept. Since there is no single parameter in the device model to expressly modify the on state current of a device, one particular parameter that does have a direct impact on Ion must be selected and the resulting values must correlate to the observed fluctuation. Because Vt has a direct impact on Ion, we can use this parameter to prove that the Ion structure will accurately reflect shifts in on state current. In essence, this is the exact opposite as the approach of the rapid Vt characterization structure. By instead maintaining a fixed VDS as well as VGS, any shift in Vt will modify the current as described in the common drain current equation. The following waveform shows how variations in Vt cause a predictable shift in Ion. From left to right, the variations seen in the plot are as follows: -5mV, +5mV, -10mV, +10mV, -15mV, +15mV, -20mV, and +20mV.

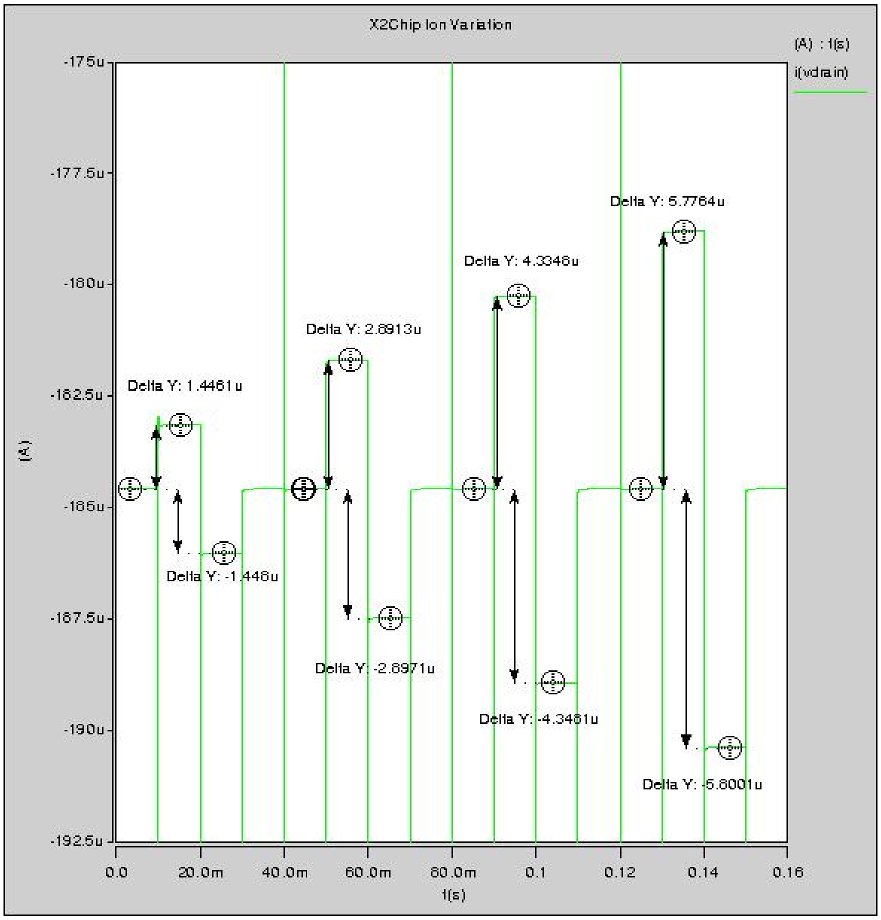


Figure 6.1.6 - Rapid Ion Variation Simulation

Section 6.1.5 SRAM Bit-cell

As discussed earlier, the bit-cell for the XChip was not an industrial grade layout and thus the values of the simulation cannot be correlated well to values of a bit-cell used in commercial SRAM designs. Moreover, showing both bitcell simulations is for the most part redundant; therefore the waveforms simulated from the commercial bitcell used in the X2Chip are presented. Refer to section 2 for a description of the process used to measure the margins discussed in this section.

The first metric shown is the read noise margin. Interestingly, even though the nominal operating voltage of both technologies is 1.0V, the read noise margin does not exhibit the same dramatic increase in current through the bit line nodes as is seen when the cell operates at 0.9V.

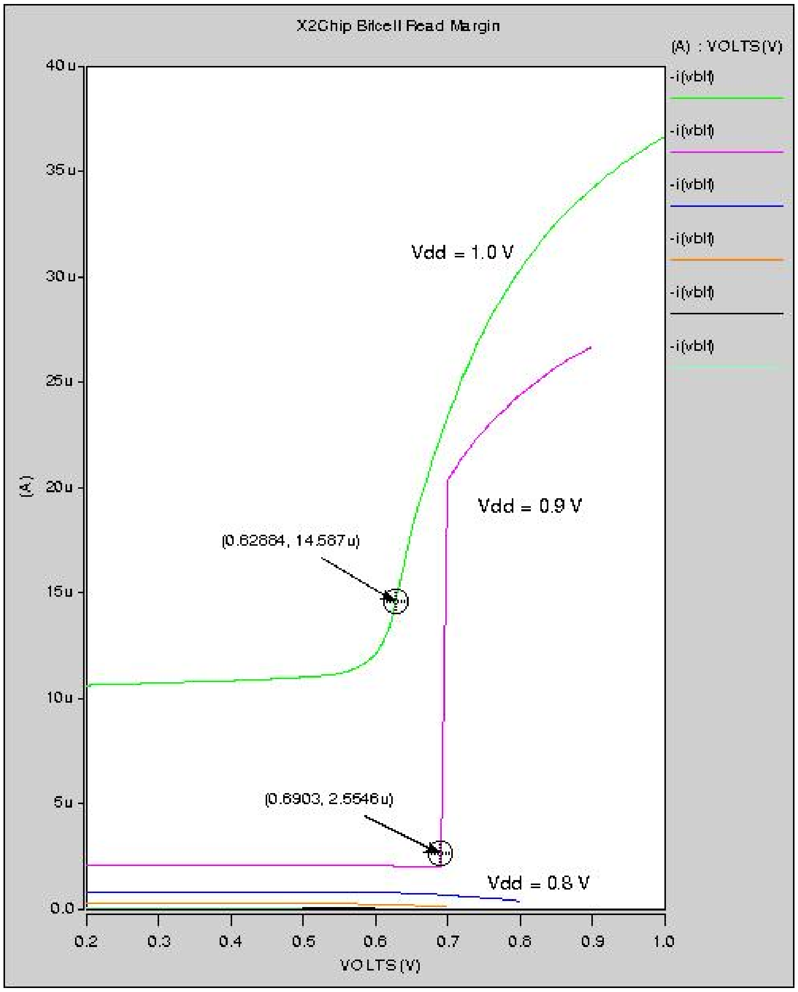


Figure 6.1.7 - Read Noise Margin Simulation

As of the writing of this work, the write noise margin was unable to be successfully simulated and measured.

Section 6.2 Measured

As of the writing of this paper, the XChip is the only available chip to perform verification on. The X2Chip will not be available for some time due to standard fabrication turn around time. The following sections present the portions of the XChip that have been measured. All values presented in this section are viable data points to compare the characteristics of one technology to another.

Section 6.2.1 Antenna Affect Diodes

An I-V characteristic curve of each antenna diode in the design is shown below.

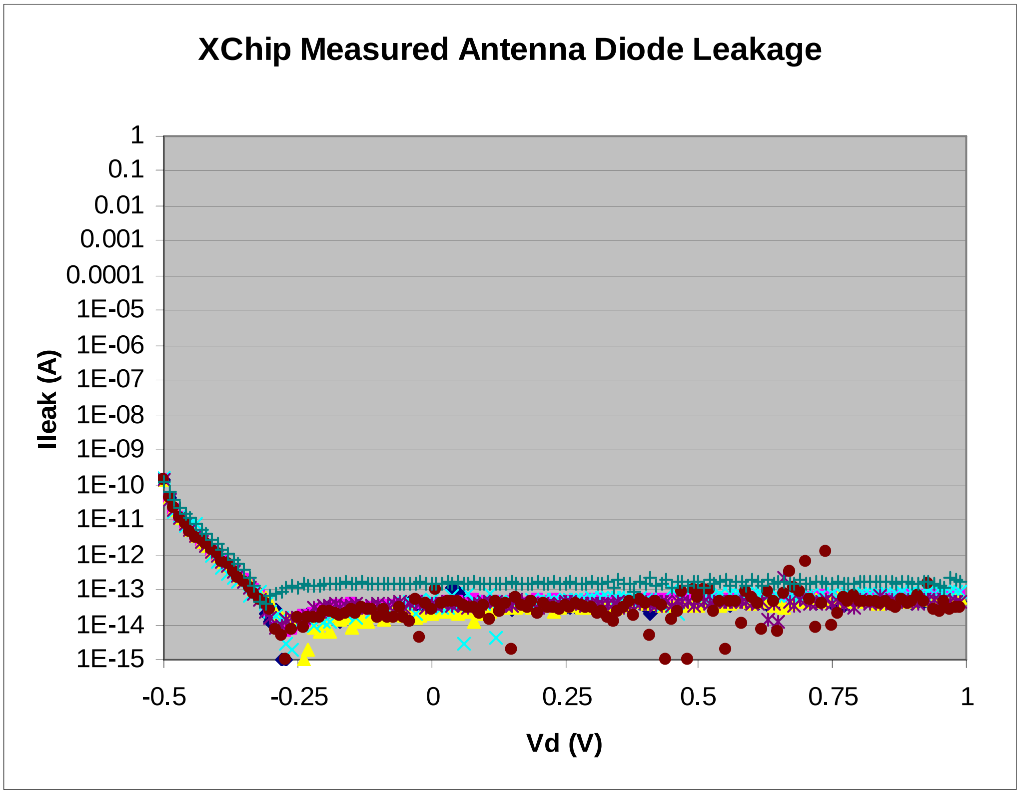


Figure 6.2.1 - Measured Antenna Diode Leakage

The above figure shows that under normal operating voltages, the leakage through any given antenna diode is in the range of 10's of pA or less. These measurements prove that the leakage through any given antenna diode in the XChips has a negligible impact on the accuracy of the measurements since the test structures themselves generally consume current in the uA range.

Section 6.2.2 Individual Transistors

Since individual transistor characterization was the primary focus of Phase I of the STEEP program, providing analysis of the performance of individual transistors in the metrics defined by the program was of the highest priority. In order to analyze the basic characteristics of Ion, Ioff, and subthreshold slope, a parameter analyzer designed to perform these sorts of tests was utilized. The following plots in this section show the overall measured trends of the 90 nm bulk technology in which the XChip was fabricated. For the purposes of this work, only one die was measured. Each length data point is defined as the average of the 10 devices for each transistor dimension on the tested die.

The first two characteristics of the transistors analyzed are the Ion and Ioff currents. The values are plotted as a function of the length of the gate at three separate supply voltages. For the purposes of the metrics established by the STEEP program, the current has been normalized to a 1um wide device, as denoted by the units of uA/um.

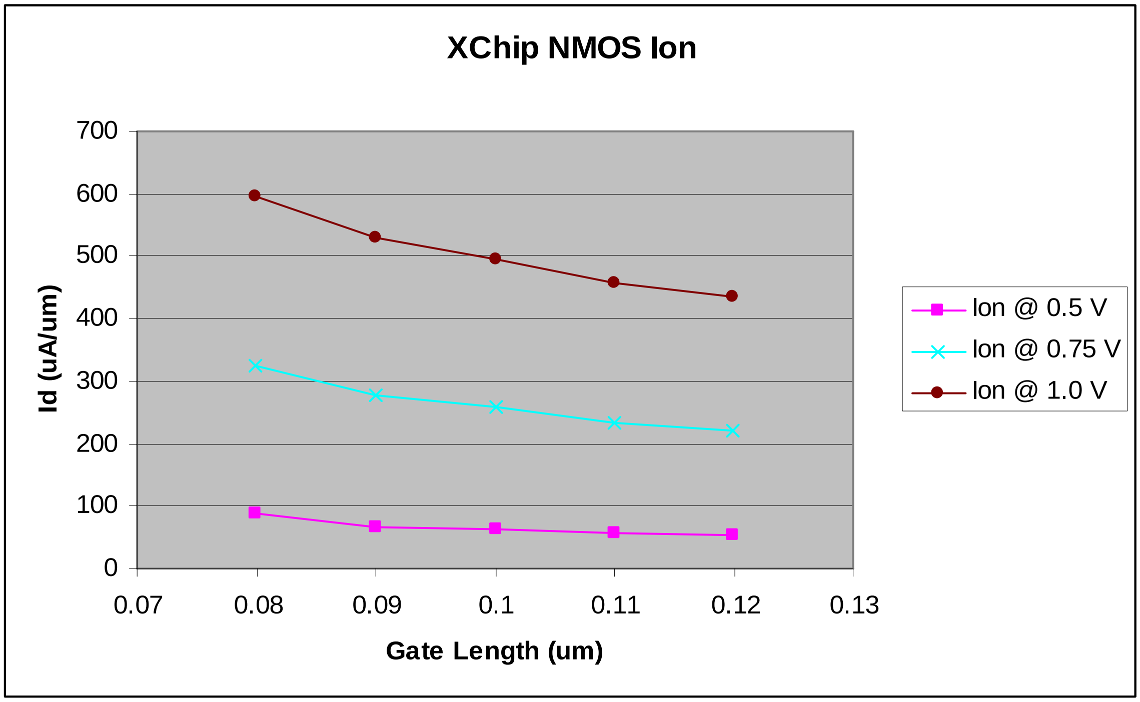


Figure 6.2.2 – Measured XChip NOMS Ion as a function of Gate Length

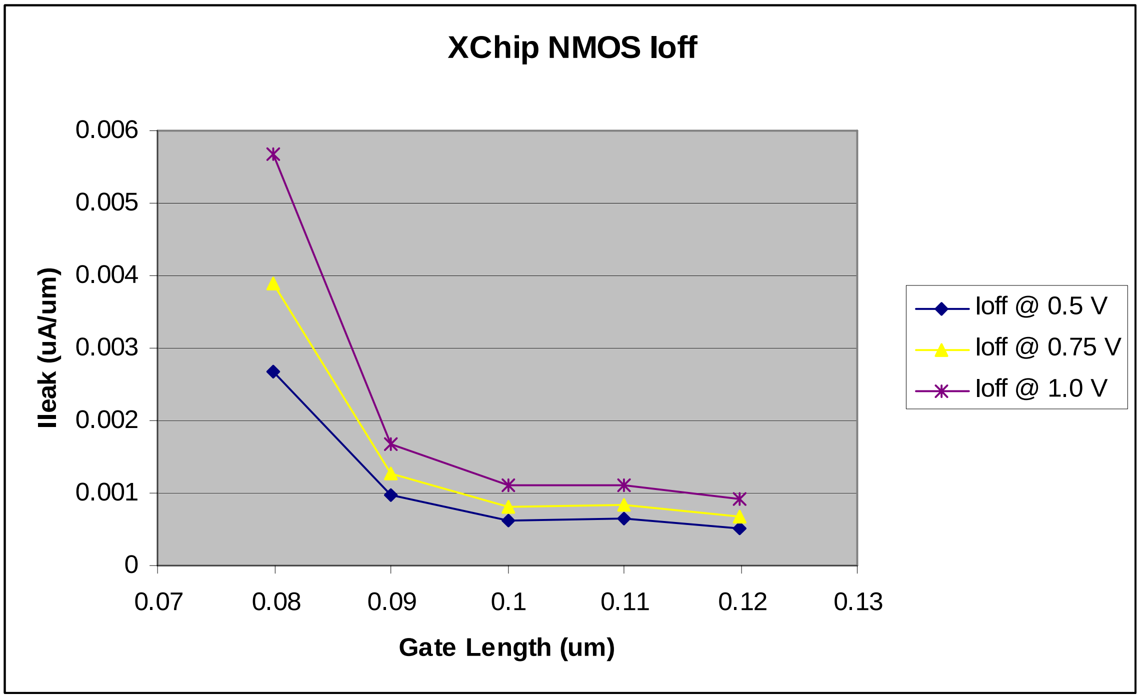


Figure 6.2.3 – Measured XChip NMOS Ioff as a function of Gate Length

Intuitively, both of these plots coincide with the general principals of the operation of transistors. As length increases, the ratio of W/L in the drain current equations becomes smaller and since drain current is directly related to this ratio, the reduction in both on and off state current makes sense. Furthermore, as illustrated in the next figure, the ratio of the Ion to Ioff currents linearly increases with the increase of length.

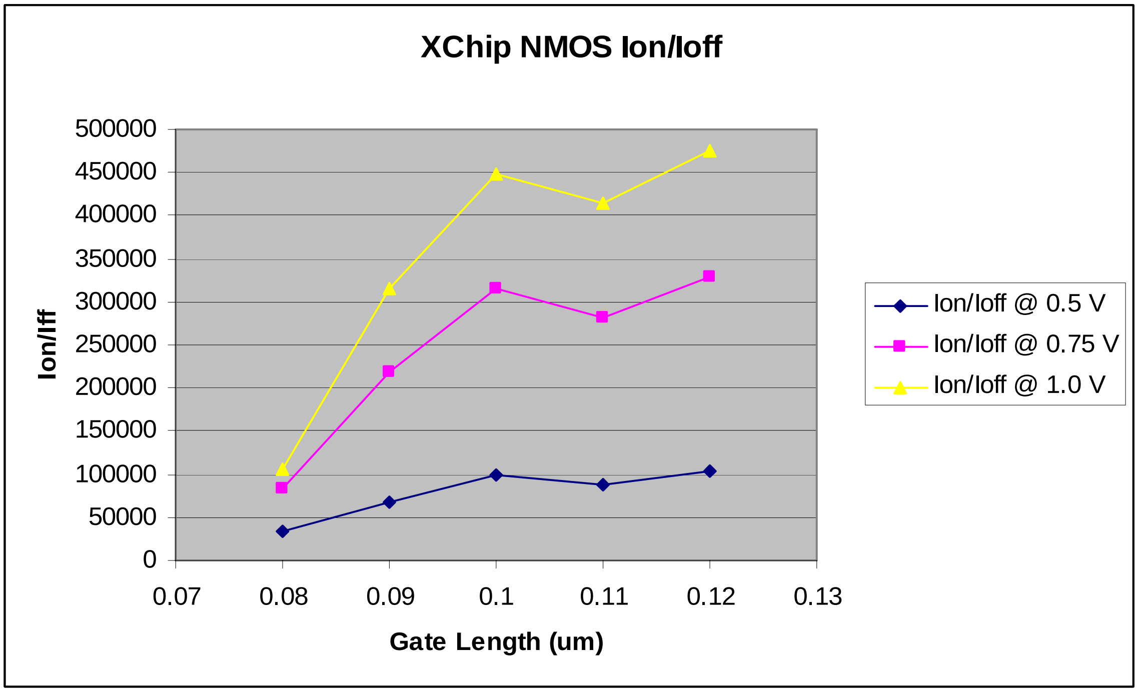


Figure 6.2.4 – Measured XChip NMOS Ion/Ioff as a function of Gate Length

One interesting observed anomaly in the above plot is that on average, all gates with a length of 110nm suffered from an increase of off state current thus resulting in the above figure showing a drop in the Ion/Ioff ratio for that particular transistor dimension. The same information, but presented as a function of supply voltage shows that as supply voltage increases, this Ion/Ioff ratio becomes more favorable.

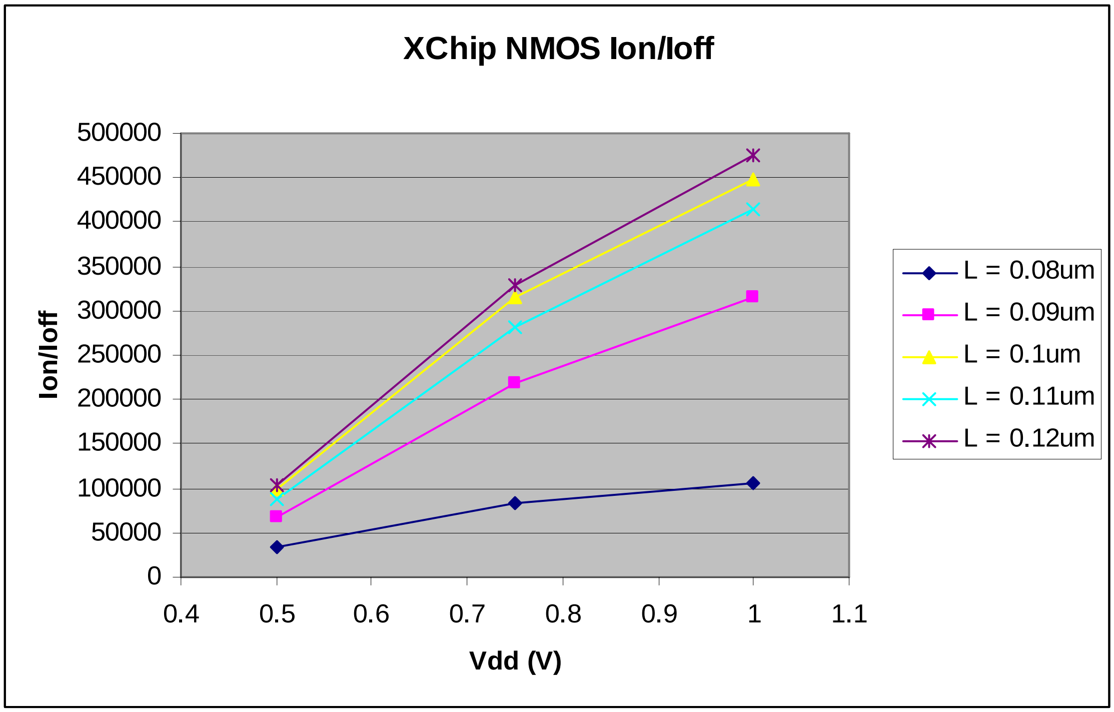


Figure 6.2.5 – Measured XChip NMOS Ion/Ioff as a function of Supply Voltage

Aside from the current draw of the transistors, another characteristic necessary to analyze for the first phase of the STEEP program is the subthreshold slope. This is calculated by finding the maximum slope of the typical I-V curve and taking its inverse. In this case, the smaller this value is, the faster the transistor switches since the value indicates the amount of voltage change necessary to increase the current through the transistor by one decade. The subthreshold value is plotted both as a function of gate length and supply voltage in the following figures. To generate the values for the subthreshold plots, all 10 devices were measured then averaged in order to reduce noise in the waveform that would introduce error into the plots.

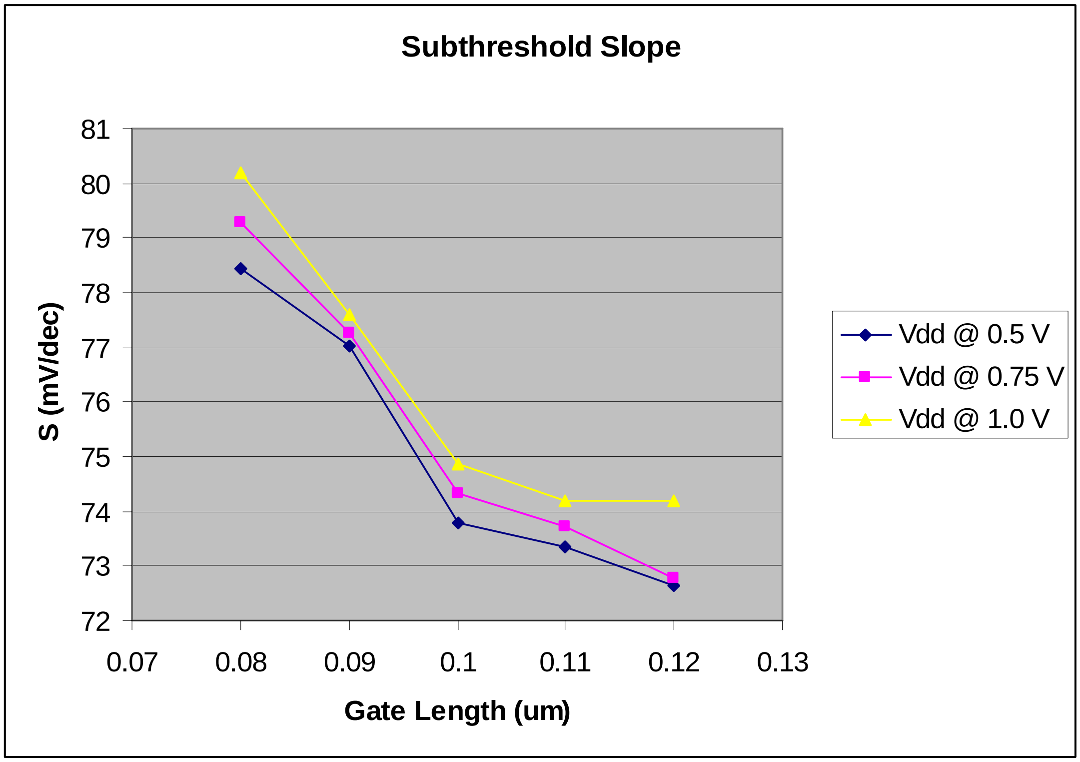


Figure 6.2.6 – Measured XChip NMOS Subthreshold Slope as a function of Gate Length

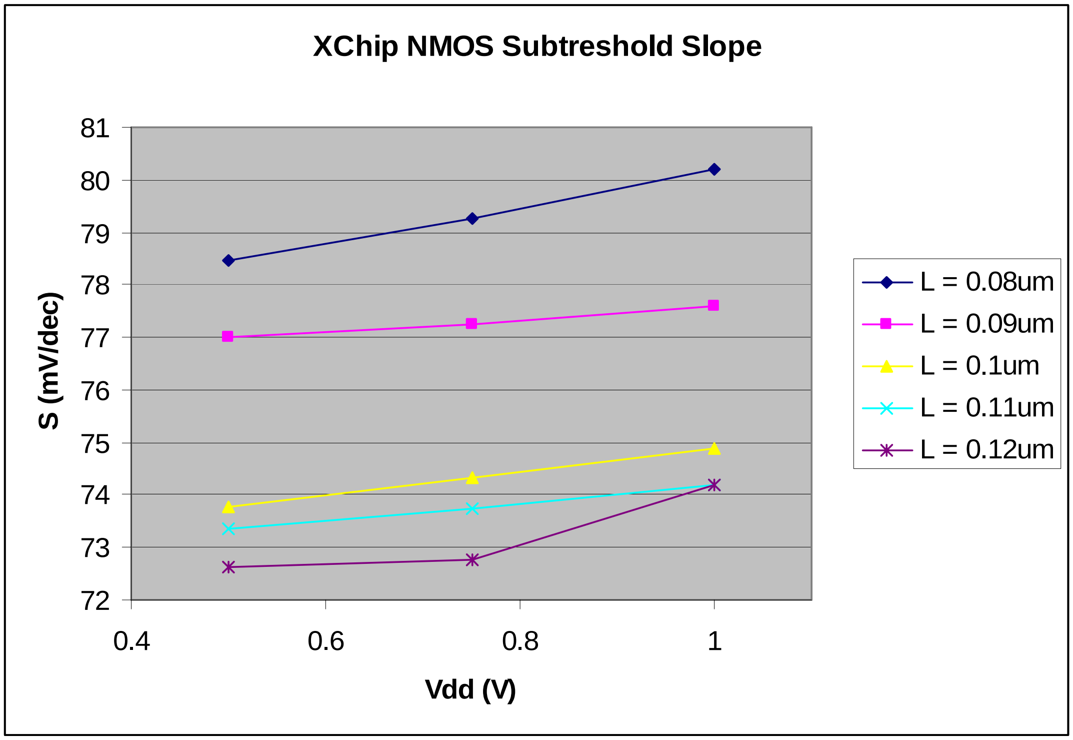


Figure 6.2.7 – Measured XChip NMOS Subthreshold Slope as a function of Supply voltage

Again, both of these figures illustrate the general trends observed with standard MOSFET devices. This plots show that regardless of the supply voltage chosen for a particular device, the subthreshold slope is closely dependent on the length of the transistor under test.

Section 6.2.3 Ring Oscillator

Screen captures of a ring oscillator working on the XChip are shown to exhibit discrepancies between simulated and measured values of the operating frequency. This discrepancy can be explained by two factors. The first is that the design kit used for the

XChip does not always perform proper callback routines to calculate attributes of a transistor such as area and perimeter of source and drain regions unless certain transistor configurations are preselected. The second attributing factor is the standard inaccuracy of schematic based simulations. However, the measurements backup the trend of reduced operating frequency as the supply voltage is reduced. At a nominal supply voltage of 1.0V, the ring oscillator is shown to operate at approximately 30.5 MHz with a current consumption of approximately 27uA.

With 420 inverters and 1 NAND, the average power consumption of each inverting stage of the ring oscillator at nominal supply voltage is approximately 65nW. Based on the dimensions of the transistors and an observed nearly 50% duty cycle, it stands to reason that a beta ratio of approximately 1.7 in this technology produces an inverter equally capable of driving either direction.

For brevity, not all waveforms at different supply voltages are shown instead, this section only contains a few waveforms in order to prove functionality and illustrate the limits of operation. The first waveform shows the operation of the ring oscillator at nominal Vdd of 1.0V followed by screen captures with Vdd at 0.7, 0.5, and lastly 0.25V.

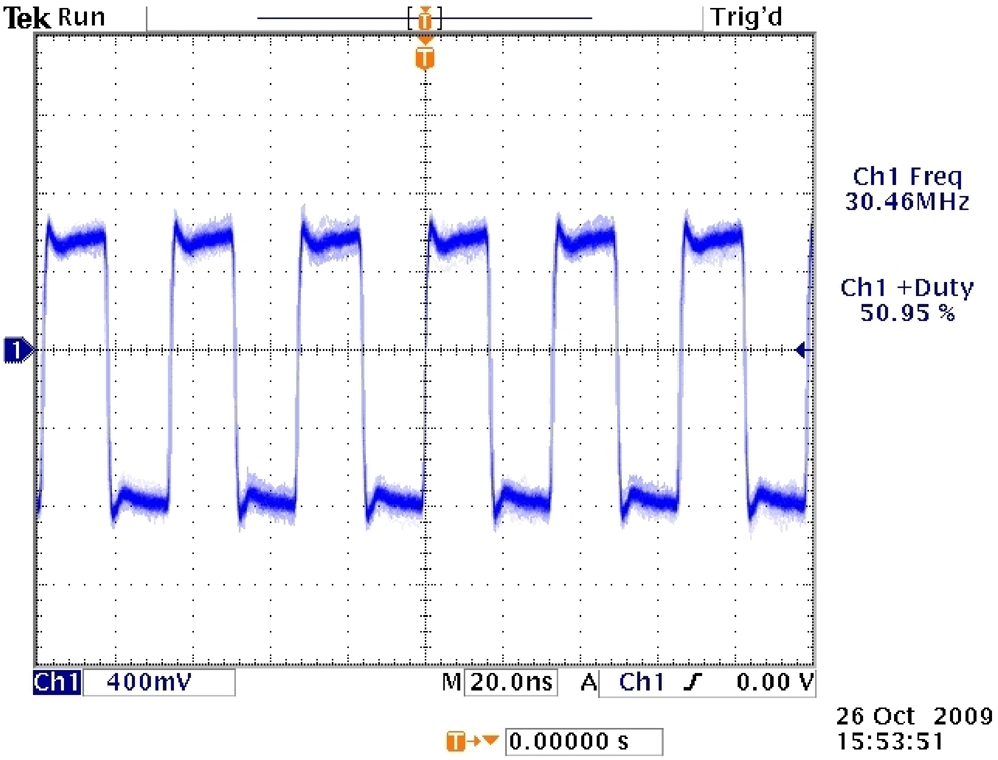


Figure 6.2.8 - Ring Oscillator at Vdd = 1.0 V

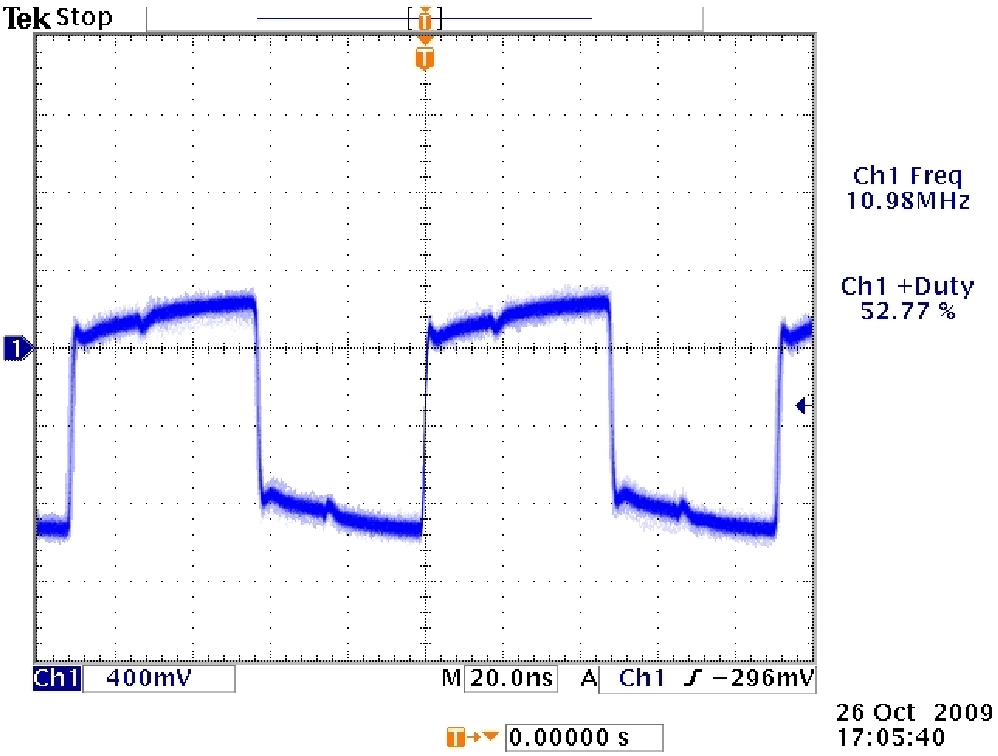


Figure 6.2.9 - Ring Oscillator at Vdd = 0.7 V

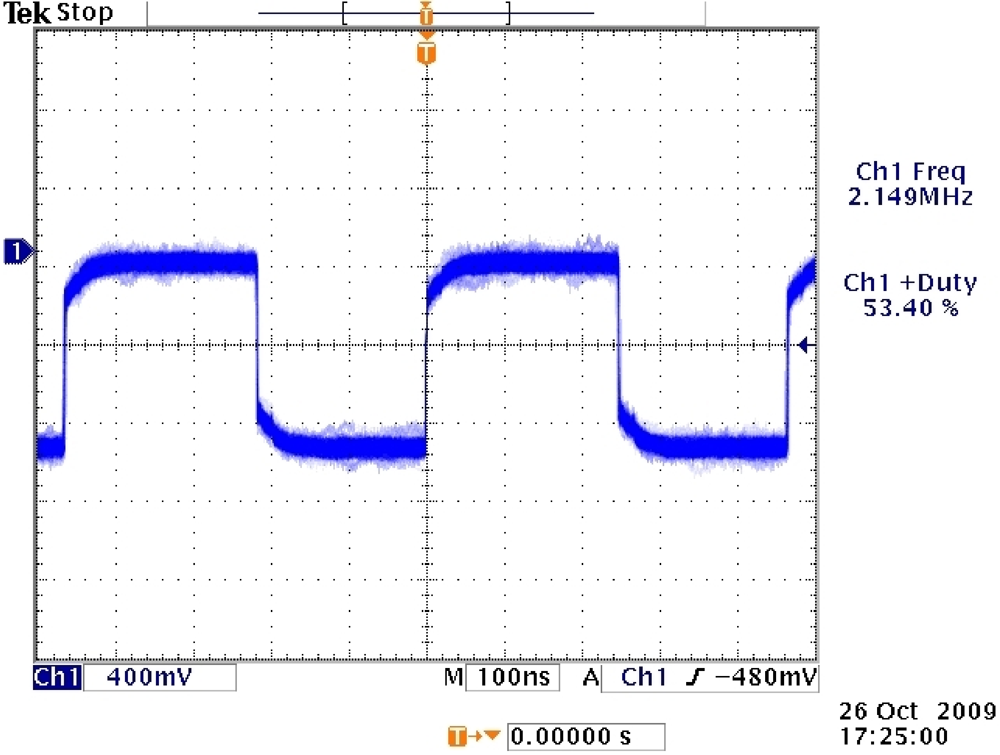


Figure 6.2.10 - Ring Oscillator Operating at Vdd = 0.5 V

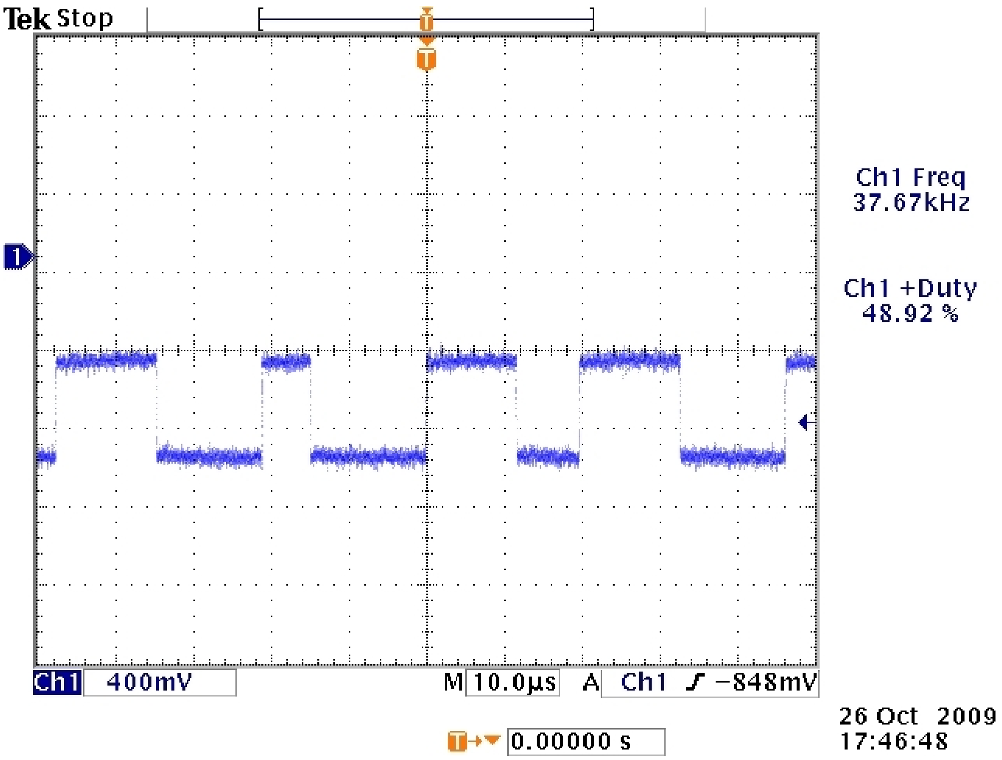


Figure 6.2.11 - Ring Oscillator at Vdd = 0.25 V

The progressive deterioration of performance is evident by the above waveforms. The ring oscillator does indeed operate as a ring oscillator to supply voltages as low as 0.3 V. However, as seen in the above waveform, once supply voltages of approximately 0.25 V are reached, the ring oscillator begins to act erratically and shortly thereafter no longer oscillates at all. The following plot shows the relationship between supply voltage and intrinsic delay in terms of total gate width.

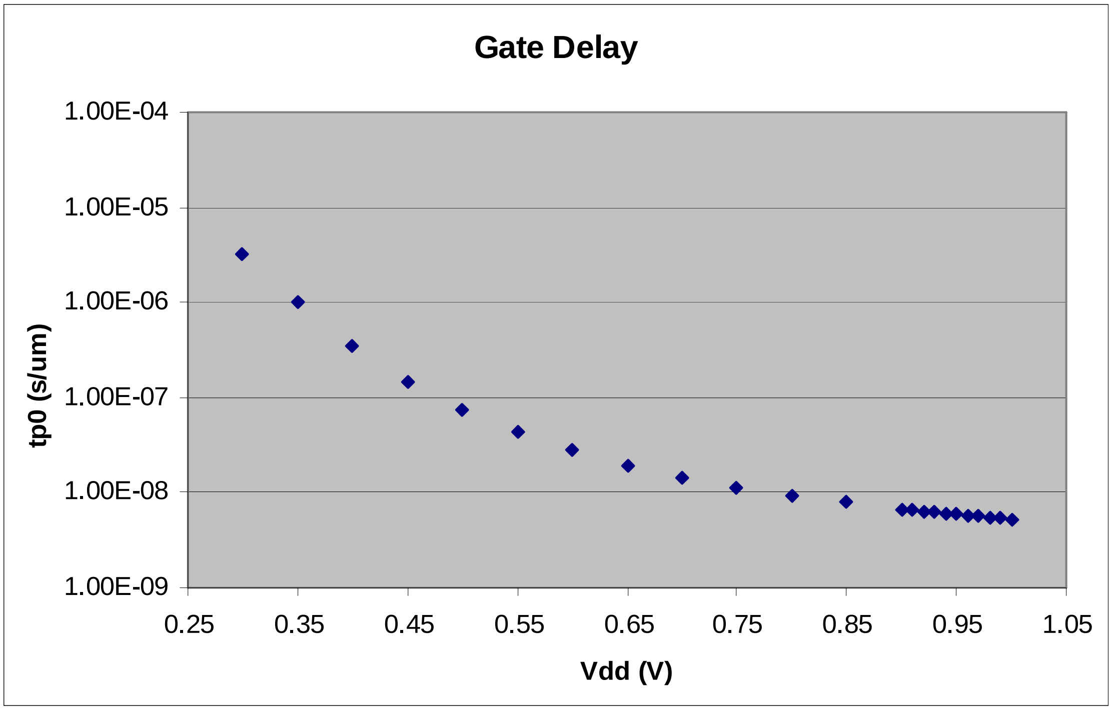


Figure 6.2.12 - Measured Gate Delay of XChip Ring Oscillator

The power consumption of the ring oscillators is also examined in the next figure. As expected the on state current consumption grows exponentially as Vdd increases. In order to better understand the overall performance of the ring oscillator the ratio of Ion/Ioff is plotted as well. Interestingly enough, the data actually shows that there is a slight reduction in this ratio at the nominal supply voltage of 1.0V. However the general trend of the line indicates an asymptotic approach to a ratio of approximately 100.

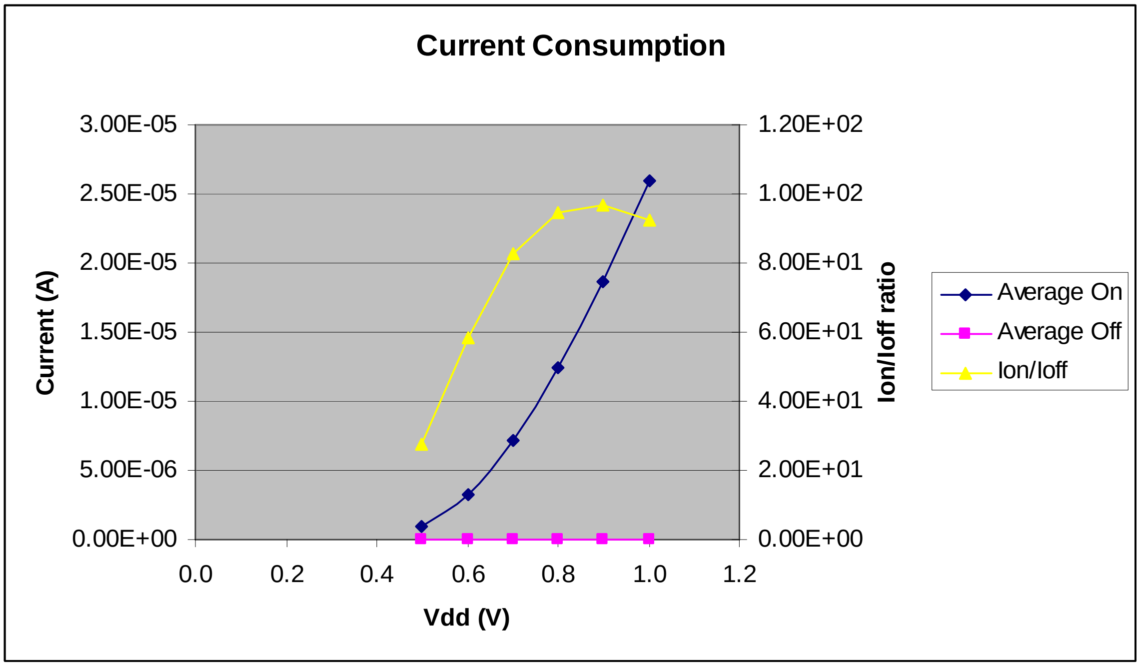


Figure 6.2.13 – Measured Current Consumption of Ring Oscillators

Section 6.2.4 SRAM Bit-cell

The same read margin as well as leakage measurements were performed on the bitcells of one of the XChip dies (refer to section 2 for the specific method for measuring the margins). The read margin, seen below, is a plot of the average read margin values for the two bitcells. The measurements indicate a read margin slightly more than 0.8 V. Because this bitcell is custom designed since an industry standard bitcell could not obtained in time for the tape-out, this read margin may not accurately reflect the state-of-the-art bitcells available from VLSI IP vendors. As a sanity check, this waveform seems reasonable compared to those seen in [6] especially considering the bitcells measured in [6] are industry standard 45nm bitcells. The strange behavior seen in the 0.9 and 1.0V data (sudden "dip" for the 0.9V and sudden rise then return for the 1.0V) could also be explained by the use of a custom bitcell. However, as of the time of this writing no data exists to explain what is happening at these points of interest.

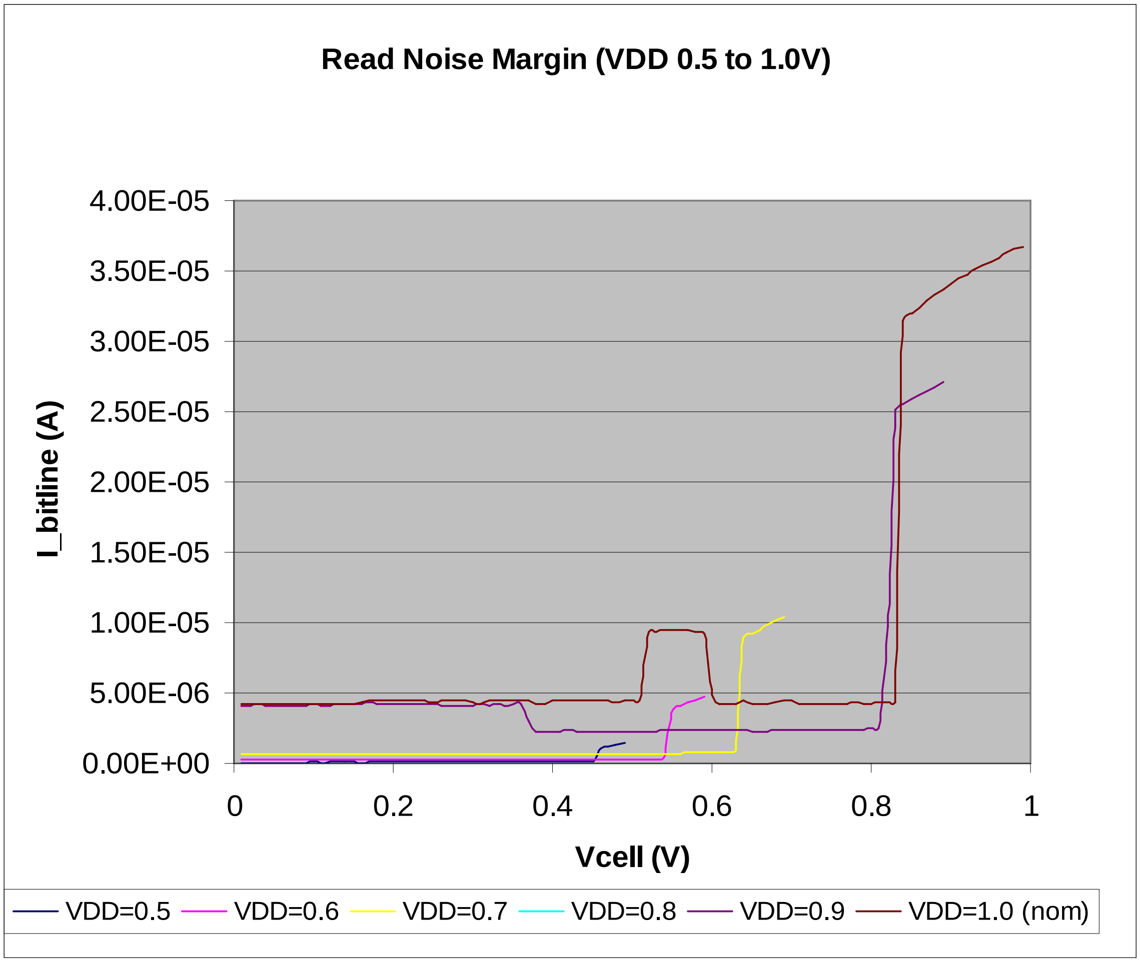


Figure 6.2.14 – Measured Read Margin of Custom Bitcell on XChip - Vdd from 0.5 V to 1.0 V

As mentioned earlier, the write noise margin was not able to be fully characterized at the time of this writing.

The final important metric identified in this SRAM bitcell is the leakage observed when the bitcell is in a hold state. As discussed earlier, the word line is unasserted and the internal VCELL is swept from 0 to 1.0 V. As observed in other aspects of the measurements, the positive correlation between leakage current and supply voltage hold true for the bitcell as well.

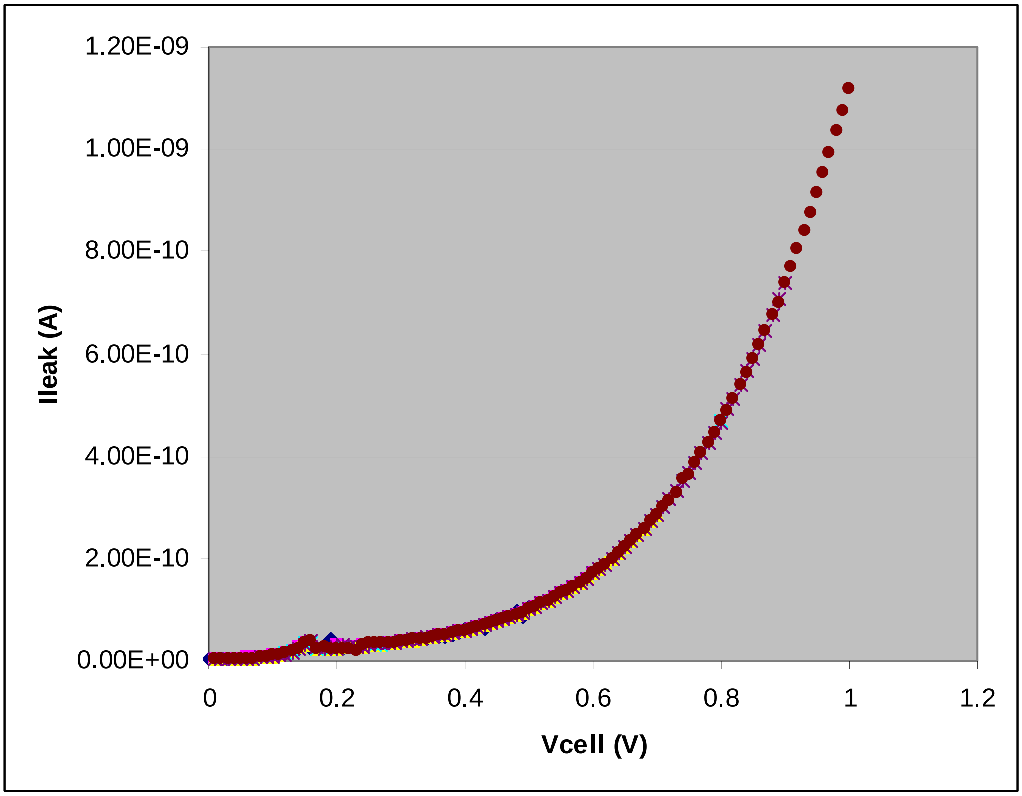


Figure 6.2.15 – Measured Leakage through Bitcell

Section 6.2.5 SRAM Yield

The following table shows the yield measurements from one die of an XChip. All 20 SRAMs on the test die were tested using the March C algorithm discussed earlier in this work. The number of bit failures for each element in the march test as well as the total number of failures can been seen in the following table. The highlighted portion of the table indicates SRAMs that were ignored for the purposes of the calculations in the table. These three SRAMs suffered some form of catastrophic failure in that their failure rate well exceeded normal values. These failures could be result of a failure within the SRAM itself, or could indicate a problem during the actual test and data gathering.

	March Element					
	1	2	3	4	5	
Padset	r0,w1	r1,w0	r0,w1	r1,w0	r0	Total
AA	35	0	10	0	0	45
AB	22	0	22	0	0	44
AC	0	0	10	0	0	10
AD	21	0	10	0	0	31
AE	32	0	0	0	0	32
AF	32	0	60	0	0	92
AG	20	0	10	0	0	30
AH	10	0	33	0	0	43
O	1766	29	179	0	0	
P	2576	4518	758	2708	75	
Q	2540	2747	2521	2557	38	
R	0	0	52	0	0	52
S	10	0	40	0	0	50
T	10	0	38	0	0	48
U	36	0	53	0	0	89
V	82	15	20	0	0	117
W	0	0	10	0	0	10
X	0	5	10	0	0	15
Y	40	0	60	0	1	101
Z	23	0	33	0	0	56
Total	373	20	471	0	1	865
Total%	0.429%	0.023%	0.541%	0.000%	0.001%	0.199%

Table 6-2 - Measured XChip SRAM Yield Results

As can be seen in the above table, when the SRAMs are generated with no redundancy or error correction, yield can be severely impacted. A failure of 865 bits out of 20k bits is not a trivial failure rate for any system requiring hardened circuitry for completely reliable performance.

Section 6.2.6 Rapid Threshold Variation (NMOS)

By far the most complex design for the XChips was the rapid characterization structures. However, the value-add of the complexity becomes apparent when analyzing the waveforms. With the current implementation, performing one sweep of any device array takes 1 second to complete (100 devices at 10ms per device). In that one second the average and standard deviation of the variation can be measured. The first waveform shown below is a screen capture of the oscilloscope used to measure the variation in threshold voltage. It can be observed that the maximum variation between any two devices is around 75mV.

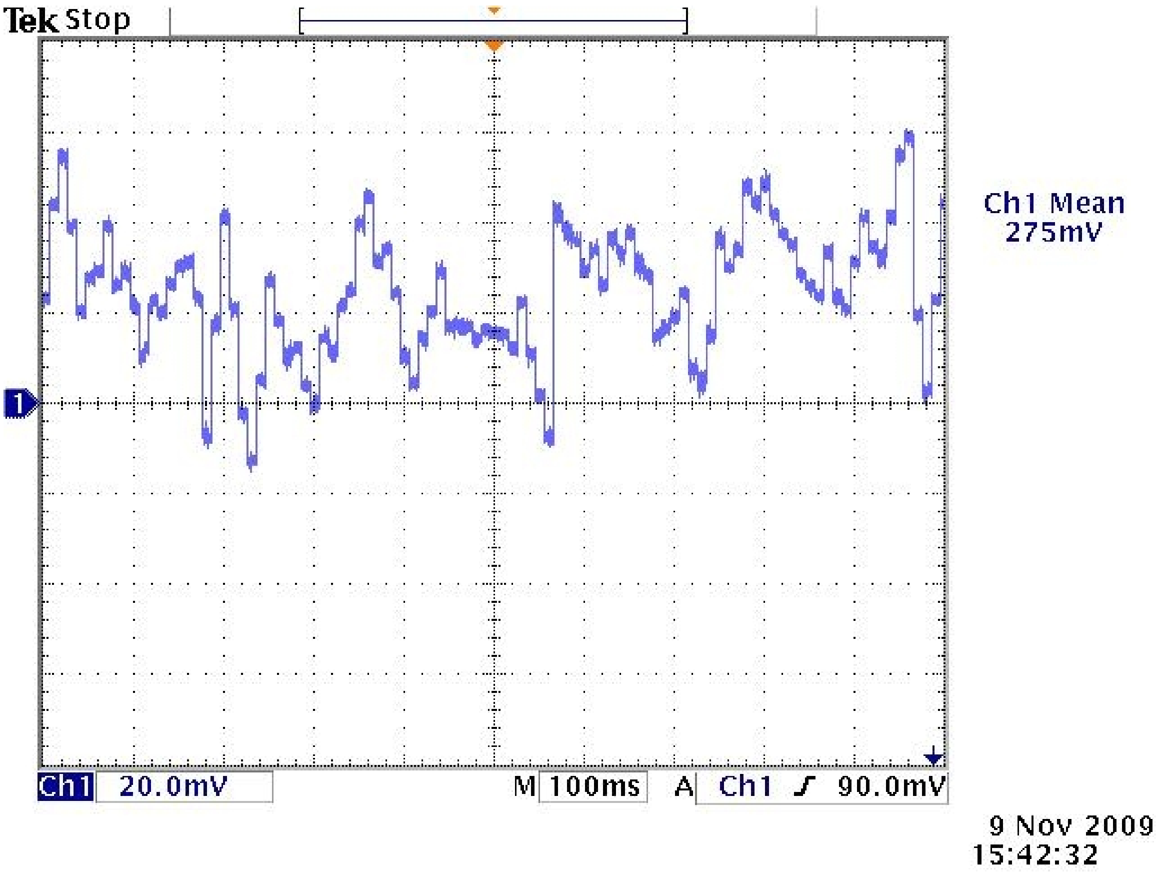


Figure 6.2.16 - Measured NMOS Treshold Variation (mean)

The following image shows the RMS value of the structure. As discussed in [5], the RMS value should prove to be equal to the standard deviation of the data sample. The image is shown in AC coupled mode in order to obtain an RMS value from the oscilloscope. In addition, 4 full cycles of the structure to shown in order to illustrate that the values measured at each device are consistent each cycle through the structure.

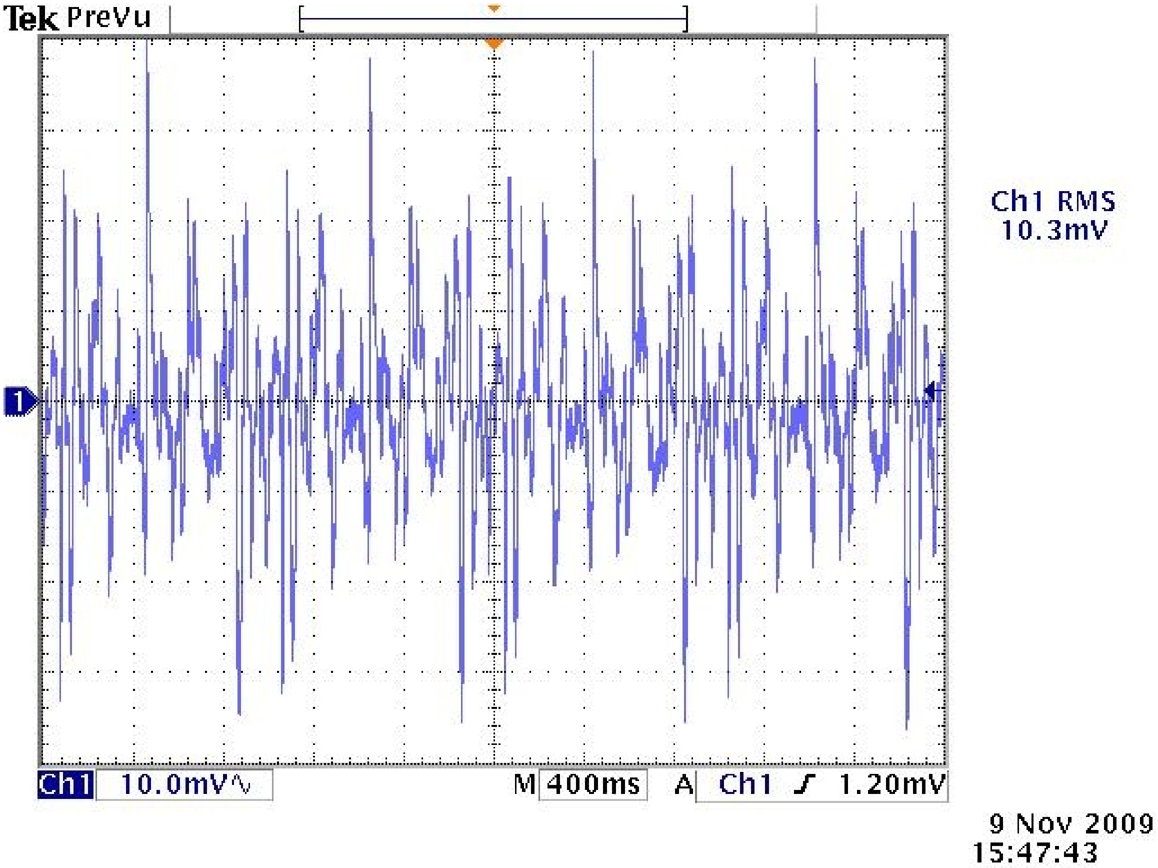


Figure 6.2.17 – Measured NMOS Threshold Variation (RMS)

Though the RMS value shown in the AC coupled waveform is in fact exactly the standard deviation of the AC coupled waveform, it is not the standard deviation of the DC coupled waveform. The oscilloscope returns an RMS value equal to the average (DC mean) when in DC coupled mode and thus the only way to return an RMS value that provides insight into the variation is by using the AC coupled mode, thus introducing the observed error. However, this error can be explained by the mechanism through which the AC coupled measurement is performed. As can be seen in the above image, all of the peaks of the AC coupled signal drift to 0V; this is intrinsic to the way AC coupling behaves. In AC coupling, essentially a capacitor is put in series with the signal to be measured and the DC value at any point in time tends to drift to 0V. Because of this, all steady values measured at each device actually drift to 0 and thus reduce the RMS value and produce the error in the measurements. This error is due to the limitation of the equipment used and not an error in the use of RMS value as the standard deviation. The true standard deviation as measured from the DC coupled waveform was calculated through a spreadsheet to be 14.076mV. The value observed in a 65nm SOI technology in [4] was 19.2mV. If the assumption is that variation worsens as device dimensions decrease, then the values observed on the XChip seem to fall in line with expectations.

Section 6.2.7 Rapid Ion Variation (NMOS)

In order to measure the variation in Ion, a constant voltage source must be supplied to the device array while the output current is measured for each device. In order to achieve this with the available resources, a resistor was placed in series between the power supply and the input pin to the structure. As individual devices are selected in the array, the variation in measured voltage across the resistor provides the necessary information to calculate the variation in Ion. The following image shows the waveform seen on the oscilloscope for this measurement.

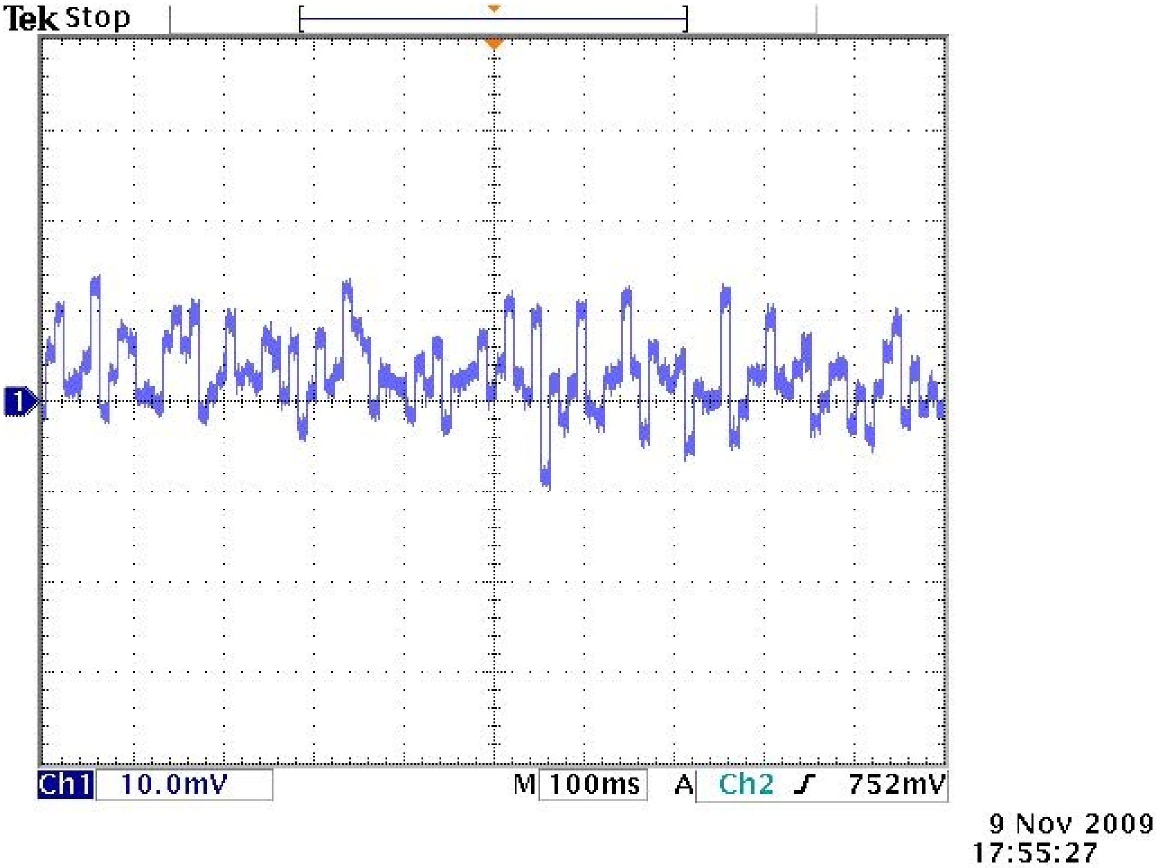


Figure 6.2.18 - Measured NMOS Ion Variation

Because the voltage displayed does not directly correlate to the current through the devices, the data from the above waveform was imported into a spreadsheet where computations of the average and standard deviation could be computed. It was calculated that the average Ion was 61.065uA with a standard deviation of 4.149uA. Similar to the threshold variation values, these values are well within a margin of error of the expected values. It should be noted that the same rapid measurement method could be utilized if current probes capable of measuring in the micro-Amp range were available. The only reason this method was not used for this work was equipment limitation.

Chapter 7 Conclusions and Future Work

The collection of test structures implemented on the XChip and X2Chip are shown in this work to provide adequate characterization of the performance of a technology node to perform comparisons among different technologies. However, as with any product of engineering, prospective improvements always exist; this final chapter discusses a few aspects of improvement that could be readily implemented into future version of such a testbed provided the time and resources are available to do so.

Section 7.1 PMOS Rapid Characterization Test Structures

Continued effort into understanding the behavior of the rapid characterization test structures for PMOS devices is necessary in order to properly analyze the variation of the PFETs in this technology. Simulations of the threshold and Ion test structures show correct functionality, but recreating the simulation in the lab results in waveforms that are unexpected. More investigation is needed to properly bias the circuit in such a manner to provide waveforms similar to those observed with the NMOS device structures.

Section 7.2 Skill Code

One portion of the project that could benefit from further work is the skill code that was used to create the pins of the tiled DUT cell array layout. This code doesn't take into consideration pins that are internally connect. For instance, the feed through path of the control logic flip-flops flows from one flip-flop to the next as well as out to the device array. In this situation a pin exists both at the point where the flip-flop drives out to the array as well as the internal connection between the flip flops. If left as is, this layout will fail LVS. For the use in the X2Chip, the clean up required to fix this problem was insignificant and merely a brief annoyance. More work could be used to help define a function that creates pins based on more specific conditions. Though out of the scope of this work, developing an entire library of skill functions used to automate certain portions of VLSI design could prove incredibly useful for a wide range of projects.

Section 7.3 March Test Pattern Conversion

Though the march test pattern generator exists and can provide stimulus patterns to be run against an SRAM, code needs to be developed to map this output to a format that is compatible with equipment available at NCSU. The particular equipment available is the HFS-9009 and significant work has already been put into a GUI front end to control the use of this device via a GPIB connection with a computer. The next step to fully characterizing SRAM yield would require a program, referred to as mtp2hfs, to be written that would take the output of mtpg and convert it into a stimulus file that can be loaded via the existing GUI into the HFS.

Section 7.4 Alternative Test Structures

There are two particular test structures of interest that, if implemented, could provide a larger sample of data to be analyzed for characterization purposes. The first is a ring oscillator structure discussed in [8] where an array of ring oscillators is created whereby each ring oscillator in the array can be individually enabled. If particular layout variations of devices wanted to be explored, creating a ring oscillator for each variant under exploration in the array would provide an excellent measure of exploring many variation impacts on frequency and power.

The second structure of interest is that found in [6] where a large array of SRAM bitcells could be measured without the overhead of extra sets of pads for each bitcell. This is achieved by creating large thick oxide pass-gates to select the array cells and measure the margins with minimal parasitic impact.

REFERENCES

[1] BAA 07-26: Steep-Subthreshold-Slope Transistors for Electronics with Extremely-Low-Power. http://www.darpa.mil/mto/solicitations/baa07-26/index.html

[2] W. G. Vandenberghe et al. "Analytical Model for a Tunnel Field-Effect Transistor", *MELECON*, 2008.

[3] D. Kim et al. "Low Power Circuit Design Based on Heterjunction Tunneling Transistors (HETTs)", *ISLPED*, 2009.

[4] K. Agarwal, S. Nassif, F. Liu, J. Hayes, K. Nowka, "Rapid Characterization of Threshold Voltage Fluctuation in MOS Devices", *IEEE Conference on Microelectronic Test Structures*, March 2007, pp. 74-77.

[5] J. Hayes, K. Agarwal, S. Nassif, "Rapid Characterization of Parametric Distributions Using a Multi-meter", *IEEE Conference on Microelectronic Test Structures*, 2008, pp. 17-20.

[6] Z. Guo et al. "Large-Scale Read/Write Margin Measurement in 45nm CMOS SRAM Arrays", *IEEE Symposium on VLSI Circuits Digest of Technical Papers*, 2008, pp. 42-43.

[7] A. J. van de Goor, "Using March Tests to Test SRAMs", *IEEE Design & Test of Computers*, March 1993, pp. 8-14.

[8] L.T. Pang, B. Nikolic, "Measurements and Analysis of Process Variability in 90 nm CMOS", *IEEE Journal of Solid-State Circuits*, May 2009, pp. 1655-1663.

APPENDICIES

Appendix A – mtpg.pl (March Test Pattern Generator)

```
#!/usr/local/bin/perl
#
# mptg.pl: March Test Pattern Generator
# author:  Philip M Iles
# email:   pmiles@ncsu.edu
#
# This script generates a march test pattern based
# on the symtax specified below
#
# "u(W0); ud(R0,W1); d(R1,W0); u(R0,W0); ud(R0)"
#
# This syntax describes the following scenario
#
# 1) write 0's to all locations in ascending address order
# 2) read a 0 and write a 1 at each location in any order
# 3) read a 1 and write a 0 in descending address order
# 4) read a 0 and write a 0 in ascending order
# 5) read a 0 in any order
#
# the script has the following parameters
#
# --wordsize or   -w: size in bits of each word; ie, width of data in
# --numberofwords -n: maximum address possible
#                     use: 0x to indicate hex; ie, 0xFFFF
#                     use: 0b to indicate binary; ie, 0b1111111111
#                     use: 0d to indicate decimal; ie, 0d1024
# --addrsize or   -s: size in bits of address input
#                     note: specify addrsize or maxaddr not both
# --binary   or   -b: output address and data in binary
# --hex      or   -h: output address and data in hex
# --decimal  or   -d: output address and data in decimal
#
#
# the output of the script will look like:
#
# W FA51 FFFF
#
# which indicates a write of 0xFFFF or all 1's is to be
# performed at address 0xFA51

package inst;
use strict;
use warnings;
use POSIX qw(ceil);
# constructor for an instruction
sub new {
       my $class = shift;
       my $self = {};
       $self->{OP}        = undef;
       $self->{ADDR}      = undef;
       $self->{DATA}      = undef;
       $self->{ADDRWIDTH} = undef;
       $self->{DATAWIDTH} = undef;
       bless($self, $class);
       return $self;
}

# getter/setter for the operation R/W
sub op {
       my $self = shift;
       if (@_) { $self->{OP} = shift };
       return $self->{OP};
```

```
}

# getter/setter for the address
sub addr {
	my $self = shift;
	if (@_) {
		$self->{ADDR} = shift;
		if($self->{ADDR} =~ m/"^0x"/){
			$self->{ADDR} = $self->{ADDR};
		} else {
			$self->{ADDR} = sprintf("%X",$self->{ADDR});
		}
	}
	return $self->{ADDR};
}

# getter/setter for the data
sub data {
	my $self = shift;
	if (@_) { $self->{DATA} = shift };
	return $self->{DATA};
}

# getter/setter for bit width
sub datawidth {
	my $self = shift;
	if (@_) { $self->{DATAWIDTH} = shift };
	return $self->{DATAWIDTH};
}

# getter/setter for bit width
sub addrwidth {
	my $self = shift;
	if (@_) { $self->{ADDRWIDTH} = shift };
	return $self->{ADDRWIDTH};
}

#
sub print {
	my $self = shift;
	my $base = shift;
	my $string = undef;
	my $datawidth = $self->datawidth();
	my $addrwidth = $self->addrwidth();
	my $formatString = undef;
	# start string with operation
	$string = $self->op();

	if(!$base){
		die "Error: inst->print(): output base not specified\n"
	} elsif($base eq "b"){
		$string .= " " . sprintf("%0${addrwidth}b", hex(  $self->addr() ));
		$string .= " " . sprintf("%0${datawidth}b", hex(  $self->data() ));
	} elsif ($base eq "h") {
		$addrwidth = ceil($addrwidth/4);
		$datawidth = ceil($datawidth/4);
		$string .= " " . sprintf("%0${addrwidth}X", hex(  $self->addr() ));
		$string .= " " . sprintf("%0${datawidth}X", hex(  $self->data() ));
	} else {
		$string .= " " . sprintf("%d", hex( $self->addr() ));
		$string .= " " . sprintf("%d", hex( $self->data() ));
	}
	$string .= "\n";
	printf($string);
	return 0;
```

```
}

package main;
use strict;
use warnings;
use Getopt::Long;

main();
sub main {
        my $wordsize      = 0;
        my $numberofwords = 0;
        my $format        = '';
        my $binary        = '';
        my $hex           = '';
        my $decimal       = '';
        my $algorithm     = '';
        my $element       = '';
        my $i             = 0;
        my $op            = '';
        my $addrwidth     = 0;

        # get all options
        GetOptions (
                'wordsize=i'  => \$wordsize,
                'numwords=s'  => \$numberofwords,
                'binary'      => \$binary,
                'hex'         => \$hex,
                'decimal'     => \$decimal,
                'algorithm=s' => \$algorithm
        );

        # error checking to make sure necessary combination of parameters is specified
        if(!$wordsize && !$numberofwords && !$format && !$algorithm){
                die "Usage mtpg.pl --wordsize 16 --numberofwords 1024 --binary --algorithm 
\"u(W0);d(R0,W1)\"\n";
        }
        # set format tag for printing
        if($binary){
                $format = "b";
        } elsif($hex) {
                $format = "h";
        } elsif($decimal) {
                $format = "d";
        } else {
                die "Error: output format not valid or unspecified\n";
        }
        if(!$wordsize){
                die "Error: wordsize (in bits) must be specified\n";
        }
        if(!$numberofwords){
                die "Error: number of words must be specified\n";
        }

        # make sure the algorithm got set
        if($algorithm eq ''){
                die "Error: must specify an algorithm\n";
        }

        # compute the width of the address bus
        $addrwidth = log($numberofwords)/log(2);

        # remove all white space for easier parsing
        $algorithm =~ s/ //g;
        my @elements = split(/;/, $algorithm);
```

```
        # go through each element in the march test
        foreach $element (@elements) {
                if($element =~ m/^ud\(/ || $element =~ m/^u\(/ ){
                        # for an u or ud perform operations ascendingly
                        for($i=0; $i<$numberofwords; $i++){
                                # parse out the direction and paren's for this operation
                                $element =~ s/.*\(//;
                                $element =~ s/\)//;
                                # get a list of all the operations and iterate
                                my @ops = split(/,/, $element);
                                foreach $op (@ops){
                                        # die of the operation is in the form of R0 or W1
                                        if(length($op) != 2){
                                                die "Error: improperly formatted operation
$op\n";
                                        }
                                        #create a new instruction and provide necessary info
                                        my $inst = inst->new();
                                        $inst->op(substr($op,0,1));
                                        $inst->addr($i);
                                        # create a binary string of the correct length
                                        my $temp = '';
                                        for(my $j=0; $j<$wordsize; $j++){
                                                $temp .= substr($op,1,1);
                                        }
                                        # convert it to octal
                                        $temp = oct("0b$temp");
                                        # now store it in hex
                                        $inst->data(sprintf("%X",$temp));
                                        $inst->addrwidth($addrwidth);
                                        $inst->datawidth($wordsize);
                                        $inst->print($format);
                                }
                        }
                } elsif($element =~ m/^d\(/) {
                        # for d perform operations descendingly
                        for($i=$numberofwords-1; $i>=0; $i--){
                                # parse out the direction and paren's for this operation
                                $element =~ s/.*\(//;
                                $element =~ s/\)//;
                                # get a list of all the operations and iterate
                                my @ops = split(/,/, $element);
                                foreach $op (@ops){
                                        # die of the operation is in the form of R0 or W1
                                        if(length($op) != 2){
                                                die "Error: improperly formatted operation
$op\n";
                                        }
                                        #create a new instruction and provide necessary info
                                        my $inst = inst->new();
                                        $inst->op(substr($op,0,1));
                                        $inst->addr($i);
                                        # create a binary string of the correct length
                                        my $temp = '';
                                        for(my $j=0; $j<$wordsize; $j++){
                                                $temp .= substr($op,1,1);
                                        }
                                        # convert it to octal
                                        $temp = oct("0b$temp");
                                        # now store it in hex
                                        $inst->data(sprintf("%X",$temp));
                                        $inst->addrwidth($addrwidth);
                                        $inst->datawidth($wordsize);
                                        $inst->print($format);
                                }
```

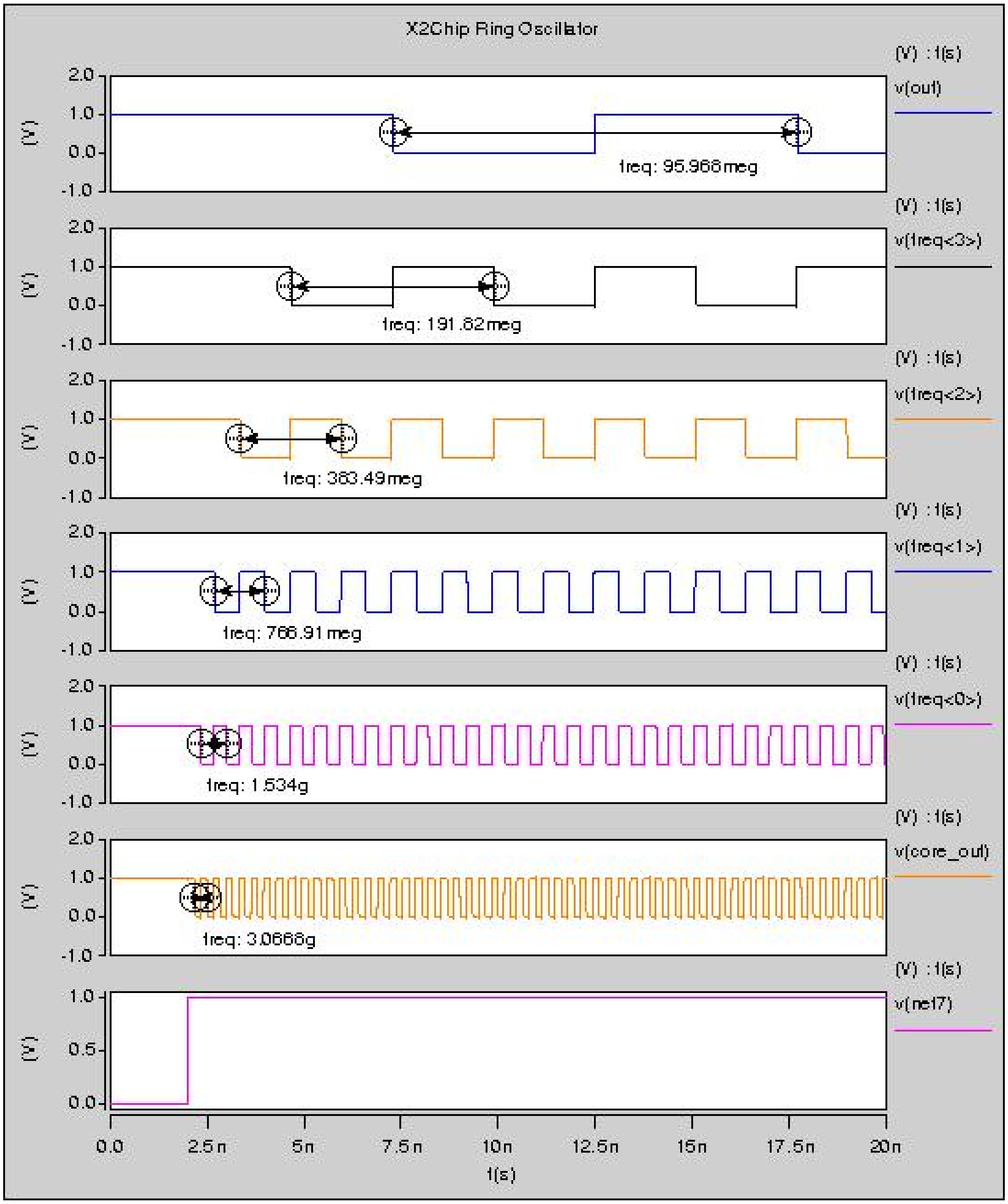
```
                        }
                } else {
                        die "Error: improperly formated element $element\n";
                }
        }
}
```

Appendix B – x2chip.il (bubblePins procedure)

```
; Author: Philip Iles (pmiles@ncsu.edu)

procedure( bubblePins( layout )

        ;we will iterate over each instance and find the pins for each
        foreach(inst layout~>instances

                ; we need to know where the instance is to add the offset for x and y
                ; compared to where the pin is within the instance
                bBox = inst~>bBox

                instllx = caar(bBox) + 0.285
                instlly = cadar(bBox) + 0.031

                ; get the names of all the net names at this level of hierarchy
                netNames = inst~>conns~>net~>name

                ; get the figures of all the pins for this instance
                figs = inst~>conns~>term~>pins~>fig

                ; get the list of terminals
                terms = inst~>conns~>term

                ; length of the lists
                numberOfPins = length(netNames)
                if(length(netNames)!=length(figs) then
                        error("differing number of netNames %d and figs %d\n",
length(netNames), length(figs))
                )

                ; lets get to work
                for(i 0 (numberOfPins-1)

                        ; get the i-th figure in the list
                        fig = nth(i figs)

                        ; the easy stuff first
                        lpp = car(fig~>lpp)
                        layer = car(lpp)
                        currentPin = nth(i netNames)
                        term = nth(i terms)
                        termDir = term~>direction

                        ; figure out the coordinates of the bBox for the pin
                        instPinbBox = fig~>bBox

                        ; lower left coordinate pair
                        instPinll = nth(0 nth(0 instPinbBox))
                        instPinllx = car(instPinll)
                        instPinlly = cadr(instPinll)

                        ; upper right coordinate pair
                        instPinur = nth(1 nth(0 instPinbBox))
                        instPinurx = car(instPinur)
                        instPinury = cadr(instPinur)

                        ; dimensions of the pin
                        width = instPinurx - instPinllx
                        height = instPinury - instPinlly
```

```
                        ; set the coordinates of the bBox of the pin to be created
                        pinbBoxllx = instllx + instPinllx
                        pinbBoxlly = instlly + instPinlly
                        pinbBoxurx = pinbBoxllx + width
                        pinbBoxury = pinbBoxlly + height

                        ; to get an idea of if the pin is horizontal or vertical
                        if(width > height then
                                orient = "R0"
                                textHeight = 0.07
                                textllx = pinbBoxllx
                                textlly = pinbBoxlly
                        else
                                orient = "R90"
                                textHeight = 0.07
                                textllx = pinbBoxurx
                                textlly = pinbBoxlly
                        )

                        ; put the coordinates into a bBox object (list)
                        pinbBox = list(list(pinbBoxllx pinbBoxlly) list(pinbBoxurx
pinbBoxury))

                        ; call the layout function to create the pin
                        leCreatePin(
                                layout
                                lpp
                                "rectangle"
                                pinbBox
                                currentPin
                                termDir
                                list("top" "bottom" "left" "right")
                        ) ; end leCreatePin

                        ; call the database function to create the label
                        dbCreateLabel(
                                layout
                                list(layer "label")
                                list(textllx textlly)
                                currentPin
                                "lowerLeft"
                                orient
                                "roman"
                                textHeight
                        ) ; end dbCreateLabel
                ) ; end for each pin
        ) ; end for each instance
) ; end bubblePins
```

Appendix C – Simulated X2Chip Ring Oscillator Demonstrating Frequency Division



Appendix D – X2Chip Verilog Test Bench for SRAM Muxing

```
`include "mySram.v"

//test bench to test the SRAM
module mySram_tb();
      wire [3:0] q;
      reg [9:0] a;
      reg [3:0] d;
      reg cen;
      reg wen;
      reg clk;
      reg [1:0] sel;

      initial begin

            $dumpfile("waves.vcd"); // save waveforms in this file
            $dumpvars;        // saves all waveforms

            $display ("time\tclk\taddr\t\td\tq\tcen\twen\tsel\n");
            $monitor ("%g\t%b\t%b\t%b\t%b\t%b\t%b\t%d", $time, clk, a,
d, q, cen, wen, sel);

            clk = 1;
            sel = 2'h0;
            cen = 1;
            wen = 1;
            a = 10'h0;
            d = 4'h0;

            #1600000 cen = 0; wen = 0; a = 10'h1; d = 4'hf; // 3.6ms
            #1000000 a = 10'h2; d = 4'ha;             // 4.6ms
            #1000000 wen = 1; a = 10'h1; d = 4'hx;          // 5.6ms
            #500000  sel = 2'h1;                      // 6.1ms
            #100000  sel = 2'h2;                      // 6.2ms
            #100000  sel = 2'h3;                      // 6.3ms
            #300000  a = 10'h2; sel = 2'h0;                 // 6.6ms
            #500000  sel = 2'h1;                      // 7.1ms
            #100000  sel = 2'h2;                      // 7.2ms
            #100000  sel = 2'h3;                      // 7.3ms
            #500000  $finish;
      end

      //clock generator
      always begin
            #500000 clk=~clk; //toggle every 0.5ms for 1ms clock
      end

      mySram sram_ut (q,a,d,cen,wen,clk,sel);
endmodule
```