Characterization, Validation & Design tuning of CMOS Memory in 0.18micron Technology

Submitted in partial fulfillment of the requirements

for the degree of

MASTER OF TECHNOLOGY

IN

ELECTRONICS & COMMUNICATION ENGG.

(VLSI DESIGN)

UNDER THE GUIDANCE OF

Mr. PIYUSH JAIN

STMICROELECTRONICS PVT. LTD.

Submitted by

Sailesh Pandey

(06MEC015)

Department of Electronics & Communication Engineering

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY OF SCIENCE & TECHNOLOGY,

AHMEDABAD 382481

CERTIFICATE

This is to certify that the M.Tech Dissertation report entitled "***Characterization, Validation and Design tuning of CMOS Memory in 0.18micron Technology***" submitted by **Sailesh Pandey (Roll No. 06MEC015)** towards the partial fulfillment of the requirements for Semester III-IV of Master Of Technology (Electronics and Communication Engineering) in the field of VLSI Design of **Nirma University of Science and Technology, Ahmedabad** at **S. T. Microelectronics, Greater Noida** is the record of the work carried out by him under our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for examination. The results embodied in this Dissertation Project Work to the best of our knowledge have not been submitted to any other University or Institute for award of any degree or diploma.

Date:

Project Guide
Mr. Piyush Jain
Senior designer
STMicroelectronics,
Greater Noida, India

Internal Project Guide
Dr. N.M. Devashrayee
Institute of Technology,
Nirma University, Ahmedabad

Project Manager
Mr. Vivek Asthana
STMicroelectronics,
Greater Noida, India

HOD
EE Department
Institute of Technology,
Nirma University, Ahmedabad

Mrs. Nutan Aggarwal
Section Manager (NVSM-FTM)
STMicroelectronics,
Greater Noida, India

Director
Institute of Technology,
Nirma University, Ahmedabad

ACKNOWLEDGEMENT

“Outstanding achievement is not possible in vacuums. It needs lot of help and assistance besides a healthy environment, luckily I have.”

First and foremost, I would like to express my hearty thanks and indebtedness to my guide ***MR. PIYUSH JAIN*** for his enormous help and encouragement throughout the course of this thesis**,** who happens to be my role model, has always given me a real example of how a researcher should be, proving ***‘Vidya Dadati Vinayam’***.

I express my soulful gratitude to ***MR. ANAND HARDI*** for their invaluable suggestion for my training. I specially want to acknowledge ***MRS. NUTAN AGGARWAL*** who gives me opportunity to do my thesis work and provide all resources required for my project work. I also want to acknowledge ***Vivek Asthana, Sachin, Nitesh Gautam, Parminder Singh, Deepti Saini, Seema Jaiswal, Swarnima Singh , Manish Patel, Chirag Gulati, Hemant, Mohit Verma, Varun, Satisbabu, Shivendera, Saurabh, Jasvinder, Naveen Batra*** *and* ***Jitendra Dasani*** without whom this project work could not have seen the daylight and help me with his constant involvement during my project tenure here.

I would like to express my sincere gratitude to ***Dr. N.D. Devashrayee (***Head of Department, M.Tech VLSI design, Nirma University, Ahmedabad***)*** for his continuous guidance, support and enthusiasm.

I acknowledge gratefully the help and suggestion of FTM people and friends, in spite of their busy schedule and huge workload, they were always eager to help me with their warm attitude and technical knowledge.

Finally, no word will be enough to express my deepest reverence to family without those entire enthusiasms and help I wouldn’t have been reach at this position, in the last the entire work presented in this thesis is dedicated to my family.

(SAILESH PANDEY)

ABSTRACT

This thesis explores the performance characterization of a Static Random Access Memory (SRAM), the design validation of the Dual port SRAM, evaluation of Eldo Optimizer, evaluation of Multi-threading & Setup for Crosstalk analysis. The SRAM cell characterization includes Static Noise Margin, Write Margin, and Discharge Rate. Design Validations include Marginality Analysis, Power Estimation, Pin Cap Measurement, Leakage Measurement, Tight Stimuli verification. Main focus on optimizing area, delay and power at circuit as well as on architectural level.

The key to low power operation in the SRAM data path is to reduce the signal swings on the high capacitance nodes like the bitlines and data lines. Clocked voltage sense amplifiers are essential for obtaining low sensing power, and accurate generation of their sense clock is required for high speed operation. We investigate tracking circuits to limit bitline and I/O line swings and aid in the generation of the sense clock to enable clocked sense amplifiers. The tracking circuits essentially use a replica memory cell and a replica bitline to track the delay of the memory cell over a wide range of range of process and operating conditions. We present the low power SRAM by using some low power methodologies like page type architecture, divided word line (DWL), selective precharging, pre-decoding scheme, two-level multiplexing.

To reduce the power dissipation due to discharge of bitlines from Vdd to 0v during the time when word line selected we made use of self-time concept. This can be done either by dummy structure approach. The dummy structure approach is more immune to process parameter variation because the dummy I/O is similar to normal I/O, therefore inter-chip variation is negligible. This SRAM has dummy column discharges through the dummy cells. This discharge is faster then normal discharge so the reset signal sense amplifier enable signal can be activated before the normal discharge exceed voltage difference between bitlines being resolve by sense amplifier.

The basic application of Dualport SRAM is in Video SRAM, which allows the memory to allocate one channel to refreshing the screen while the other is focused on changing the images on the screen. Since video memory chips are used in much lower quantities than main memory chips, they tend to be more expensive

Eldo optimizer tool, help in optimization of different circuits, element parameters & device lengths, widths & area. This uses Eldo tool for providing inputs & simulation. We had used this tool in above Dualport Memory cell analysis & got good results. This tool can be very much helpful reduce the time of designing any circuits. The Eldo Multi-threading methodology is used to distribute work & launch these child processes on different processors so the simulation time is drastically reduce.

Crosstalk analysis setup uses Hsim tool for the check of static & dynamic crosstalk & noise-sensitivity estimation. The effect is measured according to user-defined thresholds for change of characteristic signal for dynamic or coupling ratio for static analysis. This is helpful in reducing manual work of finding the nodes where crosstalk can be occurs.

CONTENTS

List of figure

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE AND SCOPE OF THE PROJECT

The electronics industry has achieved a phenomenal growth over the last few decades, mainly due to the rapid advances in integration technologies and large-scale systems design.

Complementary metal oxide semiconductor (CMOS) digital integrated circuits are the enabling technology for the modern information age. Because of their intrinsic features in low-power consumption, large noise margins, and ease of design, CMOS integrated circuits have been widely used to develop random access memory (RAM) chips, microprocessor chips, digital signal processor (DSP) chips, and application-specific integrated circuits (ASIC) chips. The popular use of CMOS circuits continues to grow with the increasing demands for low power, low-noise integrated electronic systems.

For VLSI circuit design, however, it is important that the design be done in the context of global optimization with proper boundary conditions. In fact, the beauty of integrated circuits is that the final design goal is the concerted performance of all interconnected transistors, and not of individual transistors. No matter how well an individual transistor performs, if the technology fails to have equally good interconnects, the total performance can be very poor due to large parasitic capacitances and resistances; these translate into a large delay in the interconnection lines between transistors or logic gates. The very important role of computer aided circuit simulation tools in VLSI design is well recognized. Computer simulation is, and will continue to be, an essential part of the design process, both for performance verification and for fine-tuning of circuits. However, the emphasis on simulation must be well-balanced with the emphasis on hands-on-design and analytical estimates, so that the significance of the later is not overwhelmed by the extensive use of computer-aided techniques.

Digital systems require the capability of storing and retrieving large amounts of information at high speeds. *Memories* are circuits or systems that store digital information in large quantity, hence are vital components in modern integrated circuits. Manufacturers of such products demand low-priced memories with low-power consumption, high-speed operation, high density, and small package size.

The semiconductor markets have embraced the fact that the architecture of the memory structure has a considerable impact on the performance of the system. Over the years, technology advances have been driven by memory designs of higher and higher density.

While designing System-on-Chip (SOC), system architects need to resolve a number of complex issues in high-performance system applications. However, one of the fundamental problems in these applications is Memories - the bottlenecks and challenges of system performance often reside in its memory architecture. As advances in memories come to life, system designers are faced with the challenge of selecting the proper memory for their application.

Memory developers have to design memories to address the issues in bandwidth, latency, density, power and cost. Unfortunately, it is not possible for a single memory technology to address all these issues with distinct advantages - this translates to an arsenal of components available for designers to design their system.

The criticality of my project work is to characterize the performance of memory cell to achieve the target to adjust the things according to international technology roadmap for semiconductor in the field of embedded memory. This is somewhat specific to user requirements.

1.2 DEFINITION OF PROBLEM

Dual port SRAM is specialized RAM which is used on video cards. It is dual ported RAM which means it can be accessed by two different devices simultaneously. It enables data to be read from video RAM and sent to the display devices. At the same time data to be written in to video RAM

The type and speed of memory used in the video card play an important role in the card's performance. Faster technologies mean improved performance in two ways. First, the card itself is faster when the memory is faster, leading to better performance overall for the Processor. Second, better memory means higher memory bandwidth, leading to support for higher resolution and color depths, and better refresh rates.

DISSERTATION INTENT

I. Performance Characterization of Dual-Port SRAM Cell.
II. Design Validations to optimize the design to access the entire four axes i.e. Area, Power, Speed and Robustness.

1.3 Dualport SRAM

This Dualport SRAM allows simultaneous access of the memory particular memory array. This Dualport memory allow two simultaneous read operation but don’t allow simultaneous two write or one read & other write operations. Basically this special high-speed type of RAM is used as video memory in which visual information is temporarily store & being transferred to the display hardware in a computer. It is used by graphics cards for rapid transfer of DATA between the card and CPU.

This Dualport SRAM memory is 8T memory which means that it uses eight transistor in memcell rather than 10T memory. So this can be design as self-time style in which the Tcklh is not constant clock goes low when the read/write operation is done.

This special-purpose RAM with two data paths for access, rather than the one path in conventional RAM. The two paths let a DPRAM board handle two functions at once: display refresh and processor access. DPRAM doesn't force the system to wait for one function to finish before starting the other, so it permits faster operation for the video subsystem.

This type of computer memory often used for video purposes. Faster than normal computer RAM, the amount of DPRAM determines how many colors a given video system can display.

CHAPTER 2

EXHAUSTIVE SURVEY OF MEMORY

2.1 LITERATURE SURVEY

Data storage devices: An overview

From the beginning of the electronics industry, storage of data has been a major point of consideration. Many storage devices have been developed by now, with various working principles and data storage techniques. In general, the data storage devices can be classified by a wide variety of aspects, but most frequently, they are divided by technology into the semiconductor types and the moving media types, which require mechanical equipment for operation.

The five basic semiconductor types are bipolar, N-channel and P-channel MOS and complementary MOS.

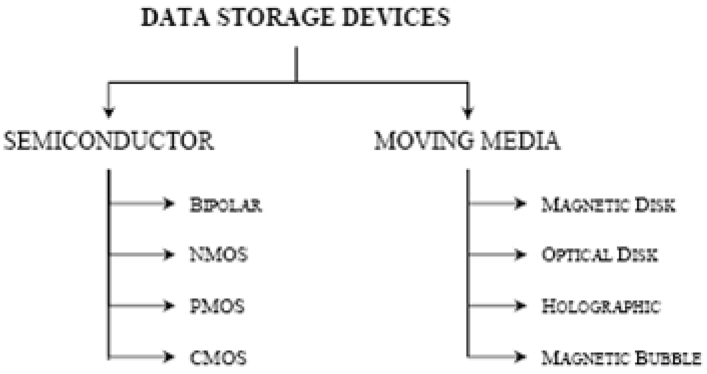


Figure. 2.1 - Basic classification of Data Storage Devices

The moving media types include magnetic disk, optical disk and holographic storage. While magnetic bubbles are not mechanical, they require equipment for supplying a magnetic field for operation and are thus considered with the mechanical types. Since the semiconductor memories are decreasing in cost per bit faster than the other types of data storage, various attempts to configure them for the disk application are occurring.

There also exist devices using various combinations of the basic semiconductor technologies such as CMOS-NMOS (Mix-MOS) and bipolar-CMOS (BI-CMOS) and devices made in exotic technologies such are gallium arsenide (GaAs).

MOS MEMORIES: INTRODUCTION

The ideal memory would be low cost, high performance, high density, with low power dissipation, random access, non-volatile, easy to test, highly reliable, and standardized throughout the industry.

The MOS memories fall into two broad categories:

- **Read-Write memories**: Dynamic RAMs and Static RAMs, allow the user both to read information from the memory and to write new information into memory while it is still in the system.
- **Read Only Memories**: ROMs, EPROM's, EEPROMs, are used primarily to store data; however, the EEPROMs can also be written into a limited number of times while in the system. Read-Only memories are non-volatile, that is, they retain their information stored in it even if the is turned off.

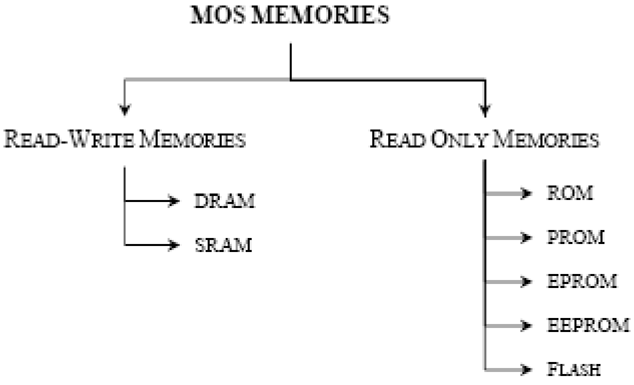


Figure. 2.2 - Major classes of MOS memories

READ-WRITE MEMORIES

Read-write random-access memories (RAM) may store information in flip-flop style circuits or simply as charge on capacitors. Because read-write memories store data in active circuits, they are *volatile*; that is, stored information is lost if the power supply is interrupted. The natural abbreviation for read-write memory would be RWM. However, pronunciation of this acronym is difficult. Instead, the term RAM is commonly used to refer to read-write random-access memories.

The two most common types of RAMs are the static RAM (SRAM) and the dynamic RAM (DRAM). Static RAMs hold the stored value in flip-flop circuits as long as the power is on. SRAM tends to be high-speed memories with clock cycles in the range of 5 to 50 ns. Dynamic RAMs store values on capacitors. They are prone to noise and leakage problems, and are slower than SRAM, clocking at 50 ns to 200 ns. However, DRAMs are much denser than SRAMs, up to four times denser in a given generation of technology.

READ-ONLY MEMORIES

Read-only memories (ROMs) store information according to the presence or absence of transistors joining rows to columns. ROMs have read speeds comparable to those for read-write memories. All ROMs are *nonvolatile*, but they vary in the method used to enter (write) stored data. The simplest form of ROM is programmed when it is manufactured by formation of physical patterns on the chip; subsequent changes of stored data are impossible. These are termed *mask-programmed* ROMs.

In contrast, *programmable* read-only memories (PROMs) have a data path present between every row and column when manufactured, corresponding to a stored 1 in every data position. Storage cells are selectively switched to the 0 state once after manufacture by applying appropriate electrical pulses to selectively open (blow out) row-column data paths. Once programmed, or *blown*, a 0 cannot be changed to 1.

Erasable programmable read-only memories (EPROM's) also have all bits initially in one binary state. They are programmed electrically (similar to the PROM), but all bits may be erased (returned to the initial state) by exposure to ultraviolet (UV) light. The packages for these components have transparent windows over the chip to permit the UV irradiation.

Electrically erasable programmable read-only memories (EEPROM's, E_2PROM, or Esquared PROM's) may be written and erased by electrical means. These are the most advanced and most expensive form of PROM. Unlike EPROM's, which must be totally erased and rewritten to change even a single bit, E2PROM's may be selectively erased. Writing and erasing operations for all PROM's require times ranging from microseconds to milliseconds. However, all PROM's retain stored data when power is turned off; thus they are termed nonvolatile.

A recent form of EPROM and E$_2$PROM is termed *Flash Memory*, a name derived from the fact that blocks of memory may be erased simultaneously. Their large storage capacity has made this an emerging mass storage medium. In addition, these types of memories are beginning to replace the role of ROMs on many chips, although additional processing is required to manufacture Flash memories in a standard CMOS technology.

MEMORY ORGANIZATION

The preferred organization for most large memories is the *random-access* architecture. The name is derived from the fact that memory locations (*addresses*) can be accessed in random order at a fixed rate, independent of physical location, for reading or writing.

The storage array, or *core*, is made up of simple cell circuits arranged to share connections in horizontal rows and vertical columns. The horizontal lines, which are driven only from outside the storage array, are called *wordlines*, while the vertical lines, along which data flow into and out of cells, are called *bitlines*.

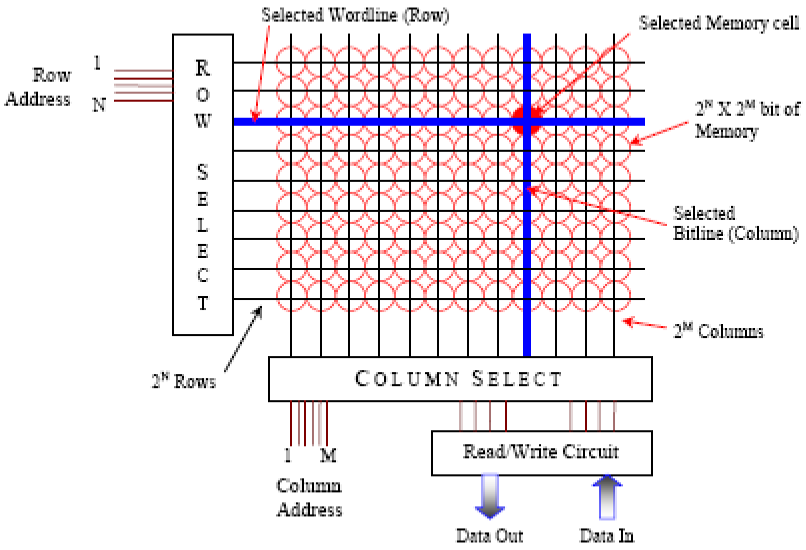


Figure. 2.3 - Typical Memory Organization

A cell is accessed for reading or writing by selecting its row and column. Each cell can store 0 or 1. Memories may simultaneously select 4, 8, 16, 32, or 64 columns in one row depending on the application. The row and column (or columns) to be selected are determined by decoding binary address information.

Memory exists as stand-alone component, but also as embedded blocks in system-onchip.

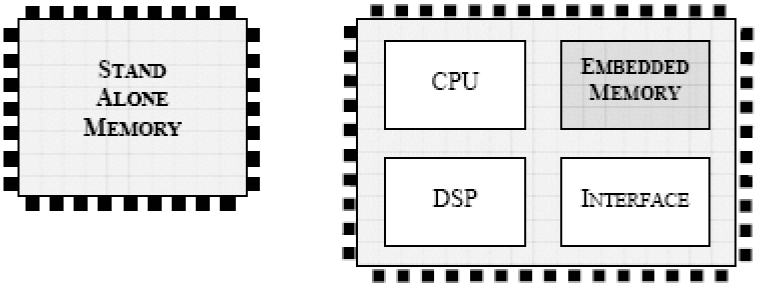


Figure. 2.4 - Memory existence as a stand-alone component and an embedded block

Memory cell circuits can be implemented in a wide variety of ways. In principle, the cells can be based on the flip-flop designs since their intended function is to store bits of data. However, these flip-flops require a substantial amount of area and are not appropriate when millions of cells are needed. In fact, most memory cell circuits are greatly simplified compared to register and flip-flop circuits. While the data storage function is preserved, other properties including quantization of amplitudes, regeneration of logic levels, input-output isolation, and fanout drive capability may be sacrificed for cell simplicity. In this way, the number of devices in a single cell can be reduced to one to six transistors.

At the level of a memory, the desired logic properties are recovered through use of properly designed peripheral circuits. Circuits in this category are the decoders, sense amplifiers, column precharge, data buffers, etc. These circuits are designed so that they may be shared among many memory cells. Read-write (R/W) circuits determine whether data are being retrieved or stored, and they perform any necessary amplification, buffering, and translation of voltage levels.

TIMING DIAGRAMS

Timing diagrams specify the minimum required and maximum expected timing requirements for system actions. The two sets of timing symbols are self-explanatory, one being the standard for timing symbols and the other older one in widespread usage.

The operation of the SRAM starts with the detection of an address change in the address register. An address change activates the SRAM circuits, the internal timing circuit generates the control clocks, and the decoders select a single memory cell.

At write, the memory cell receives a new datum from the data input buffers; at read, the sense amplifier detects and amplifies the cell signal and transfers the datum to the output buffer. Data input/output and write/read are controlled by output enable OE and write enable WE signals. A chip enable signal CE allows for convenient applications in clocked systems.

In some systems, power consumption may be saved by the use of the power down signal PD. The power down circuit controls the transition between the active and standby modes. In active mode, the entire SRAM is powered by the full supply voltage; in standby mode, only the memory cells get a reduced supply voltage. In some designs, the memory-internal timing circuit remains powered and operational also during power down.

BASICS OF READING TIMING DIAGRAMS

It is important to understand the characteristics of the memories, to understand them better. The best indicator of these characteristics is the data sheet specification for the particular memory.

In a memory system, there are signals going from the processor via the bus into the inputs of the memory and signals coming from the outputs of the memory onto the bus and to the processor.

- ❖ Inputs from the system processor to the memory include:
  - ➢ Addresses, which indicate the memory locations selected.

- ➢ Write enable, which chooses between read and write mode and controls writing of new information into the memory.
- ➢ Chip select(s), which select one memory out of several in a system. If a chip select is off, the memory is deselected.
- ➢ Output enable, which can be used to control the output buffer.
- ➢ Data input(s), to be written into the memory

❖ Outputs from the memory include

- ➢ Data output(s) being read from the memory

Some memories such as SRAM with wider data path can have a common pin for input and output.

READ CYCLE TIMINGS

A timing diagram for a basic read cycle during which the system reads out information that is stored in a static RAM (in fig. 2.5) for a wide bus, common I/O SRAM with chip enable, and output enable functions. It consists of

- ▪ System selects the RAM by turning the chip select on (~CS low).
- ▪ System sets the correct addresses (A set).
- ▪ System turns the output enable on (~OE low).

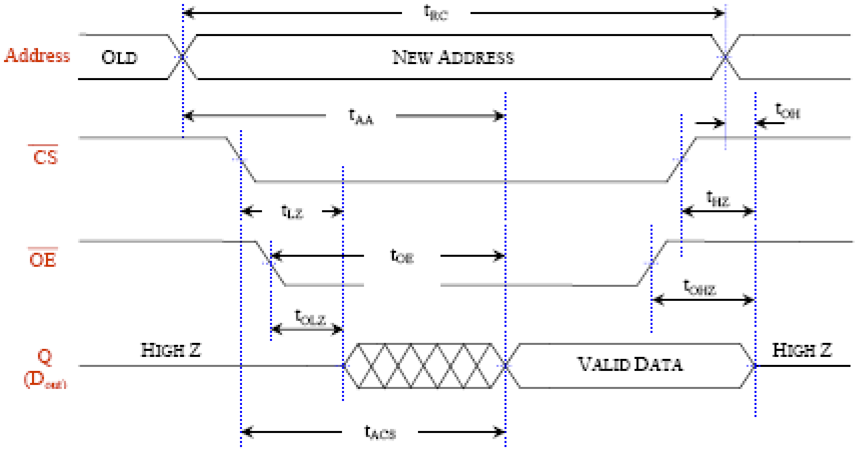


Figure. 2.5 - Typical read cycle diagram for SRAM

- System must make sure that the time that old data from other sources is still on the common I/O bus is less than the minimum of output enable low to output active (t_{OLZ}) or chip enable low to output active (t_{LZ}).
- The system must wait a minimum time of address access time (t_{AA}) in order to be sure of correct data.

The expansions of the acronyms for read timing parameters are

T_{RC} : Read Cycle Time

T_{AA} : Address Access Time

T_{ACS} : Chip Enable Access Time

T_{OE} : Output Enable Access Time

T_{OH} : Output Hold from Address Change

T_{LZ} : Chip Enable Low to Output Enable

T_{OLZ} : Output Enable Low to Output Enable

T_{HZ} : Chip Enable High to Output Enable - Z

T_{OHZ} : Output Enable High to Output Enable - Z

After the maximum required wait time (which is the minimum time the system must wait) the system may read the information stored in the memory.

WRITE CYCLE TIMINGS

A simple write cycle for changing data in an SRAM (writing into it) is shown below.

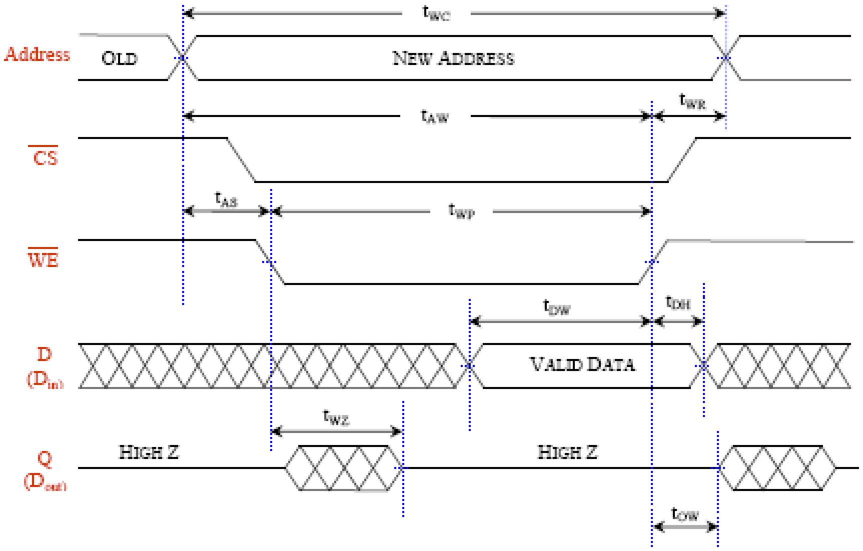


Figure. 2.6 - Typical write cycle diagram for SRAM

It consists of

- System sets the correct addresses (A set).
- System selects the RAM by turning the chip select on (~CS low).
- System waits a minimum required amount of time after changing the addresses for the RAM to do internal ‘set up’ of the addresses t_{AS}, and then turns the write enable on (~WE low).
- System waits a minimum required amount of time after turning on the write enable (t_{WZ}) for the memory to disable the data output driver' Q' in preparation for using these lines for data input.
- System inputs the new data and waits a minimum required amount of time for the memory to write the data before turning off the write enable (t_{DW}).
- System waits a minimum required amount of time after turning the write enable on before turning it off (t_{P}). This is to be sure the write enable pulse width is wide enough for correctly writing the data into the RAM.
- System waits a minimum required amount of time after turning the write enable Off
    - Before changing the data (t_{DH}). This is called the 'data hold time', and it ensures that the data is stable during the entire write cycle.
    - Before changing addresses to start the next cycle (t_{WR}). This is called the 'write recovery time' and ensures that the addresses are stable during the entire write cycle.
    - Makes sure the data have disappeared before the RAM turns the data output drivers back on (t_{OW}).

The expansions of the acronyms for read timing parameters are

T_{WC} : Write Cycle Time

T_{AS} : Address Setup Time

T_{AW} : Address Valid to End of Write

T_{WP} : Write Pulse Width

T_{DW} : Data Valid to End of Write

T_{DH} : Data Hold Time

T_{WZ} : Write Enable Low to Output High - Z

T_{OW} : Write Enable High to Output Active

T_{WR} : Write Recovery Time

The RAM is now ready to begin the next cycle. It should be clear that the minimum timings are periods when the system must wait for the RAM to do something, and the maximum timings are guaranteed limits within which the system will act.

Minimum access times for reading and writing are not necessarily the same, but for simplicity of design, most systems specify a single time for both reading and writing. For semiconductor read-write memories, the read access time is typically 50 to 80% of cycle time.

CRITICAL TIMING PATH

The critical path determining cycle times comprises the delays through the

1. Row address buffer 2. Row address decoder
3. Wordline 4. Bitline
5. Sense amplifier and 6. Output buffer circuits

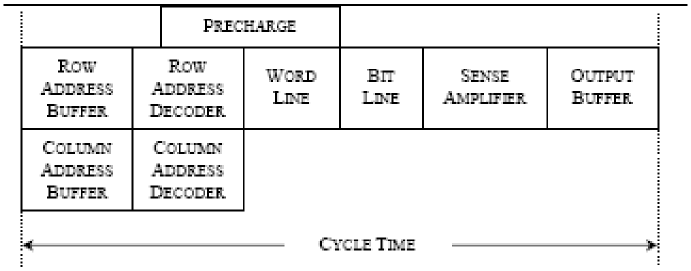


Figure. 2.7 – Critical Timing Path for SRAM

Precharge and initiation times for sensing as well as column address buffer and decoder delays can be hidden in the critical timing of an SRAM.

The memory clock cycle time, is the minimum time needed to complete successive read or write operations. Maximum read access time should not exceed the memory cycle time since there are write setup operations needed before each memory operation. The cycle time is essentially the reciprocal of the time rate at which address information is changed while reading or writing at random locations.

FUNCTIONAL SRAM CHIP MODEL

Memories are said to be static if no periodic clock signals are required to retain stored data indefinitely. Memory cells in these circuits have a direct path to V_{DD} or Gnd or both. Read-write memory cell arrays based on flip-flop circuits are commonly referred to as Static RAM's or SRAM's.

A functional block diagram for the SRAM chip is shown below

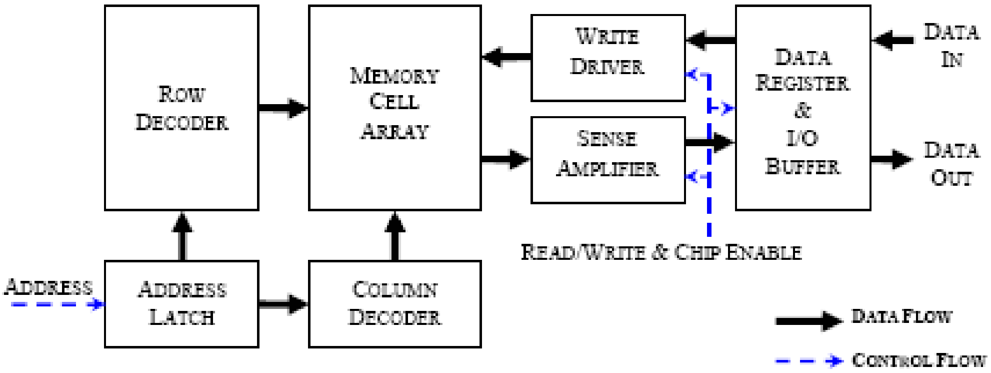


Figure. 2.8 – Functional SRAM chip model

1.The address latch block, receives the address.

2. The higher order bits of the address are connected to the row decoder, which selects a row in the memory cell array.

3. The lower order address bits go to the column decoder, which selects the required columns. The number of column selected depends on the data width of the chip, that is the number of data lines of chip, which determines how many bits can be accessed during a read or write operation

4. When the read/write line indicates *read operation*, the contents of the selected cells in the memory cell array are amplified by the sense amplifiers, loaded in the data register & presented on the data-out line(s).

5. During a *write operation* the data on the data-in line(s) are loaded into the data register & written in to the memory cell array through the write driver. Usually the data-in & data-out lines are combined to form bidirectional data lines, thus reducing the number of pins on the chip.

6. The chip-select line enables the data register, together with read/write line, the write driver.

2.2 SRAM CELL DESCRIPTION

The basic static RAM cell is consists of two cross-coupled inverters and two access transistors. The access transistors are connected to the wordline at their respective gate terminals, and the bitlines at their source/drain terminals.

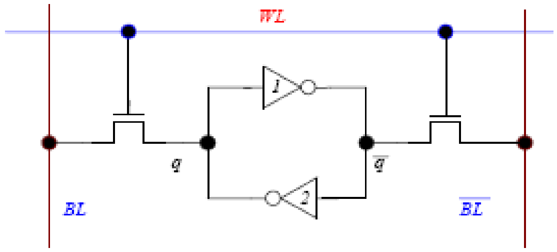


Figure. 2.9 – Basic SRAM cell

The wordline is used to select the cell while the bitlines are used to perform read or write operations on the cell. Internally, the cell holds the stored value on one side and its complement on the other side. For reference purposes, assume that node q holds the stored value while node ~q holds its complement. The two complementary bitlines are used to improve speed and noise rejection properties.

VOLTAGE TRANSFER CHARACTERISTICS

The Voltage Transfer Characteristics (VTC) conveys the key cell design considerations for read and writes operation. In the cross-coupled configuration, the stored values are represented by the two stable states in the VTC.

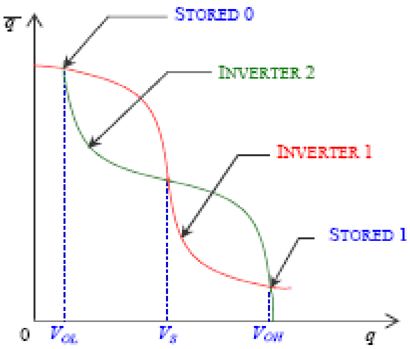


Figure. 2.10 – Voltage Transfer Characteristics for Basic SRAM cell

The cell will retain its current state until one of the internal nodes crosses the switching threshold, Vs. When this occurs, the cell will flip its internal state. Therefore, during a read operation, its current state must not be disturbed, while during the write operation the internal voltage is forced to swing past Vs to change the state.

SRAM ARRAY OPERATION

In an array the row select lines, or wordlines, run horizontally. All cells connected to a given wordline are accessed for reading or writing. The cells are connected vertically to the bitlines using the pair of access devices to provide a switch able path for data into and out of the cell. Two column lines, b and ~b, provide a differential data path. In principal, it should be possible to achieve all memory functions using only one column line and one access device, but due to normal variations in device parameters and operating conditions, it is difficult to obtain reliable operation at full speed using a single access line. Therefore, the symmetrical data paths b and ~b are usually used.

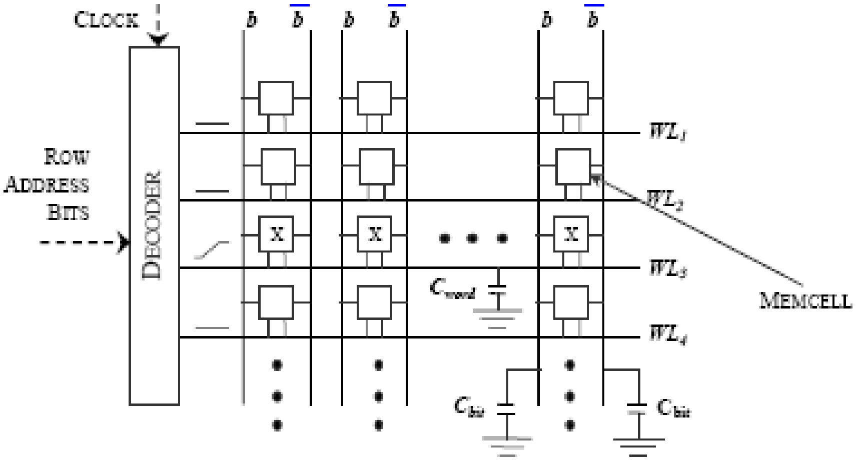


Figure. 2.11 – Wordline and Dual Bitline configuration

Row selection in CMOS memory is accomplished using the decoders. For synchronous memories, a clock signal is used in conjunction with the decoder to activate a row only when read-write operations are being performed. At other times, all wordlines are kept low. When one wordline goes high, all the cells in that row are selected. The access transistors are all turned on and a read or write operation is performed. Cells in other rows are effectively disconnected from their respective wordlines.

The wordline has a large capacitance, C_{word} that must be driven by the decoder. It is comprised of two gate capacitances per cell and the wire capacitance per cell:

$$C_{word} = (2 \text{ X gate cap} + \text{wire cap}) \quad \text{X} \quad \text{no. of cells in row}$$

Once the cells along the wordline are enabled, read or write operations are carried out. For a read operation, only one side of the cell draws current. As a result, a small differential voltage develops between b and ~b on all column lines. The column addresses decoder and multiplexer select the column lines to be accessed. The bitlines will experience a voltage difference as the selected cells discharge one of the two bit-lines. This difference is amplified and sent to output buffers.

It is noted that the bitlines also have a very large capacitance due to the large number of cells connected to them. This is primarily due to source/drain capacitance, but also has components due to wire capacitance and drain/source contacts. Typically, a contact is shared between two cells.

The total bitline capacitance, C_{bit}, can be computed as follows:

$$C_{bit} = (\text{source/drain cap} + \text{wire cap} + \text{contact cap}) \text{ X no. of cells in column}$$

During a write operation, one of the bitlines is pulled low if 0 is to be stored, while the other one is pulled low if 1 is to be stored. The requirement for a successful write operation is to swing the internal voltage of the cell past the switching threshold of the corresponding inverter. Once the cell has flipped to the other state, the wordline can be reset back to its low value.

The design of the cell involves the selection of transistor sizes for all six transistors (rather being symmetric only three transistors M1, M_3, and M_5 or M_2, M_4, and M_6) to guarantee proper read and write operations. The goal is to select the sizes that minimize the area, deliver the required performance, obtain good read and write stability, provide good cell read current, and have good soft error immunity.

READ OPERATION

For a “0” is stored on the left side of the cell, and a “1” on the right side in the 6T RAM cell, M_1 is on and M_2 is off. Initially, b and ~b are precharged to a high voltage around V_{DD} by a pair of column pull-up transistors. The row selection line, held low in the standby state, is raised to V_{DD} which turns on access transistors M_3 and M_4. Current begins to flow through M_3 and M_1 to ground. The resulting cell current slowly discharges the capacitance C_{bit}. Meanwhile, on the other side of the cell, the voltage on ~b remains high since there is no path to ground through M_2.The difference between b and ~b is fed to a sense amplifier to generate a valid low output, which is then stored in a data buffer.

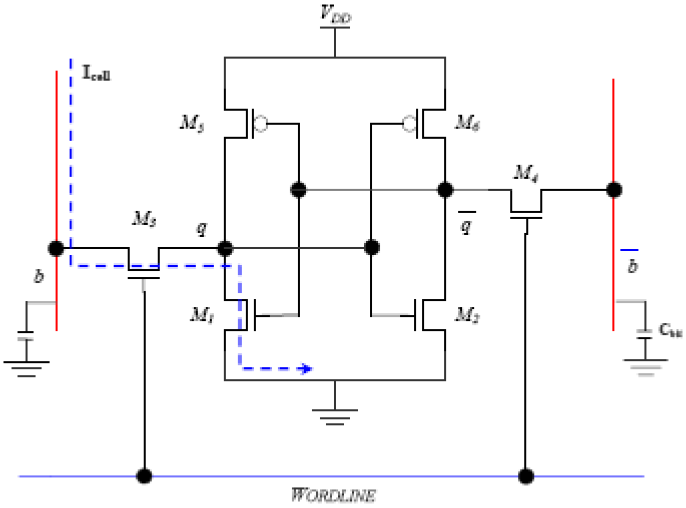


Figure. 2.12 – Read Operation in SRAM cell

Upon completion of the read cycle, the wordline is returned to zero and the column lines can be precharged back to a high value.

When designing the transistor sizes for read stability, it is ensured that the stored values are not disturbed during the read cycle. The problem is that, as current flows through M_3 and M_1, it raises the output voltage at node q which could turn on M_2 and bring down the voltage at node ~q. The voltage at node ~q may drop a little but it should not fall below V_S. To avoid altering the state of the cell when reading, the voltage at node q is controlled by sizing M_1 and M_3 appropriately. This is accomplished by making the conductance of M_1 about 3 to 4 times that of M_3 so that the drain voltage of M_1 does not rise above V_{TN}. In theory, the voltage should not exceed V_S, but this design must be carried out with due consideration of process variations and noise. In effect, the read stability requirement establishes the ratio between the two devices.

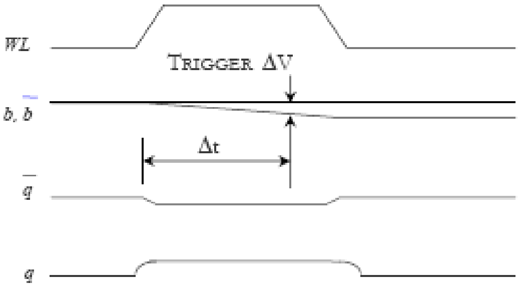


Figure. 2.13 – Read Operation waveforms

The other consideration in the read cycle design is to provide enough cell current to discharge the bitline sufficiently within 20 to 30% of the cycle time. Since the cell current, I_{cell}, is very small and the bitline capacitance is large, the voltage will drop very slowly at b. The rate of change of the bitline can be approximated as follows:

$$I_{cell} = C_{bit} \frac{dV}{dt}$$

$$\therefore \quad \frac{dV}{dt} = \frac{I_{cell}}{C_{bit}}$$

Clearly, I_{cell} controls the rate at which the bitline discharges. If a rapid full-swing discharge is desired, I_{cell} is made large. However, the transistors M_1 and M_3 would have to be larger. Since there are millions of such cells, the area and power of the memory would be correspondingly larger. Instead, a different approach is taken, attaching a sense amplifier to the bitlines to detect the small difference, ΔV between b and ~b and produce full-swing logic high or low value at the output. The trigger point relative to the rising edge of the wordline, Δt, for the enabling of the sense amplifier is chosen by based on the response characteristics of the amplifier. Then

$$I_{cell} = C_{bit} \frac{\Delta V}{\Delta t}$$

This leads to the cell current value, which, in turn, determines the final transistor sizes for M1 and M3. Alternatively, if the transistor sizes are determined to optimize the cell area, then the corresponding delay is computed as

$$\Delta t = C_{bit} \frac{\Delta V}{I_{cell}}$$

In practice, the device sizes are controlled by the RAM cell area constraints. As a rule of thumb, typically **$W_1 \approx 1.5$ X W_3** and then the sizes are optimized to provide the proper noise margin characteristics.

WRITE OPERATION

The operation of writing 0 or 1 is accomplished by forcing one bitline, either b or b, low while the other bitline remains at about V_{DD}. For SRAM cell taken above, to write 1, b is forced low, and to write 0, ~b is forced low.

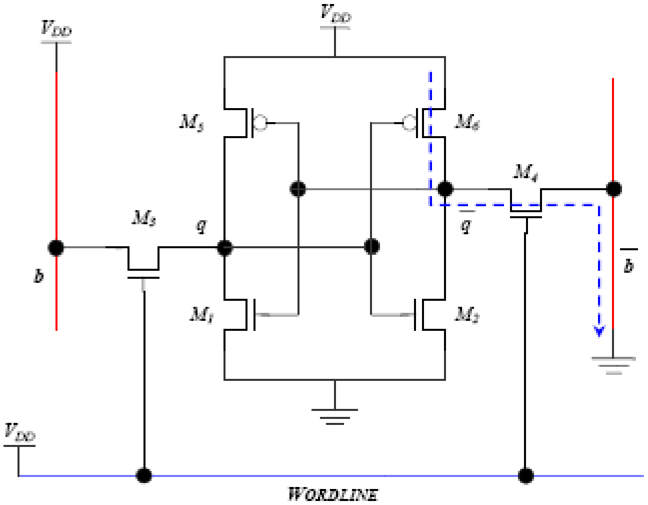


Figure. 2.14 – Write Operation in SRAM cell for writing 1

The cell must be designed such that the conductance of M_4 is several times larger than M_6 so that the drain of M_2 is pulled below V_S. This initiates a regenerative effect between the two inverters. Eventually, M_1 turns off and its drain voltage rises to V_{DD} due to the pull-up action of M_5 and M_3. At the same time, M_2 turns on and assists M_4 in pulling output ~q to its intended low value. When the cell finally flips to the new state, the row line can be returned to its low standby level.

The design of the SRAM cell for a proper write operation involves the transistor pair M_6-M_4. When the cell is first turned on for the write operation, they form a pseudo- NMOS inverter. Current flows through the two devices and lowers the voltage at node ~q from its starting value of V_{DD}. The design of device sizes is based on pulling node ~q below V_S to force the cell to switch via the regenerative action.

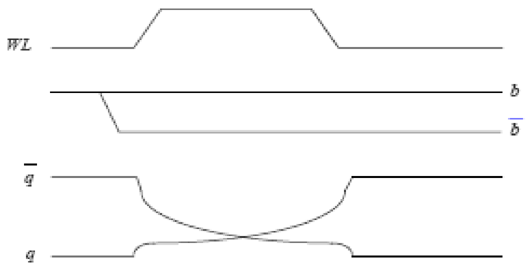


Figure. 2.15 – Write Operation waveforms

In the switching process is note that the bitline ~b is pulled low before the wordline goes up. This is to reduce the overall delay since the bitline will take some time to discharge due to its high capacitance.

The pull-up to pull-down ratio for the pseudo-NMOS inverter can be determined by writing the current equation for the two devices and setting the output to V_S. To be conservative, a value much lower than VS should be used to ensure proper operation in the presence of noise and process variations. Based on this analysis, a rule of thumb is established for M_6-M_4 sizing: $W4 \approx 1.5 \text{ X } W6$

The two ratios M_1:M_3 and M_2:M_4 are only estimates. The actual values will depend on a number of factors such as area, speed, and power considerations.

2.3 POWER CONSUMPTION IN SRAM

With continuous advancements in technologies, the SRAM memories have undergone changes with respect to following parameters:

- Decrease in geometric cell size
- Increased transistor density
- Higher complexities of the peripheral & control circuitry and
- High frequency

Such circuits consume an excessive amount of power and generate increased amount of heat. In case of reduced power processors, memories contribute significantly to the system level power consumption by taking a share of 43%-50%.

The circuits with more power dissipation are more susceptible to run-time failure and reliability problems. In addition, at increased temperatures, high power processors tend to create several silicon failures. As per studies, component failure rate double every 10°C increase in temperature.

The solution for the problem is either to pursue expensive packaging or apply cooling strategies. However, another better option is to restrict the extensive heat generation. For it, the main power consuming areas are studied and efforts are being focused to minimize the same at the extreme nodes.

The power consumption in SRAM can be divided in two modes of its operation i.e. *Active* and *Standby*. The power consumption in active mode is in the following sections

- The core area, it is the main location of power consumption in the SRAM memory.

P_{core} =bitlines of no. voltage applied e capacitance bitline

- The I/O section which uses power during precharge, multiplexer toggling, sensing and output driving also have a significant percentage of power consumption.
- The others sections which include predecoded line toggling and remaining periphery, the control and row decoder section have a small usage of power.

During the standby mode, the power consumption is very low which is used for the purpose of data retention. The main source in this mode is the leakage current in the Memcell. Static currents from other sources are negligible, sense amplifier also being disabled.

The following techniques can be deployed for low power operation of SRAM memories:

1. Capacitance reduction of wordline and bitlines. This helps in reduction of main power consumption in the active mode of operation of memory.
2. Leakage current reduction by utilizing higher threshold voltage devices in the core. This factor helps in reduction of power usage in both of the active and standby modes.
3. Operating voltage reduction. This also needs to improve the periphery circuits accordingly.
4. AC current reduction by using new decoding schemes.
5. DC current reduction by improving pulse operation techniques for wordlines and periphery.

The analysis of SRAM based on various architectures focuses the first point above, i.e. the nerve point of maximum power consumption.

Reduction in power dissipation provides following advantages:

- Better system efficiency is achieved.
- Performance of the system is improved.
- Reliability in enhanced.
- Overall cost is reduced.

2.4 ULTRA HIGH DENSITY IN SRAM

From user’s point of view all circuitry other than the memory core is redundant. So to have a high-density memory, it is required to have a smaller logic for the control, row decoders and the I/O buffers.

Density of memories has a direct link with the technology in use. Each technology has its defined limits for the minimum area of various types of memcells. A 6T, 8T, single and dual ports are few of the types available. Use of a smaller and dense memcell has to affect the density of the memory more than small percentage gains by reducing the peripheries. However, for receiving gains in density some penalties in speed are to be faced. So a balanced approach, as per the requirement in the system, to get the required feature of the memory.

A total technology change, i.e. from 130nm to 65nm or further moving to 45nm, all together the revolutionary change comes in the sizes and densities of the memories. This brings a total change all over the area and density, also affecting the periphery circuitry significantly. In this case, there is an achievement in terms of operating speed of memories also.

In few technologies, shrinking technology is also used, where to save area, the whole memory is shrinked to a fixed percentage. 10% shrinkage gives 10% reduction in both of the lengths and widths. Hence, a total gain of 19% in area in achieved.

The memory named as DP180nano (Dual Port 180nano) because of some methodology used in this design such as port sharing, ultra high dense memory cell and replica structure with no dummy row.

2.5 LOW POWER APPROACHES IN SRAM

This chapter presents the latest development in low-power circuit techniques and methods for static random access memories. The key techniques in power reduction in both active and standby modes are: capacitance reduction by using divided word line structure, selective precharging scheme, pulse word line, ac current reduction by multistage decoding, operating voltage reduction coupled with low power sensing with sense amplifier

Sources of SRAM Power

There are different sources of active and standby (data retention) power present in SRAMs. The active power is the sum of the power consumed by memory core and the periphery components in SRAM. The standby power of an SRAM has a major sources represented as leakage current.

Techniques for low-power operation

In order to significantly reduce the power consumption in SRAMs all contributors to the total power must be targeted. The most efficient techniques used in recent memories are:

- Capacitance reduction of word-lines and the numbers numbers of cell connected to them, data lines, IO lines and decoders
- DC current reduction by using new pulse operation techniques for word lines, periphery circuits and sense amplifiers
- Operating voltage reduction
- Leakage current reduction by utilizing multiple threshold voltage etc.

Capacitance reduction

The largest capacitive elements in a memory are word-line, bitline, and datalines each with numbers of cell connected to them. Therefore, reducing the size of these lines can have significant impact on power consumption reduction. A common technique used often in large memory is called Divided Wordline (DWL) which adopts a two stage hierarchical row decoder structure as shown in figure below

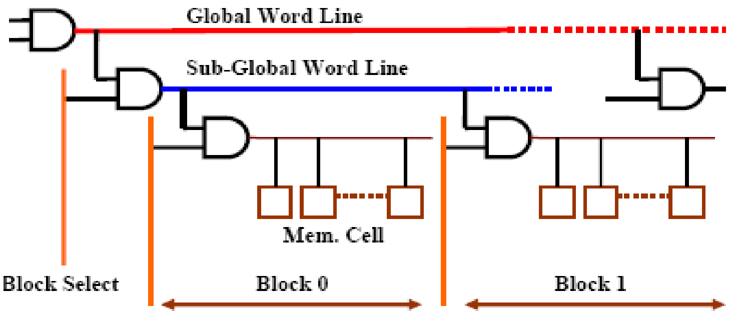


Figure 2.16: Divided Wordline Structure (DWL)

The number of sub-wordlines is connected to one main word line in the dataline direction is generally four, substituting the area of main row decoder with the area of local row decoder. DWL features two-step decoding for selecting one wordline, greatly reducing the capacitance of the address lines to a row decoder and the wordline RCdelay

Selective Precharge off principle

The selective precharge technique was recommended in low-power SRAM to reduce the dynamic power consumption , it can be applied at two different level of hierarchy: at the block level, by precharging off only for the block to be read, and at the bit-line level by precharging off only for the bitlines to be read. Selective precharge off at both the level can be implemented in our low power SRAM compiler. No extra decoder is needed since we use same decoder for the precharging and multiplexer part. Of course, the area of the precharge part is slightly increased by the height of the precharge bus.

2.6 ANALYSIS OF SRAM ARCHITECTURE

Fast low power SRAM's have become a critical component of many VLSI chips. This is especially true for microprocessors, where the on-chip cache sizes are growing with each generation to bridge the increasing divergence in the speeds of the processor and the main memory. Simultaneously, power dissipation has become an important consideration due to both the increased integration and operating speeds, as well as due to the explosive growth of battery operated appliances.

BASIC ARCHITECTURE

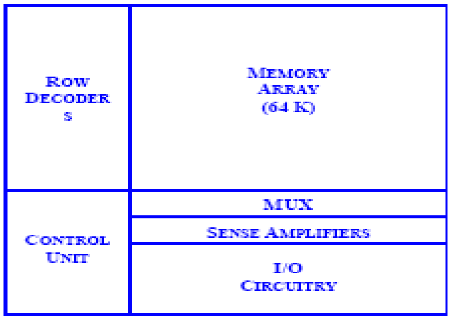


Figure. 2.17 – Basic 256 X 256 SRAM Architecture

In the conventional architecture when the selected word line is high, all the cells connected to the wordline in the row are active. When the word line is high, all the cells connected to the wordline become active -- thus dissipation increases. In the basic architecture, the two major factors contribute to the read access are the bit access time & the word line access time. When the size of the SRAM increases, the number of cells connected to the word line increases--the load is reduced. Therefore, the wordline delay increases because of the increase in the wordline capacitance. These two factors can be improved by reducing the bit line capacitance & the word line capacitance, but this is achieved only after using a different architecture.

SPLIT CORE ARCHITECTURE

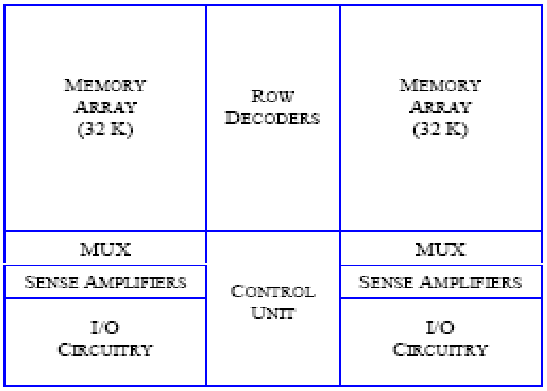


Figure. 2.18 – Split Bank Architecture for 64K SRAM

In this type of architecture, reduction is performed by splitting the matrix in smaller blocks. The resulting architecture is called Split-Core architecture. The reduction in the RC delay is observed because of the split bank, but here too the activation of a wordline activates the entire cell in both of the core areas. So certainly, there is need of a different architecture, which could also provide some advantage in terms of power dissipation.

PAGE TYPE ARCHITECTURE

In split-core, architecture although the bank is split is two parts but the word line activates the cell in both and no gain in power is observed. Thus to reduce the run length of the word lines, a new architecture is analyzed.

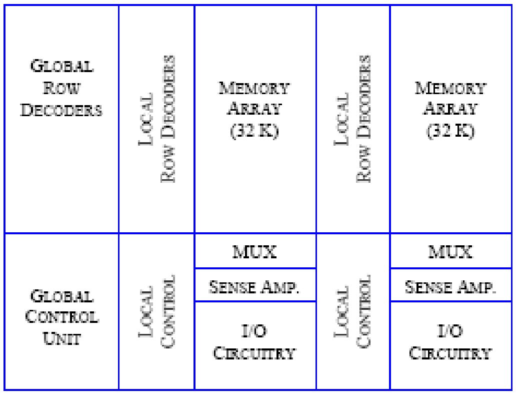


Figure. 2.19 – Page type Architecture for 64K SRAM

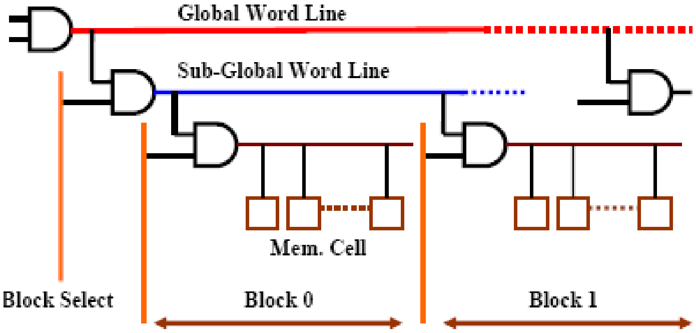


Figure. 2.20 – Page type Architecture with Global and Local Sections

Here the control unit and row decoder sections are divided in global and local sections. This benefits by activation of cells only one page and thus the wordline capacitance is reduced.

BANK ARCHITECTURE

This technique to reduce the run length of the bitlines and divided core structure helps in gain in both of speed and power.

Here the control and the Input/Output sections are divided. So for a selected wordline, the cells of only one bank are activated. Also in case of bitline, the numbers of cells activated are reduced. Thus, a significant improvement is observed in case of wordline cap and the bit line cap.

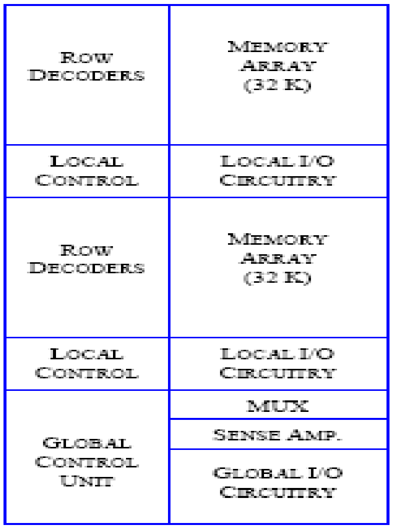


Figure. 2.21 – Bank type Architecture for 64K SRAM

There is also reduction in the power consumption to a very significant value. But in this type of architecture, the area used is more and hence a less dense memory is obtained.

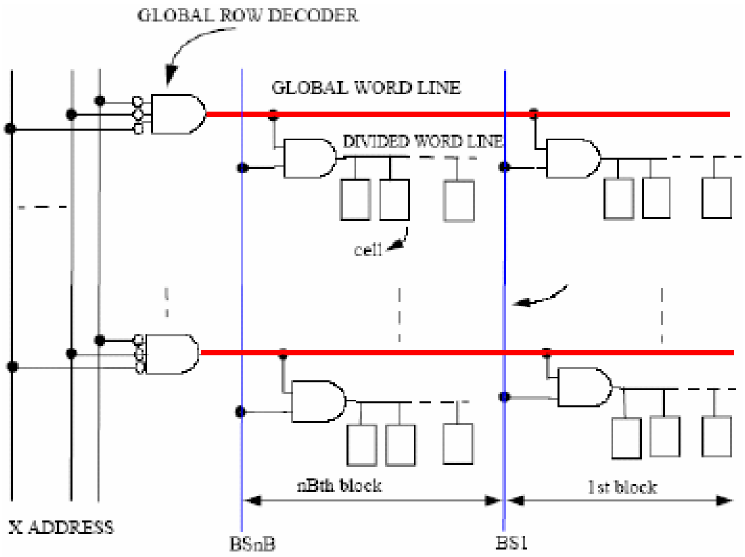


Figure. 2.22 –Bank type Architecture with Global and Local Sections

CHAPTER 3

SYSTEM ANALYSIS & DESIGN

3.1 USER REQUIREMENTS

The design of a large Dualport SRAM into a single chip driver for TFT-LCD represents a challenge due to:

- Increased memory area and transistor density, but with limited impact on device's yield
- Very low power consumption

The increased number LCD rows and column at same frame rate requires a faster access time Likely larger panels such a QVGA (Quarter Video Graphics Array) will use motion pictures as well, making the memory access even more aggressive. Moreover motion picture access may use separate access to the motion and at the same time new issues have to be addressed such as:

- Supply noise effects
- Dense memory cell

Finally, the user requirement is to design a Dual Port 180nano Video **(DP180nano)** SRAM

3.2 DETAIL LIFE CYCLE OF THE PROJECT

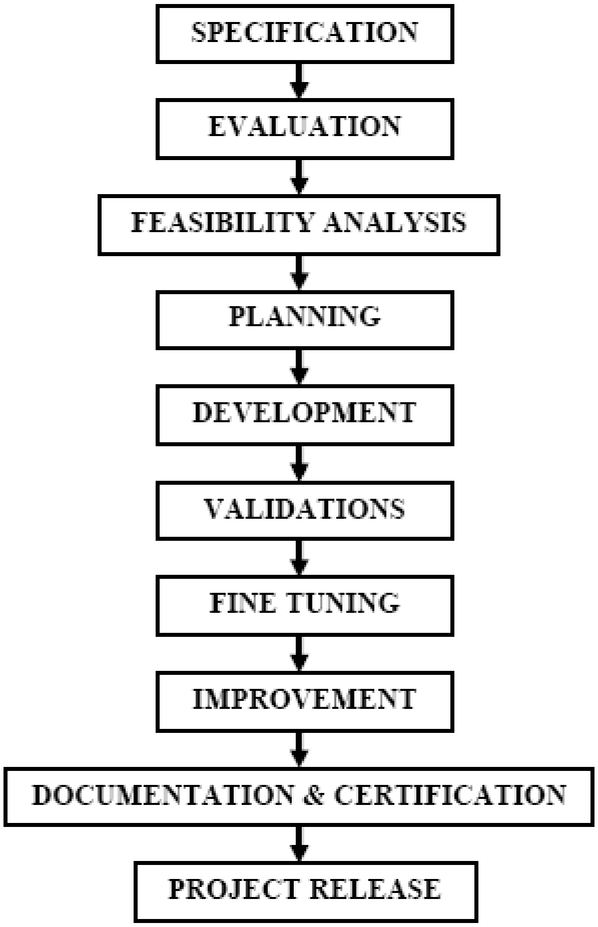


Figure: 3.2: Detail life cycle of the project

CHAPTER 4

IMPLEMENTATION

4.1 INTRODUCTION

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

With the migration toward low supply voltages in low power SRAM designs, threshold and supply voltage fluctuations will begin to have larger impacts on the speed and power specifications of SRAM's. The technique represented based on replica circuits which minimize the effect of operating conditions' variability on the speed and power. Replica memory cells and bitlines are used to create a reference signal whose delay tracks that of the bitlines. This signal is used to generate a sense clock with minimal slack time and control wordline pulse widths to limit bitline swings.

In details, low power circuit designers have been continually pushing down supply voltages to minimize the energy consumption of chips for portable applications. The same trend has also applied to low power SRAM's in the past few years; while the supply voltages are scaling down at a rapid rate, to control sub-threshold leakage, the threshold voltages have not scaled down as fast, which has resulted I corresponding reduction of the gate overdrive for the transistors. With the fluctuations is not expected to decrease in future submicron devices, the delay variability of low power circuits across process corners will increase in the future. The large delay spreads across process corners will necessitate bigger margins in design of the bitline path in an SRAM, and also will result in larger bitline power dissipation and loss of speed. This problem can be mitigated by using a self-timed approach to designing the bitline path, based on delay generators which track the bitline delays across operating conditions. Bitline power can be further minimized but controlling the wordline pulsewidth to be just wide enough to guarantee the minimum bitline swing development. This type of bitline swing control can be achieved by a precise pulse generator that can match the bitline delay. Lowpower SRAM's also use clocked sense amplifier to limit the sense power. These are either the current mirror type or cross-coupled latch type designs. In the former, the sense clock turns on the amplifier sometime before the sensing, to set up the amplifier in high gain region. To reduce power, the amount of time the sense amplifier is on should be minimized. In the latch-type amplifiers, the sense clock starts the amplification, and hence the sense clock needs to track the bitline delay to ensure correct and fast operation.

Fundamentally, the clock path needs to match the data path to ensure fast and low-power operation. The data path starts from the local block select and/or global wordline, and goes through the wordline driver, memory cell, and bitline to the input of the sense amps. The clock path often starts from the local block select or some clock phase, and goes through a buffer chain to generate the sense clock. The delay variations in the former are dominated by the bitline delay since the memory cells are made out of minimum sized devices and are more vulnerable to process variations.

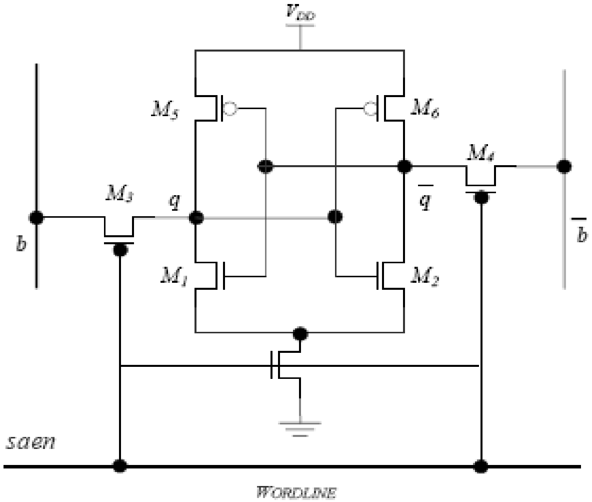


Figure 4.1: Latch-Type Sense Amplifier

Enough delay margins has to be provided to the sense clock path for worst case conditions, which reduces the average case performance.

REPLICA STRUCTURE

The replica delay stage is made up by two types of structure. First, by taking the dummy bitline capacitance value is fraction of main bitline capacitance and the tuning of capacitance will be used for tracking. Second, the replica delay stage is made up of a memory cells called dummy discharge cells (DDC) connected to a dummy bitline whose capacitance is equal to the main bitline capacitance. The number of dummy discharge cells to discharge the dummy bitline is determined by the required bitline swing for proper sensing. For the clocked voltage sense amplifiers we use (Fig. 4), the minimum bitline swing for correct sensing is around a tenth of the supply. An extra column in each memory block is converted into the dummy column by cutting its bitline pair to obtain a segment in which the tuned number of dummy discharge cells is used to discharge dummy bitline (Fig. 4.2).

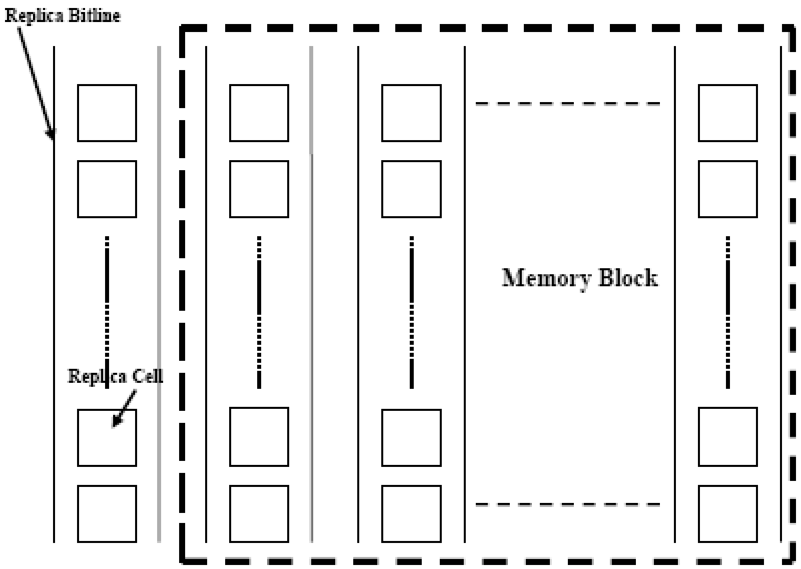


Figure 4.2: Design of Replica Bitline Column

The replica bitline has a similar structure to the main bitlines in terms of the wire and diode parasitic capacitances. The replica memory cell is programmed to always store a zero so that, when activated, it discharges the replica bitline. The delay from the activation of the replica cell to the 50% discharge of the replica bitline tracks that of the main bitline very well. The delays can be made equal by fine tuning of the replica bitline height using simulations. The replica structure takes up only one additional column per block, and hence has very little area overhead. The circuits to control the sense clock and wordline pulsewidths are shown in Fig. 4.3.

The block decoder activates the replica delay cell (*node fwl*). The output of the replica delay cell is fed to a buffer chain to start the local sensing, and is also fed back to the block decoder to reset the block select signal. Since the block select pulse is ANDed with the global wordline signal to generate the local wordline pulse, the latter's pulsewidth is set by the width of block select signal. It is assumed that the block select signal does not arrive earlier than the global wordline. The delay of the buffer chain to drive the sense clock is compensated by activating the replica delay cell with the unbuffered block select signal.

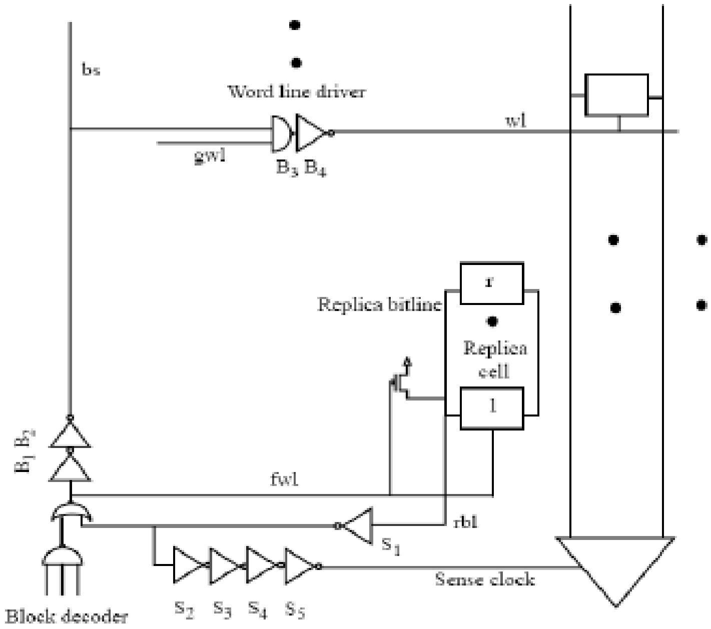


Figure 4.3: Control circuits for sense clock activation and word line pulse control

The delay of the five inverters in the buffer chain, S1–S5, is set to match the delay of the four stages, B1–B4, of the block select to local wordline path (the sense clock needs to be a rising edge). The problem of delay matching has now been pushed from having to match bitline and inverter chain delay to having to match the delay of one inverter chain to a chain of inverters and an AND gate. The latter is easier to tackle, especially since the matching for only one pair of edges needs to be done. A simple heuristic for matching the delay of a rising edge of the five-long chain, S1– S5, to the rising delay of the four-long chain, B1–B4, is to ensure that the sums of falling delays in the two chains are equal, as well as the sum of rising delays.

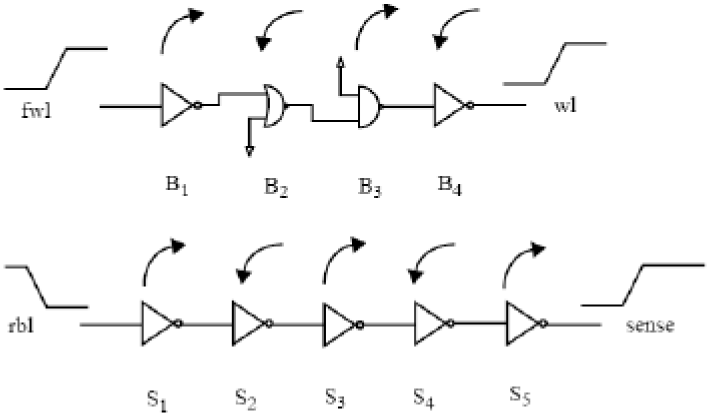


Figure 4.4: Delay matching of two buffer chains

The S chain has three rising delays and two falling delays, while the B chain has two rising and falling delays. This simple sizing technique ensures that the rising and falling delays in the two chains are close to each other, giving good delay tracking between the two chains over all process corners. The delay from *fwl* (see Fig. 4.3) to minimum bitline swing is t_{Bchain} + $t_{bitline}$ and the delay to the sense clock is $t_{Replica}$ + t_{Schain} delay. If $t_{bitline}$ equals $t_{Replica}$ and t_{Bchain} equals t_{Schain}, then the sense clock fires exactly when the minimum bitline swings have developed.

4.2 MEMORY CELL CHARACTERIZATION

Memory cell stability analysis means memory cell has to be tested for its reliability or ruggedness in storing bits. For example any stored bit should not change due to change in any parameters. Major causes which can change the stored bit are

- Noise
- Temperature
- Design of memcell, here due to different sizes of pull down transistors differential nose comes up. In this case one transistor is more affected than other by noise.
- Process variation also affects the stability of memcell. When process varies model parameters of the transistors also varies accordingly.

Before studying the memcell stability, first of all we find out worst case conditions for stability analysis. It is essential to consider the stable SRAM cells when CMOS technology is scaled down to deep-submicron dimensions .Six key parameters which can lead to the change in the stored information that need to be checked in the designing of a memory cell are:

- **Ground Bounce**
- **SNM**
- **Write margin**
- **Discharge rate**
- **Leakage**

GROUND BOUNCE

When the BIT and BITB are 1 (precharge) and the initial voltages at node A and node B are “0” and “1”and then WL is switched ON. At node “A” there will be a voltage

$$\mathbf{Va= Rm2/\ (Rm1+Rm2)*\ Vdd}$$

This voltage is known as Ground Bump from the design point of view it is kept as low as possible. In any case it shouldn’t exceed the threshold voltage of M4.

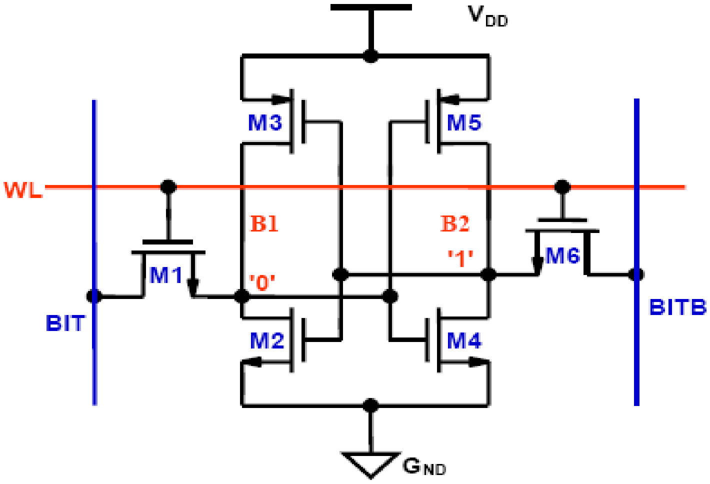


Figure 4.5: Conventional 6T CMOS SRAM Cell

STATIC NOISE MARGIN (SNM)

The cell stability is generally expressed in terms of the static Noise Margin (SNM) which determines the sensitivity of the SRAM to process tolerances and operating conditions. Thus "static noise" in this context indicated disturbances caused by the interference, spurious and transistor mismatch and offsets. For data retention while reading in noidy environment, we need to keep some margin while memcell designing and this should be more than 10% of v_{dd}.

In the figure above, memcell is said to have SNM = V_{noise} if,

V_a(Ground bump) + $V_{noise} < V_t$ (threshold voltage of pulldown transistor)

Till this criterion is satisfied the memcell will not flip. As soon as $V_a + V_{noise}$ becomes more than the threshold voltage of the pulldown transistors then the memcell flips. Thus care should be taken that the transistor sizes are designed properly.

METHODOLOGY FOR MEMCELL STABILITY

- While reading the memcell there'll be a bump on internal nodes. This bump should not be more than V_t, which can cause flipping.
- WL is switching on for long time. Check if the internal node voltage settles of keep on charging. Now the WL is switched off. The memcell shouldn't flip when the WL is switched off.

WORST CASE CONDITIONS FOR SNM

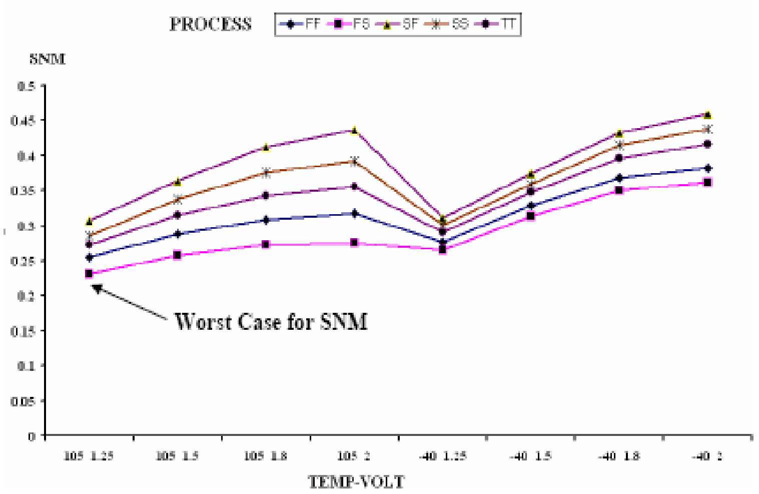


Figure 4.6: Static noise margin at different PVT

As shown in figure the worst case i.e. case at which chances of flipping memcell is max. For SNM worst case would be on fast NMOS, slow PMOS, maximum voltage, and maximum temperature in given combinations.

- Higher Temperature

High temperature will cause voltage coefficient *(V_t)* of the transistor to decrease. So less voltage at internal node B1 will be able to flip the cell.

- High Vdd

When WL will be on higher the Vdd, higher the voltage at node B1, more easily the memcell will flip. So the higher Vdd will help in flipping.

- Max NMOS

The current carrying capacity of max NMOS will be more. So maxN will pull-down the node voltage at B1 and B2 soon.

- Min PMOS

Since the current carrying capacity will be less so it'll not hold 1 at node B2 for longer time

PROCEDURE FOR NOISE ANALYSIS

Noise immunity of the memcell was tested by adding two noise sources as shown in Figure

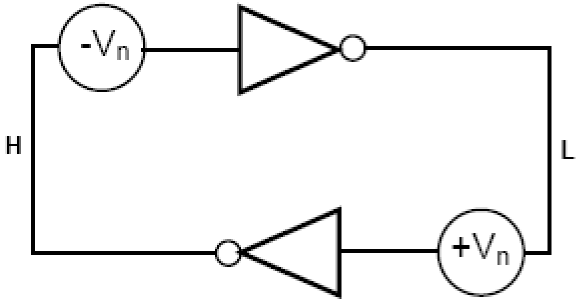


Figure 4.7: Memcell with static noise voltage sources

Stability of memory cell with the noise sources is shown in figure above. Here both oise sources are aiding in flipping the memcell. In above figure at node B1 "0" and at node B2 "1" was being stored. So noise source Vn will aiding '0' and subtracting from '1' causing less immune memcell to be flipped. Static noise is dc disturbance such as offsets and mismatches due to processing and variations in operating conditions. The SNM of the flip-flop is defined as the maximum value of Vn that can be tolerated by the flip-flop before changing states. A SRAM should be designed such that under all conditions some SNM is reserved to cope with dynamic disturbances caused by alpha-particles, crosstalk, voltage supply ripple and thermal noise. I did transient analysis to find the Static Noise Margin with a linearly increasing noise source. The value of noise source at which memcell flips is the static noise margin of memcell

WRITE MARGIN

Write Margin is defined as the maximum voltage required at the Bitline to write a "0".Initially both the BITT and BITF are precharged to Vdd. When the wordline is turned *ON* the voltage at the Bitline (say BITF) corresponding to the node at which "0" has to be written is slowly brought down and the voltage at the node B is checked. The voltage at Bitline is noted for which the voltage at the node is 10% of the wordline voltage. This is called the Write Margin of the Memcell.

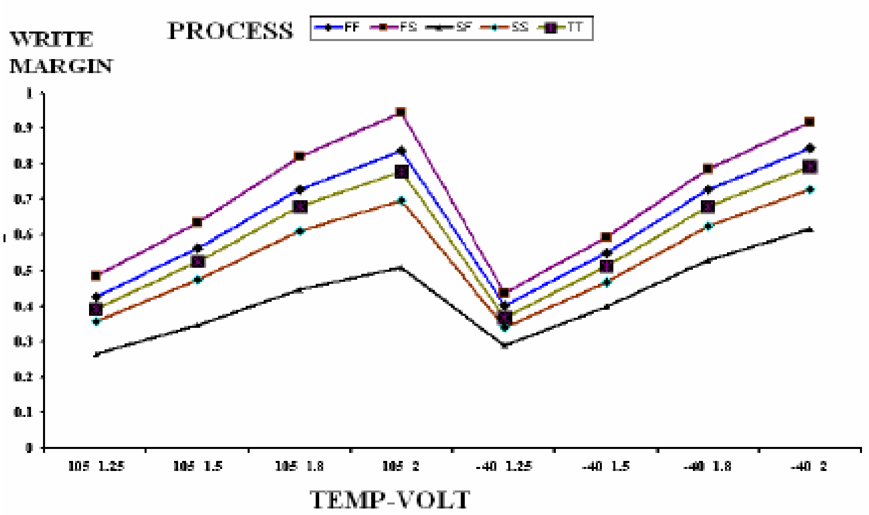


Figure 4.8: Write Margin at different PVT

The worst write margin of the memcell is on slow NMOS and fast PMOS, the maximum bit line voltage to writing the value with respect to wordline on.

DISCHARGE RATE

The rate at which the Bitline discharges is called the Discharge Rate. There are two points to be taken care while calculating Discharge Rate:

- Additional load on Bitline from
  - I/O Cell
  - Strap Cell
  - Dummy Row Cell
- Domination of fringe capacitance for small height memory. As the technology changes the height of the memory cell changes, thus the capacitance per memory cell also changes. Lesser the height, lower is the capacitance. But effectively the fringe capacitance remains the same and thus it dominates.

Discharge Rate is calculated by keeping the wordline as "ON" and BITT and BITF at VDD. The discharge rate is a prime parameter to decide the rate at which the bitline discharges directly affect the access time of memory. Generally we are looking for the high discharge rate keeping improvement in other parameter.

LEAKAGE

We generally assume that in "*OFF*" condition MOS transistors have impedance between its drain to source and thus there is no flow of current. But, in practical circuits there exists very high impedance between the source and drain in the "*OFF*" state. Thus, leakage in this context is calculated as the current flowing through the access transistor when the wordline is "*OFF*" and the bitlines are precharged. This is the case to access the memcell in an array where the cell access through bitline pair will contribute the current from the active memcell as well as from the current through the inactive memcells. The leakage should be low to make the discharging of bit line faster. The higher the leakage will create the worst case for sense amplifier for Vdiff creation.

4.3 COST AND BENEFIT ANALYSIS

Whenever we are talking about the cost analysis of integrated circuits then the three main points for cost analysis would be considered i.e. area, power and speed but in case of memory the fourth parameters which contribute cost is robustness. The methodologies have been mentioned in methodologies section used to make the product, cost and benefit efficient. The port sharing methodology, ultra high dance memory cell and the replica structure with no dummy rows made the system area efficient. Different methods used to reduce the bitline swing like page type architecture, pulsed word line, isolation of sense amplifier from bitlines, replica structure, selective recharging, divided word line structure etc. which contributes the cost benefits in terms of power.

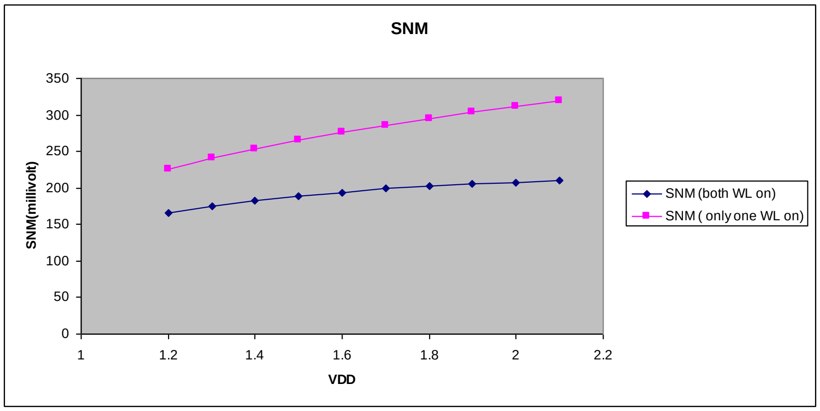
CHAPTER 5

EVALUATION AND TESTING

5.1 MEMCELL CHARACTERIZATION RESULTS

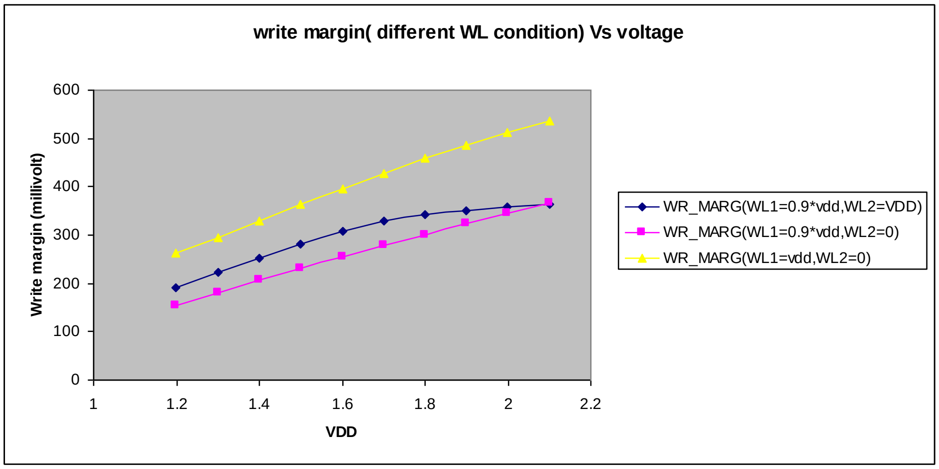
BITCELL STATIC NOISE MARGIN (SNM)

at Temperature 150C for Process FS



WRITE-MARGIN

for Process SF at Temperature 150



5.2 STIMULI GENERATION APPROACH

STIMULI FOR TIMING/MARGINALITY ANALYSIS

The critical path modeling is used to verify the functionality of the SRAM. To save the simulation times but still remains a simple and exact view of the SRAM performance. It will give the exact view of timing and power information. As name suggests the critical path means the path through which signal suffers maximum delay and the path of maximum power consumption. For verification purpose, it is sufficient to perform the read/write operation/cycles to check the functionality of SRAM. During the stimuli writing we have to make sure that the all operations on the corner occurs to check the worst and best timings and power consumption information during read/write cycles.

STIMULI FOR POWER CHARACTERIZATION

A power stimulus is used to put the operation in such a manner that the maximum possible toggling would occur in given time window of each operation to find the worst power consumption. Also take care about the individual signal toggling power contribution in systems power consumption. The stimuli used for power characterization in memory are different from the stimuli used for timing and marginality characterization. Following powers to be calculated on CAD for power characterization of memory: dynamic read power, dynamic write power, standby read power, standby write power, leakage power, and rest of the individual signals power.

STIMULI FOR LEAKAGE CHARACTERIZATION

Leakage power is amount of power consumed when none of input nodes are toggling and all the input signals are in stable state. The stimulus used to put in such a manner that after the operation performed in which maximum signal toggling, then after the slack time when all signals are relaxed, no signal toggling effect is there. This is the situation to calculate the leakage power in memory.

STIMULI FOR PIN CAPACITANCE MEASUREMENT

The stimulus used to calculate the pin capacitance is written in such a manner that the signal toggling occurs to find the capacitance at rising and falling edge and then calculate the average of those. This activity is performed for all the input pins individually to find out their pin capacitance.

TIGHT STIMULI GENERATION APPROACH

A tight stimulus is used to cross check the timings generated during the timing analysis. Here we have to ensure that the all operation to be performed on all available locations keeping the tight timings. It is generally used to check the timings and predefined operations to be performed acceptably in tense situations.

5.3 LIMITATION OF THE SYSTEM

A well known technique used to improve power efficiency is to partition a memory array in to smaller memory blocks so that only the addressed block is activated. At the architecture level we are using page type architecture, which is beneficial for low power, but area is a limited by this approach. For each page we are using separate IO blocks. As our main emphasis on area constraint so page type architecture is a limitation of this project.

CHAPTER6

EVALUATION WORKS

6.1 ELDO OPTIMIZER

The Eldo optimizer is a general-purpose electrical circuit optimization program. The optimizer will calculate the value of parameters (the optimization variables) in the circuit such that the behavior or the characteristics of the circuit conform as close as possible to the specifications. The optimizer can achieve a simultaneous improvement in AC, DC, Transient domain, Steady-State and Modulated Steady-State analyses.

The designer specifies the design objectives and the optimizer will adjust the component parameters of the circuit (such as resistor or capacitor values, the β value of a transistor, widths and lengths of a MOSFET) in order to meet a specified electrical performance. Optimization can be applied to:

- Circuit parameters
- Model parameters
- Element parameters
- Device lengths, widths, areas, and peripheries.

The parameter values must conform to the manufacturing limits, process limits, or discrete device values. To achieve this, restrictions can be specified on the design parameters. Various constraints (or inter-relations) can be specified, for example, circuit parameters must be non-negative, or must not violate upper boundaries. In addition, more complicated constraints can be specified, for example, how components physically interact with each other to produce non-linear relations.

The process of identifying the *objective*, *variables*, and *constraints* for a specific problem is known as *modeling*. The construction of an appropriate model is the first step (sometimes the most important) in the optimization process. If the model is too simplistic, it will not generate useful insights into the practical problems.

6.1.1 PROBLEM STATEMENT

In order to use the Eldo optimizer, the designer must provide the following information:

- The *nominal circuit*, identical to the circuit provided for simulation in the netlist format. The user must have a working netlist.
- The *design variables*, the designer must specify those parameters that may be optimized by the optimizer in its search for an optimal solution. The designer's selection of variables is accomplished by making minor modifications to the working netlist, using the **.PARAMOPT** command.
- The *design objectives* may be any quantity represented by a real value, a combination of quantities that are generated by multiple sweep commands, or multiple increment step on circuit parameters.

The information that the user has to provide is referred to as the *problem statement*. The designer provides a model for the optimization problem; this may be the minimization or maximization of functions (extracted measures) subject to the *constraints* on its variable.

For example, the **.PARAMOPT** command is used to specify the length `l` and the width `w` of a MOS transistor:

```
.PARAMOPT
+ l = (10u, 2u, 100u)
+ w = (60u, 2u, 200u)
```

Exploiting the results of optimization

After the optimization algorithm has been applied, the designer must be able to recognize whether it has succeeded in its task of finding a solution. This can be accomplished by analyzing the results of the optimization.

- There are optimality conditions or the final diagnostics for checking that the current set of variables is a solution of the problem. If the optimal conditions are not satisfied, they may give useful information on how the current estimate of the solution can be improved.

- There are also the values of the extracted measures printed during the optimization at the end of the process. This represents the *post-optimization* analysis.

The designer will find this kind of information within the output results of Eldo (*.chi* file) and the Eldo optimizer (*.otm* file). The script `viewotm` can be invoked to extract the desired results. This script can be invoked from the command line as follows:

```
            viewotm -f <circuit name>.otm <other arguments>
```

6.1.2 OPTIMIZATION COMMANDS

The following are commands that can be used for optimization:

- .EXTRACT
- .OBJECTIVE
- .OPTIMIZE
- .PARAMOPT

EXTRACT

```
•  OBJECTIVE_INFO:=
    GOAL=MINIMIZE|MAXIMIZE     [WEIGHT=RVALUE]
    | GOAL=RVALUE [WEIGHT=RVALUE]
   | EQUAL=RVALUE
   | {LBOUND=RVALUE | UBOUND=RVALUE}
   | LBOUND=RVALUE UBOUND=RVALUE
```

The OBJECTIVE_INFO argument describes the category of the design objective when optimization is required. When this argument defines a MINIMIZE or MAXIMIZE objective, the specified expression will be minimized or maximized. The argument GOAL=RVALUE defines a soft constraint on the measure, while the LBOUND, UBOUND and EQUAL define hard constraints. The specification of the arguments GOAL, EQUAL, LBOUND, and/or UBOUND is mandatory for optimization.

OBJECTIVE

The following is a list of reasons of using the **.OBJECTIVE** command:

- Some implicit rules in the **EXTRACT** construct are not well fitted for optimization. For example, when several **ALTER** blocks are used with optimization, the **.EXTRACT** commands are simply added to each **ALTER** blocks. By default, the **.OBJECTIVE** command has a local scope.
- It is easier to separate the optimization block, the circuit statements, and the plot statements.

Notes

- The additional data such as the goal values, the lower/upper bounds, and the weight numbers are sent to the optimizer when initializing the optimization process. They have constant values during the whole process.
- Excepting the design objectives specified with **GOAL=MINIMIZE|MAXIMIZE**, which play a specific role in the problem statement, the goal values and the bounds have *multi-point* extension. This feature is related to data driven analysis.
- Note that some of the arguments in **OBJECTIVE_INFO** are mutually exclusive. For instance, an objective cannot be specified at the same time as equality and as an inequality. The user may think of as an extracted measure having a unique descriptor that gives its role within the problem.
- In general not all equality and inequality constraints are perceived by designers in the same way. It leads to classify constraints as either *hard* or *soft*.

The term *hard constraints* means refers to the constraints the designer considers as most essential, i.e. they have to be satisfied. *The designer does not want them to take part in any subsequent design trade-off.*

OPTIMIZE

Command syntax

The optimization specification acting on all the analyses specified in the circuit netlist is done using the following command:

.OPTIMIZE
+ [**QUALIFIER**=VALUE {, QUALIFIER=VALUE}]
+ [**PARAM**=LIST_OF_VARIABLES|*]
+ [**RESULTS**=LIST_OF_MEASURES|*]
+ [**OUTER**=LIST_OF_PARAMETERS]

Parameters

- **QUALIFIER**

  The name of the corresponding configuration argument

- **PARAM** = LIST_OF_VARIABLES

  List of comma-separated variables to be tuned, specified with the **PARAMOPT** command.

- **RESULTS** = LIST_OF_MEASURES

  List of comma-separated measures to be optimized, specified with **EXTRACT** commands.

- **OUTER**=LIST_OF_PARAMETERS

  List of comma-separated design parameters specified with the **PARAM** command and used in a **STEP PARAM** command.

Eldo optimizer/SQP arguments

The performance of the solver (the SQP algorithm) is controlled by a number of parameters. For specific situations it is possible to specify non-standard values for some or all of the parameters.

- **MAX_ITER**=IVALUE

  Maximum number of iterations permitted for optimization. Default 1000.

- **MAX_SIMUL**=IVALUE

  Maximum number of circuit simulations allowed. Default 99999.

- **TOL_OPT**=RVALUE

Tolerance on optimality conditions. This parameter specifies the accuracy to which the final iterate to approximate a solution of the problem. **TOL_OPT** can be considered as indicating the level of accuracy desired in the design functions at the solution. Specifying only the value **TOL_OPT** has the same effect as setting separately both the tolerances **TOL_GRAD**=**TOL_OPT** and **TOL_FEAS**=10-2 x **TOL_OPT**. Default value is 10-4.

- **TOL_FEAS**=RVALUE

Tolerance on feasibility conditions. An iterate is said to satisfy the feasibility conditions if FEAS(x) ≤ TOLFEAS × SIZE (x), where SIZE(x) represents a scaling quantity taking into account the norm of the solution vector. Default value is 10-6.

- **TOL_GRAD**=RVALUE

Tolerance on the measure of *criticality* of the current iterate. An iterate is said to be *critical* if OPTIM(x) ≤ TOLGRAD ≤ SIZE (λ) where SIZE(λ) is a scaling factor representing the sensibility of the current solution with respect to changes of problem data. Default value is 10-4.

PARAMOPT

The specification of the optimization variables is realized with an extension of the **.PARAM** command. These variables will be denoted by a vector of real values of dimension N:

$$x = (x(1), x(2), ..., x(N))$$

The vector x is also associated with vectors of the lower and upper bounds denoted by x_l, x_u. An additional vector δ will be associate to the discretization (or resolution) of the variables x.

Command syntax

The design variables must be specified through the **.PARAMOPT** command:

```
    .PARAMOPT VARIABLE_NAME=(
 + [INIT_VALUE,]
 + {LOWER_BOUND | LOWER_PERCENT% },
 + {UPPER_BOUND | UPPER_PERCENT% }
 + [, INCREMENT])
```

Parameters

- `VARIABLE_NAME`

  Name of the design variable(s). This parameter can be one of the following:

PARAMETER_NAME |

P(PARAMETER_NAME) |

E(DEVICE,PARAMETER) |

M(MODEL_NAME,MODEL_PARAMETER) |

- `INITIAL_VALUE`

  Initial value of the design variable. It is optional in some situations.

- `LOWER_BOUND`, `UPPER_BOUND`

  Lower and upper bounds specified for the design variable. Unbounded (or free variables) must be specified using the star character `*`, for example:

```
.PARAMOPT VAR1=(1.0, 0.0, *)
.PARAMOPT VARIABLE_NAME=(INIT_VALUE, *, *)
```

- `LOWER_PERCENT%`, `UPPER_PERCENT%`

  Percentages of the initial value.

```
.PARAMOPT VAR1=(5.0, 10%, 35%),
```

  specifies that the effective lower and upper bounds are $x_l^{(1)}$ =5.0 × (1 – 0.1) and $x_u^{(1)}$ = 5.0 × (1 + 0.35).

- `INCREMENT`

  Specifies a discretization for the final value of the design parameter (“Discretization of design variables”). Optional.

For example, to optimize the length **`L`** and width **`W`** of a MOS transistor, the parameter

```
.PARAMOPT
          + L = (10U, 2U, 100U, 0.01U)
          + W = (60U, 2U, 200U, 0.01U)
```

Here, the optimization is initiated with a length of 10µm and a width of 60µm the length is allowed to change between 2µm and 100µm by steps that are multiples of 0.01µm. Similarly the width can to take values between 2µm and 200µm that differ from the initial value (60µm) by a multiple of 0.01µm.

6.1.3 Discretization of design variables

Figure 6.1-1 illustrates such a situation. The continuous box represents the feasible domain specified by the upper and lower bounds, and the black bullets are the feasible points where the final parameters are allowed to lie.

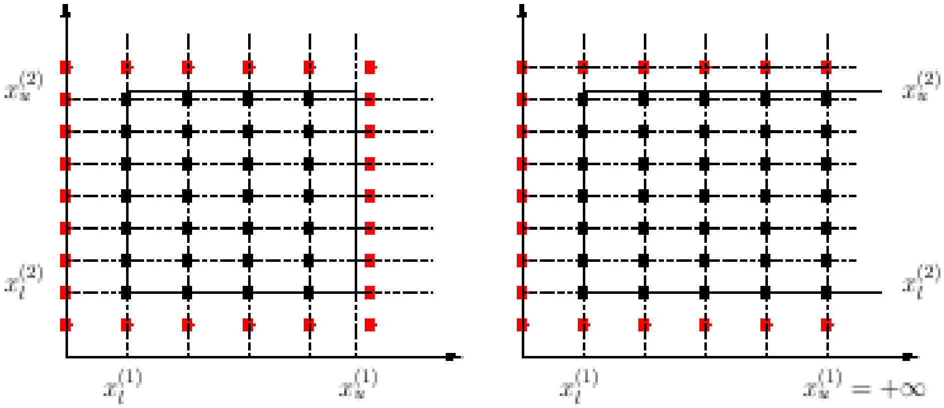


Figure 6.1-1. Discretization of final parameters

For example, when the lower bound is a finite number, the set of discretized points are as follows:

$$\{x_1^{(i),}, x_1^{(i)} + \delta, x_1^{(i)} + 2\delta, \ldots..\}$$

Where $\delta_i > 0$ is the given increment. When the lower bound is infinite $(x_l^{(i)} = \infty)$, the grid is started from the upper bound if this one is finite. The set of discretized points is then:

$$\{x_u^{(i)}, x_u^{(i)} - \delta_i , x_u^{(i)} - 2\delta_I , \ldots\ldots\}$$

When a design parameter is unbounded $(x_l^{(i)} = -\infty$ and $x_u^{(i)} = \infty)$, the requirement of discretization is not considered.

6.1.4 OPTIMIZATION OPTIONS

The following are a list of options that are used for optimization:

- **OPSELDO_ABSTRACT**

  Generates a summary table of simulations containing parameter and extract values for each run. This option has no effect on the *.otm* file, only the *.chi* file is affected.

- **OPSELDO_DETAIL**

  If this option is set to **NONE**, only the last run and the nominal run will be stored in the generated files (*.wdb, .cou, .aex*) and no other simulation information will be displayed on the standard output. When set to **ALL**, simulation information for all runs will be stored. Default is **NONE**

- **OPSELDO_NETLIST**

  Generates a netlist modified from the original input file, which contains the optimized parameter values but also every parameter set under #ifdef statements.

- **OPSELDO_OUTER**

  Allows a reverse behavior of optimization and sweep simulations(**.TEMP**, **.DATA**, or **.STEP**). A full optimization will be performed for each set of sweep parameters.

6.1.5 ROBUST OPTIMIZATION USING CORNERS

This example illustrates the combination of optimization and **.ALTER** commands. For this purpose a new architecture based on the concepts of multi-context of simulation and multinetlist optimization has been developed.

Problem definition

Nominal optimization focuses on finding the *best* design parameters for one *nominal* operating condition of the circuit. Typically the power supply, the ambient temperature, and the process technology are given their nominal value, and the best set of design parameters is found by the optimizer, given its targets. If the circuit has to operate under a variety of operating conditions, nothing guarantees that this will still be the case, if the design parameters are simply set to these optimal nominal values. Maybe if the power supply level is slightly changed, the circuit will fail.

'Robust optimization' focuses on finding the 'best' set of parameters that fulfill the specifications across a certain range of operating conditions. For example, robust optimization would be performed on a circuit that must operate between 1.7 and 1.9V, a temperature range of -25C to 100C, and accommodate variations in the process (defined by the 'corner' device model libraries). In this case the optimization targets might be upper or lower bounds on certain characteristics (for example the DC consumption has to be 'lower than 50∝A, in all operating conditions). Targets can be set as targets for the average value of a given specification.

Operating conditions are conveniently defined with **.ALTER** commands in Eldo. In each **.ALTER** command, a specific combination of parameters defining the operating conditions (typically the power supply level and the temperature) and the corner device model library, can be defined.

The optimization commands (design variables definition, design objective definitions, and the optimize command) from the main netlist are then interpreted to span the main netlist conditions *and* the various combinations defined through the **.ALTER** sections. Obviously these types of optimizations are usually more costly than simple nominal optimizations. Eldo can distribute the necessary simulation on multi-processor machines, thus potentially accelerating the process.

Smooth and non-smooth problems

A difficult situation arises when the functions corresponding to the design objectives do not have continuous derivatives or are not continuous at all. In this case, methods for smooth problems will encounter difficulties.

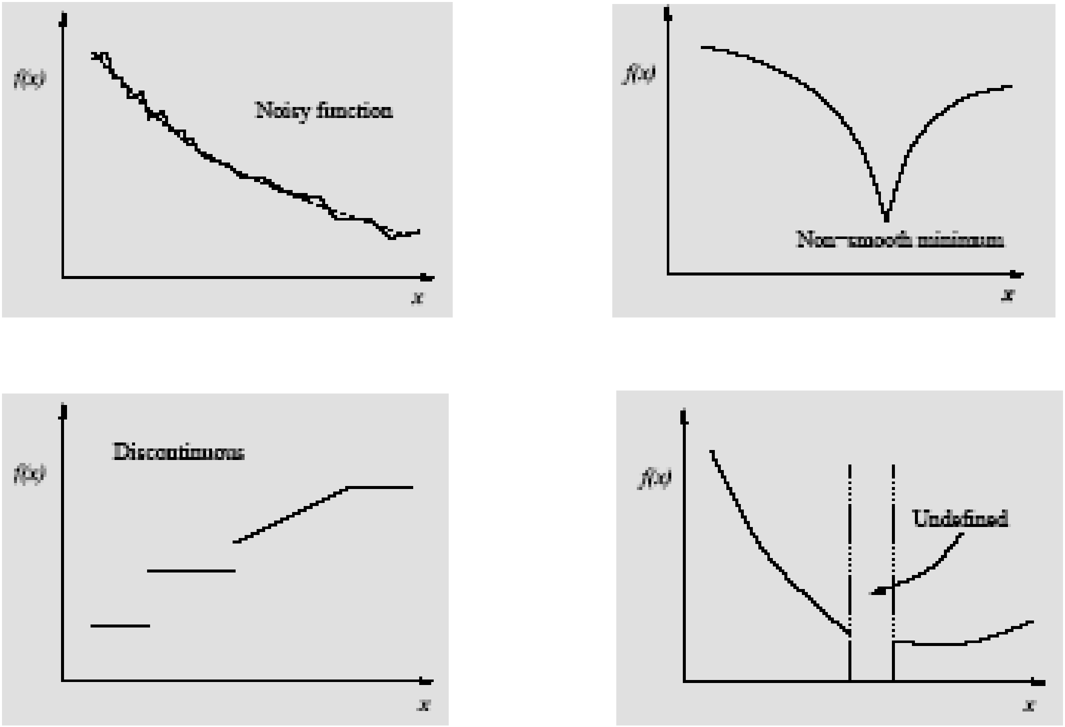


Figure 6.1-2. Examples of non smooth problems

It is assumed in the optimization problem, that the functions are continuous with continuous derivatives on some domain (or sufficiently smooth). Figure 6.1-2 illustrates some of these difficulties:

- The first case represents a “noisy function” with an overall “trend” plotted with dashed lines. It is related to the effect of features such as adaptive algorithms and stopping tests in iterative methods inside the simulation. The Eldo optimizer will fail to make any progress because local descent directions may point uphill.
- The second example is related to the use of functions such as ABS(.), MIN(.) or MAX(.) that are not differentiable in the common sense.
- In general, the last two cases will lead to serious difficulties. The “discontinuous” case probably involves functions that are not numerical in nature, or the discontinuities are the result of features such as table look-ups and switches.

Global and Local Optimization

The fastest optimization algorithms only search for a *local* solution at a point at which the objective function is smaller than all other feasible points in its vicinity. They don't always find the best of all such minima that is the *global* solution (see Figure 6.1-3). The reason for this is that practical methods for finding global optima are too expensive in all but the most specialized cases.

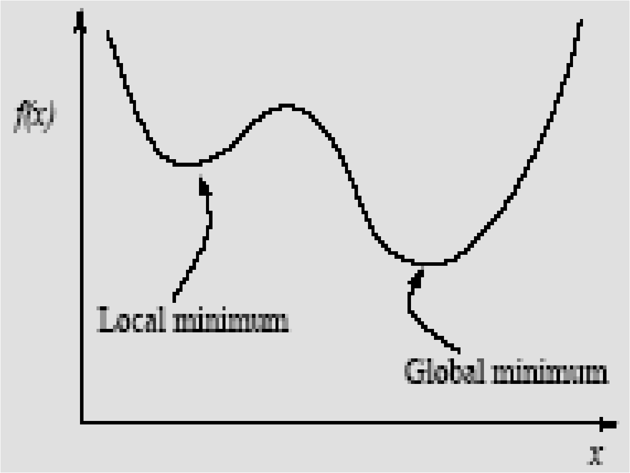


Figure 6.1-3. Different types of minima

General non-linear problems may possess local solutions that are not global solutions. Global solutions are highly desirable, however, they are usually difficult to identify and even more difficult to locate. Unless very strong assumptions are made about the functions that define the optimization problem in question, characterizations of global minima are almost impossible, so even if one is found the user may never know. The algorithm implemented in the Eldo optimizer on Newton iterations.

Role of the weight numbers

The numbers $\mu_m^{(i)}$ are *positive* weight values attached to each design objective. For instance, the weight $\mu_m^{(i)}$ can be thought of as quantifying the user's desire to make $f_m^{(i)}(x)$ small. The user would take $\mu_m^{(i)}$ large if the user wants $f_m^{(i)}(x)$ to be small. The ratio $\mu_m^{(i)} / \mu_m^{(j)}$ can be interpreted as the relative weight of the *i*th objective compared to the *j*th objective.

These remarks can help users that want to change the weights in order to get the lower values of a chosen objective, for example the *k*th. To find an optimal point which trades off the *k*th objective, users can increase the weight on the *k*th objective.

Eldo optimizer/SQP Method

The Eldo optimizer/SQP optimizer, which is based on a *Sequential Quadratic Programming* method (SQP), using an augmented Lagrangian approach (AL) for solving the sequence of quadratic problems (QP) involved in the SQP iterations. It is efficient for solving small and medium scale problems belonging to the following classes:

- unconstrained and bound constrained problems (using only the statements `**GOAL**=**MINIMIZE**|**MAXIMIZE**` and `**GOAL**=rvalue)`
- systems of non-linear equations (using only the statement `**EQUAL**=rvalue)`
- general (equality and inequality) constrained problems.

Before continuing our presentation, some useful definition and mathematical notations are given. The design variables are denoted by:

x =(x(1), x(2), ..., x(N))

a vector of real numbers of dimension *N*. Therefore the space of variables is denoted by $\Re^N$. *v* The scalar product is denoted by:

$\langle \boldsymbol{u} \mid \boldsymbol{v} \rangle = \sum_i \boldsymbol{u}^{(i)} \boldsymbol{v}^{(i)}$ onto the space $\Re^N$, and its associated norm is ||u||.

A simplified structure of the *k*th iteration can be depicted as:

- Step Computation: determine a direction of search s_k (by solving a tangent quadratic subproblem $(QP)_k$)
- Step Assessment: find $\alpha_k > 0$ in order to minimize a chosen merit function $x \to \phi(x)$ along the line $\alpha \to x_k + \alpha s_k$,
- Update: $x_{k+1} = x_k + \alpha_k s_k$.

This approach is referred to as a *descent method* since the search direction sk satisfies a descent property:

$$\langle s_k \mid \nabla\phi_k \rangle < 0$$

where the vector $\nabla\phi_k$ represents the gradient of the chosen merit function. The role of the line search algorithm is to *dampen* the displacement s_k. However, the most important feature of this algorithmic process is:

Role of tolerances in Eldo optimizer

This section gives some indications on the role of the arguments **TOL_GRAD**, **TOL_FEAS**, and **TOL_OPT**.

Suppose a value of x that is a local minimizer of f(x) in the interval $a \leq x \leq b$ is to be obtained:

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{Subject to} & a \leq x \leq b \end{array}$$

or using a SPICE formulation:

```
.OPTIMIZE
* Minimize Statement
.OBJECTIVE EXTRACT_INFO LABEL=F
+ {$MACRO|FUNCTION}
+ GOAL=MINIMIZE
* Design variable specification
.PARAMOPT X=(X0, A, B)
```

The optimality conditions OPTIM(x) in the definition of the **TOL_GRAD** argument are:

$$\mathrm{OPTIM}(x) \qquad \begin{array}{l} f'(x) \text{ if } a < x < b \\ \min(f'(x),0) \text{ if } x = a \\ \max(f'(x),0) \text{ if } x = b \end{array}$$

Consider the first case in Figure 6.1-4. The minimum is the point $x = b$. The blue vector represents the derivative f'(x) at point b (this is the opposite of the steepest descent direction -f'(x) at point b). The derivative f'(x) is negative, then is the number max(f'(x),0)=0. When OPTIM(x) is zero or very small this indicates that the point x is an optimum. The point x is optimal when the absolute value of OPTIM(x) is less than **TOL_GRAD**.

Consider the second case in Figure 6.1-4. Point y is an unconstrained minimizer of f, since it lies strictly in the interval [a,b]. The optimality condition is then OPTIM(y)=f'(y) which is the slope of f at point y. OPTIM(y) equals zero. Note that the previous definition of the optimality conditions are related only to bound constrained minimization problems.

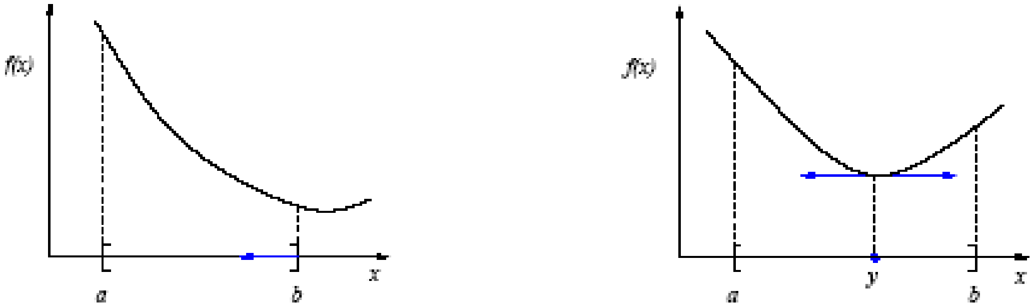


Figure 6.1-4. Illustration of the optimality conditions

For instance, suppose that a constraint function $c^{(i)}(x)$ is computed for some relevant *x* and if the first six digits are known to be correct. A constraint should be considered as active at its upper bound (or lower bound), if the magnitude of the difference between the values $c^{(i)}(x)$ and c_u (i) (or $c_l^{(i)}$ respectively) is less than some tolerance of order 1.0 × 10-6. This tolerance, δ, specifies how accurately the constraints should be satisfied. It defines the maximum absolute violation in non-linear constraints at a feasible point.

A constraint is considered satisfied if its violation does not exceed the tolerance δ. The feasible region for the constraints $c_{l_l}^{(i)} \le c_I^{(i)}(x) \le c_u^{(i)}$ is shown in Figure 6.1-5 .

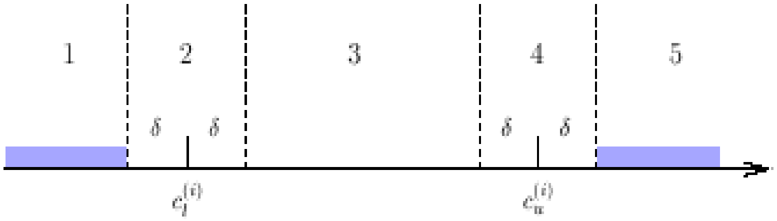


Figure 6.1-5. Illustration of the constraints $c_{l_l}^{(i)} \le c_I^{(i)}(x) \le c_u^{(i)}$

The constraints are considered satisfied if $c^{(i)}(x)$ lies in the region 2, 3, or 4, and inactive if $c^{(i)}(x)$ lies in region 3. The constraint $c_l^{(i)} \le c^{(i)}(x)$ is considered active in region 2, and violated in region 1. Similarly, $c^{(i)}(x) \le c_u^{(i)}$ is active in region 4, and violated in region 5. For equality constraints $c_l^{(i)} = c_u^{(i)}$, regions 2 and 4 are the same, and region 3 is empty. The default value is appropriate when the constraints contain data about the accuracy.

Note that specifying an appropriate tolerance on feasibility **TOL_FEAS** may lead to several savings, by allowing the optimization procedure to terminate when the difference between function values along the search direction becomes as small as the absolute error in the values.

Computation of finite-difference derivatives

The finite differences in the Eldo optimizer are used. Finite differencing is an approach to calculate the approximate derivatives whose motivation comes from Taylor's theorem. Like many software packages, the Eldo optimizer performs automatic calculation of finite differences whenever the simulator is unable to supply the code to compute exact derivatives.

A popular formula for approximating the partial derivative $\partial F / \partial x$ at a given point is the *forward differences* or *one-sided differences.* The parameter denoted by hf is used, this controls the interval used to estimate the gradients of the function F by forward differences:

$$\frac{\partial F}{\partial x} \approx \frac{F(x+h_f)-F(x)}{h_f}$$

One-sided difference estimates are used to ensure feasibility with respect to an upper or lower bound on *x*. If *x* is close to an upper bound, the trial intervals will be negative. The final interval is always positive.

- An approximation to the derivative of F can be obtained by evaluating the function F at N + 1 points and performing some elementary arithmetic.
- The resulting gradient estimates should be accurate to $O(h_f)$ unless the functions are badly scaled.

6.2 MULTI - THREADING

6.2.1 MTHREAD

Activates multi-threading for a single DC or TRAN simulation. Eldo will share computer resources on a multi-processor machine. Eldo will make use of all the possible CPUs on the machine. It will share the work between the different CPUs in order to speed-up simulation. Note that the CPUs should not already be in use, otherwise simulation will be slower. Statistics, generated at the end of simulation, show how many CPUs have been used for the current simulation. This number will also be printed out at the beginning of the TRAN simulation.

Eldo will make use of the multi-thread capability of the machine it is running on only if all the following conditions are met:

- The MTHREAD flag has been set (via option or at invocation)
- The machine is a multi-cpu machine
- The circuit contains devices which are thread-safe
- The circuit contains more than 100 devices.

If the circuit does not contain any thread-safe device, MTHREAD will be ignored. Thread safe devices are BJT, diode, and MOSFET exclusively. A few models of these categories are not thread-safe, and are therefore handled in a single processor. Eldo will notify when it cannot apply thread optimization on models. Thread-safe and non-thread-safe models can be used in the same circuit.

If a circuit contains less than 100 devices, MTHREAD is disabled because multi-threading might slow down the simulation when there are not enough devices to distribute between the processors.

.option MTHREAD

6.2.2 USETHREAD=VAL

Activates multi - threading for a single DC or TRAN simulation. Eldo will share computer resources on a multi-processor machine. Eldo will make use of the number of CPUs as specified with this option. The number specified can exceed the number of CPUs available, but this is not recommended. This option is equivalent to the Eldo -usethread # command line flag.

```
.option usethread = val
```

6.2.3 .MPRUN

Multi-Processor Simulation

```
.MPRUN [ALL|HOST={host[(nbjobs)]}|FILE=filename}][NBLICENSES=val]
 +[MAX_NBJOBS=val][CLEAN=YES|NO][QUEUE=YES|NO][SETENV=YES|NO]
 + [VIEW_COMMAND=YES|NO] [CHECK_DELAY=val] [INIT_FILE=filename]
 + [DEFAULT_INIT=YES|NO] [NETWORK_DIR=directory]
 + [CD_WORKDIR=YES|NO] [LOGFILE=YES|NO]
 + [SHELL_SYNTAX=(source_cmd, setenv_cmd, setenv_sep, init_anacad_ext)]
 + [USE_LOCAL_HOST=YES|NO] [FLAT=YES|NO]
 + [FILE_PREFIX=(name1,name2,...,nameX)]
 + [USE_SSH=YES|NO] [SSH_OPTIONS="{<options>}"]
```

This command is used to run multi-threading simulations on one multi-processor machine, or on many machines. .ALTER, .MC, .TEMP, .STEP, .DATA and .OPTIMIZE are distributed by this command.

The child processes are launched through a rsh call which by default inherits the environment variables used in .INCLUDE or .LIB statements. Temporary results are stored in subdirectories named

```
<NETLIST_NAME>.part<X>
```

where X is an incrementing counter. By default and unless they are in use temporary directories are removed once the simulation is complete.

Notes

1. In all cases, Eldo will warn you if a host cannot be used. This will happen when:

- You don t have the permissions to see the machine and/or to write in the working directory. This error must be fixed by the system administrator before the .MPRUN command can be used.

- Too many hosts are given in the command line compared to the number of tasks that can be distributed.

2. To use the .MPRUN command with both Linux and Sun platforms, you must provide an initialization script (INIT_FILE) and use the DEFAULT_INIT=NO keyword.

3. Ensure there are enough runs (more than three) from .STEP, .ALTER, and other commands to activate distribution of jobs on different processors.

The first and last processes are reserved for the main processor or CPU (say, processor #1) and the rest of the processes are distributed to any of the available processors including the first one.

In simulations with only a few processes, the .MPRUN command is not practical. Consider a temperature sweep with a command .TEMP -30 30 90. This case has only three temperatures which are not enough to trigger multi-processor runs since the algorithm runs the first and last process (in this case .TEMP -30 and .TEMP 90) in single processor mode. The rest of the processes are distributed to other processors or machines. In this case, only one processor will be used and the simulation will not take advantage of the multi-processor feature.

4. The netlist must be in a shared directory (a place visible from the other machines on the network).

5. If using local installations of Eldo, the installation patches (not the binaries) must be strictly identical on all machines.

6. Users need to be able to rlogin to other machines without supplying password. If a user needs a password to perform an rlogin to other servers, when Eldo attempts to rlogin to other systems to launch tasks it will fail because the other machines require passwords and Eldo cannot supply them.

7. The priority of distriuted commands is as follows (high to low):

.Alter, .Temp,.Step/.MC,.Data/.optimize. if you have ten runs for .Step but only two runs for .Alter only two then eldo will distribute the ten runs (from .Step) on the available machines.

Parameters

- ALL This is the default. Keyword specifies Eldo to run the simulation on all the processors of the machine. Eldo will find the number of processors of the machine and distribute the tasks between them. If the machine has one processor, Eldo will run the simulation normally without taking this command into consideration.
- HOST={host1[(nbproc1)] host2[(nbproc2)]... hostN[(nbprocN)]}

  Keyword specifies Eldo to run the simulation on the list of machines: host1, host2, hostN (commas are optional). Eldo will distribute the tasks on the list of machines specified. nbjobs is an optional parameter that explicitly tells Eldo the maximum number of jobs that can be submitted on this machine. On multi-processor stations, this number should be the number of processors.
- FILE=filename Specifies the name of a file which contains a list of machine names (with a number of processors if needed). The file can have any extension or no extension at all. The first line of the file is read. Example file contents:

  ```
  pluton kebra(3)
  morkai(2)
  nao cochise
  ```
- NBLICENSES=val Specifies the maximum number of licenses that this job can use. It must be greater than 1 to be taken into account, since the parent process always takes its own license.
- MAX_NBJOBS=val Specifies the maximum number of jobs that can be submitted for all machines.
- CLEAN[=YES|NO] Default value is YES. This specifies Eldo to remove temporary files created in any child process subdirectories, together with removing the subdirectories themselves. If keyword is set to NO, the temporary files are not removed.
- QUEUE[=YES|NO] Instructs the system to wait for the release of a license if one is not immediately available. A consequence is that the parent process will hang until its child process has finished. Default value is NO.

- LOGFILE[=YES|NO] This controls the redirection of the standard output of sub-processes. The default value is YES, which means that Eldo dumps the standard output in <NETLIST_NAME>.log in the temporary directories. "
- USE_LOCAL_HOST[=YES|NO] Setting this option to NO tells Eldo that it cannot use the local host to perform some simulations. Default value is YES (the local host is the machine on which the main process has been launched).
- FILE_PREFIX=(name1,name2,...,nameX) In case of splitting .ALTER statements, a user may want to rename the output files of each run. If a flag is given after .ALTER, it is used as the name. Output names can also be redefined with this FILE_PREFIX option. For example if a netlist contains two .ALTER statements and the following .MPRUN option FILE_PREFIX=(first,second, third), output files for each run will be:
  1. for the netlist before .ALTER statements: *first.chi, first.wdb*, etc.
  2. for the netlist after the first .ALTER statement: *second.chi, second.wdb*
  3. for the netlist after the second .ALTER statement: *third.chi, third.wdb*

Examples

```
.MPRUN HOST=host1, host2 NBLICENSES=2 QUEUE=YES
```

Specifies Eldo to run the simulation on both the host1 and host2 machines. A maximum of two licenses can be used by this simulation. If a license is not immediately available, the system will wait for the release of a license.

```
 .MPRUN HOST=host3(2) host4
+ INIT_FILE=script.shell DEFAULT_INIT=NO
```

Specifies Eldo to run the simulation on both the host3 and host4 machines, also with two processors of host3 specified as running the simulation. A script file *script.shell* will be sourced before running child processes. This script will replace any other default initialization file.

.MPRUN and external dispatchers

```
 .MPRUN DISPATCHER= [LSF |
+ (dispatcher_name, install_check_cmd, submission_cmd]
+ [REMOVE_QUOTE=YES|NO] [DISPATCHER_OPTIONS=options]
+ [NBLICENSES=val] [MAX_NBJOBS=val] [CLEAN=YES|NO]
+ [QUEUE] [SETENV] [VIEW_COMMAND] [CHECK_DELAY=val]
+ [INIT_FILE=filename [DEFAULT_INIT=YES|NO]]
+ [USE_LOCAL_HOST=YES|NO] [FLAT=YES|NO]
+ [FILE_PREFIX=(name1,name2,...,nameX)]
```

```
.MPRUN
+ DISPATCHER_TEMPLATE=command_line
+ [USE_LOCAL_HOST=YES|NO] [FLAT=YES|NO]
+ [FILE_PREFIX=(name1,name2,...,nameX)]
```

A dispatcher is software which shares and manages computer resources across a network. Instead of a basic remote shell command, the dispatcher uses various criteria (memory usage, CPU charge) to determine which machine is suitable to run the simulation on. The second syntax above is useful for .ALTER dispatching only.

- DISPATCHER=LSF

Specifies Eldo to use LSF as external dispatcher. This syntax is a shortcut to DISPATCHER=("LSF","bsub -V","bsub").

- DISPATCHER=(dispatcher_name,install_check_cmd, submission_cmd)

dispatcher_name

name of the software (required for print purpose only).

install_check_cmd

a command that can prove the software is accessible.

submission_cmd

the command that submits a job to the engine.

- REMOVE_QUOTE[=YES|NO]

This tells Eldo how to manage double quotes inside the DISPATCHER_OPTIONS argument. If defined, it is mandatory to set if before the DISPATCHER_OPTIONS argument. Default value for LSF is YES, and NO for other dispatchers.

- DISPATCHER_OPTIONS=options

This keyword allows sending extra options to the dispatcher submission command. The dispatcher options must be enclosed in ( ), { }, " " or ' . If enclosed in ( ) or { } Eldo may add some spaces between items. To keep the exact syntax, use " " or ' ' if the options contains some double quotes.

- DISPATCHER_TEMPLATE

This option allows to completely overriding the usual MPRUN mechanism. It is useful for .ALTER dispatching only. The user is in charge of providing a valid shell command which will dispatch jobs. This command can use predefined variables which are substituted by Eldo before executing the command. These variables are:

%NETLIST_NAME%

stands for the name of the netlist %RUN_NAME% stands for the name of each run (may be redefined with FILE_PREFIX)

%RUN_NUMBER%

stands for an absolute run counter (starting from 1) %MPRUN_OPTIONS% is a mandatory variable which represents internal options added by Eldo.

Example with LSF

```
.MPRUN DISPATCHER=LSF
 + DISPATCHER_OPTIONS=(-q myqueue -m "host1 host2")
+ MAX_NBJOBS=5
```

Specifies Eldo to use the LSF management system as a dispatcher for the remote jobs. The simulation will be run on both the host1 and host2 machines. A maximum of five jobs will be submitted to LSF. -q myqueue is the LSF option which controls the Batch Queue to which the jobs will be submitted.

Observation

Table : 6.2.1 comparision of simulation time

S.no	Case	Simulation time	Conclusion
1	Netlist containing Two .ALTER & 1000 MC	Approx 19 hrs	Conventional method
2	Two .ALTER & 1000 MC + MPRUN	Approx 8 hrs	according to mprun's characteristic it distribute jobs with respect to .ALTER. So it uses only two processors.
3	Netlist containing only one .ALTER(block of above netlist) & 1000 MC + .MPRUN	Approx 1½ hrs	In this mprun distribute jobs according to .MC & uses upto 40 – 45 processors

From the above evaluations we can conclude that we can reduce our simulation time very drastically. But require a smart approach (with giving corrected value for MAX_NBJOBS & NBLICENSES).

Table : 6.1.2 priority of distributing command

S.no	no. of ALTER	no. of STEP	no. of TEMP	no. of MC	observation & Conclusion
1	0 -4	5	5	50	distribution according to ALTER
2	0 -1	5	5	50	same
3	0 -9	5	5	4	according to ALTER
4	no	5	5	50	distribution according to temp
5	no	5	8	50	according to temp
6	no	8	5	50	according to temp
7	0 -9	10	10	9	according to alter

According to my observation the preferences given by MPRUN to distributing commands are:-

.ALTER > .TEMP > .STEP > .MC

CHAPTER 7

CROSS-TALK SETUP

SIGNAL INTEGRITY CHECKS

These CircuitCheck commands are designed to help check for static and dynamic crosstalk and noise-sensitivity estimation.

7.1 Dynamic Crosstalk Analysis

Dynamic crosstalk refers to the parasitic effects which lead to distortion of digital signals in the post-layout netlist. The effect is measured according to user-specified thresholds for change of characteristics of the signals. Violations are reported in the CircuitCheck result file.

In Figure 7.1 there are two overlapped waveforms of the signal. The leading one is the waveform in the pre-layout netlist. The second is the waveform found in the post-layout netlist. It somewhat differs from the first due to parasitic effect.

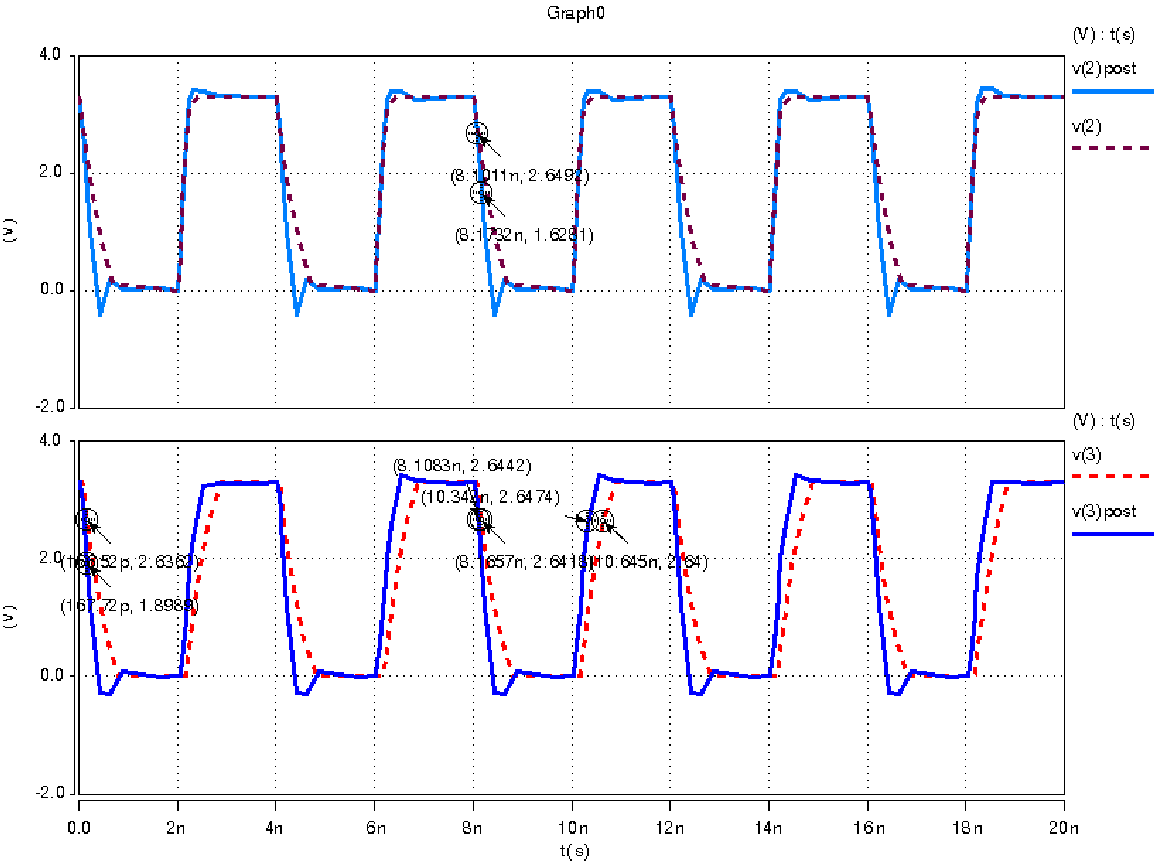


Figure7.1 comparing post-layout & prelayout

cckDXtalk syntax

cckDXtalk ccFile=<ccfile_name> **noccFile**=<noccfile_name>
[**nodeListFile**=<nodelist_name>]
[**node**=<node_name>]* [**skipnode**=<nodeName>]* [**slope_rel**=10]
[**slope_abs**=1e-9]
[**slope_vhth**=80] [**slope_vlth**=20] **[delay_rise_th**=50]
[**delay_rise_abs**=1e-9]
[**delay_fall_th**=50] [**delay_fall_abs**=1e-9]
[**separate_file**=[0|1]] [**extract_signals**=[0|1]] [**bc_wc**=[0|1]]

SYNTAX DEFINITIONS

- **ccFile**

  Optional. ccFile specifies the post-layout fsdb file name.

- **noccFile**

  Optional. noccFile specifies the pre-layout fsdb file name.

- **nodeListFile**

  Optional.

- **spfFile**

  Optional. spfFile specifies the \SPF file name.

- **node**

  Optional

- **skipnode**

  Optional

- **slope_rel**

  Relative slope in percentage. Unit is %. The DEFAULT is 10.

- **slope_abs**

  Absolute slope. Unit is V/ns. The DEFAULT is 1.

- **slope_vhth or slope_high**

  High threshold voltage of output. Unit is %.The DEFAULT is 80, meaning that 80% of HSIMVDD.

- **slope_vlth or slope_low**

  Low threshold voltage of output. Unit is %. The DEFAULT is 20, meaning that 20% of HSIMVDD.

- **delay_rise_th or delay_rise_thr**

  Rising threshold in percentage. Unit is %. The DEFAULT is 50, which means 50% of HSIMVDD.

- **delay_rise_abs**

  Absolute value of rise time. Unit is second(s). The DEFAULT is 1ps.

- **delay_fall_th or delay_fall_thr**

  Falling threshold in percentage. Unit is %. The DEFAULT is 50, which means 50% of HSIMVDD.

- **delay_fall_abs**

  Absolute value of fall time. Unit is second(s). The DEFAULT is 1ps.

- **separate_file**

  If set to 1, output to a separate file. The DEFAULT is 0.

- **extract_signals**

  If set to 1, extract signals which violate cckDXtalk rules from ccFile and noccFile into a separate fsdb file. The DEFAULT is 0.

- **bc_wc or wc_bc**

  If set to 1 best/worst violations of a signal in the simulations are reported. The DEFAULT is 0.

Signal Edge Characteristics

The signal edge characteristics for the waveform shown in Figure 7.1, Pre-/Post layout Waveform Node Differences are described below.

THRESHOLDS

- **Delay_rise_th and delay_fall_th**

  Thresholds for the rising and falling edges. Normally these two values are 50% of VDD. Their simulation time at the point where the signal crosses the thresholds are tr and tf.

- **Slope_vhth and slope_vlth**

  Defines the signal edges. The DEFAULT value of slope_vhth is 80% of VDD and 20% of VDD for slope_vlth

RISING SLOPE

- **Srising**

  The positive average slope of a signal crossing from slope_vlth to slope_vhth as shown by the following syntax: srising=(slope_ vhth-slope_ vlth)/(tr_ end-tr_ start).

FALLING SLOPE

- **Sfalling**

  The negative average slope of a signal crossing from slope_vhth to slope_vlth as shown by the following syntax: sfalling=(slope_ vlth-slope_ vhth)/(tf_ end-tf_ start).

Validation is measured based on four criteria:

- **Rising Edge Delay**

  Rising edge delay measures the time delay of a rising edge. It measures the difference of tr, i.e., tr_delay between pre-layout and post-layout waveforms. If the difference, in second, is greater than the value specified by delay_rise_abs, a Warning is issued into the result file.

- **Falling Edge Delay**

  Falling edge delay measures the time delay of a falling edge. It measures the difference of tf, i.e., tf_delay between pre-layout and post-layout waveforms. If the difference, in second, is greater than the value specified by delay_fall_abs, a Warning is issued into the result file.

- **Slope Relative Error**

  Slope relative error measures the relative error of rising/ falling slopes between pre-layout and post-layout waveforms. The error is calculated using the formula (slopepost_ layout-slopepre_ layout)*100% / (slopepre_ layout).If the absolute value of the error is greater than the value specified by slope_rel, a Warning is issued into the result file.

- **Slope Absolute Error**

  Slope absolute error measures the difference of rising/ falling slopes between pre-layout and post-layout waveforms. If the difference, in V/ns, is greater than the) value specified by slope_abs, a Warning is issued into the result file.

Usage Flow Methods

Various methods of executing dynamic crosstalk checking:

METHOD 1.

The first method of executing dynamic crosstalk checking is to execute HSIM separately. Using this approach, simulation results must be prepared for both pre-layout and post-layout netlists in advance. The result files are denoted as noccFile.fsdb and ccFile.fsdb respectively. The nodes of interest have to be output using .print command for both cases.

The analysis needs an optional node list file such as nodelist.txt. The node list file is a text file containing a list of analysis nodes itemized one node name per line. If the file is not given, the node specifier in cckDXtalk will specify which nodes are to be analyzed.

Dynamic crosstalk analysis is eventually driven by adding cckDXtalk into the CircuitCheck command file of the netlist. This is both a pre- or post-layout netlist file and run HSIM with this revised netlist. Figure 7-2, shows the detailed analysis flow.

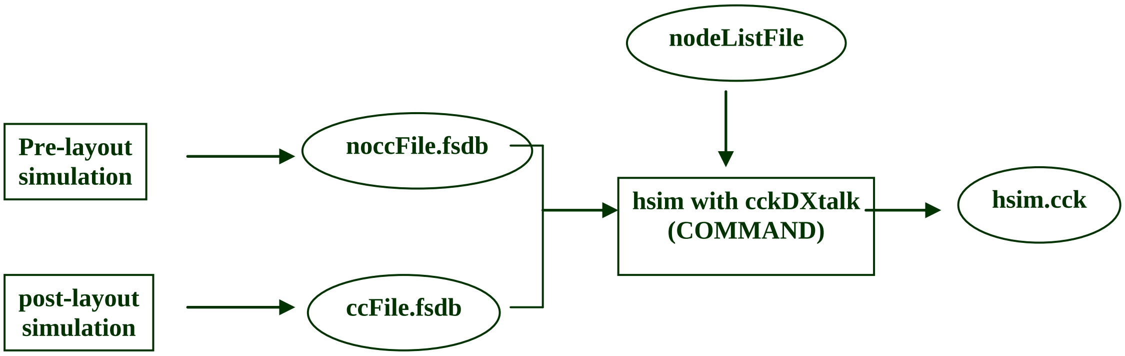


FIGURE 7-2.Dynamic Cross Talk Detailed Analysis Flow

METHOD 2.

The second method of executing dynamic crosstalk checking is to run cckDXtalk in batch mode. Using this flow, only spfFile must be specified while not specifying ccFile and noccFile. The command will automatically execute HSIM twice.

- The first run simulates the pre-layout netlist.
- The second run simulates the post-layout netlist with the specified SPF file and carries out the analysis.

Nodes found that are to be analyzed are automatically added to output files as shown in following example.

Example 1:

Using cckDXtalk when ccFile and noccFile are given when ccFile and noccFile are both given, add cckDXtalk to the CircuitCheck command file using the following syntax:

```
cckDXtalk ccFile=post.fsdb noccFile=pre.fsdb
nodeListFile=nodelist.txt\
delay_rise_th=45 delay_rise_abs=1.12n \
slope_vlth=14 slope_vhth=80 slope abs=1 slope_rel=10 \
delay_fall_th=60 delay_fall_abs =0.001n \
separate_file=1
```

Following is typical report sample from **cckDXtalk**:

```
* ----------------------------------------------------
* Dynamic Cross Talk Check
* noccFile = pre.fsdb
* ccFile = post.fsdb
* nodeListFile = nodelist.txt
* referenced vdd = 1.5 V
* slope_rel = 10%
* slope_abs = 1 V/ns
* slope_vlth = 14%
* slope_vhth = 80%
* delay_rise_th = 55%
* delay_rise_abs = 1 ns
* delay_fall_th = 60%
* delay_fall_abs = 0.001 ns
* separate_file = 1
* ----------------------------------------------------
Signal      sim  time(ns)  delay time(ns)  slope rel(%)   slope \abs(v/ns)
v(oup)        0.375            -             28.3             3.69
v(oup)        3.96          0.232(F)        -47.7             8.73
v(oup)        7.36             -              34              4.49
v(oup)        11            0.232(F)        -47.7             8.72
v(out_b)      0.28             -              34              4.49
v(out_b)      2.71          0.232(R)        -47.7             8.72
.......
find 24 rise delay violation(s).
find 89 fall delay violation(s).
find 174 absolute slope violation(s).
find 174 relative slope violation(s).
Total of 437 dynamic cross talk violation(s).
Total of 6 node checked.
```

7.2 Static Crosstalk Noise Analysis -Estimate Noise Glitches

Due to the ever increasing density in a chip design, the coupling capacitor between signal nets is an important topic to study. Aggressor nets can induce noise on victim nets. If the noise glitch is too large, it will trigger the circuit to change state unexpectedly. Furthermore, with the decrease of power supply voltage (VDD) such as from 3V to 2.5V to 1.8V, the noise glitch requires more attention and needs to be addressed.

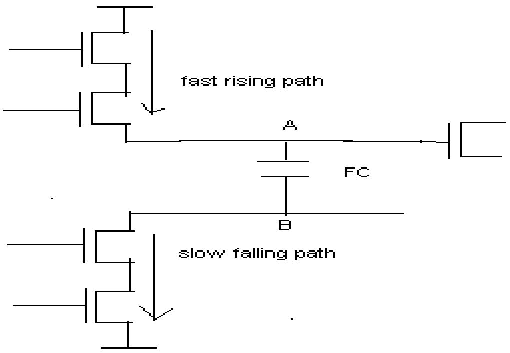


FIGURE 7-3.Fast Rising vs. Slow Falling Path

Slow falling path. The total = Rise noise glitch is
Capacitance along the path is Vr = (FC / Total_Cf)*VDDs

If FC is a floating capacitor with a value exceeding a specific threshold; and from node A, the fast rising paths are located; from node B, slow falling paths are located. The fast rising at A will create a rising noise glitch at node B. The noise bump is approximated as (FC / Total_Cf) * VDD, where FC is the coupling capacitance; Total_Cf is the total capacitance on the slow falling path.

Similarly, fast falling paths and slow rising paths may exist at the two ends of a floating capacitor. The noise glitch on the slow path needs to be computed.

The following specifications are needed to analyze crosstalk:

SYNTAX DEFINITIONS

cckXtalkFloatingCap	cap
cckXtalkRiseVolt	vr
cckXtalkFallVolt	vf
cckXtalkByWL	[0|1]
cckXtalkModelWLratio	<model=m> <min=d1> <max=d2>
cckXtalkByRC	[0|1]
cckXtalkRiseTimeConst	<min=t1> <max=t2>
cckXtalkFallTimeConst	<min=t3> <max=t4>
cckXtalkAtNode	<subckt=s> <node=nd>
cckXtalkSkipNode	<subckt=s> <node=nd>
cckXtalkSkipElem	<subckt=s> <inst=e> <pattern=p>
cckXtalkReportCouplingCapRatio	<CC/Ctotal ratio>
cckXtalkReportAggressorNode	node = <node_name>(supports wildcard)

CircuitCheck permits nodes in a design to be selectively analyzed as described in the following steps.

Step 1. Specify the coupling capacitance threshold (cckXtalkFloatingCap=fcap). Any floating capacitor with a value larger than the specified threshold is considered in the crosstalk analysis.

STEP 2. Specify the following:

- Rising noise glitch threshold (cckXtalkRiseVolt=vr)
- Falling noise glitch threshold (cckXtalkFallVolt=vf)
  Nodes are reported if they are impacted by the coupling capacitance and the resulting glitches are larger than vr/vf.

STEP 3. The fast and slow paths ending at the coupling capacitors need to be identified. Fast paths have low impedance and slow paths have high impedance. CircuitCheck looks for the impact from a fast path to a slow path.

Two methods for detecting the fast and slow paths are:

- Width-to-Length (W/L) Ratio using *cckXtalkByWL*=1
- RC-delay using *cckXtalkByRC*=1

Note : These two commands are mutually exclusive:
Either *cckXtalkByWL* or *cckXtalkByRC* may be set to =1, but not both.

a. **W/L Ratio Method** (*cckXtalkByWL*=1)

The MOSFET models that need to be considered along the paths must be specified. In this model, the following apply:

- If a path's smallest W/L ratio is larger than the maximum ratio specified, the path is treated as a fast path.
- If a path's smallest W/L ratio is smaller than minimum ratio specified, then this path is treated as a slow path.
- *ckXtalkModelWLratio* model=m1 min=d1 max=d2 specifies the min/max W/L ratios for model m1.

b. **RC-delay Method** (*cckXtalkByRC*=1)

The rising and falling time constant must be specified. In this model, the following apply:

Rising Time Calculation

- If a path's rising time constant is smaller than minimum rise time constant, then this path is a fast rising path.
- If this rising time constant is greater than the maximum rise time constant, the path is a slow path.
- *cckXtalkRiseTimeConst* min=t1 max=t2 specifies the delay range for rising path. Any rising path with a delay greater than t2 is a slow path.

Falling Time Calculation

- For falling paths, the minimum and maximum time constants are used to decide if a falling path is fast or slow. The rc-delay calculation is same as the method described in *cckRCDlyPath*.

The following three files are created:

- *hsim.cckxtk* **or** *out_file.cckxtk*

  These files show path-pairs on the interested floating capacitors. In each pair, one path is fast and the other is slow.

- *hsim.cckvr* **or** *out_file.cckvr*

  Path-pairs that cause a large rising noise glitch will be reported in these files.

- *hsim.cckvf* **or** *out_file.cckvf*

  These files contain path-pairs that cause large falling glitch, hence smaller vf.

Examples 3:

1. *cckXtalkAtNode node* = xam*

   CircuitCheck only analyzes the floating capacitors with two ends in xam.

2. *cckXtalkSkipNode subckt* = ram node=*

   CircuitCheck analyzes all the nodes except those in all the instantiation of subckt ram.

Example 4: **Crosstalk Aggressor Static Check**

Using the following syntax, a check is conducted by going through the x1.a* victim nodes to see if coupling capacitance is connected. If *YES*, and CC has CC/Ctotal value is greater than the specified ratio (0.2), then the other end of CC terminal is reported as the aggressor signal. The checking result will be stored in a report file with the *.cckXtkNdCap* extension.

cckXtalkReportCouplingCapRatio 0.2

cckXtalkReportAggressorNode node=x1.a*

Example 5:

```
cckXtalkFloatingCap        1.e-12
cckXtalkRiseVolt            0.02
cckXtalkFallVolt            0.3
cckXtalkByWL                1
cckXtalkModelWLratio  model=nx   min=12  max=18
cckXtalkModelWLratio  model=px   min=16  max=24
cckXtalkModelWLratio  model=px2 min=10  max=20
```

In Example 5, any floating capacitor with value larger than 1 pF will be considered. The W/L ratio is used to determine if a path is fast or slow. The three MOSFET models to be considered are:

- model = nx The min/max ratio for this model is 12 and 18. Any nx type of transistor with W/L ratio larger than 18 will likely to be part of a fast path. This is a fast path=low impedance.
- model = px
- model = px2

When a floating capacitor FC is considered, its two nodes are A and B. From node A, the paths are as follows:

- 2 rising paths to VDD
- 3 falling paths to GND

For each rising path from A, the smallest W/L ratio on the path must be found. If the px2 MOSFET has the smallest W/L ratio, then this W/L ratio is compared with the given min /max values. If its W/L is larger than 20, then this path is a fast path. If its W/L is smaller than 10, then it is a slow path. Then examine the falling paths from node A and use the min/max W/L ratios of model nx to decide if a falling path is slow or fast.

To find the fast and slow paths, a similar search is accomplished from node B.

After this is accomplished, the impact on node B from node A and node A from node B is computed. Therefore, for each selected floating capacitor, a set of fast-path and slowpath pairs is identified. All pairs are reported in either the *hsim.cckxtk* or *out_file.cckxtk* files.

To calculate the rising and falling glitch, the following paths are used:

- Fast paths ending at A
- Slow paths ending at B

Reports of the rising and falling glitch on B are saved in the following files:

- Rising Glitch > 0.02 *hsim.cckvr* or *out_file.cckvr*
- Falling Glitch > 0.3 *hsim.cckvf* or *out_file.cckvf*

***hsim.cckxtk* Output Sample**

The hsim.cckxtk output sample contains pairs of slow and fast paths as shown in the following:

```
; Report format:
; Serial_num Capacitor_name Cap_1st_node_name
Cap_2nd_node_name Cap_value \
; minWL_mos_name_on_low_imp_path
minWL_mos_name_on_high_imp_path
; Low_imp_path:
; Vsrc_node_name
; Elem_name Elem_type L/W or Resistance
; High_imp_path:
; Vsrc_node_name
; Elem_name Elem_type L/W or Resistance
Total number of low and high impedance path pairs for
floating capacitors=2912
User defined values:
Floating capacitance threshold=1e-012(F)
Use MOSFET WL ratio to find low/high impedance paths.
nxmin=12, max=18
pxmin=16, max=24
px2min=10, max=20
1 c6x gx1 wl8 4.51e-012 mx7|mn1 mx55|mi30
low_imp_path:
gnd
mx7|mn1nmos0.4/5
* mx7|xi38|mn1@3 nmos0.4/5
* mx7|xi38|mn1@2 nmos0.4/5
* mx7|xi38|mn1@5 nmos0.4/5
```

CHAPTER 8

FUTURE WORKS

8.1 FUTURE WORK/REQUIREMENTS

There are several ways how future work can be based on the presented results. Our replica structure implementation suggests that it is possible to built an *efficient fast and low power memory* without making any modifications to the existing system except the additional dummy column to replicate the normal column. Although our system contains all necessary mechanisms to support the dual port at a time, there are a few more issues concerning the overall *policy* that deserve further investigation. Some of these issues are discussed below.

Clearly, a deeper exploration of the effects of the parameter on the basic memory cell is needed as the technology is shrinking. In particular, a better understanding of the memory cell parameters which are used to characterize the performance, speed and density. Alternative solution attempts for the low power outlined in the thesis section 4.3 should also be investigated further. Thus, better methods for characterizing the basic memory cell are a rewarding topic for further research. Ideas from theory revision might turn out to be useful in this context.

The Eldo optimizer is very powerful tool for designing purposes. But require some modification in tool because it can only calculate parameter which produces local optimized objective set by the user. Due to this we require to change the variable parameters to achieve the efficient output. Require to develop full flow of the procedure or methodology so every person uses it efficiently.

The cross-talk analysis setup is very helpful for generating the nodes which suffer from cross-talk & which can harm our design. It is alright with the static check but in case of dynamic analysis we require a healthy discussion about type of stimuli use & where to apply these stimuli.

8.2 APPLICATIONS

Dualport SRAM is a special purpose memory often used for video purposes. Faster than normal computer RAM, the amount of Dualport SRAM determines how many colors a given video system can display. This Dualport SRAM will be used in single chip driver for LCD display.

The optimizer can reduce the designing time so; it can be very helpful to the designers. There are some designs which require huge affords or designing may be very tidiest then it reduce lots of designer affords.

The multi-threading is totally devoted to reduce the simulation time of any big Eldo netlist. This facility can reduce the simulation time by 60 -80% of the original simulation time. It basic applications can be come when we have to simulate a very big design.

The cross-talk analysis can be very powerful setup for generating those nodes (victim, aggressors) which violating user specified threshold. It also reduces the burden of estimating the cross-talk prone nodes manually.

CHAPTER 9

CONCLUSION

CONCLUSION

In this thesis we looked at design trade offs for dual port 180nano (DP180nano) SRAMs. The effort required for DP180nano memory start from basic dual port SRAM cell design to system level. Different methodologies are used to make the system area and power efficient. Mainly the port sharing technique reduces the large amount of area of the system and the rest of area is minimized by using the ultra high density memory cell and the replica structure without dummy rows.

The key to low power operation in the SRAM data path is to reduce the signal swings in the high capacitance nodes like the bitlines and the data lines. Clocked voltage sense amplifiers are essential for obtaining low sensing power, and accurate generation of their sense clock is required for high speed operation. An energy efficient way to obtain low voltage swings in the bitlines is to limit the word line pulse width, by controlling the pulse width of the block select signal. The pulse widths are regulated by the aid of a replica delay element which consists of a replica memory cell and a replica bitline and hence tracks the delay of the memory cell over a wide range of process and operating conditions. The pulse width is controlled by discharging a replica line which has the same capacitance as the worst case data line. This replica line is also used to trigger the amplifiers at the receiving end of these lines.

The optimizer can reduce the designing time so; it can be very helpful to the designers. There are some designs which require huge affords or designing may be very tidiest then it reduce lots of designer affords. The multi-threading is totally devoted to reduce the simulation time of any big Eldo netlist. This facility can reduce the simulation time by 60 -80% of the original simulation time. It basic applications can be come when we have to simulate a very big design. The cross-talk analysis can be very powerful setup for generating those nodes (victim, aggressors) which violating user specified threshold. It also reduces the burden of estimating the cross-talk prone nodes manually.

REFERENCE

Bibliography

[1] E. Seevinck, F. J. List, and J. Lohstroh, “Static-noise margin analysis of MOS SRAM cells,” *IEEE J. Solid-State Circuits,* vol. SC-22, pp. 748-754, Oct. 1987.

[2] A. J. Bhavnagarwala, T. Xinghai, and J. D. Meindl, “The impact of intrinsic device fluctuations on CMOS SRAM cell stability,” *IEEE J. Solid-State Circuits,* vol. 36, pp. 658-665, Apr. 2001.

[3] M. Yamaoka, “0.4-V Logic-Library-Friendly SRAM array using rectangular-diffusion cell and delta-boosted-array voltage scheme,” *IEEE J. Solid-State Circuits,* vol. 39, pp. 934-940, June 2004.

[4] A. Hardi, B. Bhaumik, P. Pradhan, R. Varambally, “ A Low Power 256 KB SRAM Design,”

[5] M. Lee, Wing-II Sze, Chii-ming M. Wu, “Static noise margin and soft-error rate simulations for thin film transistor cell stability in a 4 Mbit SRAM Design,” © *1995 IEEE.*

[6] ST internal site

[7] Neil H. E. Weste, Kamran Eshraghian, *the system perspective* PRINCIPLES OF CMOS VLSI DESIGN, Second Edition.

[8] Sung-Mo Kang, Yusuf Leblebici, CMOS Digital Integrated Circuits *Analysis & Design,* Third Edition..

[9] Bhardwaj S. Amrutur and Mark A. Horowitz, “A Replica Technique for Wordline and Sense Control in Low-Power SRAM’s.” *IEEE J. Solid State Circuits,* Vol. 33, No. 8, August 1998.

[10] Evert Seevinck, Frans J. List, J. Lohstroh, “Static Noise Margin Analysis of MOS SRAM Cells,” *IEEE J. Solid State Circuits,* Vol. SC-22, No. 5, pp. 748-754, October 1987.

[11] Martin Margala “Low-Power SRAM Circuit Design,” @ IEEE, 1999.