8-PORT S-RAM MEMORY CELL, WITH 8 WRITES OR 16 READS SIMULTANEOUSLY

BY

RAGHU KOPPANATHI

Bachelor of Engineering

Osmania University

Andhra Pradesh

India

2002

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May,2005.

8-PORT S-RAM MEMORY CELL, WITH
8 WRITES OR 16 READS
SIMULTANEOUSLY

Thesis Approved:

Dr.Louis.G.Johnson

Thesis Advisor

Dr.Yumin Zhang

Dr.R.G.Ramakumar

Dr.Gorden Emslie

Dean of the Gradute College

Acknowledgements

This section of my thesis is the most difficult part of all the work I have done so far, I was mulling on what to write in this section for more than a week. Initially, I thought this would be the easiest of all, but it turned out the other way, specially expressing every thing in words, Ohh booy!!! This is one heck of a job. I would like to take this opportunity to sincerely thank all the people responsible for every bit of their contributions whatsoever, for what I am.

Foremost, I would like to express my gratitude to my thesis advisor Dr.Louis.G.Johnson, for his guidance, whose inspiration and ceaseless enthusiasm for the topic explored in this thesis, has meant a great deal to me. The topic indeed a suggestion of his, has improved my understanding on the design of Single port and Multi-Port Memory cells. I would like to sincerely thank him for sharing his understanding and knowledge in this field. I take this opportunity to sincerely appreaciate for his guidance, patience and being so supportive in all my difficult times in my masters study. In addition, I express my appreciation and thanks to other member of the committee, Dr.Yuming Zhang and Dr. R.RamaKumar for having served on my committee. Their thoughtful comments and questions were greatly valued.

And I would also like to thank all my research group members, roommates and friends for being a surrogate family and continued support during my many years in Stillwater, Oklahoma. And above all, I would like to thank my loving parents, my father Mr.K.R.Vara Prasad, my mother Mrs.K.Adi lakshmi and my sister Ms.K.Revathi for their support and encouragement althrough out my career. They have been my source of inspiration and motivation. Thank you Dad and Mom.

Table of Contents

List of Figures

List of Tables

List of Acronyms

RAM	Random Access Memory or Read Write Memory
SRAM	Static Random Access Memory or Read Write Memory
DRAM	Dynamic Random Access Memory or Read Write Memory
CMOS	Complementary Metal-Oxide Semiconductor
NMOSFET	n-channel Metal Oxide Semiconductor Field-Effect Transistor
PMOSFET	p-channel Metal Oxide semiconductor Field-Effect Transistor
PMOS	p-channel Metal Oxide semiconductor
NMOS	n-channel Metal Oxide semiconductor
BSIM3v3	Berkeley Short-channel insulated gate FET model, Version 3
SPICE	Simulation Programs with integrated circuit emphasis

Nomenclatures

$\mathbf{V_{dd}}$	Supply Voltage (5v)
Gnd	Ground (0v)
$\mathbf{V_{inv}}$	Inverter Threshold Voltage
$\mathbf{V_{cell}}$	Voltage at memory cell storage node in single port, 2-SRAM, and J-SRAM Models of SRAM.
$\mathbf{V_{cellbar}}$	Voltage at memory cellbar storage node in single port, 2-SRAM, and J-SRAM Models of SRAM.
$\mathbf{V_{AB}}$	Voltage at memory AB storage node in 8-SRAM model.
$\mathbf{V_{ABbar}}$	Voltage at memory ABbar storage node in 8-SRAM model.
$\mathbf{V_{ABCD}}$	Voltage at memory ABCD storage node in 4-SRAM model.
$\mathbf{V_{ABCDbar}}$	Voltage at memory ABCDbar storage node in 4-SRAM model.
$\mathbf{W_p}$	p-channel MOSFET width
$\mathbf{W_n}$	n-channel MOSFET width
$\mathbf{W_a}$	access transistor, n-channel MOSFET width
$\mathbf{V_{GS}}$	Gate-Source Voltage
$\mathbf{V_{DS}}$	Drain-Source Voltage
$\mathbf{V_{tn}}$	n-channel MOSFET threshold voltage
$\mathbf{V_{tp}}$	p-channel MOSFET threshold voltage

Chapter 1

Introduction

1.0 Memory Arrays

Memories arrays are systems or array of elements capable of storing digital data or information in large quantity, ranging from bytes to giga bytes. Today, memories are available in different forms like SRAM, DRAM, SDRAM, EPROM, EEPROM, Flash and (memories based on ferroelectric elements) FRAMs or FeRAMs depending on applications. Semiconductor memories may be divided into three categories:

- Random access memory
- Serial access memory
- Content access memory

Random access memory is accessed with an address and has latency independent of the address (access time independent of the physical location of the data). In contrast, serial access memories are accessed sequentially and are address independent (no address) and content addressable memories has some latency associated with reading or writing of a particular datum.

Random access memories are sub-divided into Read/Write memory (RAM-volatile memory) and Read only memory (ROM-Nonvolatile memory). Read/Write access memory may store data or information in flip-flop style circuits or simply as a charge on capacitors. Two most common types of RAMs based on structure are Static RAM and Dynamic RAM. Static RAMs hold the stored value in flip-flop style circuits as long as power is on; and tend to be high speed memories with clock cycle range from 5 to 50ns. Dynamic RAMs store data on capacitors, are prone to noise and leakage problems. They tend to be slower than SRAMs, clocking at 50ns to 200ns. Both types of memory may be further divided into static-load, synchronous, and asynchronous categories. Static-load memories require no clock. Synchronous RAMs or ROMs require a clock edge to enable memory operation. Asynchronous RAMs recognize address changes and output new data after any such change. Within general classification of random access memory, read/write memory (RAMs) or read only memory (ROMs). ROMs usually have a write time much greater than their read time (programmable ROMs have write time of the order of few milliseconds), while RAMs have very similar read and write times [2].

1.2 Research Interest

With the advent of technologies, today's advanced computer processor designs requires complex functional units to process more than 2 source operands and 1 destination operand, to be interfaced with the same register file. For this either the register file can be partitioned to process those dedicated functional units or design a register file capable of processing more number of inputs simultaneously [7]. The 6-T SRAM (Fig.3.1) model can be used to design an array of multiport memory cell. But, the readability and stability of the cell would be of prime issue; especially if the number of ports is more than “three”. With the increase in the number of ports, correct readability and writability would be possible only with, sizing of n-channel MOSFET transistors in the memory cell. Sizing the transistor properly to enable read and write to the memory cell with multiport is a challenge with limited area constrain. Increase in number of ports and increase in the size of n-channel MOSFET would increase the effective capacitance on the cell and cellbar node, this would deteriate the stability of the cell. The other important issue is noise immunity of the cell, within which correct data can be read out and stored. This noise immunity of the cell deteriates as the number of ports on the cell increases, thereby decreasing the stability of the cell. Noise immunity is also defined as noise margin, is calculated as the difference in the cell node voltage and inverter threshold voltage of the cross coupled inverter in the memory cell. These two issues namely the transistor sizing with increase in the number of ports on the cell and stability of the cell in terms of noise margin are the prime issue that are dealt with in the study of 8-port SRAM memory cell. Various designs were worked on to increase the stability of the memory cell by reduce the effective capacitance on the cell node with a reasonable noise margin as a prime goal.

1.3 Key Design Issues

8-port SRAM memory cell has serious design issues which cannot be neglected for stability and readability of memory cell. My thesis extensively deals with two issues in design of 8-port SRAM. Firstly, large bitline capacitance with increase in the number of ports coupled to the cell and cellbar node. This bitline capacitance accounts for drain capacitance of all the access transistors coupled to memory cell, its presence is also felt even when the access transistor or pass transistors (as referred by some authors) is in cutoff region. To drive this large bitline capacitance, transistors have to be sized in memory cell. There is a tradeoff between transistor size in the memory cell and the chip area.

Secondly, Noise margin (or noise immunity) of the cell is a major concern. Noise margin decreases as increase in number of ports on the memory cell. Noise either from power lines, cross-coupling capacitance, or mutual inductance of wires would result in instability in the memory cell. Major concern in my work is the power line noise from Vdd and Gnd.

The problem encountered with the design of multi-port SRAM cells demands a design which focuses on maximizing the noise margin for the given number of ports. The noise margin of the cell is interdependent with various factors of the cell. There always exists a tradeoff between noise margin of the cell and the speed (read access time) of the cell. Often noise margin is compromised to meet high speed requirements of the cell. In my work an optimum transistor widths are chosen to satisfy both the noise margin and speed requirements of the 8-port memory cell.

CADENCE virtuoso layout editor is used for making a layout of 8-port SRAM memory cell design. The new BSIM3v3 equations are used to derive the equations for unknown node voltages in the memory cell.

1.4 Literature Review

Main source of literature review was IEEE Journal of Solid State circuits. In the field of memory design, most of the work is in the stage of research, not many papers are written about, inparticular on multiport SRAM designs. The conventions many authors follow differs from one another; the author in [10] calls the memory on itanium 2 processor as 20 port capable of doing 8 writes and 12 reads simultanesously. This memory is capable of doing read and write in a same clock cycle. This is made possible by separate read and write controls circuits for the same cell. And few design concerns in the multiport cell have been discussed quite a long time back, the readability of the memory simulataneously through all the ports, this is discussed in [12], [13], and [14]. These authors have generalized that for a multiport SRAM, width of the n-channel MOSFET in the inverter loop should be large to read simultaneaously. The details of how large the transistor be, Dr.L.G.Johnson has explained inhis lecture notes and provided mathematical equations to support the theory [6], [7]. The author of [10] has explained about the stability of the cell, permit simultaneous read from the same location through all the ports. Christian Reichling have provided step by step procedure to design a single-port SRAM cell and have also given equations for calculating noise margin of the cell, which is called as static noise margin. [9] Discussed about various noise sources in Pentium processor; a main source he considers are power line supply, cross-coupling capacitance in the adjacent metal lines, mutual inductances, and interconnects. But, because of the limitation of the layout extracter, in my thesis I would be considering power lines as the main source of noise into the memory cell. And Dr.Johnson, in his notes of memory cell and register files [7] and [8] talks extensively about the necessary criterion for readability and writability into a memory cell. And the theory he proposes is well supported by mathematical equations. He has provided an excellent design methodology to design "m" multiport SRAM memory cell, and register files. He also explains about the design of sense amplifier, decoders, and write drivers and precharge, which are integral parts in design of register files.

1.5 Thesis organization

Chapter 2 describes the characteristics of the basic building block in memory cell- inverter, cross coupled inverters that form the flip-flop style circuit (latch). It also extensively describes about the DC characteristics of an inverter, regions of operation, noise margin and calculation of noise margin in an inverter.

Chapter 3 explains in detail the operation and structure of a single-port and multiport SRAM cell, describes about the criteria necessary to satisfy readability and writability into a single port and multiport SRAM memory cell. And briefly describes about noise margin in a memory cell.

Chapter 4 introduces various designs of 8-port SRAM cell; it extensively explains the understanding and working of the proposed designs. Readability and writability of the memory cell, worst case scenarios are discussed. And theoretical equations are derived for all the four designs.

Chapter 5 has an introduction to register file, and briefly emphasis the importance of sense amplifier, row decoders, write drivers and pre-charge circuits.

Chapter 6 provides theoretical results obtained by application of the theoretical equations described in chapter 4, and the simulation results obtained of the design layouts are also listed. Simulation is done using CADENCE SPECTRE simulator.

Chapter 2

CMOS Inverter DC characteristics

2.0 Introduction

Today, CMOS technology is the dominant semiconductor technology for microprocessors, memories, and application specific integrated circuits (ASICs). CMOS (Complementary Metal-Oxide Semiconductors) both N-type and P-type transistors are used to realize logic functions. CMOS has several key advantages over NMOS and bipolar technologies. CMOS offers low power dissipation, relatively high speed, high noise margins, and will operate over a wide range of source and input voltages (provided the source voltage is fixed).

2.1 Complementary CMOS Inverter-DC characteristics

The DC transfer characteristics of a circuit relate the output voltage to the input voltage, assuming the input changes slowly enough that the capacitances have plenty of time to charge or discharge. In digital logic levels specific ranges of input and output voltages are defined as valid “0” and “1” (low and high respectively).

The simplest device in digital design, complementary CMOS inverter is realized by the series connection of a p-channel MOSFET pullup and n-channel MOSFET pulldown device as shown in the Fig. 2.1. It's worth spending sometime in the DC analysis of a CMOS inverter, more because of the fact that it's a basic building block for digital logic circuits and memory cells. This analysis can also be extended to explain the behavior of more complex gates. MOSFET characteristics (both n-& p-channel) in different regions of operations cutoff, linear, and saturation can be referred from appendix A.

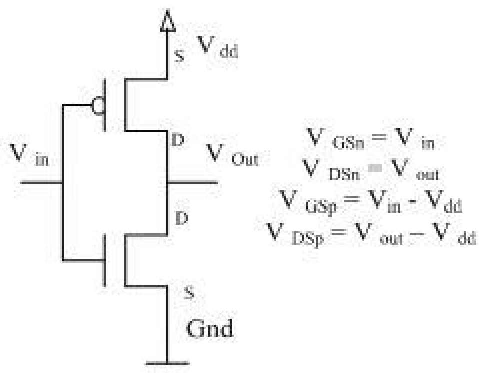

Fig.2.1 Schematic of CMOS Inverter

The objective is to find the variation in output voltage (V_{out}) as a function of the input voltage (V_{in}). This can be done graphically, analytically or through simulations. Given V_{in}, we must find V_{out} subject to the constraint that $I_{Dn} = | I_{Dp} |$. The operation of the CMOS inverter can be divided into five regions as indicated in Fig. 2.2 [5]. In region A, the n-channel MOSFET transistor is cutoff and p-channel MOSFET is in ohmic, so the p-channel MOSFET transistor pulls the output to V_{dd}.

$V_{in} < V_{Tn}$; Where V_{Tn} is nMOS threshold Voltage

$$I_{Out} = 0 \Rightarrow I_{Dn} = | I_{Dp} |$$
$$I_{Dn} = 0; \Rightarrow | I_{Dp} | = 0$$
$$\Rightarrow V_{Out} = V_{dd}$$

In region B, the n-channel MOSFET transistor is in saturation and p-channel MOSFET is in ohmic, pulling the output down. In region C, both the transistors are in saturation (note that ideal transistors are in the region C for $V_{in} = V_{dd} / 2$). We assume that an MOS device in saturation behaves like an ideal current source being independent of V_{ds}. In reality, as V_{ds} increases I_{ds} also increases slightly; thus region C has a finite slope. The significant factor to be noted is that in region C we have two current sources in series, which is an unstable condition. This makes the output transition very steep; the relation defined in this region is particularly useful since it provides the basis for defining the inverter threshold, or switching threshold V_{inv}.

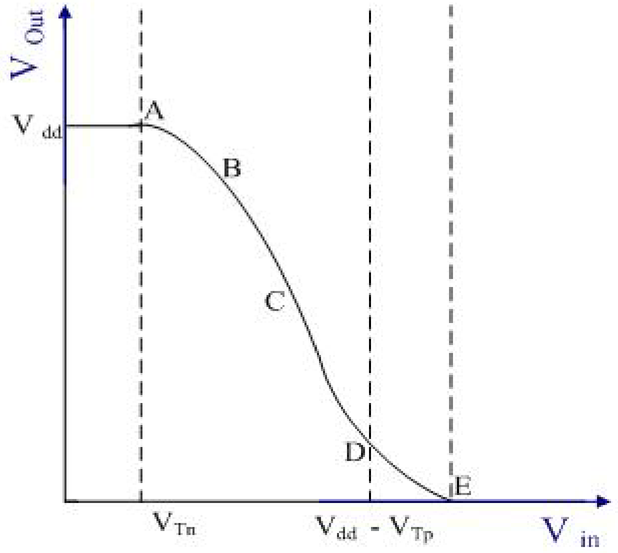

Fig.2.2 Region of operations in a CMOS Inverter

In CMOS inverter DC characteristics, point where the input voltage is equal to the output voltage is inverter threshold (or switching) voltage, we assume $V_{in} = V_{inv}$.

From this region of operation, both n-channel & p-channel MOSFETs are in saturation

$$I_{Dn} = |I_{Dp}| \qquad \text{(Since } I_{Out} = 0\text{)}$$

$$\frac{\beta_n}{2} V_{DSsatn}(V_{GSn} - V_{Tn}) = \frac{\beta_p}{2} V_{DSsatp}(V_{GSp} - V_{Tp})$$

$$\frac{\beta_n}{2}\left(\frac{E_{Satn} L_n (V_{in} - V_{Tn})^2}{(V_{in} - V_{Tn}) + A_{bulkn} E_{satn} L_n}\right) = \frac{\beta_p}{2}\left(\frac{E_{Satp} L_p (V_{in} - V_{dd} - V_{Tp})^2}{|V_{in} - V_{dd} - V_{Tp}| + A_{bulkp} E_{satp} L_p}\right) \text{---2.1}$$

$$where, \beta_n = \mu_e\left(\frac{\varepsilon_{ox}}{t_{ox}}\right)\left(\frac{W_n}{L_n}\right); \beta_p = \mu_p\left(\frac{\varepsilon_{ox}}{t_{ox}}\right)\left(\frac{W_p}{L_p}\right)$$

2.1 Holds good, considering source/drain parasitic resistance is zero ($R_{ds} = 0$). The new version of BSIM3v3 (appendix A) considers, R_{ds} as a finite parasitic drain source resistance parameter in the analysis of transistor current characteristics. As per BSIM3v3 the above equality can be re-written as [3];

$$\frac{\beta_n}{\beta_p}\left[(V_{in} - V_{Tn}) - A_{bulk}.V_{DSsatn}\right]\left(1 + \frac{V_{in} - V_{DSsatn}}{V_{An}}\right)$$
$$= \left[\left(V_{dd} - V_{in} - V_{Tp}\right) - A_{bulk}\cdot|V_{DSsatp}|\right]\left(1 + \frac{V_{dd} - V_{in} - |V_{DSsatp}|}{V_{Ap}}\right) \qquad \text{---- 2.2}$$

Where V_{DSsatn} & $|V_{DSsatp}|$ are calculated as in table A.1.2 & table A.2.2 (appendix A).

In region D, the p-channel MOSFET transistor is in saturation and n-channel MOSFET is in ohmic. And in region E, p-channel MOSFET is in cutoff and n-channel MOSFET in ohmic, the n-channel MOSFET pulls the output down to Gnd (0 V).

$$V_{in} > V_{dd} + V_{Tp} = V_{dd} - |V_{Tp}|$$

$\Rightarrow$ n-MOSFET ohmic, p-MOSFET cutoff

$$I_{Out} = 0 \Rightarrow I_{Dn} = | I_{Dp} |$$
$$I_{Dp} = 0; \Rightarrow I_{Dn} = 0$$
$$\Rightarrow V_{DSn} = 0$$
$$\Rightarrow V_{Out} = 0$$

Also, notice that the inverter's current consumption is zero (ignoring the leakage current) and no static power, when the input is within a threshold voltage of the V $_{dd}$ or Gnd rails. This feature is important for low-power operation.

2.2 Inverter Threshold Voltage (V $_{inv}$).

Inverter threshold voltage (V $_{inv}$) is defined as the input voltage of a CMOS inverter at which output rapidly changes between high and low. In DC inverter characteristics this is the voltage at which input voltage is equal to output voltage. The value of this voltage can be calculated analytically and by simulation.

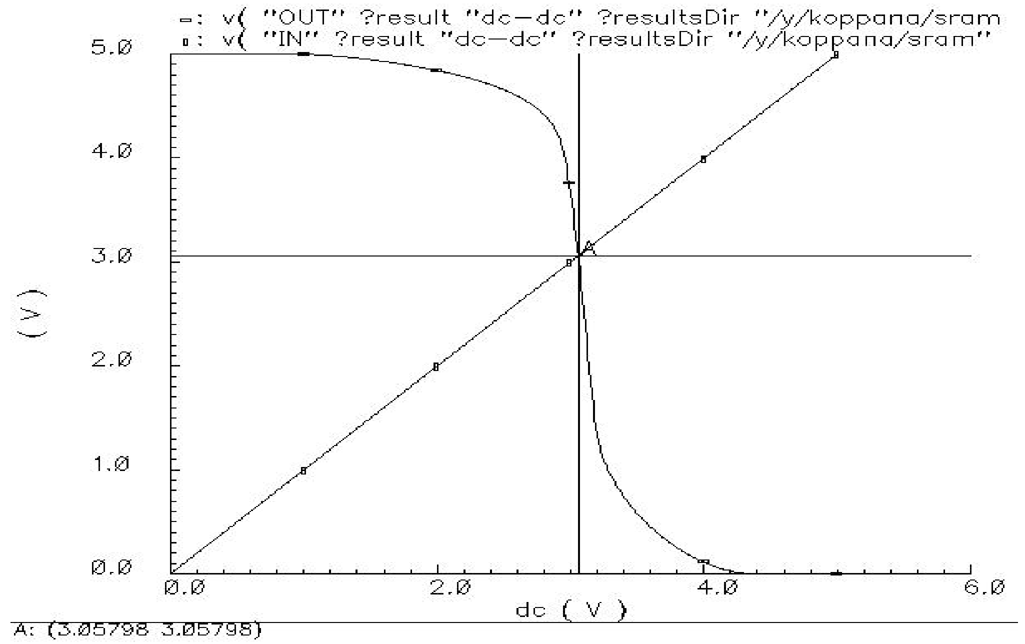

Fig.2.3 CMOS Inverter DC characteristics

Analytically, threshold voltage of an inverter can be calculated in the region where both p-channel and n-channel MOSFET transistors are in saturation. That is in the region C as indicated in fig.2.2. The value V_{inv} can be calculated from the equation as shown below; this is a cubic equation in V_{in} [4], [5].

$$\frac{\beta_n}{2}\left(\frac{E_{Satn}L_n(V_{in}-V_{Tn})^2}{(V_{in}-V_{Tn})+A_{bulkn}E_{satn}L_n}\right)=\frac{\beta_p}{2}\left(\frac{E_{Satp}L_p(V_{in}-V_{dd}-V_{Tp})^2}{|V_{in}-V_{dd}-V_{Tp}|+A_{bulkp}E_{satp}L_p}\right) \quad \text{--- 2.3}$$

Substituting the values of β_n & β_p in the above equation in terms of widths of p- & n-channel MOSFET and replacing $V_{in} = V_{inv}$; with $R_{ds} = 0$

We get; $$\frac{W_p}{W_n}=\left(\frac{(V_{inv}-V_{Tn})^2}{(V_{inv}-V_{Tn})+A_{bulkn}E_{satn}L_n}\right)\left(\frac{|V_{inv}-V_{dd}-V_{Tp}|+A_{bulkp}E_{satp}L_p}{(V_{inv}-V_{dd}-V_{Tp})^2}\right) \quad \text{--- 2.4}$$

We get threshold voltage V_{inv} by solving 2.3 or 2.4

Considering BSIM3v3 model inverter threshold voltage can be found to be as in equation (5) in this model, R_{ds} is considered a finite parasitic source/drain resistance.

$$\frac{W_n}{W_p}\left[(V_{inv}-V_{Tn})-A_{bulk}.V_{DSsatn}\right]\left(1+\frac{V_{inv}-V_{DSsatn}}{V_{An}}\right)$$
$$=\left[(V_{dd}-V_{inv}-V_{Tp})-A_{bulk}\cdot|V_{DSsatp}|\right]\left(1+\frac{V_{dd}-V_{inv}-|V_{DSsatp}|}{V_{Ap}}\right) \quad \text{--- 2.5}$$

This 3rd order equation 2.5 in V_{inv} can be solved to get threshold voltage either using mathematica or any other software. This equation can be approximated in terms of widths of p- & n-channel FETs, W_p & W_n respectively and threshold voltage V_{inv}, we observe that $V_{inv} \alpha \frac{W_p}{W_n}$. We can generalize; increase in W $_p$ would increase V $_{inv}$ and increase in W $_n$ would decrease V $_{inv}$. Threshold voltage V $_{inv}$ can also be found by doing a transient response simulation. This value is at the intersection of transient response and V $_{in}$ =V $_{Out}$ line as shown in the Fig.2.2. The test bench for simulating these characteristics is presented in the Appendix C.

2.3 Noise Margin

Noise Margin is related to the DC input-output voltage characteristics. This parameter allows you to determine the allowable noise, noise voltage either on the input gate or noise through the power lines (V $_{dd}$ or Gnd) that will not affect the output signal. The specification most commonly used to describe noise margin (or noise immunity) use two parameters; the low noise margin, NM $_L$ is defined as the difference in magnitude between the maximum low output voltage of the driving gate and the maximum input low voltage recognized by the driven gate [2]. Thus

$$NM_L = | V_{ILmax} - V_{OLmax} | \quad \text{--- 2.6}$$

The high noise margin, NM $_H$ is the difference between the minimum high output voltage of the driving gate and the minimum high input voltage recognized by the receiving gate. Thus

$$NM_H = | V_{OHmin} - V_{IHmin} | \quad \text{--- 2.7}$$

Where,

V $_{IHmin}$ = minimum high input voltage

V_{ILmax} = maximum low input voltage

V_{OHmin} = minimum high output voltage

V_{OLmax} = maximum low output voltage

For the purpose of calculating noise margin, the transfer characteristics of a typical inverter and the definition of voltage levels V_{IL}, V_{OL}, V_{IH}, and V_{OH} are shown in the Fig.2.4 [2].

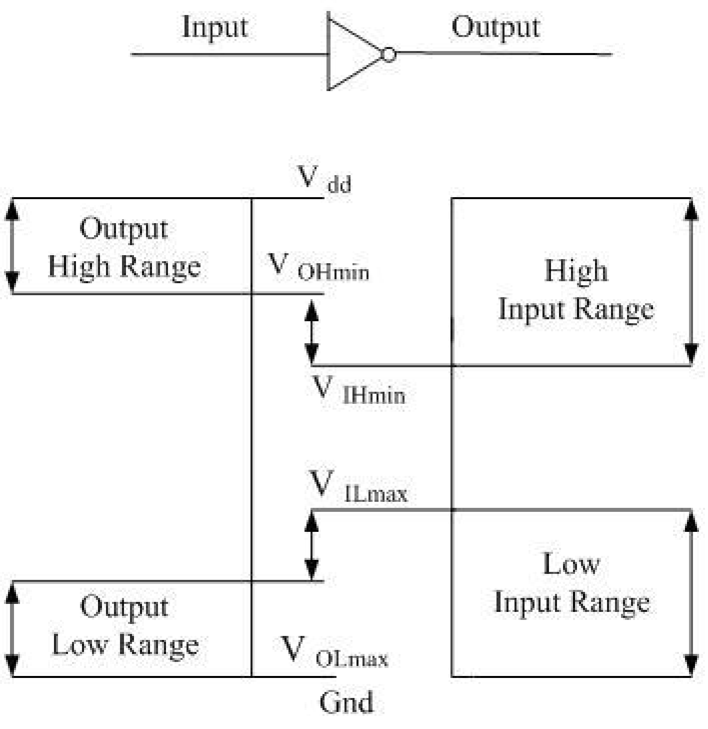

Fig.2.4 Logic range levels "high" and "low" in an Inverter

Logic levels are defined at the unity gain point where the slope is -1. Note that the output is slightly degraded when the input is at its worst legal value; this is called noise feedthrough or propagated noise. Quite often, noise margins are compromised to improve speed. Noise sources tend to scale with the supply voltage, so noise margin are best given as a fraction of the supply voltage.

To maximize noise margins and allow a capacitive load to charge and discharge in equal times by providing equal current source and sink capabilities, it may be desired that for $\beta_p = \beta_n$, the inverter threshold voltage V_{inv} is $V_{dd} / 2$. An inverter has a stronger pMOS transistor. Therefore, if the input is $V_{dd} /2$, we would expect the output will be greater than $V_{dd} /2$. As the beta ratio is changed, the switching threshold moves. However, the output voltage transition remains sharp. Gates are usually skewed by adjusting the widths of transistor while maintaining minimum length for speed.

2.4 Summary

CMOS inverter has an important role in memory systems. CMOS Inverter threshold voltage V_{inv} is voltage at which output rapidly changes between high and low (this is also referred to as switching threshold or trip point). Ratio of the widths of the p-channel to n-channel MOSFET can be approximated to $V_{inv,}$ increase in p-channel MOSFET width increases $V_{inv.}$ This approximation plays an important role in attaining required V_{inv} for cross-coupled inverters in memory cells.

Chapter 3

Static RAM Memory

3.0 Introduction

Semiconductor memories, particularly SRAM, are widely used in electronic system. The memory cells used in RAMs can be divided into static structures and dynamic structures. Static RAM structures use latched storages like cross coupled inverters, while dynamic RAM structures use dynamic storage of charge on a capacitor. Static RAMs tends to be much faster than the dynamic RAMs, but tend to be much larger in chip size area and are more costly per bit stored. Many efforts have been made to improve the efficiency of the SRAM, in terms of noise margin (or noise immunity), speed of the memory cell (read access time), and area (density) of the memory cell. There is always a tradeoff between all these constrains, the manufacturer has to sacrifice one constrain or the other depending on the application. For example; to optimize speed (read access time) for an application, manufacturer might have to sacrifice on his noise margin, there always exists a tradeoff.

3.1 Single Port-SRAM cell

The most commonly used in ASIC (Application Specific Integrated Circuit) memory is the 6-transistor cross-coupled inverter circuit as shown in the Fig.3.1. The data in the cell can be read from or written into, through the bitlines. On read/write memory, each port can be read-only, write-only, or capable of both read and write. There are many variations of these circuits to achieve varying density/speed/noise-margin requirements.

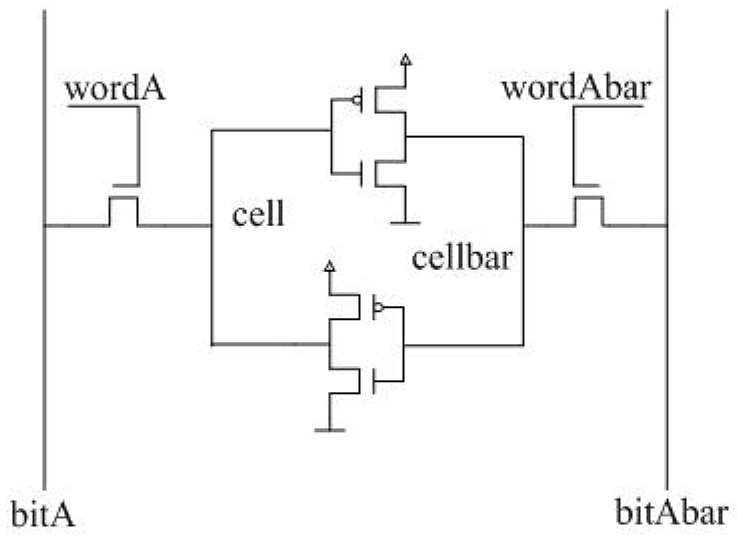

Fig.3.1 SRAM memory cell 6-transistor model

SRAM Memory cell retains data with a pair of cross coupled inverters (forming a latch); this is easy way of storing data. The inverter pair retains the data as long as there is no interruption of power supply (unlike dynamic RAMs, where data refreshing is necessary). In my convention number of access transistor (n-channel MOSFET) connected to cell or cellbar node define the number of ports of the cell. In this case it is a single-port memory cell, Fig.3.1 has one access transistor coupled each on cell and cellbar node, capable of simultaneous two single ended reads or one double ended write.

Typical small memory array architecture consists of 2^n words of storage of 2^m bits each. For example, to store a 8-bit value we need 8 SRAM cells in a row and to store 8, 8-bit values we need 8 rows of SRAM cell with each row having 8 SRAM cells (2^3 rows and 2^3 columns). Thus this memory array can store 2^6 or 64 bits.

Each wordline (wordA and wordAbar) is enabled through row decoders, through which an access transistor is made conductive, thereby accessing the location in the memory array. Bit and bitbar lines are connected to writedriver circuit to drive the large capacitance coupled to the memory array which hold bit and bitbar line low or high ("0" or "1" respectively). Writedrivers force a "1" or "0", when doing a write so that the data in the selected memory cell is overwritten with a stored data. Sense amplifiers are connected to each bitline to sense the change in the value on the bitlines during a read cycle. A simple inverter can be used to sense the change in the value on each bitlines enabling to do single ended reads.

The cell contains a pair of cross-coupled inverters and an access-transistor for each bitline. The nodes of the cross-coupled inverters are labeled as cell and cellbar. The gate of the access transistor is labeled as wordA on the cell node, and wordAbar on the cellbar node. The wordlines are asserted to read through or write into the cell by row decoders. This memory cell is capable of performing two simultaneous reads or one double-ended write through the ports available (2 read ports and 1 write port). The 6T cell achieves its compactness at the expense of more complex read write circuitry in the cells. This is a good tradeoff with shorter wires and hence low power consumption in large SRAM circuits.

3.1.1 Reading from SRAM cell.

SRAM memory cell drives the bit and bitbar lines during a read, memory cell has two complementary values stored on the cell and cellbar node which can be read out. Data from the memory can be read through the bitlines connected to the drains of the access transistor coupled to the nodes. Memory location can be accessed through the bitlines by enabling the access transistor, high or low on the bitlines is detected by the sense amplifier during a read cycle. In this arrangement it is possible to read two different values out of the memory cell and cellbar nodes. This is single-ended read on bit and bitbar lines. In some cases an analog differential amplifier is used to sense the difference in high and low on bit and bitbar lines to read the data out, this is not done in this design because we do a single ended read through the bit and bitbar lines.

The cross-coupled inverter arrangement enables us to read either a “0” or “1” from the complementary cell nodes on the memory. Reading a stored “0” from the memory cell node through the bitlines is an interesting situation compared to reading a “1” from the memory cell node, there is a possibility of overwriting stored “0”, wherein the large bitlines capacitance on the memory column provide large currents to raise the cell node voltage high “1”. The total bitline capacitance account for drain capacitance of the access transistors even when are not conducting, this situation must be avoided. For this criterion to overcome, the cell node voltage should be lower than the cross-coupled inverter switching threshold V_{inv}. Schematic Fig.3.2 shows the operating transistor while reading a stored “0” on cell node.

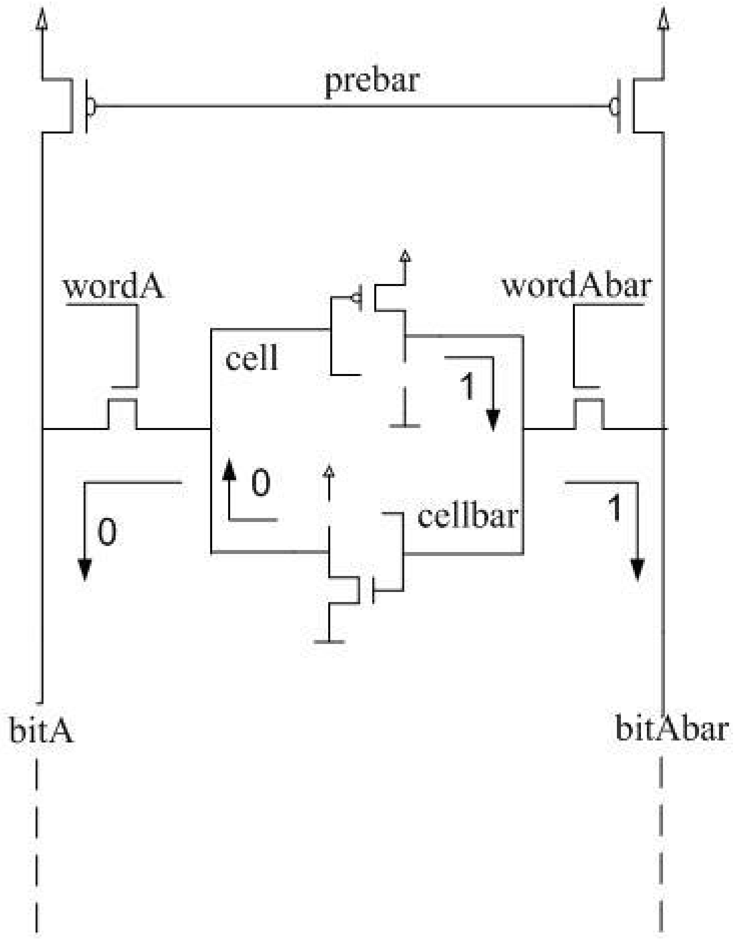

Fig.3.2 Schematic of operating transistors while reading "0" and "1"

Consider reading a "1" from the cellbar node through the bitbar lines, bitlines must be initialized (precharged) to some non-zero initial voltage before reading starts. This is a required condition for readability, to ensure bitbar lines are not low enough to overwrite the stored "1". And apart from this access transistor (n-channel MOSFET) through which we try to read a high "1" from the cellbar node is not good at passing a high "1" we read $V_{dd} - V_{Tn.}$ Thus cellbar node (storage node reading a "1") should be greater than inverter switching voltage V_{inv} to avoid the switching of the inverter state, in turn memory state. Schematic Fig.3.2 shows operating transistors while reading a stored "1" on cellbar node.

The key aspect of the precharge RAM read cycle is the timing relationship between the RAM address, the precharge pulse, and enabling the word decoder. If the word-line assertion precedes the end of the precharge cycle, the RAM cells on the active wordline will see both bit lines pulled high and the RAM cells may flip state (overwriting the stored value in the memory cell). Thus, for reading precharge the bitlines high and then enable the word-line decoder. The precharge circuit (bit or bitbar pull up circuit) may use a p-channel MOS transistor or an n-channel MOS transistor for increasing the read access time of the memory but at the expense of V_{Tn} [2].

3.1.2 Writing into SRAM cell

Writing into SRAM cell, is to store data in memory cell node, by raising the node voltage high or low (desired value). The large bitline capacitance and the memory cell are both trying to drive the bit and bitbar lines. The row address decoders are enabled to turn the access transistor ON, trying to force a value onto the cell or cellnode. Access transistors being an n-channel MOSFET are good at passing a “0”. Trying to force a high “1” on the cell node through an access transistor is not a good idea, as it would pass $V_{dd} – V_{Tn}$. For writing a “1” into the cell, the best alternative is to pass a “0” through the cellnode node (complementary node) enabling the “1” through cross coupled inverters on the cell node. This is referred to as a double-ended write.

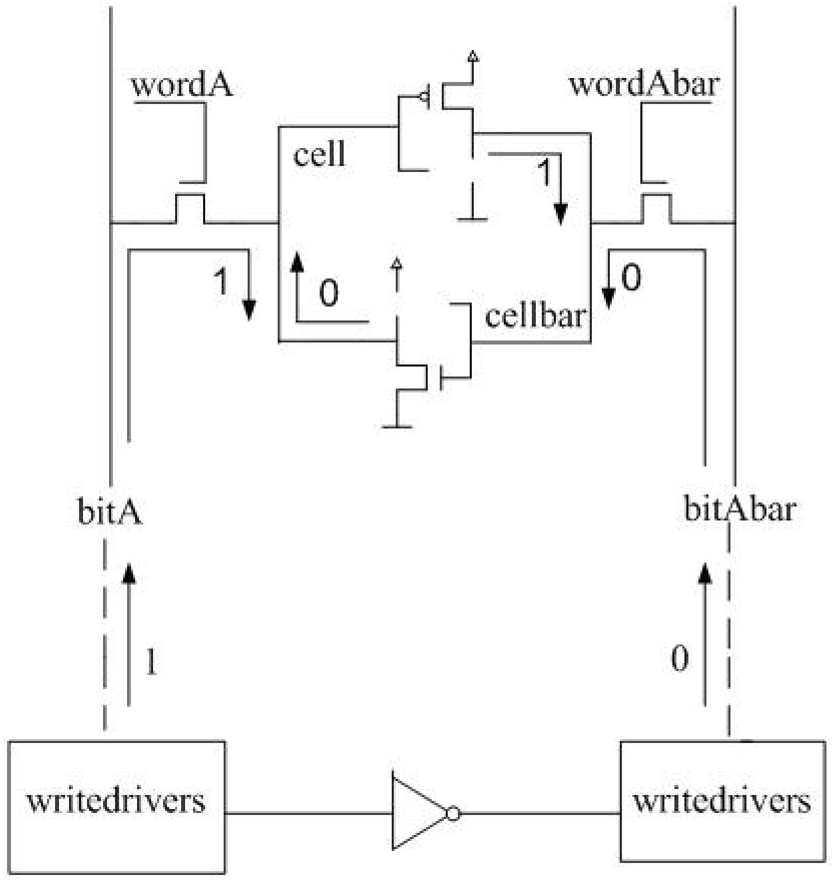

Fig.3.3 Schematic of operating transistors while writing a "1" over stored "0".

Writing a "0" is not a serious situation as access transistor is good at passing "0" and this can be done by enabling the gate of the transistors by row decoders of that memory location, but writing a "1" is. Therefore a complementary drive on bit and bitbar lines is necessary so that a strong "0" appears on either node of the cross coupled inverter. We always design the memory cell in such a way that a "1" on the bitline can never overwrite a "0" in the storage node and a "0" on the bitbar lines will always overwrite a "1" in the storage node. For this criterion to satisfy, the memory node voltage should be lower than the cross-coupled inverter switching voltage V $_{\mathrm{inv.}}$ To store a value in memory cell (in other words overwrite a previously stored data) the writedriver must be sized to drive a strong “0” to flip the data state in the cross coupled inverters. Fig.3.3 shows the schematic of the transistor that are operating while writing a “1” over a stored “0”.

3.2 Multi-Port SRAM cell

Multi-port SRAM cells are capable of writing into or reading from the cell through “m” (more than one) number of ports (access transistors) coupled to the cell node. From the earlier discussion of single-port SRAM cell, memory cell requires bit and bitbar lines to do a data write into the cell and read from the cell through the bitlines. With the standard SRAM cell (Fig. 3.1) it is possible to either do two simultaneous reads (bit and bitbar lines to read different data) or write one datum (bit and bitbar lines must have the same datum). Memory’s are used in processors, which require an interface to the registers that can quickly read two input operands (source operands) and one output operand (destination operand). If pipelining is done so that simultaneous reads and writes are not necessary, then a register file based on the static RAM with 2 simultaneous read or 1 double-ended write is adequate for interfacing single processing unit.

More advanced computer processor design requires that several processing units should be connected to the same register file. One approach has been to partition the register file into parts that are dedicated to each hardware units. The problem is many instructions and much time is wasted in moving the data between registers. The true advantage of processor design are achieved more efficiently when many inputs and outputs are provided to each register in the register file so that many of the hardware units can use several registers simultaneously. The simplest thing to do is to add additional pair of bitlines as in a normal multi-port SRAM [6], [7].

A multiport SRAM can be designed simply by adding access transistors to the memory cell by coupling them to the cell and cellbar nodes. There by increasing the read ports by number of access transistors connected to the nodes and write ports, half the number of transistors connected. To generalize, say if they're "m" pair of access transistors connected to the memory node, then we have "2m", read ports and "m" write ports in memory cell. This holds only if memory cell is accessed with a double ended write and a single ended read. With an addition of each pair we get additional two reads or one write. For example, in Fig.3.4 "m=4", this would give access to 8 simultaneous reads or 4 double ended writes to the memory cell. The register file of 4 pair of bitlines or 8 bitlines would usually operate with reads done on all the bitlines at the same time and writes are done at the same time, never a mixture of read and write [8].

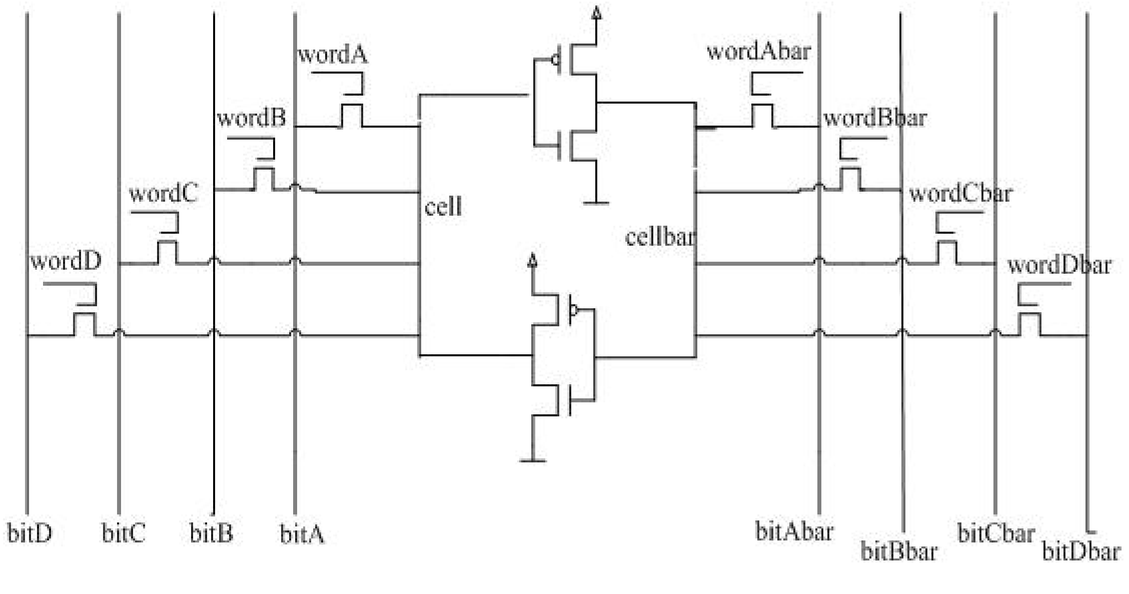

Fig.3.4 Four-port SRAM memory cell schematic

This increase in the access to the memory cell decreases the stability of the cell, due to the added capacitance to the nodes from the "m" source of the access transistors. This in turn decreases the attainable noise margin (discussed in section 3.3) for the cell, there by decreasing immunity of the cell to the noise from various sources (power supply noise, interconnect cross capacitance noise, charge leakage noise, and mutual inductance noise). My work would concentrate on better designs of multiport SRAM that would sustain a reasonable amount of noise from the power supplies. To attain this transistor must be carefully sized to ensure correct operation of the memory cell.

3.2.1 Reading from a Multi-Port SRAM cell

Criterion for reading from the multi-port SRAM is the same as reading from a single port SRAM cell.With increase in numer of ports memory cell access for reading from a same location through all the available bitlines is acceptable and possible, which is not possible in a single port SRAM Memory cell. But in this situation, the cell and cellbar node have to drive all the precharged high capacitance bitlines simultaneously. Since all the bitlines are precharged high before reading, reading a “0” from the memory cell is the worst case scenario. This situation must be taken care as to avoid precharged capacitive bitline not to overwrite a stored “0” in the memory cell. This high capacitance is capable of supplying a current, which can flip the state of the memory state. The n-channel MOSFET in the cross-coupled inverter would try to pull down the “m” bitlines through the access transistors. This might overwrite the stored “0” in the memory cell. By reading a “0” from the memory cell from all the bitlines, the bit-lines would try to discharge through n-channel MOSFET in the memory cell. For this, the n-channel MOSFET should be wide enough to handle this large current. This implies that as the number of ports increase in a multi-port SRAM the size of the n-channel MOSFET increases accordingly.

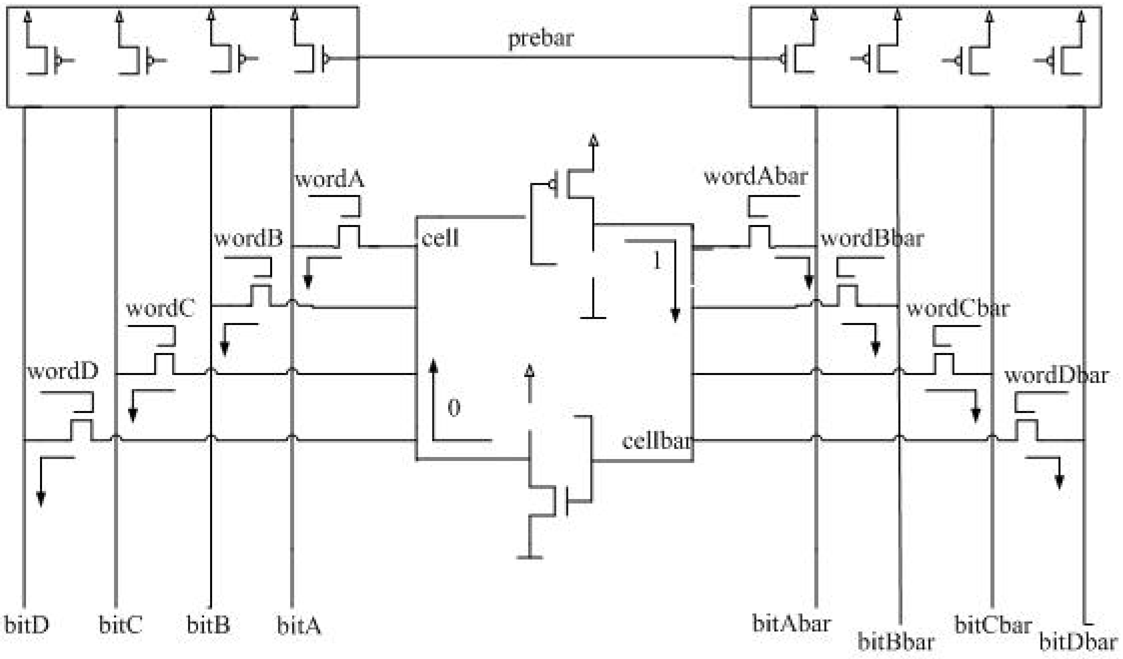

Fig.3.5 Schematic of operating transistors while reading "0" and "1" from cell and cellbar node respectively, through all the ports.

An analytical solution is derived for the cell node voltage in this situation. Considering a case with m = 4; four access transistors coupled to the cell node conduct and operate in saturation region and the pull down n-channel MOSFET in the cross coupled inverter conduct in ohmic or linear region (refer appendix A),[8]. We have,

$$(linear) I_{DSn,lin} = \frac{I_{DS,lin,R_{ds=0}}}{1+\frac{R_{ds}.I_{DS,lin,R_{ds=0}}}{V_{DS}}}$$

$$(saturation) I_{DSA,sat} = \frac{\varepsilon_{ox}}{t_{ox}}.W.v_{sat}.\left[\left(V_{GS}-V_{Tn}\right)-A_{bulk}.|V_{DSsat}|\right]\left(1+\frac{V_{DS}-V_{DSsat}}{V_A}\right)$$

With "n" referring to n-channel MOSFET of the pulldown cross coupled inverters and "A" refers to access transistor. The equation to be solved;

$$I_{DSn} = 4 * I_{DSA} \qquad \text{--- 3.1}$$

This would give a 3rd order equation in two variables V_{Cell} and W_A / W_n. For known values of W_A and W_n we can calculate the value of V_{Cell}.

As mentioned above as the number of ports increases the width of the n-channel MOSFET increases to drive the large capacitance on the bitlines while reading memory nodes. But, increasing the width of an n-channel MOSFET decreases the threshold voltage of the inverter thereby decreasing the noise margin. This is a serious problem to be taken care in general applications susceptible to noise, so we propose various designs to meet both constrains for the multiport SRAM design and to achieve reasonable noise margin for used technology.

3.2.2 Writing into a Multi-Port SRAM cell

Criterion for writing into the multi-port SRAM is the same as writing into a single port SRAM memory cell. Irrespective of number of ports the writes are always double ended writes into a memory cell. Writing into a memory cell would mean writing into a memory location through bit and bitbar line with writedrivers forcing a value into the memory. And writing into a same memory location through 2 or more different bitlines would result in an unpredictable result and this should be avoided. That is a write into memory location is possible with a double ended write through only one pair of bitlines (say bitA and bitAbar). Overwriting a stored “0” with “1” on a memory node is a critical case to be considered because of the same reason that prevailed in single-port SRAM, the access transistor is good at passing a “0” and poor at passing a “1”. To ensure a “0” be overwritten with a “1”, we pass a “0” through the access transistor (good at passing a

"0") at the complementary cell node. Fig.3.5 shows schematic of operating transistors conducting when passing a "0" through cellbar node to overwrite a stored "1".

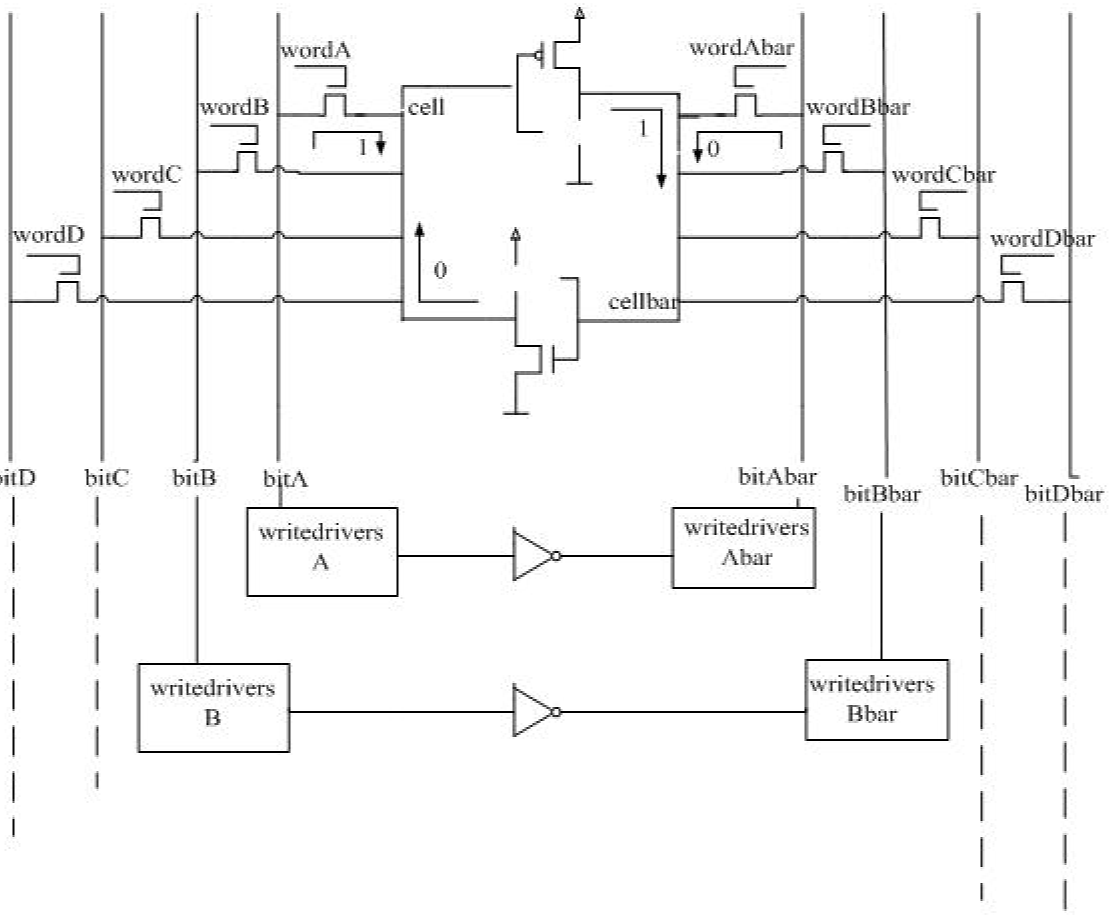

Fig.3.6 Schematic of operating transistors, overwriting a stored "0".

Let's assume initially the cell node and cellbar node in memory cell has "0" and "1" stored respectively in it. Overwriting a stored "0" in the cell node can be accomplished by overwriting a stored "1" on the cellbar node; this is double ended write. In this situation the write-drivers try to force a "0" into cellbar node, eventually the drain-source voltage of the access transistor connected to this node would be small and operate in ohmic region and the p-channel MOSFET in the pull-up has a large voltage, operating in saturation region. Thus, the access transistor is more conductive and forces a “0” in cellbar node. Analytically this can be expressed as;

$$(linear)I_{DSA,lin} = \frac{I_{DSA,lin,R_{ds=0}}}{1 + \frac{R_{dsA}.I_{DSA,lin,R_{ds=0}}}{V_{DSA}}}$$

$$(saturation)I_{DS,sat} = \frac{\varepsilon_{ox}}{t_{ox}}.W.v_{sat}.\Big[\big(|V_{GS} - V_{Tp}|\big) - A_{bulk}.|V_{DSsat}|\Big]\left(1 + \frac{|V_{DS} - V_{DSsat}|}{V_A}\right)$$

$$(linear)I_{DSA} = I_{DSp}(saturation) \qquad \text{--- 3.2}$$

This quadratic equation has two variables V $_{\text{Cellbar}}$ and W $_{\text{p}}$ / W $_{\text{A}}$, known parameters and widths of the transistors are substituted to solve unknown V $_{\text{Cellbar}}$.

3.3 Noise Margin in a Memory cell

Noise margin in a memory cell is different from the noise margin of an inverter (section 2.3), they are related in terms of noise immunity for a device, but are defined differently. Noise margin of a memory cell is maximum noise voltage (disturbance) that a cell can accept without flipping the memory state, through the bitlines or the power lines.

Noise margin (NM) is calculated mathematically as the difference of inverter threshold voltage (cross-coupled inverter) to the memory cell node voltage. We have read noise margin and write noise margin, read noise margin is calculated as the difference of inverter threshold voltage to the memory cell node voltage (node reading a “0”).

$$NM_{read} = V_{inv} - V_{cell,read-node} \qquad \text{--- 3.3}$$

Write noise margin is calculated as the difference of inverter threshold voltage to the memory cell write node voltage (complementary node).

$$NM_{write} = V_{inv} - V_{cell,write-node} \qquad \text{--- } 3.4$$

Note: the cell read-node and write-node mentioned in the mathematical equations for noise margin calculations imply to the node through which memory is read or written into. Recall, irrespective of number of ports the writes are always double ended writes into a memory cell. Hence, we call write-node in the mathematically equation for write noise margin as a complementary node. In general, we calculate noise margin with respect to the maximum of the two node voltages cell and cellbar.

In general, we would like to have maximum noise margin possible for any memory cell irrespective of number of ports. But, in practice there is a limitation for achievable noise margin for a given memory cell (single or multi-port). From the equations 3.3 and 3.4, we observe that for attaining maximum noise margin we would like to have large inverter threshold voltage V_{inv} than the cell node voltage. Recall that, for attaining large V_{inv} in a memory cell we would like to have large p-channel MOSFET. Consider the case of a Multi-port SRAM; the design constrains to scale the width of n-channel MOSFET in cross-coupled inverter to drive the large bitline capacitance for a stable memory cell cannot be neglected. By satisfying these conditions we obtain an optimum noise margin. Optimum noise margin depends on the technology, for example; for a 5V, 0.6 μm technology, acceptable noise margin would be in the range of 1V to 2V as compared to 0.3V to 0.5 V noise margin for a 1.3V, 0.25 μm technology. For any SRAM designer, the best interest would be to attain an optimum noise margin with a reasonable speed (read access time). Note that there always exists a trade-off between noise margin and read access time of a memory cell. For an application, to optimize speed (read access time) manufacturer might have to sacrifice on his noise margin and vice-versa.

3.4 Summary

This chapter has extensively dealt with single-port and multi-port SRAM design, stability of the memory cell, necessary criterion for read and writes and noise margin of the memory cell. As a remainder, writing into multi-port SRAM cell is very similar to writing into a single port SRAM cell. Reading of the multi-port SRAM cell is different from the single-port SRAM cell only when we try to read the same location through all the ports. In the design perspective NMOS transistor in the cross coupled inverter should be large enough to handle the capacitance associated with reading through all the ports.

Chapter 4

8-Port SRAM Memory cell

4.0 Introduction

In the previous chapter, we extensively discussed about single port SRAM and multiport SRAM memory cell, their stability issues, and readability and writeability criterions. As an example, we considered 4-port SRAM cell, and studied its transistor level behavior. In this chapter, we will apply the same criterion discussed for multiport SRAM on 8-port SRAM memory cell. We will extensively study about; how the bitlines capacitance effect the readability and writeability of the memory cell, and we propose various designs of 8-port cells capable of simultaneous 8 writes or 16 reads (the designs defer in how the capacitance is distributed in the memory cell, still preserving the idea of basic inverter loop), and compare their performances with respect to noise margin. You will be fascinated to know, at the end of the chapter how easy it is to control the design aspects of the 8-port memory cell.

4.1 8-Port Memory cells

As mentioned in section 3.2, multiport SRAM can be designed simply by adding required number of access transistor on either node to the memory cell. This is one possible way of designing a multiport SRAM, but not the best design possible as we would see in the later part of the chapter. The main issue in designing a multiport SRAM, is the bitline capacitance and the drain capacitance of the access transistor, coupled to the memory cell array, which would make up to an order of few femto farads; >240fF (1fF $=10^{-15}$F) in case of 32 word 8-port SRAM memory array. The drain capacitance coupled to the bitlines cannot be neglected, because the effect of drain capacitance is felt even if the transistor is in cutoff region. A simple 8-port SRAM design schematic is as shown in the Fig 4.1.

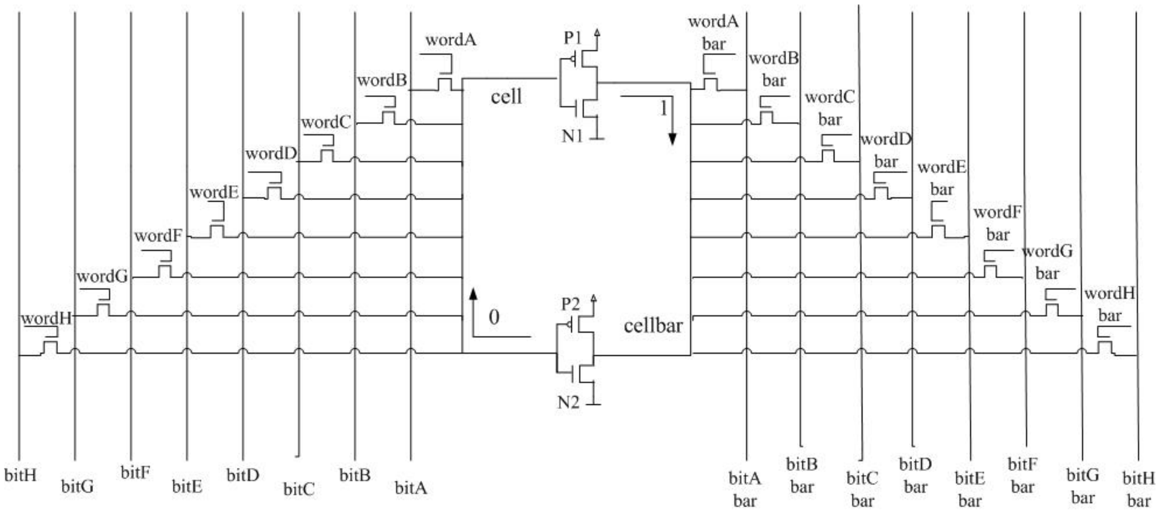

Fig.4.1 Schematic of 8-Port SRAM Memory Cell

As from the schematic of 8-port SRAM memory cell, it is clear that the large capacitance at the cell and cellbar node and the bitline capacitance along with drain capacitance associated with the access transistor is of high magnitude, read access time, noise margin of the cell are dependent on how this large capacitance is charged and discharged in the memory array [7]. Typical small memory array architecture consists of 2^{n} words of storage of 2^{m} bits each. For example, to store a 32-bit value we need 32 SRAM cells in a row and to store 32, 32-bit values we need 32 rows of SRAM cell with each row having 32 SRAM cells (2^{5} rows and 2^{5} columns). Thus this memory array can store 2^{10} or 1024 bits.

Consider a case, where in the memory is accessed through all the ports simultaneously, for example: trying to access the memory cell in Fig.4.1 through all the ports available on the cell node or cellbar node simultaneously. In this situation as it is evident from the schematic, memory cell has to drive all the bitlines connected to the node, for this to be possible theoretically we would prefer to have proportionally large n-channel MOSFET to the number of ports [7]. Given the liberty in terms of silicon area, designer would prefer to have an infinitely large n-FET to improve the performance of the memory cell. But, such a large FET is not practical, not just in terms of the silicon area available, but also the speed of the memory would have to be sacrificed. For this, we have to consider few design issues to overcome this large capacitance with an optimum n-FET width. For which, we propose four designs for 8-port SRAM memory cell, capable of 16 reads or 8 writes simultaneously. These designs vary with the distribution of capacitance on the memory node, with preserving the idea of inverter loop in the memory cell. The four 8-port SRAM designs are named as follows:

- 2-SRAM
- 4-SRAM
- 8-SRAM

- J-SRAM

Typical SRAM circuits have two inverters (cross coupled) in the memory cell, in the proposed design we have inverters forming a closed loop-a latch like formation within which the data is stored in the memory. In the 2-SRAM model of 8-port memory cell, 2 inverters are cross-coupled similar to typical memory cell with 8 access transistors or pass transistors (referred by some authors) on either side of the memory nodes namely cell and cellbar, schematic of which is shown in Fig.4.1. In the 4-SRAM model of 8-port memory cell, 4 inverters are coupled to each other forming a latch as shown in Fig.4.4; where in the effective 8-ports of the memory cell are accessed with 4 access transistors coupled on each available node (namely ABCD, ABCDbar, EFGH, and EFGHbar) on the memory. In the 8-SRAM model of 8-port memory cell, 8 inverters are coupled as shown in Fig.4.5; where in effective 8-ports are accessed with 2 access transistors coupled on each available node on the memory. And J-SRAM model is designed more effectively and is innovative of all the designs. This model is named after Dr.L.G.Johnson (Dr.J's design). This design has essentially 6 inverters in the memory cell, where in the inverters in the block effectively form buffer like structure, and the other inverters are designed in such a way that that the memory can handle the capacitance and the resistance on the cell and cellbar node, schematic of which is shown in Fig.4.6. Notice that the distribution of access transistors on the memory nodes is similar to 2-SRAM model, but at the end of chapter you find, this design is more effective than the other designs proposed. Its performance is better in terms of noise margin and read access time, when compares with all other designs.

4.2 2-SRAM, 8-port static memory cell

2-SRAM model with 20 transistors, has 2 inverters in the memory cell coupled to each other as cross-coupled inverters (refer to Fig.4.1). Cross coupled inverters form a latch like structure and stores the data in the memory. In this design the memory nodes are labeled as cell and cellbar, to which the access transistors (n-channel MOSFETs) are connected, 8 each on cell and cellbar nodes. The gates of which are labeled as follows; on cell node the gates are labeled as wordA, wordB, wordC, wordD, wordE, wordF, wordG and wordH. And on the cellbar node the access transistor gates of which are labeled as wordAbar, wordBbar, wordCbar, wordDbar, wordEbar, wordFbar, wordGbar and wordHbar. These gates are enabled to access the memory, i.e. either to read from or write into the memory locations, memory is enabled to do a single ended read or double ended write (refer to section 3.2.1 and 3.2.2.

In a typical register file with memory array, sense amplifier, row decoders, column decoders, and precharge. Each location in a memory array can be accessed through row decoders connected to the gates of the access transistors. The drain of the access transistor are connected to the bitlines (data lines), the bitlines are labeled as shown in the Fig 4.1. Bitline connected to the drain of the transistor (with gate, wordA) is labeled as bitA. Similarly the rest of the bitlines on the cell node side are labeled as bitB, bitC, bitD, bitE, bitF, bitG and bitH. And complementary bitlines on the cellbar node are labeled as bitAbar, bitBbar, bitCbar, bitDbar, bitEbar, bitFbar, bitGbar and bitHbar.

Design criteria for reading from and writing into a multiport SRAM as described in section 3.2.1 and 3.2.2 are applicable for the design of 2-SRAM model as well. Schematic of the model while reading through all ports from the memory is shown in

Fig.4.2. for better analysis and understanding of the model, consider the worst case scenarios while reading from the memory is, reading a “0” from a multiport SRAM through all ports (namely bitA, bitB, bitC, bitD, bitE, bitF, bitG and bitH) simultaneously, this is because, in this situation all the capacitance existing on any given node is considered, and all the bitlines are precharged high before reading begins, so there is a possibility of overwriting the stored memory data. So to avoid, the voltage on the memory node must be smaller than the inverter threshold voltage to avoid overwriting the “0” stored. We can calculate cell node voltage by considering the regions of operation of transistors, wherein the access transistor is in saturation and the n-channel MOSFET is in linear mode of operation. With reference to appendix A, sections 3.2.1 and 3.2.2, equating the source-drain current equation of the access transistor and n-channel MOSFET, we can calculate the cell node voltage has follows:

$$(linear) I_{DSn,lin} = \frac{I_{DS,lin,R_{ds=0}}}{1+\frac{R_{ds}.I_{DS,lin,R_{ds=0}}}{V_{cell}}}$$

$$(saturation) I_{DSA,sat} = \frac{\varepsilon_{ox}}{t_{ox}}.W.v_{sat}.\left[\left(V_{dd}-V_{cell}-V_{Tn}\right)-A_{bulk}.|V_{DSsat}|\right]\left(1+\frac{V_{dd}-V_{cell}-V_{DSsat}}{V_A}\right)$$

$$I_{DSn} = 8 * I_{DSA} \qquad \text{--- 4.1}$$

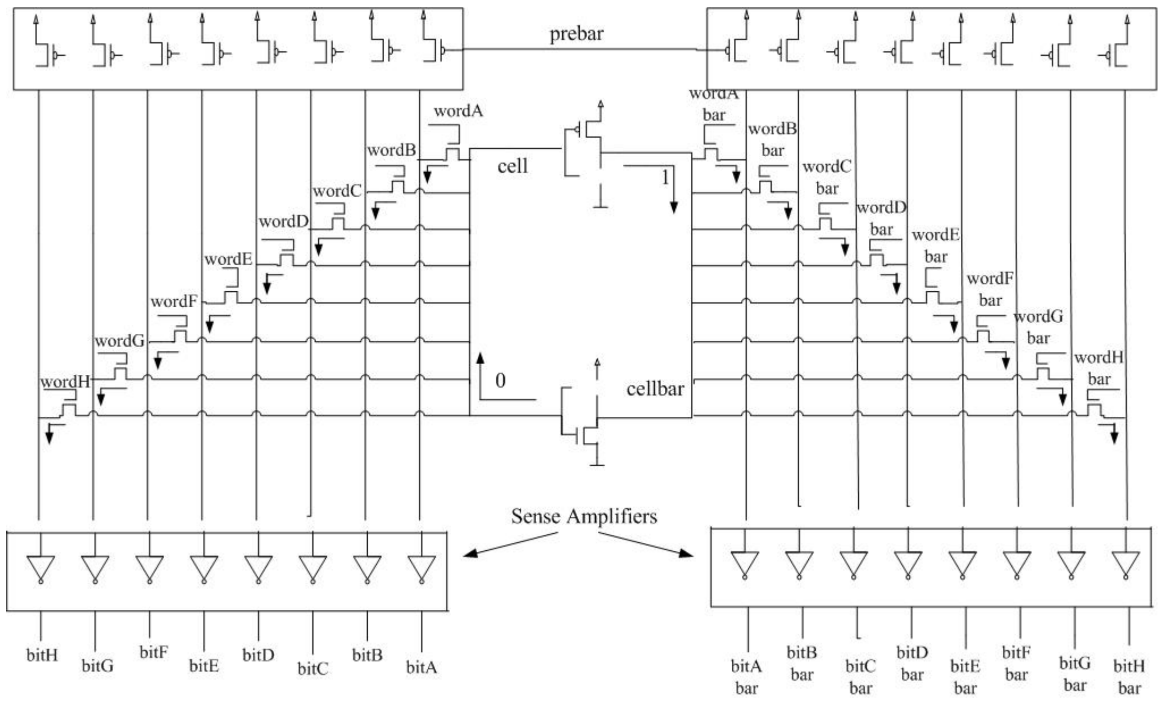

Fig.4.2 Schematic of operating transistors while reading “0” and “1” from cell and cellbar node respectively, through all the ports.

Writing criteria in an 8-port SRAM or multiport SRAM is the same, irrespective of the number of ports available. This is because; it is possible only to write into one memory location by double ended write at one time, in a memory array. Trying to write into a memory location through more than one port (double ended write) would result in trying to store some junk data, which neither the memory can differentiate between a high and low nor the user. Schematic of writing a “1” into the memory cell node is shown in the fig 4.3. Cellbar voltage can be calculated from the given equations substituting for V_{GS} and V_{DS}. In this case, the access transistor is in linear mode of operation and the p-channel MOSFET is in saturation.

$$(linear) I_{DSA,lin} = \frac{I_{DSA,lin,R_{ds=0}}}{1 + \frac{R_{dsA} . I_{DSA,lin,R_{ds=0}}}{V_{cellbarA}}}$$

$$(saturation) I_{DSp,sat} = \frac{\varepsilon_{ox}}{t_{ox}} . W . v_{sat} . \left[\left(| V_{dd} - V_{Tp} | \right) - A_{bulk} \cdot | V_{DSsat} | \right] \left(1 + \frac{| V_{dd} - V_{cellbar} - V_{DSsat} |}{V_A} \right)$$

$$(linear) I_{DSA} = I_{DSp} (saturation) \qquad \text{--- 4.2}$$

V $_{cell}$ and V $_{cellbar}$ are calculated from the equations 4.1 and 4.2 respectively. And inverter threshold voltage of the memory cell is calculated from the equation 2.5, in chapter 2. I have used mathematica for calculating V $_{cell,}$ V $_{cellbar}$ and V $_{inv}$ the procedure and results of which, are shown in appendix D.1 Notice that in equation 4.1 the number "8" on the right hand side of the inequality relates to the number of access transistors connected to the memory node.These equations can be generalized to calculate V $_{cell}$ for any multiport SRAM cell as shown under, where in "m" denote number of ports in my convention:

$$(linear) I_{DSn,lin} = \frac{I_{DS,lin,R_{ds=0}}}{1 + \frac{R_{ds} . I_{DS,lin,R_{ds=0}}}{V_{DS}}}$$

$$(saturation) I_{DSA,sat} = \frac{\varepsilon_{ox}}{t_{ox}} . W . v_{sat} . \left[\left(V_{GS} - V_{Tn} \right) - A_{bulk} \cdot | V_{DSsat} | \right] \left(1 + \frac{V_{DS} - V_{DSsat}}{V_A} \right)$$

$$I_{DSn} = m * I_{DSA} \qquad \text{--- 4.3}$$

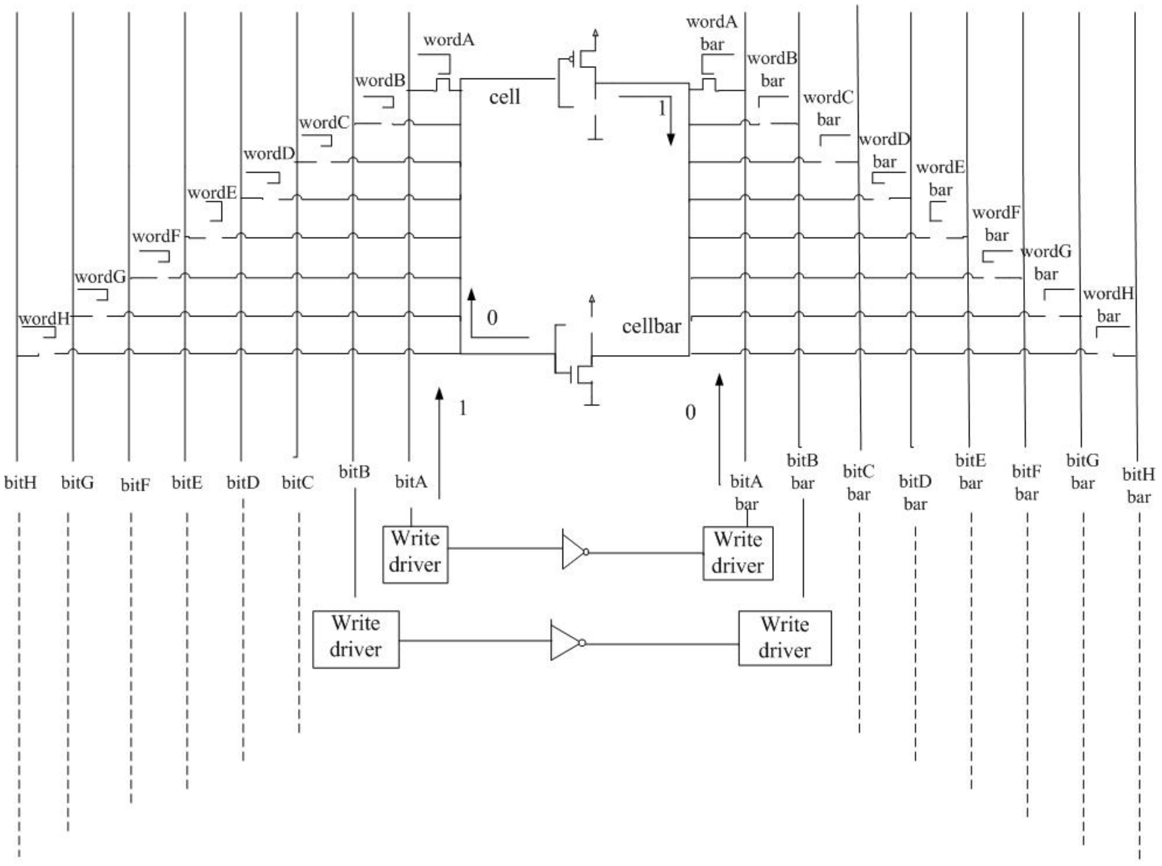

Fig.4.3 Schematic of operating transistors, overwriting a stored "0".

Solving this equation, by replacing gate-source, and source-drain voltages of access transistor and source-drain voltage of the n-channel MOSFET in the inverter loop (in this case transistors N2) to the corresponding voltages would give us V $_{\text{cell.}}$ and we get V $_{\text{cellbar}}$ by solving the below equation:

$$(linear)I_{DSA,lin} = \frac{I_{DSA,lin,R_{ds=0}}}{1+\frac{R_{dsA}.I_{DSA,lin,R_{ds=0}}}{V_{DSA}}}$$

$$(saturation)I_{DS,sat} = \frac{\varepsilon_{ox}}{t_{ox}}.W.v_{sat}.\left[\left(|V_{GS}-V_{Tp}|\right)-A_{bulk}.|V_{DSsat}|\right]\left(1+\frac{|V_{DS}-V_{DSsat}|}{V_A}\right)$$

$$(linear)I_{DSA} = I_{DSp}(saturation) \qquad \text{--- 4.4}$$

Now that we have, $V_{cell,}$ $V_{cellbar}$ and V_{inv} it's easy for us to calculate the noise margin of the memory cell. Noise margin (NM) is calculated mathematically as the difference of inverter threshold voltage (cross-coupled inverter) to the maximum of memory cell node voltages ($V_{cell,}$ $V_{cellbar}$). The results calculated from the theoretical equations are listed in table 6.2 and the simulations results (cadence SPECTRE simulator) are listed in table 6.1 refer to appendix C.2, for the simulation test bench and netlist generated from the layout of the design. The design layout is made using CADENCE virtuoso layout editor with AMI 06 µm technology. The layout of which is shown in Fig.6.1

4.3 4-SRAM, 8-Port Memory cell

4-SRAM model of 8-port static memory cell with 24 transistors has four inverters in the memory cell; these inverters are connected to each other in a closed loop, forming latch-alike structure enabling to store data in the memory. This cell stores data as long as power supply is ON, like any other volatile memory elements (SRAM). The nodes of the memory cell are labeled as ABCD, ABCDbar, EFGH, and EFGHbar, each node is connected with 4 access transistors or pass transistors and is shown in the Fig 4.4. The gates of the access transistors connected to node "ABCD" are labeled as wordA, wordB, wordC, and wordD, the gate of the transistors to node "ABCDbar" are labeled as wordAbar, wordBbar, wordCbar, and wordDbar, the gates of the transistor to node EFGH are labeled as wordE, wordF, wordG, and wordH and finally the transistor gates to node EFGHbar are labeled has wordEbar, wordFbar, wordGbar and wordHbar. And the bitlines connected to corresponding drains of the access transistor are labeled as bitA, bitB, bitC, bitD, bitE, bitF, bitG, bitH, bitAbar, bitBbar, bitCbar, bitDbar, bitEbar, bitFbar, bitGbar and bitHbar.

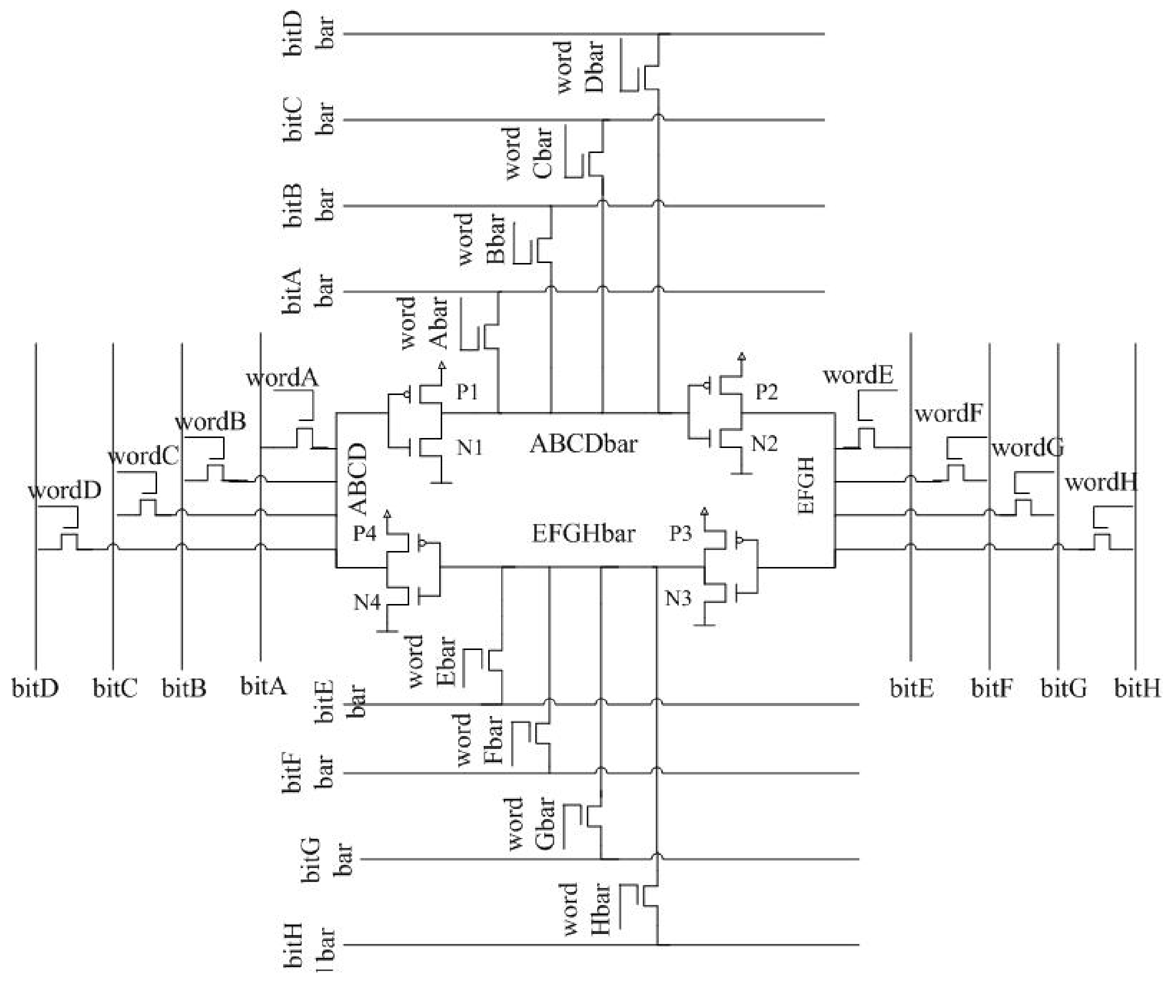

Fig.4.4 Schematic of 4-SRAM model of 8-port memory cell.

Design criterion for reading from and writing into the memory cell discussed in section 3.2.1 and 3.2.2 are applicable in this case as well with some changes with respect to the number of ports available on each node. The difference in this design and the latter design (2-SRAM) is evident from the schematics of both; it's how we distribute the capacitance on the nodes of the memory cell. In this design, we couple 4 inverters in a closed loop, enable or disable access transistors to do a read or write from 16 read ports and 8 write ports. With this form of arrangement, memory node would affectively feel the presence of 4 access transistors which can be enabled to read from or write into the memory. The transistors P1, P2, P3, P4, N1, N2, N3, N4 and access transistors are sized accordingly to satisfy the read and write criterion of the memory cell [6].The read and write criterions discussed for multiport SRAM design is applicable, from the generalized equations for memory nodes (refer to section 4.2). We get, V_{ABCD} by solving equation 4.5, the operating transistors are with gates labeled as wordA, wordB, wordC, wordD, N4, P1, N2, and P3 (refer Fig.4.4) Node voltage ABCD, when reading from the memory through 4 ports simultaneously is given as under:

$$(linear) I_{DSn,lin} = \frac{I_{DS,lin,R_{ds=0}}}{1 + \frac{R_{ds} . I_{DS,lin,R_{ds=0}}}{V_{ABCD}}}$$

$$(saturation) I_{DSA,sat} = \frac{\varepsilon_{ox}}{t_{ox}} . W . v_{sat} . \left[\left(V_{dd} - V_{ABCD} - V_{Tn} \right) - A_{bulk} . | V_{DSsat} | \right] \left(1 + \frac{V_{dd} - V_{ABCD} - V_{DSsat}}{V_A} \right)$$

$$I_{DSn} = 4 * I_{DSA} \qquad \text{--- 4.5}$$

And, we get $V_{ABCDbar}$ by solving equation 4.6, where in the access transistor operates in linear mode and the p-channel MOSFET in the inverter loop operates in saturation. In this case, say when trying to write into memory through access transistor wordA and wordAbar (being double ended write) on node ABCDbar (complementary node- forcing a “0”) the operating transisitor are P1, N2, P3, N4 and access transistor with gates wordA and wordAbar (refer Fig.4.4).

$$(linear) I_{DSA,lin} = \frac{I_{DSA,lin,R_{ds=0}}}{1 + \frac{R_{dsA} . I_{DSA,lin,R_{ds=0}}}{V_{ABCDbar}}}$$

$$(saturation) I_{DSp,sat} = \frac{\varepsilon_{ox}}{t_{ox}} . W . v_{sat} . \left[\left(| V_{dd} - V_{Tp} | \right) - A_{bulk} . | V_{DSsat} | \right] \left(1 + \frac{| V_{dd} - V_{ABCDbar} - V_{DSsat} |}{V_A} \right)$$

$$(linear) I_{DSA} = I_{DSp} (saturation) \qquad \text{--- 4.6}$$

Similarly, all other node voltages can be calculated from the same equations, fortunately voltages are ought to be same, with access transistors of same dimensions on all the ports. Now that we have $V_{ABCD,}$ $V_{ABCDbar,}$ and V_{inv} available, we can calculate noise margin and compare it with simulation results. The theoretical and simulation results are tabled in 6.4 and 6.3 respectively, and the simulation test bench for SPECTRE and netlist generated from CADENCE VIRTUOSO LAYOUT EDITOR can be referred from the appendix C.3. Layout of which is shown in Fig 6.3

4.4 8-SRAM, 8-port memory cell.

In the design of 8-port SRAM memory cell with 32 transistors, memory cell has 8 inverters coupled to each other forming a closed loop, so the name 8-SRAM. Each input and outputs nodes of an inverter are labeled as memory nodes, to which access transistors (pass transistors) are connected as shown in Fig.4.5. Memory nodes are labeled as AB, ABbar, CD, CDbar, EF, EFbar, GH, GHbar respectively, access transistors connected to these nodes with their drain connected to the bitlines and source connected to memory node, the gates of which are labeled as wordA, wordB, wordC, wordD, wordE, wordF, wordG, wordH, wordAbar, wordBbar, wordCbar, wordDbar, wordEbar, wordFbar, wordGbar, and wordHbar. The bitlines connected to the memory node AB through the drain of the access transistors are labeled as bitA and bitB. Similarly, the rest of the bitlines are labeled as bitAbar, bitBbar, bitC, bitD, bitCbar, bitDbar, bitE, bitF, bitEbar, bitFbar, bitG, bitH, bitGbar, and bitHbar as shown in the schematic. We would consider the two worst case scenarios, reading simultaneously from all the available ports on the memory node and writing a “0” over a stored “1” (double ended write). Looking at these scenarios we find that for readability and writability into an 8-SRAM memory cell is same with any multiport SRAM cell so considering the generalized equations 4.3 and 4.4, consider, reading from the memory node AB through access transistor with gates wordA and wordB coupled. The operating transistors while reading through the access transistors simultaneously are N8, P1, N2, P3, N4, P5, N6, and P7 in the memory cell, and the access transistors with gates wordA and wordB coupled to the memory node AB (refer to Fig.4.5). At node AB, the access transistors are in saturation and the transistor N8 is in linear mode of operation. Node voltage V_{AB} can be calculated by solving equation 4.7.

$$(linear) I_{DSn,lin} = \frac{I_{DS,lin,R_{ds=0}}}{1 + \frac{R_{ds} . I_{DS,lin,R_{ds=0}}}{V_{AB}}}$$

$$(saturation) I_{DSA,sat} = \frac{\varepsilon_{ox}}{t_{ox}} . W . v_{sat} \cdot \left[\left(V_{dd} - V_{AB} - V_{Tn} \right) - A_{bulk} \cdot | V_{DSsat} | \right] \left(1 + \frac{V_{dd} - V_{AB} - V_{DSsat}}{V_A} \right)$$

$$I_{DSn} = 2 * I_{DSA} \qquad \text{--- 4.7}$$

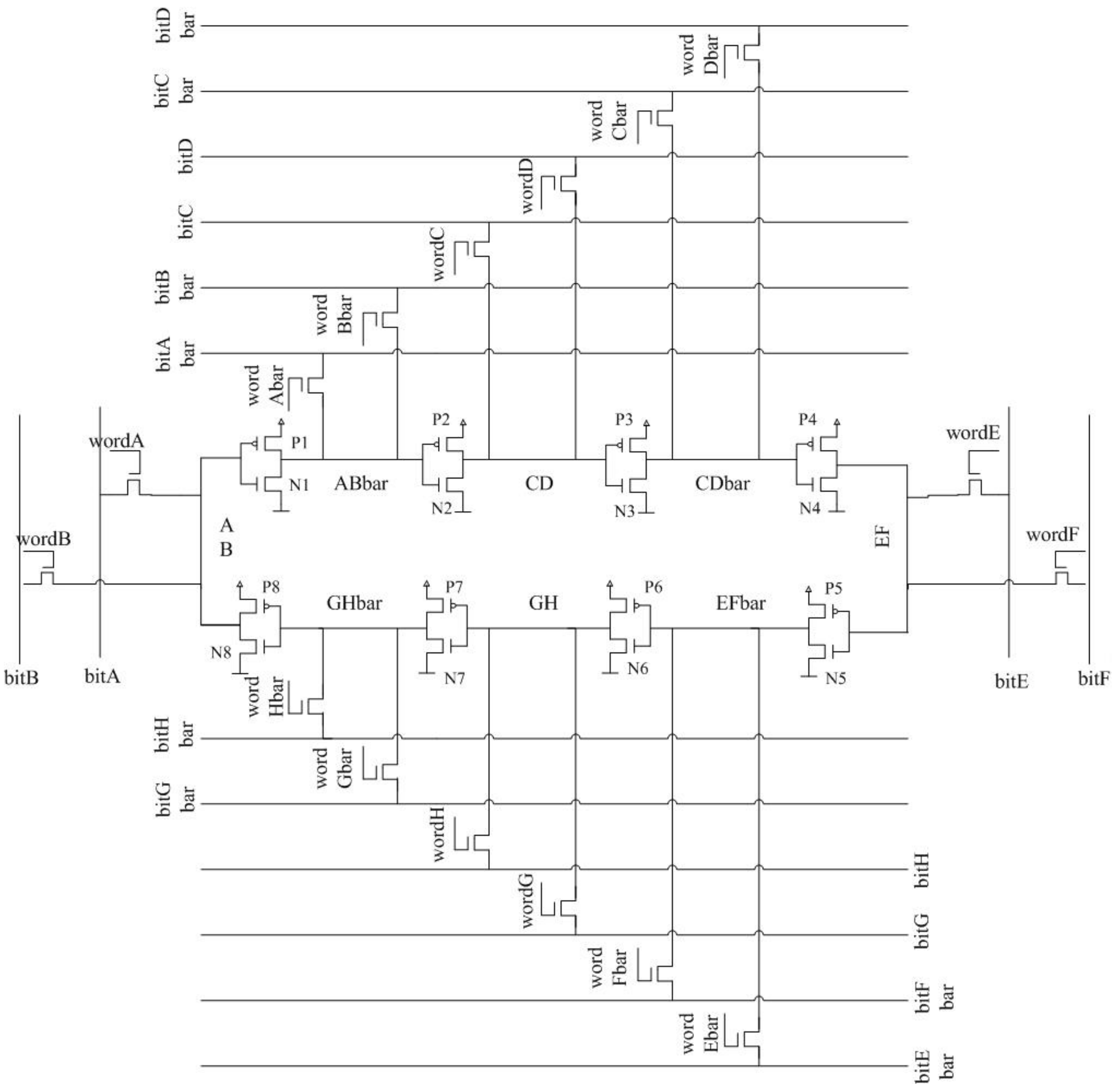

Fig.4.5 Schematic of 8-SRAM model of 8-port memory cell

Similarly, consider writing a “0” over stored “1” by double ended write into the memory. The transistors conducting in this condition are transistors enabled to do a double ended write say wordA and wordAbar, the transistors in the memory loop P1, N2, P3, N4, P5, N6, P7, and N8. With in which, transistor with gate wordAbar operate in linear mode and transistor P1 operate in saturation (refer to Fig.4.5) are considered. We get V_{ABbar} by solving equation 4.8

$$(linear)I_{DSA,lin} = \frac{I_{DSA,lin,R_{ds=0}}}{1+\frac{R_{dsA}.I_{DSA,lin,R_{ds=0}}}{V_{ABbar}}}$$

$$(saturation)I_{DSp,sat} = \frac{\varepsilon_{ox}}{t_{ox}}.W.v_{sat}.\left[\left(|V_{dd}-V_{Tp}|\right)-A_{bulk}.|V_{DSsat}|\right]\left(1+\frac{|V_{dd}-V_{ABbar}-V_{DSsat}|}{V_A}\right)$$

$$(linear)I_{DSA} = I_{DSp}(saturation) \quad \text{--- 4.8}$$

Now that we have V_{AB}, V_{ABbar}, and V_{inv} available, we can calculate noise margin and compare it with simulation results. Similarly, we can calculate cell node voltages for the rest of the available nodes in the memory. The theoretical and simulation results are tabled in 6.6 and 6.5 respectively, and the simulation test bench for SPECTRE and netlist generated from CADENCE VIRTUOSO LAYOUT EDITOR can be referred from the appendix C.4. Layout of which is shown in Fig 6.5

4.5 J-SRAM, 8-Port memory cell

J-SRAM model of 8-port memory cell has 6 inverters in the inverter cell loop and 16 access transistors connected to the memory cell enabled or disabled accordingly for accessing the memory, which makes upto 28 transistors. This design is named after Dr.J (Dr. Louis.G.Johnson) as J-SRAM. The 6 inverters in the memory cell are coupled to form a closed inverter loop, preserving data in the memory. The nodes of the memory are labeled as V_{cell} and $V_{cellbar}$ and the internal nodes in the memory are labeled as A, B, C,and D. “Eight” access transistors are connected each on cell and cellbar node in the memory cell, the gates of which are labeled as bitA, bitB, bitC, bitD, bitE, bitF, bitG, and bitH on the cell node, and bitAbar, bitBbar, bitCbar, bitDbar, bitEbar, bitFbar, bitGbar, and bitHbar on the cellbar node as shown in the schematic in Fig.4.5.

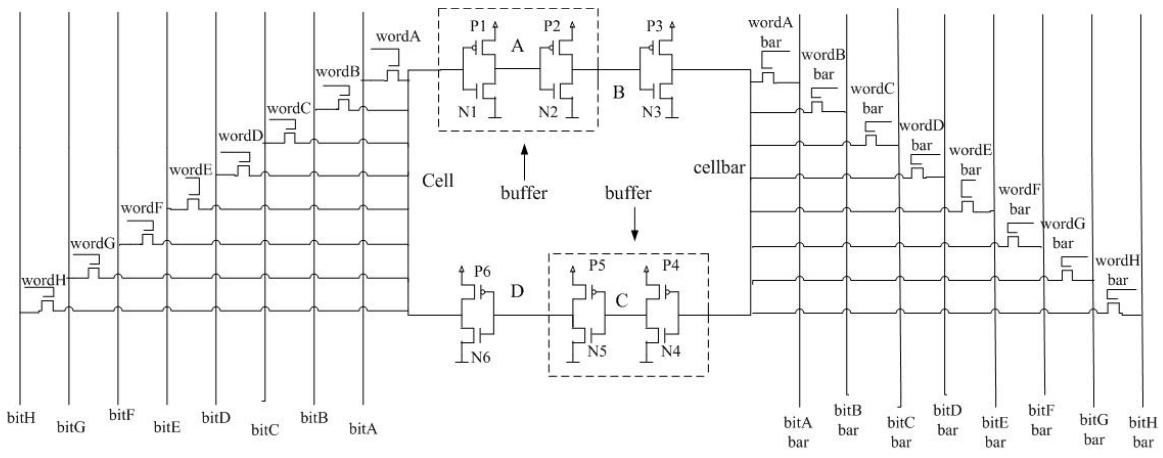

Fig.4.5 Schematic of J-SRAM model of 8-port SRAM memory cell

This design looks similar in the sense of how the memory is accessed through cell and cellbar nodes to the 2-SRAM design proposed in the beginning of this chapter. Noticeable differences from the schematic diagram are the buffer circuits. The purpose of the buffer circuit is as follows:

1. Buffers are designed (sized) so that the inverter threshold voltage is higher than ideal value of Vdd/2, this is made possible by sizing the p-channel MOSFET's in the inverters.
2. Sizing the p-channel MOSFET's increase the noise margin of the cell (increase in inverter threshold voltage increases noise margin) holding cell node voltage marginally the same.
3. Buffers share the capacitive load, when reading through all the ports simultaneously on the memory nodes and bitline capacitance, without which the n-channel MOSFET's N3 and N6 would be proportionally large to the number of ports.

With sizing of transistors in the buffers, transistors in the cell (P3, N3, P6, and N6), and access transistor appropriately it is observed that all the performance parameters (namely

Noise margin, Read access time) are far better than all the other design's in 8-port memory cell. It is observed by simulations that, the J-SRAM operates with a read access time of 1.5ns with a noise margin of over 0.5V in a 5 V, AMI 06µm technology. Refer to appendix C.4 for the test bench and the results of which are listed in chapter 6.

Reading and writing criteria are the same as any other design proposed; the worst case scenario is similar to other design. The two worst cases encountered are as follows:

1. While reading a "0" from the memory cell simultaneously through all the available ports on that node.
2. Writing a "0" over a stored "1" in the memory cell, while doing a double ended write.

Considering case 1, we can calculate the cell node voltage, for which the operating transistors are: access transistors which are enabled to read from the cell node, namely with gates labeled as wordA, wordB, wordC, wordD, wordE, wordF, wordG, and wordH and transistor N6, P1, N2, P3, N4, P5 in the inverter loop (refer to Fig.4.5). For calculating cell node voltage, 8 access transistors are in saturation and transistor N6 is in linear mode, the drain-source currents of which are same. Therefore equating the drain-source current equation of an n-channel MOSFET operating in saturation and linear modes would give cell node voltage. This is same as in the generalized equation discussed in section 4.2, notice that "m = 8". Therefore the V cell is calculated from the equation 4.9.

$$I_{DSn} = 8 * I_{DSA} \qquad \text{--- 4.9}$$

$$(linear) I_{DSn,lin} = \frac{I_{DS,lin,R_{ds=0}}}{1 + \frac{R_{ds}.I_{DS,lin,R_{ds=0}}}{V_{cell}}}$$

$$(saturation) I_{DSA,sat} = \frac{\varepsilon_{ox}}{t_{ox}}.W.v_{sat}.\left[\left(V_{dd} - V_{cell} - V_{Tn}\right) - A_{bulk}.|V_{DSsat}|\right]\left(1 + \frac{V_{dd} - V_{cell} - V_{DSsat}}{V_A}\right)$$

Now consider the other worst case scenario, writing a "0" over a stored "1", in this situation, transistors operating on cellbar node are transistor with gates as wordAbar and wordA (double ended write), and P3 in the memory cell. Access transistors wordA and wordAbar operate in linear mode and P3 is in saturation, equating the drain-source current equation as shown in equation 4.10, solving the equality would give cellbar node voltage.

$$(linear) I_{DSA,lin} = \frac{I_{DSA,lin,R_{ds=0}}}{1 + \frac{R_{dsA}.I_{DSA,lin,R_{ds=0}}}{V_{cellbarA}}}$$

$$(saturation) I_{DSp,sat} = \frac{\varepsilon_{ox}}{t_{ox}}.W.v_{sat}.\left[\left(|V_{dd} - V_{Tp}|\right) - A_{bulk}.|V_{DSsat}|\right]\left(1 + \frac{|V_{dd} - V_{cellbar} - V_{DSsat}|}{V_A}\right)$$

$$(linear) I_{DSA} = I_{DSp}(saturation) \qquad \text{--- 4.10}$$

Now that we have $V_{cell,}$ $V_{cellbar,}$ and V_{inv} (refer to section 2.4) available, we can calculate noise margin and compare it with simulation results. The simulation results are tabled in 6.7 and 6.8 respectively, and the simulation test bench for SPECTRE and netlist generated from CADENCE VIRTUOSO LAYOUT EDITOR can be referred from the appendix C.4. Layout of which is shown in Fig.6.7

4.6 Summary

As a remainder, we considered two worst case scenarios in all the designs proposed; firstly trying to read a “0” from the memory through all the available ports on the memory node. Secondly, trying to force a “0” on a previously stored “1” through a complementary node, enabled by double ended write. We observed that the write criterion for single port memory cell and multiport memory cell is the same and while reading through a muliport memory cell, one has to consider the effect of all the ports available on that particular node.

We have formulated, theoretical equations for calculating memory node voltages for various designs proposed, the results of which are tabled in chapter 6. Theoretical values are compared with simulation results obtained from simulation (CADENCE SPECTRE) of the layout made, using CADENCE VIRTUOSO LAYOUT EDITOR.

Chapter 5

Register Files

5.0 Introduction

This chapter would briefly explain the need of register files in modern computer architecture, why modern RISC processors designers are in lookout for more effective and better performing multiport memory cells, and also introduces other logic circuits in the memory array that contribute to better performance of the register file.

5.1 Need of Register Files

In modern computer processors design, register files are considered an important component for implementing registers. In Today's simple processor applications, typical word sizes are 32 or 64 bits (4 or 8 bytes)to process integer values or floating point values as well, modern RISC processor will have atleast 32 registers to store temporary data. Computer processors work efficiently when there are enough registers so that only occasionally data need to be transferred from register and memory. Register files are like memory in that only limited numbers of connections (ports) are provided to each register and the same connection is shared by different registers. Note that internal register differ from main memory in that many different buses connect to the registers but usually one bus connect to the memory element, this causes dataflow bottle neck (the Von Neuman Bottleneck).This reduces the cost of the memory element but makes it unsuitable for connecting to the processing units [7].

Register files are fast RAMs with multiple reads and write ports. Modern processors often need multiple ports to handle multiple simultaneous instructions. For example, the itanium 2 processors issues upto 6 integer instructions in a cycle, each of which require two source registers and one destination register [1]. For this a register file should have mutliports in the memory cell array, which enables the processor to process simultaneous instructions. All designs of 8-port memory cells discussed in earlier chapter serve the purpose of modern RISC processor, which requires 32 bit register or 64 bit registers depending on application. Memory arrays are designed for these purposes which are capable of storing large data and to process 32 bit or 64 bit registers.

5.2 Design of Multi-port Memory Arrays (Register File).

As an example consider a register file cell described in section 4.2 with 8 pair of bitlines. This memory cell is usually operated to read from all the ports simultaneously or write into the memory through a single bitline pair, to do a single ended read or double ended write. This design allows 16 single ended reads and 8 double ended writes through the available pair of bitlines. In this design it is acceptable to read from the memory through all the bitlines simultaneously but, writing into the same location through one or more bitlines is not acceptable. Writing into a different location is acceptable but not into same location.

Now, same memory location or different location in a memory cell does not make any sense, but when talking of memory location in an array of memory cell does make sense. Typical small memory array architecture consists of 2^n words of storage of 2^m bits each. For example, to store a 32-bit value we need 32 SRAM cells in a row and to store 32, 32-bit values we need 32 rows of SRAM cell with each row having 32 SRAM cells (2^5 rows and 2^5 columns). Thus this memory array can store 2^{10} or 1024 bits, which serve the purpose of storing some of data or process 32 bit register.

Accessing different memory location in a memory array is possible by selecting desired memory cell in the memory array. Apart from this there are two other issues; how do you read data out of the memory and how is data written into the memory. These three issues are briefly explained as under;

1. How is the memory location selected? This is made possible by design of *row decoders* connected to the gates of the access transistors. The gate of the access transistors are enabled or disabled according to the need. For example in Fig.4.1 memory cell is selected to write into or read from the memory, say through portA (wordA), then the decoder connected to portA is enabled and the memory cell is either read from or written into. For this case we design the simplest decoder, is a collection of AND gates using true and complementary versions of the address bits.
2. How is the data readout of the memory? The bitlines are the available source through which the memory can be readout, the bitlines can have any value between a high (V_{DD}) or low (Gnd). For this there must be a mechanism to detect the variation with respect to some values between high and low or a reference

value to sense the change. The purpose is served by *sense amplifier* connected to the bitlines; sense amplifier can be either a simple digital inverter or an analog differential amplifier, in this case a simply inverter works. The design specification is that: reference voltage set for the sense amplifier should be the same as the inverter threshold voltage in the inverter cell. We choose an inverter for the purpose of single ended read, if we were to read the difference in voltage between two bitlines then we would have preferred an analog CMOS differential Amplifier [6].

3. How is the data read into the memory? The bitlines are the only means of communication with the memory, either when writing into or reading from the memory. When writing into a memory the bitlines have to drive the memory cell high or low. This is made possible by external circuit called *write drivers*. These drivers are essentially an n-channel MOSFET connected to the bit and bitbar lines which force a low into the memory. N-channel MOSFETs used as write drivers can be proportionally sized (large transistor are possible as these occupy separate chip area) and a high is forced into the memory with the large p-channel MOSFET connected to the bit and bitbar lines [6].

Apart from the design aspects of the multiport SRAM memory cell, design of sense amplifiers, write drivers, row decoders and pre-charge transistor play an equally important role in designing a register file.

5.3 Summary

Register files are main building blocks in modern computer architecture, modern RISC processors has 32 or 64 bit register. For this the need of multiport register cell are the need of the hour. And the performance of the cell is of utmost interest and concern, for this understanding the operation of the cell is necessary. Apart from the register cell, external circuit designs like the sense amplifier, decoders, write driver and precharge are also design concerns, and their better performance improves register cell performance.

Chapter 6

Results and Discussion

6.0 Introduction

Basic aim of this thesis is to understand the working, and design an 8-port SRAM memory cell. In chapter 3, we extensively discussed about working principles and stability issues while reading from and writing into a single port SRAM and multi-port SRAM cells. We looked into various design issues which were a cause of concern in designing a multiport SRAM. In chapter 4, we proposed four different designs of 8-port SRAM memory cells, which tried to overcome the problem concerning design issues. And we were successful in doing so. In this chapter, we would compare the simulation results and theoretical results obtained in this process, and look for the design metrics which has the maximum possible noise margin for an 8-port SRAM memory cell.

Finally the design factors of the multiport SRAM memory cell are discussed in detailed. Within a limited area, transistors in the deisgn are sized to attain maximum noise margin. Layouts of the designs are made using CADENCE VIRTUOSO LAYOUT EDITOR in an AMI 0.6 μm technology process, and the netlist is extracted for simulation. The tool used for simulation is CADENCE SPECTRE.

6.1 Various designs of 8-port SRAM Memory cell

In chapter 4, we proposed various designs for 8-port SRAM memory cell. My primary goal in designing these memory cells is to understand their working and attain maximum noise margin possible within a limited area of 1600µm^2. The four designs models proposed for 8-port SRAM memory cell in chapter 4 are as follows, they are namely;

1. 2-SRAM
2. 4-SRAM
3. 8-SRAM
4. J-SRAM

In the following section, I would list results obtained, theoretically and simulation results for all the designs.

6.2 2-SRAM model of 8-port SRAM memory cell

In this design the memory nodes are labeled as cell and cellbar, to which the access transistors (n-channel MOSFETs) are connected, 8 each on cell and cellbar nodes. The gates of which are labeled as follows; on cell node the gates are labeled as wordA, wordB, wordC, wordD, wordE, wordF, wordG and wordH. And on the cellbar node the gates of which are labeled as wordAbar, wordBbar, wordCbar, wordDbar, wordEbar, wordFbar, wordGbar and wordHbar.

Here is a list of theoretical and simulation results of cell, cellbar voltages, inverter threshold voltage, and noise margin of the memory cell. Simulation results are shown in table 6.1 and theoretical values are listed in table 6.2. The theoretical calculation results are shown in the table 6.2, and refer to appendix C.2.for the test bench and netlist.

Wncell (in µm)	Wpcell (in µm)	Wn (in µm)	Vinv.sim (in volts)	Vcell.sim (in volts)	Vcellbar (in volts)	NM,cell (in volts)
9.60	1.2	1.5	1.46092	1.89974	1.96448	-0.50356
12.0	1.2	1.5	0.98198	1.18102	1.19837	-0.21639
13.2	1.2	1.5	0.95883	1.02116	1.03281	-0.07397
14.4	1.2	1.5	0.93889	0.81370	0.82495	0.11394
15.6	1.2	1.5	0.92148	0.73160	0.73448	0.18700
16.8	1.2	1.5	0.90590	0.67812	0.68103	0.22486
18.0	1.2	1.5	0.89237	0.63491	0.63665	0.25574
15.9	1.2	1.5	0.91746	0.71630	0.71630	0.19765

Table 6.1 Simulation results of 2-SRAM memory cell, V $_{cell}$, V $_{cellbar}$, V $_{inv}$, & NM$_{, cell}$

Wncell (in µm)	Wpcell (in µm)	Wn (in µm)	Vinv (in volts)	Vcell.th (in volts)	NM,cell (in volts)
9.60	1.2	1.5	1.27303	1.39939	-0.12636
12.0	1.2	1.5	1.21590	1.18289	0.03301
13.2	1.2	1.5	1.09723	1.09723	0.09599
14.4	1.2	1.5	1.02285	1.02285	0.15036
15.6	1.2	1.5	0.95769	0.95769	0.19785
16.8	1.2	1.5	0.90020	0.90020	0.23954
18.0	1.2	1.5	0.84911	0.84911	0.30232
15.9	1.2	1.5	0.94266	0.94266	0.20877

Table 6.2 Theoretical results of 2-SRAM memory cell, V $_{cell}$, V $_{inv}$, & NM$_{, cell}$

We observe from the simulation results (refer table 6.1) that with increase in the width of the n-channel MOSFET in the inverter loop, there is a considerable increase in noise margin. This supports the theoretical analysis that was studied in the section 3.2.1 and 3.2.2. The layout of the design was made with transistor widths giving maximum possible noise margin in a limited area of 1600 µm^2, which is made with CADENCE and is as shown in the fig. 6.1 the noise margin is calculated as follows:

$$NM_{storage-node} = V_{inv} - V_{[\max(cell,cellbar)]}$$

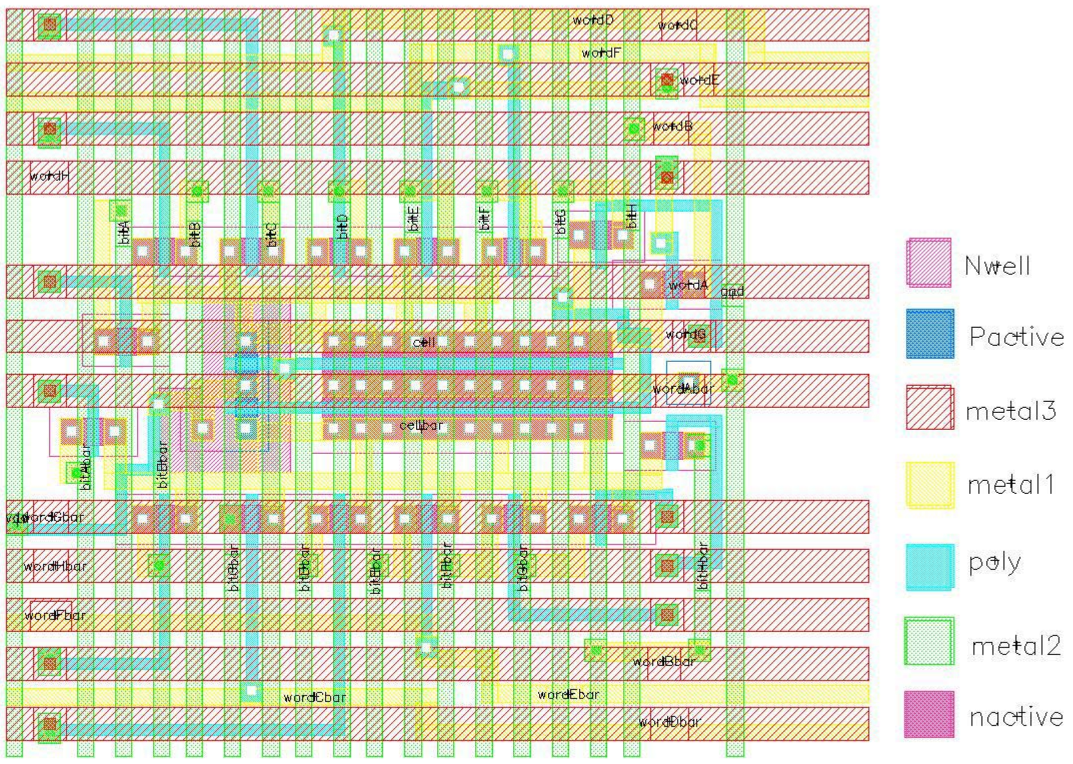

Fig.6.1. CADENCE Layout of 2-SRAM model of 8-port memory cell.

The layout dimensional specifications are as under, and the simulation waveform for data on the cell node is shown in Fig.6.2. for calculation of noise margin in the cell, we need cell node or the storage node voltage, we read the storage node voltage from the data wavform shown in the fig as under; notice that the peak in the voltage, while reading a "0" is measured at every node as indicated in the figure rounded in black.we get a noise margin of 0.1974 V for these specified widths of transistors in thedesign.

Wncell (in µm)	Wpcell (in µm)	Wn (in µm)	Vinv.sim (in volts)	Vcell.sim (in volts)	Vcellbar (in volts)	NM,cell (in volts)
15.9	1.2	1.5	0.91746	0.71630	0.71630	0.19765

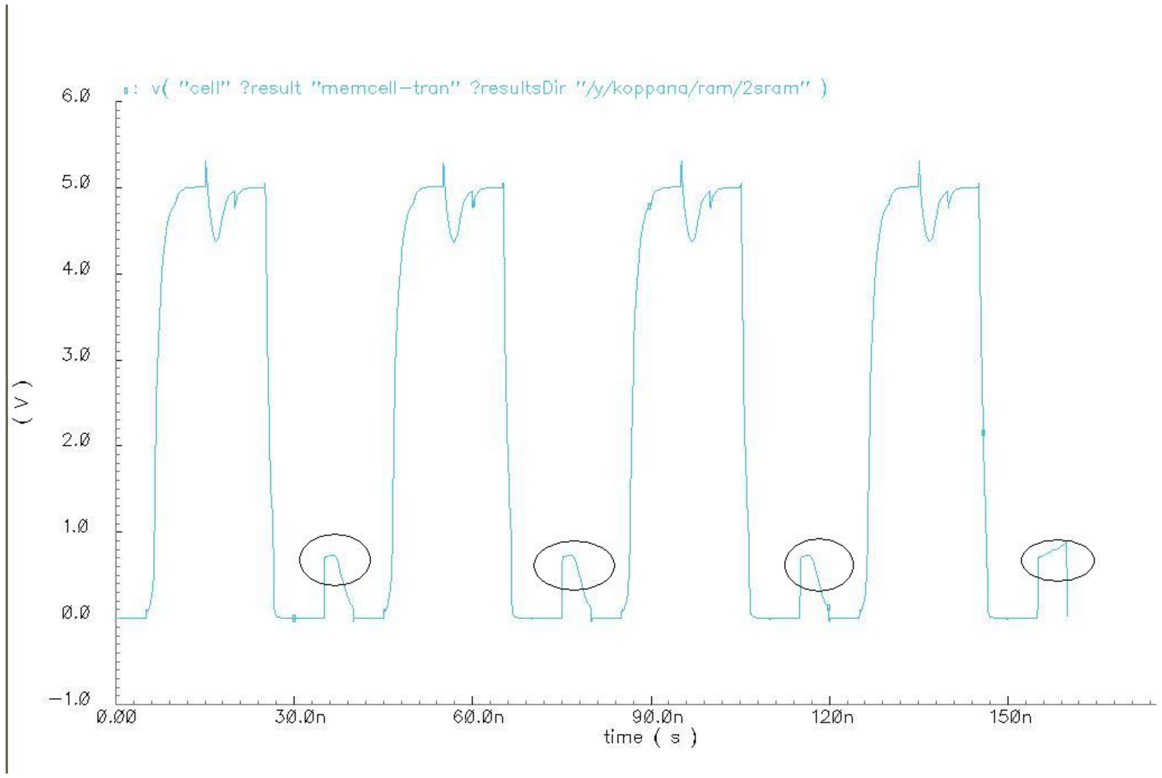

Fig 6.2 Data on the cell node of 2-SRAM model

6.3 4-SRAM model of 8-port SRAM memory cell

The storage nodes of the 4-SRAM memory cell are labeled as ABCD, ABCDbar, EFGH, and EFGHbar, each node is connected with 4 access transistors or pass transistors (as referred by some authors) is shown in the Fig 4.4. The gates of the access transistors connected to node "ABCD" are labeled as wordA, wordB, wordC, and wordD, the gates of the transistors to node "ABCDbar" are labeled as wordAbar, wordBbar, wordCbar, and wordDbar, the gates of the transistor to node EFGH are labeled as wordE, wordF, wordG, and wordH and finally the transistor gates to node EFGHbar are labeled has wordEbar, wordFbar, wordGbar and wordHbar. And the bitlines connected to corresponding drains of the access transistor are labeled as bitA, bitB, bitC, bitD, bitE, bitF, bitG, bitH, bitAbar, bitBbar, bitCbar, bitDbar, bitEbar, bitFbar, bitGbar and bitHbar.

Here is a list of theoretical and simulation results of ABCD, ABCDbar voltages, inverter threshold voltage, and noise margin of the memory node. Simulation results are shown in table 6.3 and theoretical values are listed in table 6.4. The theoretical calculation and its procedure are listed in appendix D.2, and refer to appendix C.3 for the test bench with the netlist.

Wncell (in µm)	Wpcell (µm)	Wn (µm)	Vinv.sim (in volts)	VABCD. (in volts)	VABCDbar (in volts)	NM,ABCD (in volts)
4.8	1.2	1.5	1.29152	1.93369	1.88252	-0.6422
6.0	1.2	1.5	1.19900	1.02021	1.01018	0.17879
7.2	1.2	1.5	1.13247	0.82587	0.82396	0.30659
8.4	1.2	1.5	1.08183	0.71480	0.71312	0.36702
9.6	1.2	1.5	1.04180	0.63451	0.63265	0.40729
10.8	1.2	1.5	1.00916	0.57125	0.57029	0.43790
11.9	1.2	1.5	0.98408	0.52387	0.52321	0.46021
13.1	1.2	1.5	0.96007	0.48050	0.48002	0.47957
14.2	1.2	1.5	0.94205	0.44325	0.44622	0.49583
15.4	1.2	1.5	0.92427	0.41470	0.41438	0.50956
15.9	1.2	1.5	0.91746	0.40271	0.40231	0.51474
16.6	1.2	1.5	0.90856	0.38577	0.38677	0.52279
16.8	1.2	1.5	0.90590	0.38287	0.38241	0.52303

Table 6.3 Simulation results of 4-SRAM memory cell, V_{ABCD}, $V_{ABCDbar}$, V_{inv}, and $NM_{,ABCD}$

We observe from the simulation results in table 6.1.that the results support the theoretical analysis on the transistor sizing of n–channel MOSFET in the memory cell we obtain maximum noise margin of 0.52303 V, with n-channel FET width of 16.8 µm in the inverter cell, the layout of which is made with CADENCE and is as shown in the fig. 6.3 the noise margin is calculated as follows:

$$NM_{storage-node} = V_{inv} - V_{[\max(cell,cellbar)]}$$

Wncell (in µm)	Wpcell (µm)	Wn (inµm)	Vinv (in volts)	VABCD. (in volts)	NM,ABCD (in volts)
4.8	1.2	1.5	1.48803	1.39939	0.08864
6.0	1.2	1.5	1.41233	1.18289	0.22944
7.2	1.2	1.5	1.35512	1.02285	0.33227
8.4	1.2	1.5	1.30991	0.90020	0.40971
9.6	1.2	1.5	1.27303	0.80343	0.46960
10.8	1.2	1.5	1.24219	0.72522	0.51697
11.9	1.2	1.5	1.21795	0.66570	0.55225
13.1	1.2	1.5	1.19493	0.61091	0.58402
14.2	1.2	1.5	1.17636	0.5680	0.60836
15.4	1.2	1.5	1.15835	0.52754	0.63081
15.9	1.2	1.5	1.15143	0.51233	0.63910
16.6	1.2	1.5	1.14226	0.49244	0.64982
16.8	1.2	1.5	1.13974	0.48703	0.65237

Table 6.4 Theoretical results of 4-SRAM memory cell, V_{ABCD}, V_{inv}, and $NM_{,ABCD}$

The layout dimensional specifications are as under, and the simulation waveform for data on the cell node is shown in Fig.6.4. For calculation of noise margin in the cell, we need cell node or the storage node voltage; we read the storage node voltage from the data wavform shown in the fig as under; notice that the peak in the voltage, while reading a "0" is measured at every node as indicated in the figure rounded in black.

Wncell (in µm)	Wpcell (µm)	Wn (µm)	Vinv.sim (in volts)	VABCD. (in volts)	VABCDbar (in volts)	NM,ABCD (in volts)
16.8	1.2	1.5	0.90590	0.38287	0.38241	0.52303

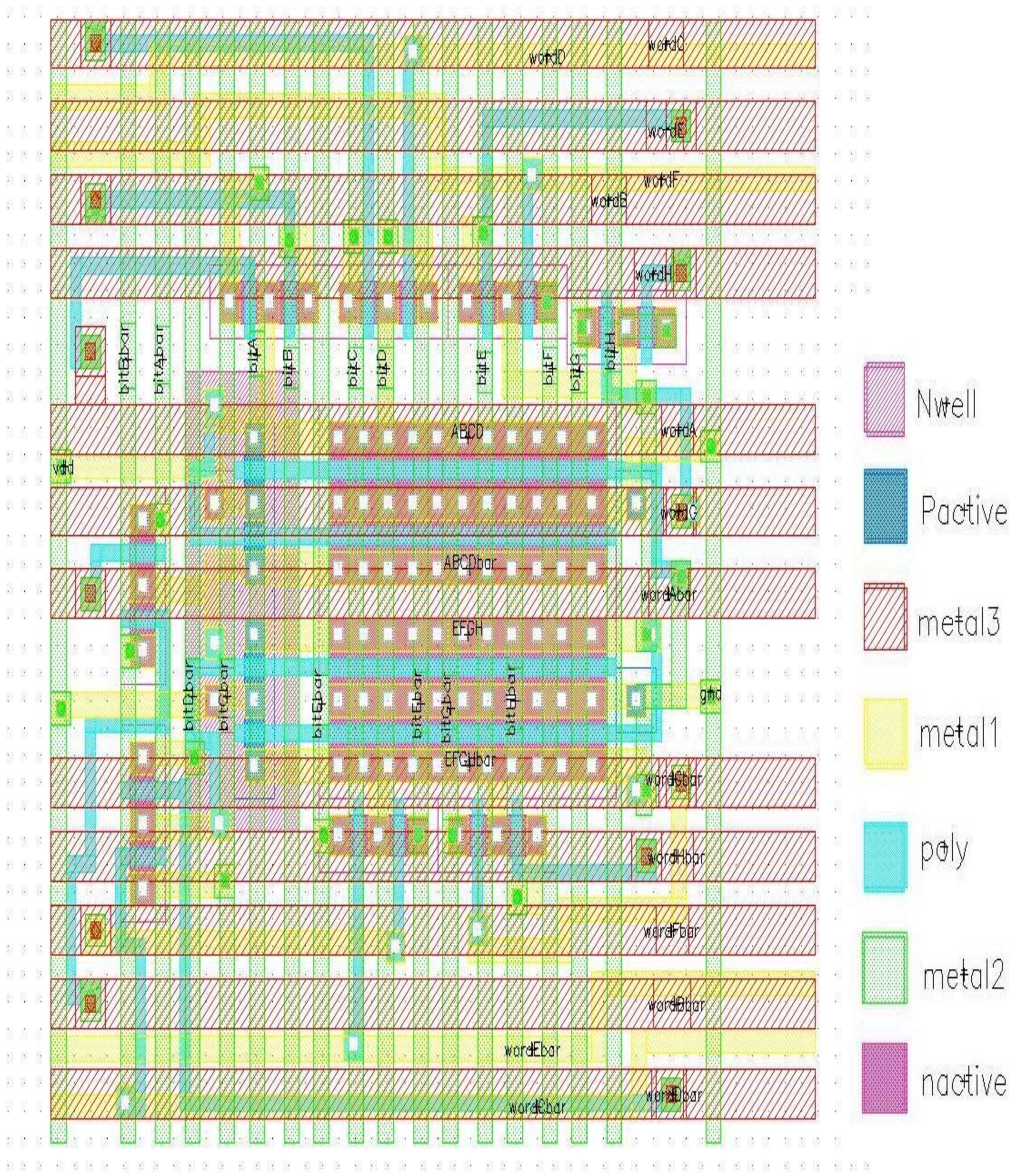

Fig.6.3. CADENCE Layout of 4-SRAM model of 8-port memory cell

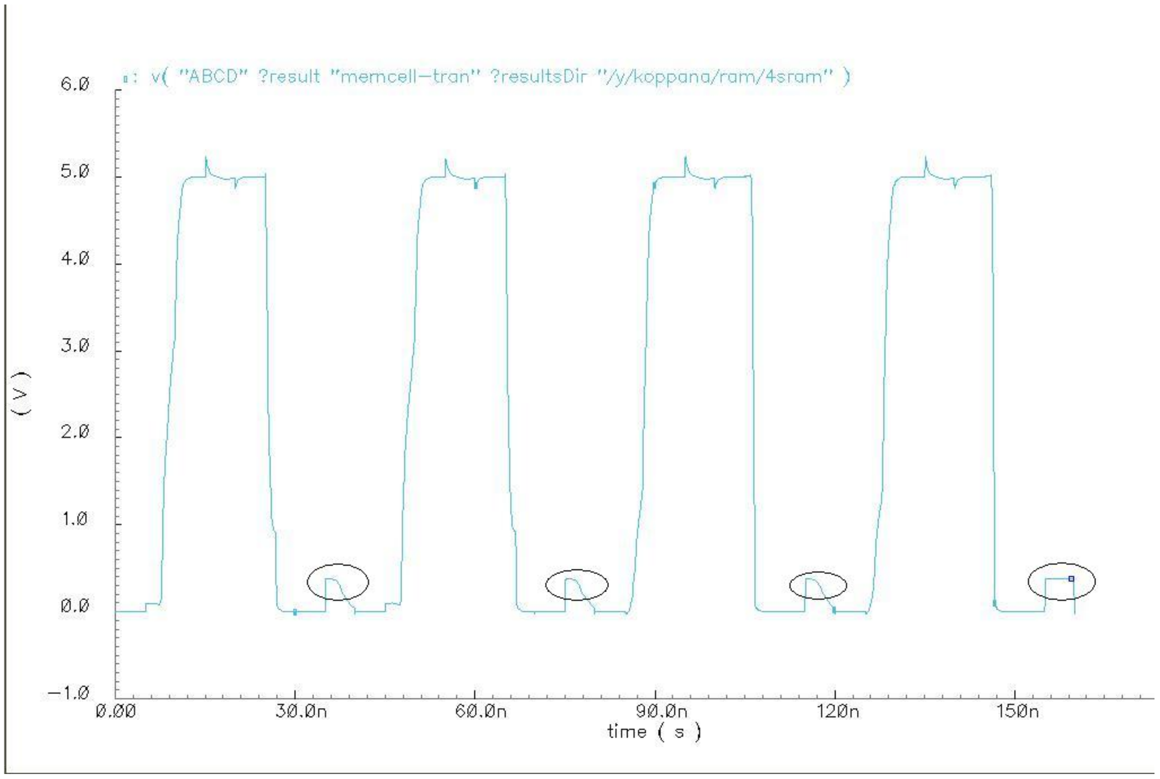

Fig 6.4 Data on the ABCD node of 4-SRAM model

6.4 8-SRAM model of 8-port SRAM memory cell

. Memory storage nodes in 8-SRAM are labeled as AB, ABbar, CD, CDbar, EF, EFbar, GH, GHbar respectively, access transistors connected to these nodes with their drain connected to the bitlines and source connected to memory node. The bitlines connected to the memory node AB through the drain of the access transistors are labeled as bitA and bitB. Similarly, the rest of the bitlines are labeled as bitAbar, bitBbar, bitC, bitD, bitCbar, bitDbar, bitE, bitF, bitEbar, bitFbar, bitG, bitH, bitGbar, and bitHbar.

Here is a list of theoretical and simulation results of cell, cellbar voltages, inverter threshold voltage, and noise margin of the memory node. Simulation results are shown in table 6.5 and theoretical values are listed in table 6.6. The theoretical calculation and its procedure are listed in appendix D.2, and refer to appendix C.4 for the test bench with the netlist.

Wncell (in µm)	Wpcell (in µm)	Wn (in µm)	Vinv.sim (in volts)	VAB (in volts)	VABbar (in volts)	NM, AB (in volts)
2.4	1.2	1.5	1.67360	1.23960	1.23726	0.4340
2.7	1.2	1.5	1.59723	1.11384	1.11107	0.48616
3.0	1.2	1.5	1.53311	1.01074	1.00907	0.52237
3.3	1.2	1.5	1.47850	0.92629	0.92428	0.55221
3.6	1.2	1.5	1.43123	0.85381	0.85365	0.57738
3.9	1.2	1.5	1.38986	0.79457	0.79417	0.59529
4.2	1.2	1.5	1.35333	0.74173	0.61080	0.6116

Table 6.5 Simulation results of 8-SRAM memory cell, V_{AB}, V_{ABbar}, V_{inv}, and $NM_{,AB}$

Wncell (in µm)	Wpcell (in µm)	Wn (in µm)	Vinv (in volts)	VAB (in volts)	NM, AB (in volts)
2.4	1.2	1.5	1.76323	1.39939	0.36384
2.7	1.2	1.5	1.71233	1.28245	0.42988
3.0	1.2	1.5	1.66817	1.18289	0.48528
3.3	1.2	1.5	1.62940	1.09723	0.52767
3.6	1.2	1.5	1.59499	1.02285	0.57214
3.9	1.2	1.5	1.56417	0.95770	0.60647
4.2	1.2	1.5	1.53637	0.90020	0.63617

Table 6.6 Theoretical results of 8-SRAM memory cell, V_{AB}, V_{inv}, and $NM_{,AB}$

We observe from the simulation results that the results support the theoretical analysis on transistor sizzing in the memory cell. Design is made with transistor widths giving maximum noise margin that is possible in with a limited area of 1600µm^2, the layout of which is made with CADENCE and is as shown in the fig. 6.5 the noise margin is calculated as follows:

$$NM_{storage-node} = V_{inv} - V_{[\max(cell,cellbar)]}$$

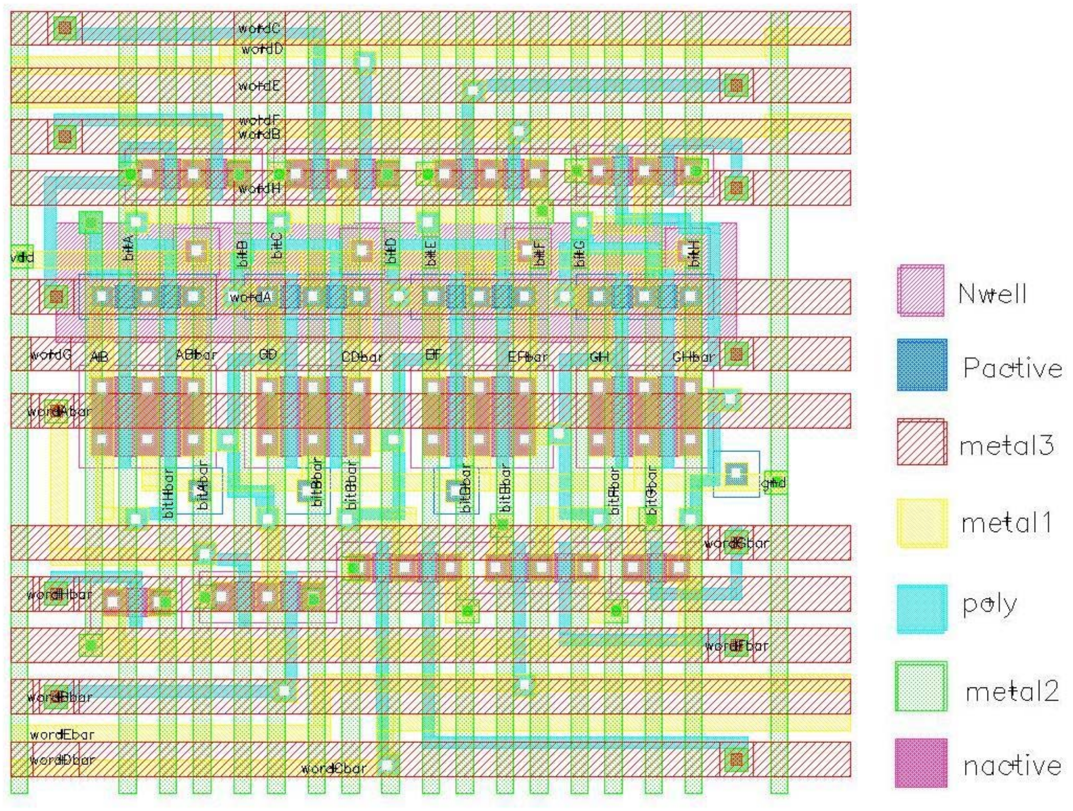

Fig.6.5 CADENCE Layout of 8-SRAM model of 8-port memory cell.

Wncell (in µm)	Wpcell (in µm)	Wn (in µm)	Vinv.sim (in volts)	VAB (in volts)	VABbar (in volts)	NM, AB (in volts)
4.2	1.2	1.5	1.35333	0.74173	0.61080	0.6116

The layout dimensional specifications are as under, and the simulation waveform for data on the cell node is shown in Fig.6.5. For calculation of noise margin in the cell, we need cell node or the storage node voltage; we read the storage node voltage from the data wavform shown in the fig as under; notice that the peak in the voltage, while reading a "0" is measured at every node as indicated in the figure rounded in black

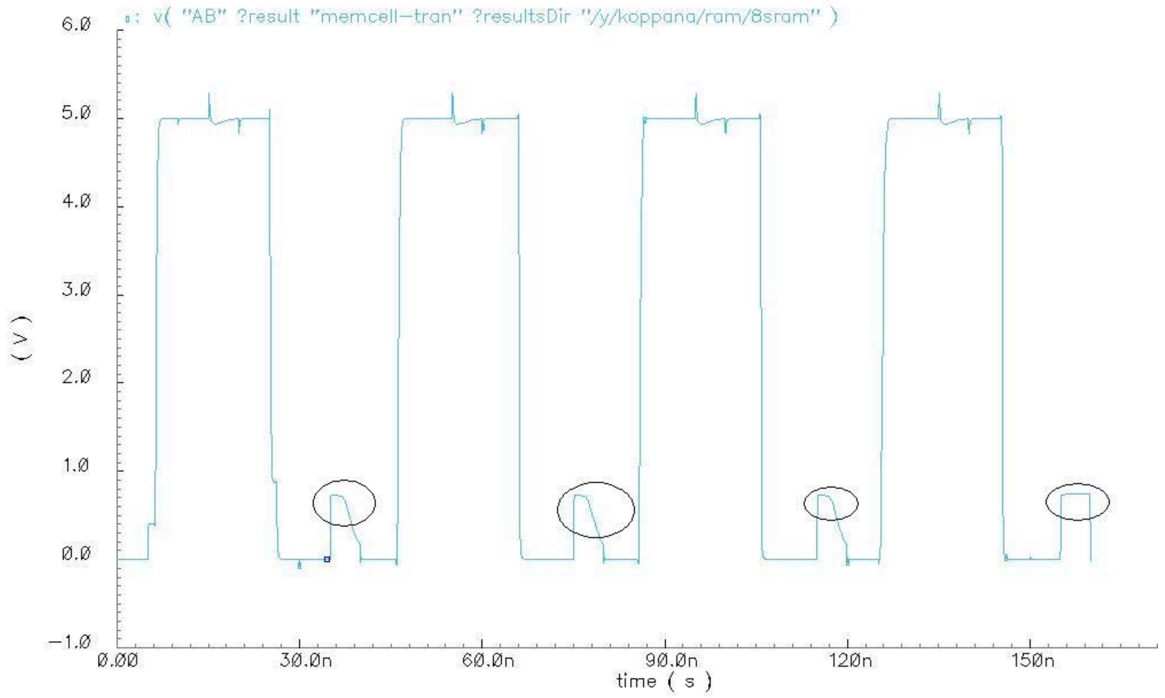

Fig.6.6 Data on AB node of 8-SRAM model

6.5 J-SRAM model of 8-port SRAM memory cell

In J-SRAM model the storage nodes of the memory are labeled as V $_{cell}$ and V $_{cellbar}$ and the internal nodes in the memory are labeled as A, B, C, and D.16 access transistors are connected to memory cell, 8 each on cell and cellbar node, the gates of which are labeled as bitA, bitB, bitC, bitD, bitE, bitF, bitG, and bitH on cell node, bitAbar, bitBbar, bitCbar, bitDbar, bitEbar, bitFbar, bitGbar, and bitHbar on cellbar node. And the gates of these transistors are labeled as wordA, wordB, wordC, wordD, wordE, wordF, wordG and wordH respectively, and wordAbar, wordBbar, wordCbar, wordDbar, wordEbar, worFbar, wordGbar and wordHbar.

This design is worked on keeping these design constriains:

1. Transistors in the first inverter are designed (sized) so that the inverter threshold voltage is higher than ideal value of Vdd/2, this is made possible by sizing the p-channel MOSFET's in the inverters.
2. Sizing the p-channel MOSFET's increase the noise margin of the cell (increase in inverter threshold voltage increases noise margin) holding cell node voltage marginally the same.
3. Transistors share the capacitive load, when reading through all the ports simultaneously on the memory nodes and bitline capacitance, without which the n-channel MOSFET's N3 and N6 would be proportionally large to the number of ports.

And you find that the buffer p-channel MOSFET is larger than the n-channel MOSFET, which would give a higher inverter threshold voltage. And the rest of the transistors are designed with minimum dimension,(except for n-channel N3 and N6) and the access transistor is made larger than normal 2.1 μm wider, compared to other designs this enable to increase the speed of the memory cell or minimize the read access time.

The layout of which is shown in fig.6.7 and the dimensions of which is given as under:

1. J-SRAM model has the got a noise margin of 0.8251 V with dimension of the W_n = 4.2 μm, W_p = 4.2 μm, W_a = 2.1 μm. and read access time for these widths is 1.5ns.

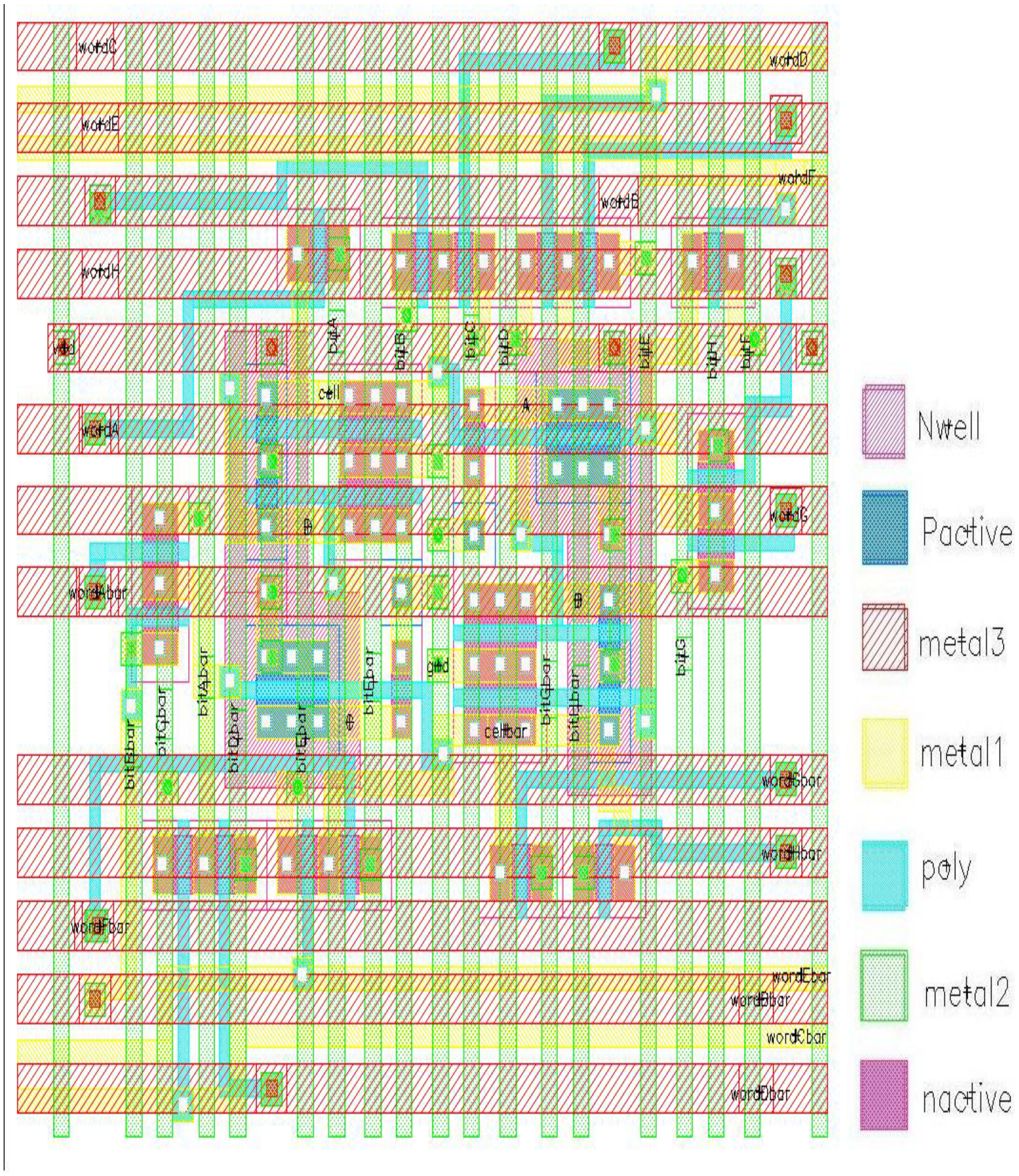

Fig.6.7 CADENCE Layout of J-SRAM model of 8-port memory cell

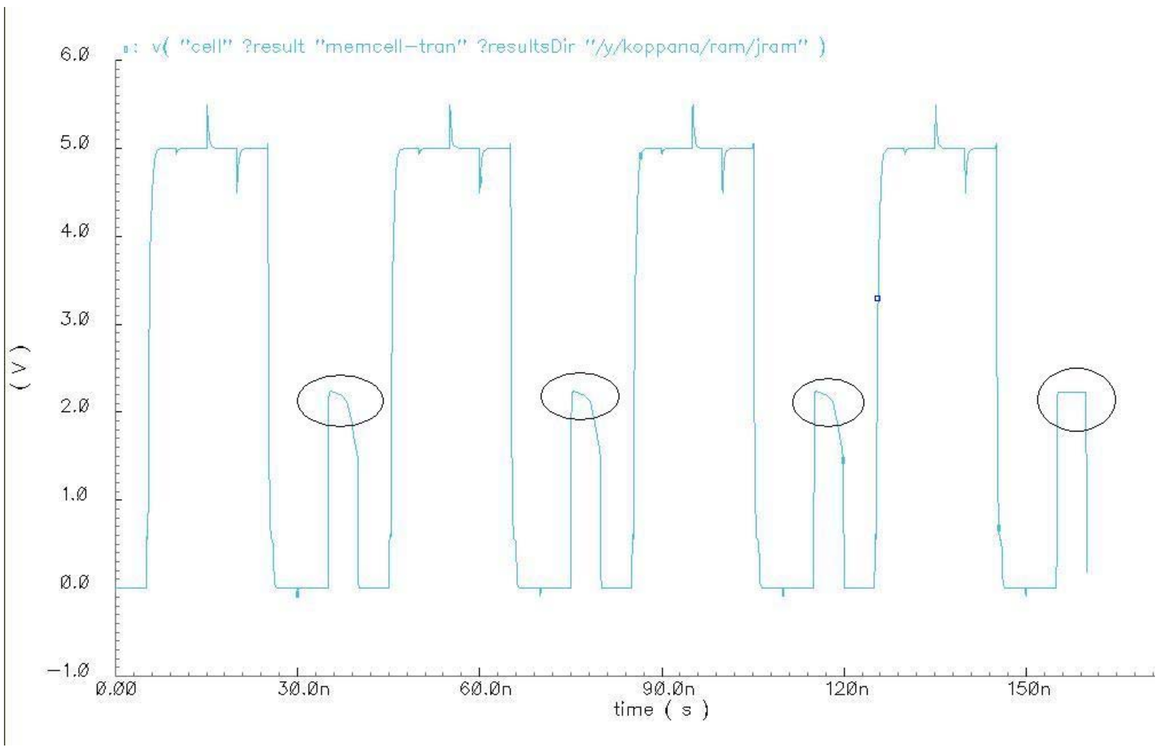

Fig.6.8 Data on cell node of J-SRAM model of 8-port SRAM cell.

Wn3 (in µm)	Wp1 (in µm)	Wn (in µm)	Vinv.sim (in volts)	Vcell.sim (in volts)	Vcellbar (in volts)	NM,cell (in volts)
4.2	4.2	2.1	3.06	2.23458	2.23490	0.8251
4.8	4.2	2.1	3.06	2.11571	2.11598	0.94429
5.4	4.2	2.1	3.06	2.00833	2.00833	1.05167
6.0	4.2	2.1	3.06	1.9103	1.91031	1.14969
6.6	4.2	2.1	3.06	1.82082	1.82224	1.23918
7.2	4.2	2.1	3.06	1.73868	1.74034	1.31966
7.8	4.2	2.1	3.06	1.66161	1.66273	1.39729
8.4	4.2	2.1	3.06	1.59254	1.59561	1.46439
9.0	4.2	2.1	3.06	1.52874	1.53141	1.52859

Table 6.7 Simulation results of J-SRAM memory cell, V_{cell}, $V_{cellbar}$, V_{inv}, & $NM_{, cell}$

6.6 Performance comparision

In this section, we would compare the performance of the all the various designs of 8-port discussed earlier: 2-SRAM, 4-SRAM, 8-SRAM, and J-SRAM memory cells. The thoeritical and simulation results are the evidence for the design performance in terms of noise margin, and read access times which are the basis of this comparison.Notice that;

1. 2-SRAM model has the got a noise margin of 0.1976 V with dimension of the W_n = 15.9 µm, W_p =1.2 µm, W_a = 1.5 µm. and read access time for these widths is 5ns.
2. 4-SRAM model has the got a noise margin of 0.52303 V with dimension of the W_n = 16.8 µm, W_p =1.2 µm, W_a = 1.5 µm. and read access time for these widths is 5ns.
3. 8-SRAM model has the got a noise margin of 0.6116 Vwith dimension of the W_n = 4.2 µm, W_P =1.2 µm, W_a = 1.5 µm. and read access time for these widths is 5ns.
4. J-SRAM model has the got a noise margin of 0.8251 V with dimension of the W_n = 4.2 µm, W_p = 4.2 µm, W_a = 2.1 µm. and read access time for these widths is 5ns.

Comparing all the results above, we notice that the J-SRAM has the best noise margin, and operates at minimum read access time with in the same area as the other layouts. The obvious preference would be J-SRAM for its innovative design. The buffers are designed so that they increase the inverter threshold voltage, there by increasing the noise margin in the memory cell. The detailed discussion of this design is done in chapter 4.

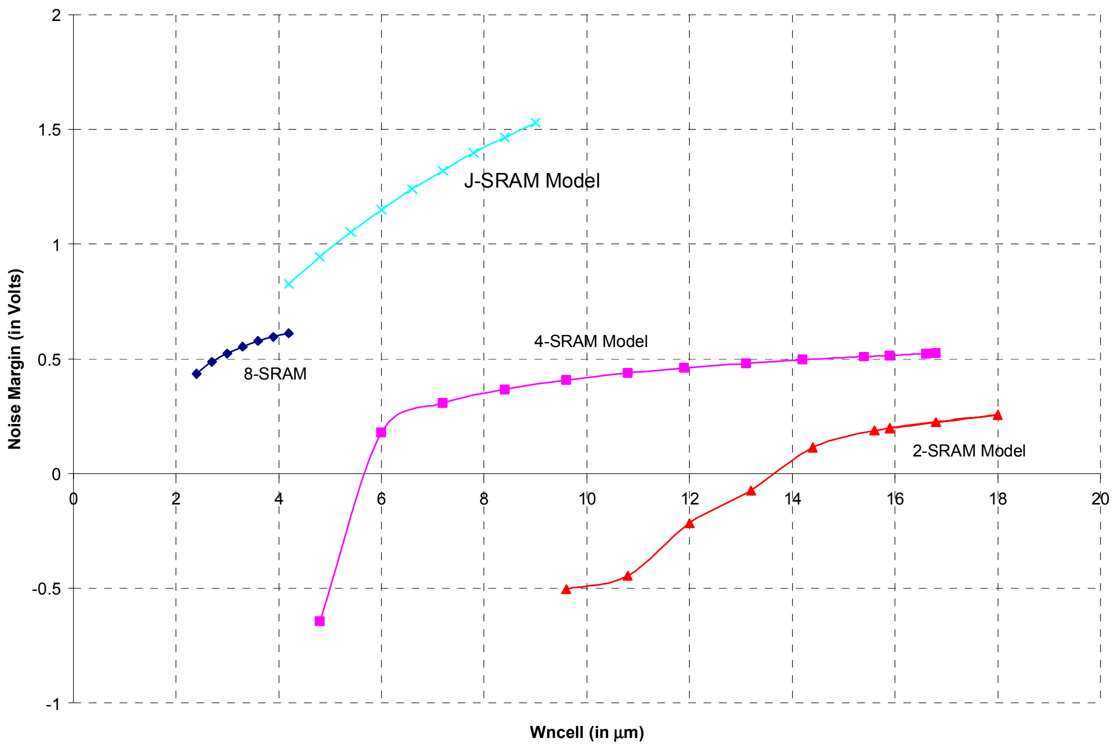

Fig 6.9 Performance comparisions with respect to attained noise margin.

6.7 Design issues of Multiport SRAM Memory cell

As a remainder, it's worth spending time discussing briefly the design constrains for multiport SRAM design. They are outlined as follows:

- Mutliport SRAM cells are used as a part of register file cells, and are used to implememt on-chip registers. And the main constrain for an on-chip register is read access time, so the memory has to be designed in a way that the speed of the register is greatest or the access time is as smallest possible for a given technology. So, being a designer, one should be in a look out for better way of designing the memory, and take all the design issues into consideration to improve the performance of the memory.

- The other factor, is the available silicon area for the design of a register file, in a multiport SRAM design, the bilines occupy maximum area, and designer have to live with this. within a limited area constrain the design, which can satisfy all the constrains, n-channel MOSFET sizing, pass transistors sizing, and main memory loop design within this area, with optimum speed and noise margin. There always exists a trade-off between area, speed, and noise margin. Apart from the bitlines, row decoders also occupy a large amount of space, for selecting memory location in a double ended write.

These issues are interrelated and interdependent, so all the design issues have to be considered for design a multiport SRAM cells.

6.8 Transistor Sizing

In the four proposed designs, 2-SRAM, 4-SRAM, 8-SRAM, and J-SRAM, transistor sizing improves the design performance. The first question that comes to mind is- by sizing which transistor the performance can be improved (in terms of read access time, noise margin). Primary aim of this thesis is to attain maximum possible noise margin in a limited area, so to attain maximum noise margin sizing n-channel mosfets in the memory cell would be best idea, then read access time for which sizing the access transistor would be productive. And when designing a register file sizing pre-charge circuit and write driver appropriately to the number of cells connected in a column is a prime issue. Each of which are discussed as under, reasons are outlines considering all the proposed design.

- Sizing N-MOSFET in memory cell that is increasing the width of the Channel.
    1. The n-mosfet that is considered here is the one conducting in the worst case scenario while reading a "0" through all the ports. The width of this

    transistor has to be scaled proportional to the number of access transistor connected on given node. This enables the memory cell to discharge the large bitline capacitances which are precharged high initially.

    2. There is a limitation to size this n-channel MOSFET, increasing it indiscriminately would decrease the inverter threshold voltage, there by decreasing the Noise margin. These two effects counter each other and usually there is no significant change in noise margin or increase is very small. The answer to this problem is J-SRAM design, where in the memory cell trip point or the inverter threshold voltage is set with a independent inverter (i.e by sizing the p-channel MOSFET) and the cell node voltage is tried to maintain low by sizing the n-channel MOSFET (independent) of the other transistors in the memory cell.

- Sizing P-MOSFET in memory cell.
    1. Increases the inverter threshold voltage thereby increasing the noise margin of the memory cell. We make use of this concept in the design of J-SRAM model where we increase the p-channel mosfet width to increase the noise margin.
- Sizing the n-mosfet access transistor connected to the nodes of the memory. Increasing the width of the channel would:
    1. Increases the capacitive loading on the bitlines which is undesirable. This leads to increase in the read noise when trying to read a "0" through all the ports.

2. Increases the capacitive loading on the wordlines, there by slowing down the row decoders used to select the memory location.
3. Decreases the read access time, which designer would like to do. Observe the width of the access transistor in J-SRAM model.

Decreasing the width of the channel would:

1. Increase the noise margin of the memory cell.
2. Increase the read access time.

6.9 Testing layouts

The layouts of various designs of 8-port SRAM memory cell are made use of CADENCE Vistuoso Layout Editor. The layouts of the designs are shown in their respective sections of the designs. The layouts are extracted and the netlist generated out of it gives us the details about the transistor sizes, paracitic capacitance, capacitance and the circuit network associated with the layout. A testbenchs used to test the layouts are listed in appendix C. Notice that the designs are tested for bitline and wordline line load of 32 bit and 32 word memory arrays. The precharge and the write driver are simulated in the test bench.

The sequence of operation used to test these memory cells are as follows:

- Writing a high “1” through port A
- Reading from all the available ports simultaneously.
- Writing low “0” through port B
- Reading from all the available ports simultaneously.
- Writing a high “1” through port C

- Reading from all the available ports simultaneously.
- Writing low “0” through port D
- Reading from all the available ports simultaneously.
- Writing a high “1” through port E
- Reading from all the available ports simultaneously.
- Writing low “0” through port F
- Reading from all the available ports simultaneously.
- Writing a high “1” through port G
- Reading from all the available ports simultaneously.
- Writing low “0” through port H
- Reading from all the available ports simultaneously.

Storage nodes voltages are monitored to calculate noise margin, and ensure correct readability and writability of the memory cell. The simulation results of various storage node outputs of the designs are shown in the figures as under:

In the figure, the spike when reading a “0” is of prime interest. The voltage spike is monitored in all the designs, for their corresponding cell node voltages. This voltage values are listed in tables 6.1, 6.3, 6.5 and 6.7 for 2-SRAM, 4-SRAM, 8-SRAM and J-SRAM models respectively for various widths of n-FET in the inverter loop, with W_a and W_p remaining at minimum dimensions.

The storage node voltages are listed in the table, similarly the complementary node voltages are also monitored, with these values the noise margin of the cell is calculated, the list of values for noise margin are listed in table 6.1, 6.3, 6.5 and 6.7. Theoritical values obtained are compared with simulation results with different width of n-channel FET in the inverter loop and the layouts are made of the the dimensions which has maximum noise margin possible within a limited area of 1600 µm^2.

6.10 Conclusion

Thus out of four working designs of 8-port SRAM design, J-SRAM has the best performance compaired with the rest of the design with a noise margin of 0.8251 V for a read access time of 5ns. The other designs namely 2-SRAM, 4-SRAM, and 8-SRAM work at this rate, but the noise margin of the design are very marginal and a small disturbance would make the memory to flip its state to a high or a low. And apart from this, the transistor dimension in the J-SRAM are close to minimal, rather than having transistors widths of the order 16 µm wide in both 2-SRAM and 4-SRAM.and 4.2 µm wide in 8-SRAM model. The noise margin can be increased to an order of 2 V in J-SRAM model by increasing the width of the n-channel mosfet; this is a trade off with area and noise margin.

Of all the proposed designs (2-SRAM, 4-SRAM, 8-SRAM, and JRAM) of 8-port SRAM memory cells. JRAM has got a better performance, then 8-SRAM, followed by 4-SRAM and 2-SRAM with 0.8251 V, 0.6116, 0.52303, and 0.19765 V respectively for read access time of 5nsec. These observations on performance are made with noise margin of the memory cell into consideration.

This thesis considers noise margin and read access time as the parameters to measure the performance of the memory cell. With these parmeters in to consideration we re-instate that the J-SRAM is the best, best bet to make one on-chip regiser file. And we have noticed that the width of the n-channel MOSFET improves the stability of the memory cell. Theoritcally the n-channel MOSFET increases proportionatly with increase in the number of ports.

6.11 Future Work

With advanced computer processor, 32 or 64 bit register for RISC processors is the need of the hour in basic memory appication. The best application would be to design a 64 bit registerfile with all the circuitry including sense amplifier (possibly differential sense amplifier) and test for noise margin and read access time. The design we would like to suggest for this study would be J-SRAM model that showed the best performance out of all the designs.

Secondly, one would like to test the design with real transistors and compare the theoretical, simulation results with the performance of the memory with real transistors.

References:

[1]. Neil H.E.Weste, David Harris. *“CMOS VLSI Design-A Circuit and System Perspective”*, 3rd Edition-Addison Wesley-2005

[2]. Neil H.E.Weste, Kamran Eshraghian. *“Principles of CMOS VLSI Design-A System Perspective”,* 2nd Edition-Prentice Hall-2000.

[3]. Forty.D.Daniel *“MOSFET modeling with SPICE: principles and practice”,* Prentice Hall 1997.

[4]. Dr.Louis.G.Johnson. *“MOS Device Characteristics (DC)-Digital VLSI Circuit Design”.* http://lgjohn.ecen.ceat.okstate.edu/5263/lectures/devchar.pdf - Feb 2005.

[5]. Dr.Louis.G.Johnson. *“Simple Gate Characteristics (DC)-Digital VLSI Circuit Design”.* http://lgjohn.ecen.ceat.okstate.edu/5263/lectures/simple.pdf - Feb 2005.

[6]. Dr.Louis.G.Johnson. *“Memory -Advanced Digital VLSI Circuit Design”.* http://lgjohn.ecen.ceat.okstate.edu/6263/lectures/memory.pdf - Sep 2004.

[7]. Dr.Louis.G.Johnson. *“Register File -Advanced Digital VLSI Circuit Design”.* http://lgjohn.ecen.ceat.okstate.edu/6263/lectures/regfile.pdf - Aug 2000.

[8]. Sunitha Manickavasakam, “Design of a 4-Port Register File”

[9]. Rajesh Kumar, *Desktop Platforms Group-Circuit Technology, Intel Corp. “Interconnect and Noise Immunity Design for the Pentium 4 processor”*. Intel Technology Journal Q1, 2001.

[10]. Eric S Fetzer, Mark Gibson, Anthony Klein, Naoim Calick, Chengyu Zhu, Eric Busta, and Baker Mohammed, *“A fully bypassed six issue integer datapath and*

register file on the itanium 2-microprocessor", IEEE journal of solid-state circuits,VOL.37, NO.11, November 2002.

[11]. Richard D. Jolly, *"A 9ns, 1.4-Gigabytes, 17-ported CMOS register file",* IEEE journal of solid-state circuits, VOL.26, NO.10, Pages 1407-1412, October 1991.

[12]. Allan L.Silburt et al., *"A 180 Mhz 0.8µm BiCMOS Modular Memory Family of DRAM and Multiport SRAM "*, IEEE journal of solid-state circuits, VOL.28, NO.3, Pages 222-232, March 1993.

[13]. Christian Reichling et al., "a Quad ported SRAM". http://www.gsi.de

[14]. Kenji Anami et al., "Design consideration of a static memory cell", IEEE journal of solid-state circuits, VOL. SC-18, No.4, Pages 414- 418, August 1983.

[15]. Spice model parameters for sub-micron technologies, http://www.mosis.org/Technical/Testdata/submicron-spice-parameters.html

Appendix A

BSIM3v3 Model Parameters

Second generation models (BSIM, HSPICE, Level 28, and BSIM2) place more emphasis on the mathematical conditioning of the transistor model equations for efficient circuit simulation. The model parameters are largely empirical in both structure and numerical values, providing little parameters information about the process technology. BSIM3v3 significantly changed the form of the model to guarantee continuous and smooth model equations; a number of empirical fitting expressions were also introduced, along with many additional parameters.

BSIM3v3 includes the extrinsic source/drain series resistance as a lumped resistance, rather than in the mobility model. The lumped resistance term R_{ds} by applying ohm's law to the linear region current expression:

In Linear region

$$I_{DS,lin} = \frac{V_{DS}}{R_{tot}} = \frac{V_{DS}}{(R_{chan} + R_{ds})},$$

$$R_{chan} = \frac{V_{DS}}{I_{DS|R_{DS}=0}},$$

$$I_{DS,lin} = \frac{\mu_e \left(\frac{\varepsilon_{ox}}{t_{ox}}\right)\left(\frac{W}{L}\right)(V_{GS} - V_T)V_{DS} - A_{bulk}V_{DS}^2/2}{1 + \frac{V_{DS}}{E_{sat}L}},$$

$$E_{sat} = \frac{2v_{sat}}{\mu_e}$$

Where; R_{chan} is the channel resistance, and R_{ds} is source/drain series resistance.

Modeling parasitic resistance in a direct method yields a complicated drain-current expression. R_{ds} , will have a more significant effect on device characteristics.

R_{ds} , parasitic source-drain resistance per unit width is given by;

$$R_{ds} = \frac{RDSW\left\{1 + PRWG\left(V_{GS} - V_T\right) + PRWB\left[\left(\phi_s - V_{BSeff}\right)^{\frac{1}{2}} - \phi^{\frac{1}{2}}\right]\right\}}{\left(10^{-6}.W\right)^{WR}}$$

Where; $RDSW$ -resistance per unit channel width, $PRWG$ -gate bias effect coefficient, and $PRWB$ -substrate bias effect (body effect). Note that the width dependence factor $RDSW$ is much appropriate than the lumped dependence factor RS and RD, it also allows a model to be valid for an arbitrary effective channel width.

Considering, factor R_{ds} we can rewrite the current equations for the linear region of operation as follows;

$$I_{DS,lin} = \frac{I_{DS,lin,R_{ds=0}}}{1 + \frac{R_{ds}.I_{DS,lin,R_{ds=0}}}{V_{DS}}}$$

Where, $I_{DS,lin,R_{ds=0}}$ is drain current when $R_{ds} = 0$;

In Saturation region

For $R_{ds} = 0$,

$$I_{DS,sat} = \frac{\mu_e}{2}.\frac{\varepsilon_{ox}}{t_{ox}}\frac{W}{L}.V_{DSsat}\left(V_{GS} - V_T\right)\left(1 + \frac{V_{DS} - V_{DSsat}}{V_A}\right)$$

$$V_{DSsat} = \frac{\left(V_{GS} - V_T\right)E_{sat}L}{\left(V_{GS} - V_T\right) + A_{bulk}.E_{sat}.L}$$

Now, consider the factor R_{ds} into the saturation region of operation we get the drain current equation as;

$$I_{DS,sat} = \frac{\varepsilon_{ox}}{t_{ox}}.W.v_{sat}.\left[\left(V_{GS}-V_T\right)-A_{bulk}.|V_{DSsat}|\right]\left(1+\frac{V_{DS}-V_{DSsat}}{V_A}\right)$$

$$|V_{DSsat}| = \frac{-b \pm (b^2-4.a.c)^{\frac{1}{2}}}{2.a}$$

where,

$$a = A_{bulk}^2.R_{ds}.C_{ox}.W_{eff}.v_{sat}\left(T\right)+\left(\frac{1}{\lambda}-1\right).A_{bulk}$$

$$b = -\left[\left(V_{GS}-V_T\right).\left(\frac{2}{\lambda}-1\right)+A_{bulk}.E_{sat}.L_{eff}+3.A_{bulk}.R_{ds}.C_{ox}.W_{eff}.v_{sat}.\left(V_{GS}-V_T\right)\right]$$

$$c = E_{sat}.L_{eff}.\left(V_{GS}-V_T\right)+2.R_{ds}.C_{ox}.W_{eff}.v_{sat}.\left(V_{GS}-V_T\right)^2$$

and,

$$\lambda = A_1.\left(V_{GS}-V_T\right)+A_2.$$

Summarizing current equations for n-channel MOSFET for regions of operation as under;

Region	I D	Limits
Cutoff	0	$V_{GS} < V_{Tn}$
Linear(Ohmic)	$I_{DS} = \dfrac{\mu_e\left(\frac{\varepsilon_{ox}}{t_{ox}}\right)\left(\frac{W}{L}\right)\left(V_{GS}-V_{Tn}\right)V_{DS}-A_{bulk}V_{DS}^2/2}{1+\frac{V_{DS}}{E_{sat}L}},$	$V_{GS} > V_{Tn}$ $V_{DS} < V_{DSsat}$
Saturation	$I_{DS,sat} = \frac{\mu_e}{2}.\frac{\varepsilon_{ox}}{t_{ox}}\frac{W}{L}.V_{DSsat}\left(V_{GS}-V_{Tn}\right)\left(1+\frac{V_{DS}-V_{DSsat}}{V_A}\right)$	$V_{GS} > V_{Tn}$ $V_{DS} > V_{DSsat}$

Table A.1.1 Drain/Source current equations for n-channel MOSFET, considering R_{ds} = 0;

Where; $E_{sat} = \dfrac{2v_{sat}}{\mu_e}$

$$V_{DSsat} = \frac{\left(V_{GS}-V_{Tn}\right)E_{sat}L}{\left(V_{GS}-V_{Tn}\right)+A_{bulk}.E_{sat}.L}$$

Region	I $_D$	Limits
Cutoff	0	$V_{GS} < V_{Tn}$
Linear	$I_{DS,lin} = \dfrac{I_{DS,lin,R_{ds=0}}}{1+\dfrac{R_{ds}.I_{DS,lin,R_{ds=0}}}{V_{DS}}}$	$V_{GS} > V_{Tn}$ $V_{DS} < V_{DSsat}$
Saturation	$I_{DS,sat} = \dfrac{\varepsilon_{ox}}{t_{ox}}.W.v_{sat}.\left[\left(V_{GS}-V_{Tn}\right)-A_{bulk}.\lvert V_{DSsat}\rvert\right]\left(1+\dfrac{V_{DS}-V_{DSsat}}{V_A}\right)$	$V_{GS} > V_{Tn}$ $V_{DS} > V_{DSsat}$

Table A.1.2 Drain/Source current equations for n-channel MOSFET, considering $R_{ds} > 0$;

Where, $R_{ds} = \dfrac{RDSW\left\{1+PRWG\left(V_{GS}-V_{Tn}\right)+PRWB\left[\left(\phi_s - V_{BSeff}\right)^{\frac{1}{2}}-\phi^{\frac{1}{2}}\right]\right\}}{\left(10^{-6}.W\right)^{WR}}$

$$\lvert V_{DSsat}\rvert = \frac{-b \pm (b^2-4.a.c)^{\frac{1}{2}}}{2.a}$$

where,

$$a = A_{bulk}^2.R_{ds}.C_{ox}.W_{eff}.v_{sat}\left(T\right)+\left(\frac{1}{\lambda}-1\right).A_{bulk}$$

$$b = -\left[\left(V_{GS}-V_{Tn}\right).\left(\frac{2}{\lambda}-1\right)+A_{bulk}.E_{sat}.L_{eff}+3.A_{bulk}.R_{ds}.C_{ox}.W_{eff}.v_{sat}.\left(V_{GS}-V_{Tn}\right)\right]$$

$$c = E_{sat}.L_{eff}.\left(V_{GS}-V_{Tn}\right)+2.R_{ds}.C_{ox}.W_{eff}.v_{sat}.\left(V_{GS}-V_{Tn}\right)^2$$

and,

$$\lambda = A_1.\left(V_{GS}-V_{Tn}\right)+A_2.$$

Summarizing current equations for p-channel MOSFET for regions of operation as under;

Region	I D	Limits
Cutoff	0	$V_{GS} > V_{Tp}$
Linear(Ohmic)	$I_{DS} = \dfrac{\mu_e \left(\dfrac{\varepsilon_{ox}}{t_{ox}}\right)\left(\dfrac{W}{L}\right)\left(\mid V_{GS} - V_{Tp} \mid\right) V_{DS} - A_{bulk} V_{DS}^2 / 2}{1 + \dfrac{\mid V_{DS} \mid}{E_{sat} L}},$	$V_{GS} < V_{Tp}$ $V_{DS} > V_{DSsat}$
Saturation	$I_{DS,sat} = \dfrac{\mu_e}{2} . \dfrac{\varepsilon_{ox}}{t_{ox}} \dfrac{W}{L} . \mid V_{DSsat} \mid \left(\mid V_{GS} - V_{Tp} \mid\right)\left(1 + \dfrac{\mid V_{DS} \mid - \mid V_{DSsat} \mid}{V_A}\right)$	$V_{GS} < V_{Tp}$ $V_{DS} < V_{DSsat}$

Table A.2.1 Drain/Source current equations for p-channel MOSFET, considering R_{ds} = 0.

Where; $E_{sat} = \dfrac{2v_{sat}}{\mu_p}$

$$V_{DSsat} = -\frac{\left(\mid V_{GS} - V_{Tp} \mid\right) E_{sat} L}{\left(\mid V_{GS} - V_{Tp} \mid\right) + A_{bulk} . E_{sat} . L}$$

Region	I D	Limits
Cutoff	0	$V_{GS} > V_{Tp}$
Linear	$I_{DS,lin} = \dfrac{I_{DS,lin,R_{ds=0}}}{1+\dfrac{R_{ds}.I_{DS,lin,R_{ds=0}}}{V_{DS}}}$	$V_{GS} < V_{Tp}$ $V_{DS} > V_{DSsat}$
Saturation	$I_{DS,sat} = \dfrac{\varepsilon_{ox}}{t_{ox}}.W.v_{sat}.\left[\left(\lvert V_{GS}-V_{Tp}\rvert\right)-A_{bulk}.\lvert V_{DSsat}\rvert\right]\left(1+\dfrac{\lvert V_{DS}-V_{DSsat}\rvert}{V_A}\right)$	$V_{GS} < V_{Tp}$ $V_{DS} < V_{DSsat}$

Table A.2.2 Drain/Source current equations for p-channel MOSFET, considering $R_{ds} > 0$;

Where, $R_{ds} = \dfrac{RDSW\left\{1+PRWG\left(V_{GS}-V_{Tp}\right)+PRWB\left[\left(\phi_s - V_{BSeff}\right)^{\frac{1}{2}}-\phi^{\frac{1}{2}}\right]\right\}}{\left(10^{-6}.W\right)^{WR}}$

$$|V_{DSsat}| = \frac{-b \pm (b^2-4.a.c)^2}{2.a}$$

where,

$$a = A_{bulk}^2.R_{ds}.C_{ox}.W_{eff}.v_{sat}\left(T\right)+\left(\frac{1}{\lambda}-1\right).A_{bulk}$$

$$b = -\left[\left(\lvert V_{GS}-V_{Tp}\rvert\right).\left(\frac{2}{\lambda}-1\right)+A_{bulk}.E_{sat}.L_{eff}+3.A_{bulk}.R_{ds}.C_{ox}.W_{eff}.v_{sat}.\left(\lvert V_{GS}-V_{Tp}\rvert\right)\right]$$

$$c = E_{sat}.L_{eff}.\left(\lvert V_{GS}-V_{Tp}\rvert\right)+2.R_{ds}.C_{ox}.W_{eff}.v_{sat}.\left(V_{GS}-V_{Tp}\right)^2$$

and,

$$\lambda = A_1.\left(\lvert V_{GS}-V_{Tp}\rvert\right)+A_2.$$

Appendix B

B.1 Technology file for AMI 06 N bsim3v3.

```
model ami06N bsim3v3 type = n
+version = 3.1          tnom    = 27            tox     = 1.41E-8
+xj      = 1.5E-7       nch     = 1.7E17        vth0    = 0.7086
+k1      = 0.8354582    k2      = -0.088431     k3      = 41.4403818
+k3b     = -14          w0      = 6.480766E-7   nlx     = 1E-10
+dvt0w   = 0            dvt1w   = 5.3E6         dvt2w   = -0.032
+dvt0    = 3.6139113    dvt1    = 0.3795745     dvt2    = -0.1399976
+u0      = 533.6953445  ua      = 7.558023E-10 ub      = 1.181167E-18
+uc      = 2.582756E-11 vsat    = 1.300981E5    a0      = 0.5292985
+ags     = 0.1463715    b0      = 1.283336E-6  b1      = 1.408099E-6
+keta    = -0.0173166   a1      = 0             a2      = 1
+rdsw    = 2.268366E3   prwg    = -1E-3        prwb    = 6.320549E-5
+wr      = 1            wint    = 2.043512E-7  lint    = 3.034496E-8
+xl      = 0            xw      = 0            dwg     = -1.446149E-8
+dwb     = 2.077539E-8  voff    = -0.1137226    nfactor = 1.2880596
+cit     = 0            cdsc    = 1.506004E-4   cdscd   = 0
+cdscb   = 0            eta0    = 3.815372E-4  etab    = -1.029178E-3
+dsub    = 2.173055E-4  pclm    = 0.6171774     pdiblc1 = 0.185986
+pdiblc2 = 3.473187E-3  pdiblcb = -1E-3         drout   = 0.4037723
+pscbe1  = 5.998012E9   pscbe2  = 3.788068E-8   pvag    = 0.012927
+delta   = 0.01         mobmod  = 1             prt     = 0
+ute     = -1.5         kt1     = -0.11         kt1l    = 0
+kt2     = 0.022        ua1     = 4.31E-9       ub1     = -7.61E-18
+uc1     = -5.6E-11     at      = 3.3E4         wl      = 0
+wln     = 1            ww      = 0             wwn     = 1
+wwl     = 0            ll      = 0             lln     = 1
+lw      = 0            lwn     = 1             lwl     = 0
+capmod  = 2            xpart   = 0.4           cgdo    = 1.99E-10
+cgso    = 1.99E-10     cgbo    = 0            cj      = 4.233802E-4
+pb      = 0.9899238    mj      = 0.4495859    cjsw    = 3.825632E-10
+pbsw    = 0.1082556    mjsw    = 0.1083618     pvth0   = 0.0212852
+prdsw   = -16.1546703  pk2     = 0.0253069     wketa   = 0.0188633
+lketa   = 0.0204965
```

B.2 Technology file for AMI 06 P bsim3v3

```
model ami06P bsim3v3 type = p
+version = 3.1            tnom    = 27             tox     = 1.41E-8
+xj      = 1.5E-7         nch     = 1.7E17         vth0    = -0.9179952
+k1      = 0.5575604      k2      = 0.010265       k3      = 14.0655075
+k3b     = -2.3032921     w0      = 1.147829E-6  nlx     = 1.114768E-10
+dvt0w   = 0              dvt1w   = 5.3E6          dvt2w   = -0.032
+dvt0    = 2.2896412      dvt1    = 0.5213085      dvt2    = -0.1337987
+u0      = 202.4540953    ua      = 2.290194E-9    ub      = 9.779742E-
19
+uc      = -3.69771E-11   vsat    = 1.307891E5     a0      = 0.8356881
+ags     = 0.1568774      b0      = 2.365956E-6    b1      = 5E-6
+keta    = -5.769328E-3   a1      = 0              a2      = 1
+rdsw    = 2.746814E3     prwg    = 2.34865E-3     prwb    = 0.0172298
+wr      = 1              wint    = 2.586255E-7   lint    = 7.205014E-8
+xl      = 0              xw      = 0            dwg     = -2.133054E-8
+dwb     = 9.857534E-9    voff    = -0.0837499     nfactor = 1.2415529
+cit     = 0              cdsc    = 4.363744E-4    cdscd   = 0
+cdscb   = 0              eta0    = 0.11276        etab    = -2.9484E-3
+dsub    = 0.3389402      pclm    = 4.9847806     pdiblc1 = 2.481735E-5
+pdiblc2 = 0.01           pdiblcb = 0              drout   = 0.9975107
+pscbe1  = 3.497872E9     pscbe2  = 4.974352E-9    pvag    = 10.9914549
+delta   = 0.01           mobmod  = 1              prt     = 0
+ute     = -1.5           kt1     = -0.11          kt1l    = 0
+kt2     = 0.022          ua1     = 4.31E-9        ub1     = -7.61E-18
+uc1     = -5.6E-11       at      = 3.3E4          wl      = 0
+wln     = 1              ww      = 0              wwn     = 1
+wwl     = 0              ll      = 0              lln     = 1
+lw      = 0              lwn     = 1              lwl     = 0
+capmod  = 2              xpart   = 0.4            cgdo    = 2.4E-10
+cgso    = 2.4E-10        cgbo    = 0             cj      = 7.273568E-4
+pb      = 0.9665597      mj      = 0.4959837    cjsw    = 3.114708E-10
+pbsw    = 0.99           mjsw    = 0.2653654     pvth0   = 9.420541E-3
+prdsw   = -231.2571566   pk2     = 1.396684E-3   wketa   = 1.862966E-3
+lketa   = 5.728589E-3
```

Appendix C

C.1 Test bench module and netlist to simulate CMOS DC Inverter characteristics

```
// power supplies
VPWR(vdd 0) vsource dc=5.0
VGND(gnd 0) vsource dc=0.0

// inputs
VIN(IN 0) vsource dc=1.5 type=pulse val0=0 val1=5\
  period=12n rise=1000p fall=1000p width=5n

opts1 options pwr=total save=all
opts2 options currents=all


// controls
//inverter tran stop=3n
dc dc dev=VIN param=dc start=0 stop=5 step=0.01
save OUT IN


// Library name: inverter_lib
// Cell name: inverter
// View name: extracted
_inst0 (OUT IN vdd vdd) ami06P w=4.2e-06 l=6e-07 as=1.44e-11 ad=1.44e-11 \
      ps=1.26e-05 pd=1.26e-05 m=1 region=sat
_inst1 (OUT IN gnd gnd) ami06N w=1.2e-06 l=6e-07 as=3.6e-12 ad=3.6e-12 \
      ps=5.4e-06 pd=5.4e-06 m=1 region=sat
_inst2 (OUT gnd) capacitor c=100e-15 m=1
```

C.2 Test bench module and netlist to simulate 2-SRAM model of 8-port Memory cell operating for a read access time of 5nsec

```
// simulate load cap on bit lines
clbitA (bitA 0) capacitor c=175f
clbitAbar (bitAbar 0) capacitor c=175f
clbitB (bitB 0) capacitor c=175f
clbitBbar (bitBbar 0) capacitor c=175f
clbitC (bitC 0) capacitor c=175f
clbitCbar (bitCbar 0) capacitor c=175f
clbitD (bitD 0) capacitor c=175f
clbitDbar (bitDbar 0) capacitor c=175f
clbitE (bitE 0) capacitor c=175f
clbitEbar (bitEbar 0) capacitor c=175f
clbitF (bitF 0) capacitor c=175f
clbitFbar (bitFbar 0) capacitor c=175f
clbitG (bitG 0) capacitor c=175f
clbitGbar (bitGbar 0) capacitor c=175f
clbitH (bitH 0) capacitor c=175f
clbitHbar (bitHbar 0) capacitor c=175f

clwordA (wordA 0) capacitor  c=145f
clwordAbar (wordAbar 0) capacitor  c=145f
clwordB (wordB 0) capacitor c=145f
clwordBbar (wordBbar 0) capacitor c=145f
clwordC (wordC 0) capacitor c=145f
clwordCbar (wordCbar 0) capacitor c=145f
clwordD (wordD 0) capacitor c=145f
clwordDbar (wordDbar 0) capacitor c=145f
clwordE (wordE 0) capacitor c=145f
clwordEbar (wordEbar 0) capacitor c=145f
clwordF (wordF 0) capacitor c=145f
clwordFbar (wordFbar 0) capacitor c=145f
clwordG (wordG 0) capacitor c=145f
clwordGbar (wordGbar 0) capacitor c=145f
clwordH (wordH 0) capacitor c=145f
clwordHbar (wordHbar 0) capacitor c=145f

// add precharge transistors
mprebitA (bitA prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitAbar (bitAbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitB (bitB prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
```

```
mprebitBbar (bitBbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitC (bitC prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitCbar (bitCbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitD (bitD prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitDbar (bitDbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitE (bitE prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitEbar (bitEbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitF (bitF prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitFbar (bitFbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitG (bitG prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitGbar (bitGbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitH (bitH prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitHbar (bitHbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06

// add write drivers
mwbitA (bitA bitAw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitAbar (bitAbar bitAbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitB (bitB bitBw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitBbar (bitBbar bitBbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitC (bitC bitCw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitCbar (bitCbar bitCbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitD (bitD bitDw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitDbar (bitDbar bitDbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitE (bitE bitEw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitEbar (bitEbar bitEbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitF (bitF bitFw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitFbar (bitFbar bitFbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitG (bitG bitGw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitGbar (bitGbar bitGbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitH (bitH bitHw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitHbar (bitHbar bitHbarw err 0) ami06N w=24.0e-06 l=0.6e-06

//power supplies
VPWR(vdd 0) vsource dc=5.0
VGND(gnd 0) vsource dc=0.0
ERROR(err 0) vsource dc=0.0



// write 1 on A, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on B, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 1 on C, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
```

```
// write 0 on D, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 1 on E, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on F, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 1 on G, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on H, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar


vpre (prebar 0) vsource type=pwl wave=[0n 0 4.9n 0 5n 5 10n 5 10.1n 0 14.9n 0 15n 5
20n 5 20.1n 0 24.9n 0 25n 5 30n 5 30.1n 0 34.9n 0 35n 5 40n 5 40.1n 0 44.9n 0 45n 5 50n
5 50.1n 0 54.9n 0 55n 5 60n 5 60.1n 0 64.9n 0 65n 5 70n 5 70.1n 0 74.9n 0 75n 5 80n 5
80.1n 0 84.9n 0 85n 5 \
90n 5 90.1n 0 94.9n 0 95n 5 100n 5 100.1n 0 104.9n 0 105n 5 110n 5 110.1n 0 114.9n 0
115n 5 120n 5 120.1n 0 124.9n 0 125n 5 130n 5 130.1n 0 134.9n 0 135n 5 140n 5 140.1n
0 144.9n 0 145n 5 150n 5 150.1n 0] \
pwlperiod=160n

vwordA (wordA 0) vsource type=pwl wave=[0n 0 5n 0 5.1n 5 9.9n 5 10n 0 15n 0 15.1n 5
19.9n 5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5
60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordAbar (wordAbar 0) vsource type=pwl wave=[0n 0 5n 0 5.1n 5 9.9n 5 10n 0 15n 0
15.1n 5 19.9n 5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordB (wordB 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 25.1n 5 29.9n 5 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n
```

```
vwordBbar (wordBbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 25.1n 5 29.9n 5 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n
5 59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordC (wordC 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 45.1n 5 49.9n 5 50n 0 55n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0  75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordCbar (wordCbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 45.1n 5 49.9n 5 50n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordD (wordD 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
65.1n 5 69.9n 5 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordDbar (wordDbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 65.1n 5 69.9n 5 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0  95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0
125n 0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5
160n 0] \
pwlperiod=160n

vwordE (wordE 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n 0
25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 85.1n 5 89.9n 5 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordEbar (wordEbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5
20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 85.1n 5 89.9n 5 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordF (wordF 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n 0
25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 105.1n 5 109.9n 5 110n 0 115n 0 115.1n 5
119.9n 5 120n 0 125n 0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordFbar (wordFbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5
20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 105.1n 5 109.9n 5 110n 0 115n 0 115.1n 5
119.9n 5 120n 0 125n 0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordG (wordG 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 125.1n 5 129.9n 5 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordGbar (wordGbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 125.1n 5 129.9n 5 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordH (wordH 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 145.1n 5 149.9n 5 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordHbar (wordHbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15.1n 5 19.9n 5 20n
0 25n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55.1n 5 59.9n 5 60n 0 65n 0 70n 0 75.1n 5
79.9n 5 80n 0 85n 0 \
90n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115.1n 5 119.9n 5 120n 0 125n 0 130n 0
135.1n 5 139.9n 5 140n 0 145n 0 145.1n 5 149.9n 5 150n 0 155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n


vbitAw (bitAw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitAbarw (bitAbarw 0) vsource type=pwl wave=[0n 0 5n 0 5.1n 5 9.9n 5 10n 0 15n 0
20n 0 25n 0 30n 0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0
90n 0 95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitBw (bitBw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 25.1n 5
29.9n 5 30n 0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0
95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitBbarw (bitBbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n

vbitCw (bitCw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitCbarw (bitCbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 45.1n 5 49.9n 5 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0
90n 0 95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n
```

```
vbitDw (bitDw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 65.1n 5 69.9n 5 70n 0 75n 0 80n 0 85n 0 90n 0
95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitDbarw (bitDbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n

vbitEw (bitEw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitEbarw (bitEbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 85.1n 5 89.9n 5
90n 0 95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitFw (bitFw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 105.1n 5 109.9n 5 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitFbarw (bitFbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n

vbitGw (bitGw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitGbarw (bitGbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 125.1n 5 129.9n 5 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n
```

```
vbitHw (bitHw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 145.1n 5 149.9n 5 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitHbarw (bitHbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n


//power calc


opts1 options pwr=total save=all
opts2 options currents=all


//
ic cell=0 cellbar=5
memcell tran stop=160.0n
save cell cellbar bitA bitAbar bitB bitBbar bitC bitCbar bitD bitDbar bitE bitEbar bitF
bitFbar bitG bitGbar bitH bitHbar


// Netlist generated from the extracted view of the layout

// Generated for: spectre
// Generated on: Apr 10 20:55:02 2005
// Design library name: ram_lib
// Design cell name: 2ram
// Design view name: extracted


// Library name: ram_lib
// Cell name: 2ram
// View name: extracted
_inst0 (gnd cellbar) capacitor c=7.03131e-15 m=1
_inst1 (wordCbar gnd) capacitor c=9.12186e-15 m=1
_inst2 (wordEbar gnd) capacitor c=9.11235e-15 m=1
_inst3 (bitH gnd) capacitor c=2.03862e-15 m=1
_inst4 (wordF gnd) capacitor c=9.34026e-15 m=1
_inst5 (wordE gnd) capacitor c=2.14158e-15 m=1
```

```
_inst6 (cell gnd) capacitor c=6.9624e-15 m=1
_inst7 (wordD gnd) capacitor c=9.3504e-15 m=1
_inst8 (gnd bitDbar) capacitor c=3.28377e-15 m=1
_inst9 (gnd bitFbar) capacitor c=2.51801e-15 m=1
_inst10 (gnd vdd) capacitor c=4.40607e-15 m=1
_inst11 (gnd bitHbar) capacitor c=3.43521e-15 m=1
_inst12 (gnd bitAbar) capacitor c=3.25206e-15 m=1
_inst13 (bitCbar gnd) capacitor c=2.63196e-15 m=1
_inst14 (bitEbar gnd) capacitor c=3.2689e-15 m=1
_inst15 (bitH gnd) capacitor c=3.01554e-15 m=1
_inst16 (bitG gnd) capacitor c=3.159e-15 m=1
_inst17 (bitF gnd) capacitor c=2.87145e-15 m=1
_inst18 (bitE gnd) capacitor c=2.06051e-15 m=1
_inst19 (bitC gnd) capacitor c=2.29688e-15 m=1
_inst20 (bitB gnd) capacitor c=2.89211e-15 m=1
_inst21 (bitA gnd) capacitor c=3.06284e-15 m=1
_inst22 (bitGbar gnd) capacitor c=2.35454e-15 m=1
_inst23 (bitBbar gnd) capacitor c=2.00112e-15 m=1
_inst24 (gnd wordDbar) capacitor c=3.05772e-15 m=1
_inst25 (gnd wordFbar) capacitor c=3.34704e-15 m=1
_inst26 (gnd wordHbar) capacitor c=3.85656e-15 m=1
_inst27 (gnd wordAbar) capacitor c=3.85794e-15 m=1
_inst28 (wordGbar gnd) capacitor c=2.7567e-15 m=1
_inst29 (wordH gnd) capacitor c=3.68064e-15 m=1
_inst30 (wordG gnd) capacitor c=2.82192e-15 m=1
_inst31 (wordE wordF) capacitor c=2.15649e-15 m=1
_inst32 (wordC gnd) capacitor c=4.15746e-15 m=1
_inst33 (wordB gnd) capacitor c=4.11612e-15 m=1
_inst34 (wordA gnd) capacitor c=2.53502e-15 m=1
_inst35 (wordBbar gnd) capacitor c=3.87486e-15 m=1
_inst36 (cellbar cell vdd vdd) ami06P w=1.2e-06 l=6e-07 as=1.08e-12 \
      ad=1.8e-12 ps=1.8e-06 pd=4.2e-06 m=1 region=sat
_inst37 (vdd cellbar cell vdd) ami06P w=1.2e-06 l=6e-07 as=1.8e-12 \
      ad=1.08e-12 ps=4.2e-06 pd=1.8e-06 m=1 region=sat
_inst38 (cellbar cell gnd gnd) ami06N w=1.59e-05 l=6e-07 as=1.431e-11 \
      ad=2.385e-11 ps=1.8e-06 pd=1.89e-05 m=1 region=sat
_inst39 (gnd cellbar cell gnd) ami06N w=1.59e-05 l=6e-07 as=2.385e-11 \
      ad=1.431e-11 ps=1.89e-05 pd=1.8e-06 m=1 region=sat
_inst40 (bitHbar wordHbar cellbar gnd) ami06N w=1.5e-06 l=6e-07 \
      as=2.25e-12 ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst41 (bitH wordH cell gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst42 (cellbar wordGbar bitGbar gnd) ami06N w=1.5e-06 l=6e-07 \
      as=2.25e-12 ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst43 (bitG wordG cell gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
```

```
_inst44 (cellbar wordFbar bitFbar gnd) ami06N w=1.5e-06 l=6e-07 \
        as=2.25e-12 ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst45 (bitF wordF cell gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
        ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst46 (cellbar wordEbar bitEbar gnd) ami06N w=1.5e-06 l=6e-07 \
        as=2.25e-12 ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst47 (bitE wordE cell gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
        ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst48 (cellbar wordDbar bitDbar gnd) ami06N w=1.5e-06 l=6e-07 \
        as=2.25e-12 ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst49 (bitD wordD cell gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
        ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst50 (cellbar wordCbar bitCbar gnd) ami06N w=1.5e-06 l=6e-07 \
        as=2.25e-12 ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst51 (bitC wordC cell gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
        ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst52 (cellbar wordBbar bitBbar gnd) ami06N w=1.5e-06 l=6e-07 \
        as=2.25e-12 ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst53 (bitB wordB cell gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
        ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst54 (cell wordA bitA gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
        ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst55 (cellbar wordAbar bitAbar gnd) ami06N w=1.5e-06 l=6e-07 \
        as=2.25e-12 ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
```

C.3 Test bench module and netlist to simulate 4-SRAM model of 8-port Memory cell operating for a read access time of 5nsec

```
//
//
// simulate load cap on bit lines
clbitA (bitA 0) capacitor c=170f
clbitAbar (bitAbar 0) capacitor c=170f
clbitB (bitB 0) capacitor c=170f
clbitBbar (bitBbar 0) capacitor c=170f
clbitC (bitC 0) capacitor c=170f
clbitCbar (bitCbar 0) capacitor c=170f
clbitD (bitD 0) capacitor c=170f
clbitDbar (bitDbar 0) capacitor c=170f
clbitE (bitE 0) capacitor c=170f
clbitEbar (bitEbar 0) capacitor c=170f
clbitF (bitF 0) capacitor c=170f
clbitFbar (bitFbar 0) capacitor c=170f
clbitG (bitG 0) capacitor c=170f
clbitGbar (bitGbar 0) capacitor c=170f
clbitH (bitH 0) capacitor c=170f
clbitHbar (bitHbar 0) capacitor c=170f

clwordA (wordA 0) capacitor  c=125f
clwordAbar (wordAbar 0) capacitor  c=125f
clwordB (wordB 0) capacitor c=125f
clwordBbar (wordBbar 0) capacitor c=125f
clwordC (wordC 0) capacitor c=125f
clwordCbar (wordCbar 0) capacitor c=125f
clwordD (wordD 0) capacitor c=125f
clwordDbar (wordDbar 0) capacitor c=125f
clwordE (wordE 0) capacitor c=125f
clwordEbar (wordEbar 0) capacitor c=125f
clwordF (wordF 0) capacitor c=125f
clwordFbar (wordFbar 0) capacitor c=125f
clwordG (wordG 0) capacitor c=125f
clwordGbar (wordGbar 0) capacitor c=125f
clwordH (wordH 0) capacitor c=125f
clwordHbar (wordHbar 0) capacitor c=125f

// add precharge transistors
mprebitA (bitA prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitAbar (bitAbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitB (bitB prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitBbar (bitBbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
```

```
mprebitC (bitC prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitCbar (bitCbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitD (bitD prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitDbar (bitDbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitE (bitE prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitEbar (bitEbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitF (bitF prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitFbar (bitFbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitG (bitG prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitGbar (bitGbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitH (bitH prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitHbar (bitHbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06

// add write drivers
mwbitA (bitA bitAw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitAbar (bitAbar bitAbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitB (bitB bitBw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitBbar (bitBbar bitBbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitC (bitC bitCw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitCbar (bitCbar bitCbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitD (bitD bitDw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitDbar (bitDbar bitDbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitE (bitE bitEw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitEbar (bitEbar bitEbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitF (bitF bitFw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitFbar (bitFbar bitFbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitG (bitG bitGw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitGbar (bitGbar bitGbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitH (bitH bitHw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitHbar (bitHbar bitHbarw 0 0) ami06N w=120e-06 l=0.6e-06

//power supplies
VPWR(vdd 0) vsource dc=5.0
VGND(gnd 0) vsource dc=0.0



// write 1 on A, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on B, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 1 on C, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on D, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
```

```
// write 1 on E, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on F, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 1 on G, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on H, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar

vpre (prebar 0) vsource type=pwl wave=[0n 0 4.9n 0 5n 5 10n 5 10.1n 0 14.9n 0 15n 5
20n 5 20.1n 0 24.9n 0 25n 5 30n 5 30.1n 0 34.9n 0 35n 5 40n 5 40.1n 0 44.9n 0 45n 5 50n
5 50.1n 0 54.9n 0 55n 5 60n 5 60.1n 0 64.9n 0 65n 5 70n 5 70.1n 0 74.9n 0 75n 5 80n 5
80.1n 0 84.9n 0 85n 5 \
90n 5 90.1n 0 94.9n 0 95n 5 100n 5 100.1n 0 104.9n 0 105n 5 110n 5 110.1n 0 114.9n 0
115n 5 120n 5 120.1n 0 124.9n 0 125n 5 130n 5 130.1n 0 134.9n 0 135n 5 140n 5 140.1n
0 144.9n 0 145n 5 150n 5 150.1n 0] \
pwlperiod=160n

vwordA (wordA 0) vsource type=pwl wave=[0n 0 5n 0 5.1n 5 9.9n 5 10n 0 15n 0 15.1n 5
19.9n 5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5
60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordAbar (wordAbar 0) vsource type=pwl wave=[0n 0 5n 0 5.1n 5 9.9n 5 10n 0 15n 0
15.1n 5 19.9n 5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordB (wordB 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 25.1n 5 29.9n 5 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordBbar (wordBbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 25.1n 5 29.9n 5 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n
5 59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordC (wordC 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 45.1n 5 49.9n 5 50n 0 55n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0  75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordCbar (wordCbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 45.1n 5 49.9n 5 50n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordD (wordD 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
65.1n 5 69.9n 5 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordDbar (wordDbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 65.1n 5 69.9n 5 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0  95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0
125n 0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5
160n 0] \
pwlperiod=160n

vwordE (wordE 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n 0
25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 85.1n 5 89.9n 5 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n
```

```
vwordEbar (wordEbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5
20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 85.1n 5 89.9n 5 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordF (wordF 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n 0
25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 105.1n 5 109.9n 5 110n 0 115n 0 115.1n 5
119.9n 5 120n 0 125n 0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordFbar (wordFbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5
20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 105.1n 5 109.9n 5 110n 0 115n 0 115.1n 5
119.9n 5 120n 0 125n 0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordG (wordG 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 125.1n 5 129.9n 5 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordGbar (wordGbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 125.1n 5 129.9n 5 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordH (wordH 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 145.1n 5 149.9n 5 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n
```

```
vwordHbar (wordHbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 145.1n 5 149.9n 5 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vbitAw (bitAw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitAbarw (bitAbarw 0) vsource type=pwl wave=[0n 0 5n 0 5.1n 5 9.9n 5 10n 0 15n 0
20n 0 25n 0 30n 0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0
90n 0 95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitBw (bitBw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 25.1n 5
29.9n 5 30n 0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0
95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitBbarw (bitBbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n

vbitCw (bitCw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitCbarw (bitCbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 45.1n 5 49.9n 5 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0
90n 0 95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitDw (bitDw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 65.1n 5 69.9n 5 70n 0 75n 0 80n 0 85n 0 90n 0
```

```
95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitDbarw (bitDbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n

vbitEw (bitEw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitEbarw (bitEbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 85.1n 5 89.9n 5
90n 0 95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitFw (bitFw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 105.1n 5 109.9n 5 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitFbarw (bitFbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n

vbitGw (bitGw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitGbarw (bitGbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 125.1n 5 129.9n 5 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitHw (bitHw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
```

```
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 145.1n 5 149.9n 5 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitHbarw (bitHbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n




//
ic ABCD=0 ABCDbar=5
memcell tran stop=160.0n
save ABCD ABCDbar EFGH EFGHbar bitA bitAbar bitB bitBbar bitC bitCbar bitD
bitDbar bitE bitEbar bitF bitFbar bitG bitGbar bitH bitHbar

//power calc

opts1 options pwr=total save=all
opts2 options currents=all


// netlist generated from the extracted view of layout.

// Generated for: spectre
// Generated on: Apr 10 20:56:29 2005
// Design library name: ram_lib
// Design cell name: 4ram
// Design view name: extracted


// Library name: ram_lib
// Cell name: 4ram
// View name: extracted
_inst0 (gnd wordDbar) capacitor c=2.46496e-15 m=1
_inst1 (gnd vdd) capacitor c=2.81826e-15 m=1
_inst2 (gnd wordFbar) capacitor c=3.11694e-15 m=1
_inst3 (wordCbar wordDbar) capacitor c=4.617e-15 m=1
_inst4 (wordCbar gnd) capacitor c=6.1047e-15 m=1
_inst5 (wordEbar gnd) capacitor c=8.75028e-15 m=1
_inst6 (wordGbar gnd) capacitor c=3.7041e-15 m=1
_inst7 (wordF gnd) capacitor c=9.02988e-15 m=1
_inst8 (wordD gnd) capacitor c=7.53864e-15 m=1
```

```
_inst9 (wordC wordD) capacitor c=2.08575e-15 m=1
_inst10 (EFGH gnd) capacitor c=3.06834e-15 m=1
_inst11 (gnd bitFbar) capacitor c=2.8064e-15 m=1
_inst12 (gnd vdd) capacitor c=3.99339e-15 m=1
_inst13 (gnd bitHbar) capacitor c=2.49363e-15 m=1
_inst14 (gnd bitAbar) capacitor c=2.70537e-15 m=1
_inst15 (bitEbar gnd) capacitor c=2.96286e-15 m=1
_inst16 (bitH gnd) capacitor c=3.19218e-15 m=1
_inst17 (bitG gnd) capacitor c=3.10179e-15 m=1
_inst18 (bitF gnd) capacitor c=2.91045e-15 m=1
_inst19 (bitE gnd) capacitor c=2.45217e-15 m=1
_inst20 (bitD gnd) capacitor c=2.40257e-15 m=1
_inst21 (bitC gnd) capacitor c=2.26089e-15 m=1
_inst22 (bitGbar gnd) capacitor c=3.31499e-15 m=1
_inst23 (bitBbar gnd) capacitor c=2.67561e-15 m=1
_inst24 (gnd wordDbar) capacitor c=2.85756e-15 m=1
_inst25 (gnd wordFbar) capacitor c=2.69211e-15 m=1
_inst26 (gnd wordHbar) capacitor c=2.76552e-15 m=1
_inst27 (gnd wordAbar) capacitor c=2.08608e-15 m=1
_inst28 (wordCbar wordDbar) capacitor c=2.1432e-15 m=1
_inst29 (wordGbar gnd) capacitor c=2.48382e-15 m=1
_inst30 (wordH gnd) capacitor c=3.15612e-15 m=1
_inst31 (wordG gnd) capacitor c=3.38502e-15 m=1
_inst32 (wordE gnd) capacitor c=3.01419e-15 m=1
_inst33 (wordC gnd) capacitor c=2.48712e-15 m=1
_inst34 (wordC wordD) capacitor c=2.5155e-15 m=1
_inst35 (wordB gnd) capacitor c=2.66397e-15 m=1
_inst36 (wordA gnd) capacitor c=2.96565e-15 m=1
_inst37 (wordBbar gnd) capacitor c=2.69694e-15 m=1
_inst38 (EFGHbar EFGH vdd vdd) ami06P w=1.2e-06 l=6e-07 as=1.08e-12 \
      ad=1.8e-12 ps=1.8e-06 pd=4.2e-06 m=1 region=sat
_inst39 (vdd ABCDbar EFGH vdd) ami06P w=1.2e-06 l=6e-07 as=1.8e-12 \
      ad=1.08e-12 ps=4.2e-06 pd=1.8e-06 m=1 region=sat
_inst40 (ABCDbar ABCD vdd vdd) ami06P w=1.2e-06 l=6e-07 as=1.08e-12 \
      ad=1.8e-12 ps=1.8e-06 pd=4.2e-06 m=1 region=sat
_inst41 (vdd EFGHbar ABCD vdd) ami06P w=1.2e-06 l=6e-07 as=1.8e-12 \
      ad=1.08e-12 ps=4.2e-06 pd=1.8e-06 m=1 region=sat
_inst42 (bitCbar wordCbar ABCDbar gnd) ami06N w=1.5e-06 l=6e-07 \
      as=1.35e-12 ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst43 (ABCDbar wordDbar bitDbar gnd) ami06N w=1.5e-06 l=6e-07 \
      as=2.25e-12 ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst44 (bitBbar wordBbar ABCDbar gnd) ami06N w=1.5e-06 l=6e-07 \
      as=1.35e-12 ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst45 (ABCDbar wordAbar bitAbar gnd) ami06N w=1.5e-06 l=6e-07 \
      as=2.25e-12 ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst46 (EFGHbar EFGH gnd gnd) ami06N w=1.68e-05 l=6e-07 as=1.512e-11 \
      ad=2.52e-11 ps=1.8e-06 pd=1.98e-05 m=1 region=sat
_inst47 (gnd ABCDbar EFGH gnd) ami06N w=1.68e-05 l=6e-07 as=2.52e-11 \
      ad=1.512e-11 ps=1.98e-05 pd=1.8e-06 m=1 region=sat
_inst48 (ABCDbar ABCD gnd gnd) ami06N w=1.68e-05 l=6e-07 as=1.512e-11 \
      ad=2.52e-11 ps=1.8e-06 pd=1.98e-05 m=1 region=sat
_inst49 (gnd EFGHbar ABCD gnd) ami06N w=1.68e-05 l=6e-07 as=2.52e-11 \
      ad=1.512e-11 ps=1.98e-05 pd=1.8e-06 m=1 region=sat
_inst50 (bitH wordH EFGH gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst51 (EFGH wordG bitG gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst52 (bitF wordF EFGH gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst53 (bitHbar wordHbar EFGHbar gnd) ami06N w=1.5e-06 l=6e-07 \
      as=1.35e-12 ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst54 (EFGH wordE bitE gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst55 (EFGHbar wordGbar bitGbar gnd) ami06N w=1.5e-06 l=6e-07 \
      as=2.25e-12 ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst56 (bitD wordD ABCD gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst57 (bitFbar wordFbar EFGHbar gnd) ami06N w=1.5e-06 l=6e-07 \
      as=1.35e-12 ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst58 (ABCD wordC bitC gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst59 (EFGHbar wordEbar bitEbar gnd) ami06N w=1.5e-06 l=6e-07 \
      as=2.25e-12 ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst60 (bitB wordB ABCD gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst61 (ABCD wordA bitA gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
```

C.4 Test bench module and netlist to simulate 8-SRAM model of 8-port Memory cell operating for a read access time of 5nsec

```
//
//
// simulate load cap on bit lines
clbitA (bitA 0) capacitor c=170f
clbitAbar (bitAbar 0) capacitor c=170f
clbitB (bitB 0) capacitor c=170f
clbitBbar (bitBbar 0) capacitor c=170f
clbitC (bitC 0) capacitor c=170f
clbitCbar (bitCbar 0) capacitor c=170f
clbitD (bitD 0) capacitor c=170f
clbitDbar (bitDbar 0) capacitor c=170f
clbitE (bitE 0) capacitor c=170f
clbitEbar (bitEbar 0) capacitor c=170f
clbitF (bitF 0) capacitor c=170f
clbitFbar (bitFbar 0) capacitor c=170f
clbitG (bitG 0) capacitor c=170f
clbitGbar (bitGbar 0) capacitor c=170f
clbitH (bitH 0) capacitor c=170f
clbitHbar (bitHbar 0) capacitor c=170f

clwordA (wordA 0) capacitor  c=125f
clwordAbar (wordAbar 0) capacitor  c=125f
clwordB (wordB 0) capacitor c=125f
clwordBbar (wordBbar 0) capacitor c=125f
clwordC (wordC 0) capacitor c=125f
clwordCbar (wordCbar 0) capacitor c=125f
clwordD (wordD 0) capacitor c=125f
clwordDbar (wordDbar 0) capacitor c=125f
clwordE (wordE 0) capacitor c=125f
clwordEbar (wordEbar 0) capacitor c=125f
clwordF (wordF 0) capacitor c=125f
clwordFbar (wordFbar 0) capacitor c=125f
clwordG (wordG 0) capacitor c=125f
clwordGbar (wordGbar 0) capacitor c=125f
clwordH (wordH 0) capacitor c=125f
clwordHbar (wordHbar 0) capacitor c=125f

// add precharge transistors
```

```
mprebitA (bitA prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitAbar (bitAbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitB (bitB prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitBbar (bitBbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitC (bitC prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitCbar (bitCbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitD (bitD prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitDbar (bitDbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitE (bitE prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitEbar (bitEbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitF (bitF prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitFbar (bitFbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitG (bitG prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitGbar (bitGbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06
mprebitH (bitH prebar vdd vdd) ami06P  w=120e-06 l=0.6e-06
mprebitHbar (bitHbar prebar vdd vdd) ami06P w=120e-06 l=0.6e-06

// add write drivers
mwbitA (bitA bitAw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitAbar (bitAbar bitAbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitB (bitB bitBw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitBbar (bitBbar bitBbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitC (bitC bitCw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitCbar (bitCbar bitCbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitD (bitD bitDw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitDbar (bitDbar bitDbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitE (bitE bitEw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitEbar (bitEbar bitEbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitF (bitF bitFw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitFbar (bitFbar bitFbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitG (bitG bitGw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitGbar (bitGbar bitGbarw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitH (bitH bitHw 0 0) ami06N w=120e-06 l=0.6e-06
mwbitHbar (bitHbar bitHbarw 0 0) ami06N w=120e-06 l=0.6e-06

//power supplies
VPWR(vdd 0) vsource dc=5.0
VGND(gnd 0) vsource dc=0.0



// write 1 on A, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on B, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
```

```
// write 1 on C, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on D, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 1 on E, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on F, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 1 on G, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on H, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar

vpre (prebar 0) vsource type=pwl wave=[0n 0 4.9n 0 5n 5 10n 5 10.1n 0 14.9n 0 15n 5
20n 5 20.1n 0 24.9n 0 25n 5 30n 5 30.1n 0 34.9n 0 35n 5 40n 5 40.1n 0 44.9n 0 45n 5 50n
5 50.1n 0 54.9n 0 55n 5 60n 5 60.1n 0 64.9n 0 65n 5 70n 5 70.1n 0 74.9n 0 75n 5 80n 5
80.1n 0 84.9n 0 85n 5 \
90n 5 90.1n 0 94.9n 0 95n 5 100n 5 100.1n 0 104.9n 0 105n 5 110n 5 110.1n 0 114.9n 0
115n 5 120n 5 120.1n 0 124.9n 0 125n 5 130n 5 130.1n 0 134.9n 0 135n 5 140n 5 140.1n
0 144.9n 0 145n 5 150n 5 150.1n 0] \
pwlperiod=160n

vwordA (wordA 0) vsource type=pwl wave=[0n 0 5n 0 5.1n 5 9.9n 5 10n 0 15n 0 15.1n 5
19.9n 5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5
60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordAbar (wordAbar 0) vsource type=pwl wave=[0n 0 5n 0 5.1n 5 9.9n 5 10n 0 15n 0
15.1n 5 19.9n 5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordB (wordB 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 25.1n 5 29.9n 5 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n
```

```
vwordBbar (wordBbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 25.1n 5 29.9n 5 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n
5 59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordC (wordC 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 45.1n 5 49.9n 5 50n 0 55n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0  75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordCbar (wordCbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 45.1n 5 49.9n 5 50n 0 55.1n 5
59.9n 5 60n 0 65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordD (wordD 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
65.1n 5 69.9n 5 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordDbar (wordDbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 65.1n 5 69.9n 5 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0  95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0
125n 0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5
160n 0] \
pwlperiod=160n

vwordE (wordE 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n 0
25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 85.1n 5 89.9n 5 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0 155.1n 5 159.9n 5 160n
0] \
pwlperiod=160n

vwordEbar (wordEbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5
20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 85.1n 5 89.9n 5 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordF (wordF 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n 0
25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 105.1n 5 109.9n 5 110n 0 115n 0 115.1n 5
119.9n 5 120n 0 125n 0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordFbar (wordFbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5
20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 105.1n 5 109.9n 5 110n 0 115n 0 115.1n 5
119.9n 5 120n 0 125n 0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordG (wordG 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 125.1n 5 129.9n 5 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordGbar (wordGbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 125.1n 5 129.9n 5 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordH (wordH 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n 5 20n
0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0 65n 0
70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 145.1n 5 149.9n 5 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vwordHbar (wordHbar 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 15.1n 5 19.9n
5 20n 0 25n 0 30n 0 35n 0 35.1n 5 39.9n 5 40n 0 45n 0 50n 0 55n 0 55.1n 5 59.9n 5 60n 0
65n 0 70n 0 75n 0 75.1n 5 79.9n 5 80n 0 85n 0 \
90n 0 95n 0 95.1n 5 99.9n 5 100n 0 105n 0 110n 0 115n 0 115.1n 5 119.9n 5 120n 0 125n
0 130n 0 135n 0 135.1n 5 139.9n 5 140n 0 145n 0 145.1n 5 149.9n 5 150n 0 155n 0
155.1n 5 159.9n 5 160n 0] \
pwlperiod=160n

vbitAw (bitAw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitAbarw (bitAbarw 0) vsource type=pwl wave=[0n 0 5n 0 5.1n 5 9.9n 5 10n 0 15n 0
20n 0 25n 0 30n 0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0
90n 0 95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitBw (bitBw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 25.1n 5
29.9n 5 30n 0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0
95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitBbarw (bitBbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n

vbitCw (bitCw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitCbarw (bitCbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 45.1n 5 49.9n 5 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0
90n 0 95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n
```

```
vbitDw (bitDw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 65.1n 5 69.9n 5 70n 0 75n 0 80n 0 85n 0 90n 0
95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitDbarw (bitDbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n

vbitEw (bitEw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitEbarw (bitEbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 85.1n 5 89.9n 5
90n 0 95n 0 100n 0 105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitFw (bitFw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 105.1n 5 109.9n 5 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitFbarw (bitFbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n

vbitGw (bitGw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n 0] \
pwlperiod=160n

vbitGbarw (bitGbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 125.1n 5 129.9n 5 130n 0 135n 0 140n 0 145n 0
150n 0 155n 0 160n 0] \
pwlperiod=160n
```

```
vbitHw (bitHw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n 0 35n
0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0 105n
0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 145.1n 5 149.9n 5 150n 0
155n 0 160n 0] \
pwlperiod=160n

vbitHbarw (bitHbarw 0) vsource type=pwl wave=[0n 0 5n 0 10n 0 15n 0 20n 0 25n 0 30n
0 35n 0 40n 0 45n 0 50n 0 55n 0 60n 0 65n 0 70n 0 75n 0 80n 0 85n 0 90n 0 95n 0 100n 0
105n 0 110n 0 115n 0 120n 0 125n 0 130n 0 135n 0 140n 0 145n 0 150n 0 155n 0 160n
0] \
pwlperiod=160n




//
ic AB=0 ABbar=5
memcell tran stop=160.0n
save AB ABbar CD CDbar EF EFbar GH GHbar bitA bitAbar bitB bitBbar bitC bitCbar
bitD bitDbar bitE bitEbar bitF bitFbar bitG bitGbar bitH bitHbar

//power calc

opts1 options pwr=total save=all
opts2 options currents=all


// netlist generated from the extracted view of the layout.

// Generated for: spectre
// Generated on: Apr 10 20:57:25 2005
// Design library name: ram_lib
// Design cell name: 8ram
// Design view name: extracted



// Library name: ram_lib
// Cell name: 8ram
// View name: extracted
_inst0 (GH vdd) capacitor c=2.04647e-15 m=1
_inst1 (gnd wordAbar) capacitor c=2.98038e-15 m=1
_inst2 (wordCbar gnd) capacitor c=8.21742e-15 m=1
_inst3 (wordEbar gnd) capacitor c=8.41476e-15 m=1
_inst4 (wordF gnd) capacitor c=7.272e-15 m=1
```

```
_inst5 (wordD gnd) capacitor c=8.04657e-15 m=1
_inst6 (GHbar gnd) capacitor c=6.8415e-15 m=1
_inst7 (gnd bitDbar) capacitor c=2.93499e-15 m=1
_inst8 (gnd bitFbar) capacitor c=2.1509e-15 m=1
_inst9 (gnd vdd) capacitor c=4.40607e-15 m=1
_inst10 (gnd bitHbar) capacitor c=2.14788e-15 m=1
_inst11 (gnd bitAbar) capacitor c=2.52612e-15 m=1
_inst12 (bitCbar gnd) capacitor c=2.2269e-15 m=1
_inst13 (bitG gnd) capacitor c=2.1554e-15 m=1
_inst14 (bitF gnd) capacitor c=2.50635e-15 m=1
_inst15 (bitE gnd) capacitor c=2.20953e-15 m=1
_inst16 (bitD gnd) capacitor c=2.23806e-15 m=1
_inst17 (bitB gnd) capacitor c=2.65565e-15 m=1
_inst18 (bitGbar gnd) capacitor c=2.79626e-15 m=1
_inst19 (bitBbar gnd) capacitor c=2.95422e-15 m=1
_inst20 (GHbar bitH) capacitor c=2.69376e-15 m=1
_inst21 (gnd wordDbar) capacitor c=2.08254e-15 m=1
_inst22 (gnd wordFbar) capacitor c=3.48546e-15 m=1
_inst23 (gnd wordHbar) capacitor c=2.45244e-15 m=1
_inst24 (gnd wordAbar) capacitor c=2.36952e-15 m=1
_inst25 (wordGbar gnd) capacitor c=2.3628e-15 m=1
_inst26 (wordH gnd) capacitor c=2.5944e-15 m=1
_inst27 (wordG gnd) capacitor c=2.12706e-15 m=1
_inst28 (wordE gnd) capacitor c=3.10602e-15 m=1
_inst29 (wordC gnd) capacitor c=2.73453e-15 m=1
_inst30 (wordB gnd) capacitor c=2.03829e-15 m=1
_inst31 (wordB wordF) capacitor c=2.22048e-15 m=1
_inst32 (wordA vdd) capacitor c=2.48985e-15 m=1
_inst33 (GHbar GH vdd vdd) ami06P w=1.2e-06 l=6e-07 as=1.08e-12 ad=1.8e-12 \
      ps=1.8e-06 pd=4.2e-06 m=1 region=sat
_inst34 (vdd EFbar GH vdd) ami06P w=1.2e-06 l=6e-07 as=1.8e-12 ad=1.08e-12 \
      ps=4.2e-06 pd=1.8e-06 m=1 region=sat
_inst35 (EFbar EF vdd vdd) ami06P w=1.2e-06 l=6e-07 as=1.08e-12 ad=1.8e-12 \
      ps=1.8e-06 pd=4.2e-06 m=1 region=sat
_inst36 (vdd CDbar EF vdd) ami06P w=1.2e-06 l=6e-07 as=1.8e-12 ad=1.08e-12 \
      ps=4.2e-06 pd=1.8e-06 m=1 region=sat
_inst37 (CDbar CD vdd vdd) ami06P w=1.2e-06 l=6e-07 as=1.08e-12 ad=1.8e-12 \
      ps=1.8e-06 pd=4.2e-06 m=1 region=sat
_inst38 (vdd ABbar CD vdd) ami06P w=1.2e-06 l=6e-07 as=1.8e-12 ad=1.08e-12 \
      ps=4.2e-06 pd=1.8e-06 m=1 region=sat
_inst39 (ABbar AB vdd vdd) ami06P w=1.2e-06 l=6e-07 as=1.08e-12 ad=1.8e-12 \
      ps=1.8e-06 pd=4.2e-06 m=1 region=sat
_inst40 (vdd GHbar AB vdd) ami06P w=1.2e-06 l=6e-07 as=1.8e-12 ad=1.08e-12 \
      ps=4.2e-06 pd=1.8e-06 m=1 region=sat
_inst41 (bitH wordH GH gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
```

```
_inst42 (GHbar wordGbar bitGbar gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst43 (GH wordG bitG gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst44 (bitFbar wordFbar EFbar gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst45 (EFbar wordEbar bitEbar gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst46 (bitF wordF EF gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst47 (EF wordE bitE gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst48 (bitDbar wordDbar CDbar gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst49 (CDbar wordCbar bitCbar gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst50 (bitD wordD CD gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst51 (CD wordC bitC gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst52 (bitBbar wordBbar ABbar gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst53 (ABbar wordAbar bitAbar gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst54 (bitB wordB AB gnd) ami06N w=1.5e-06 l=6e-07 as=1.35e-12 \
      ad=2.25e-12 ps=1.8e-06 pd=4.5e-06 m=1 region=sat
_inst55 (AB wordA bitA gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=1.35e-12 ps=4.5e-06 pd=1.8e-06 m=1 region=sat
_inst56 (bitHbar wordHbar GHbar gnd) ami06N w=1.5e-06 l=6e-07 as=2.25e-12 \
      ad=2.25e-12 ps=4.5e-06 pd=4.5e-06 m=1 region=sat
_inst57 (GHbar GH gnd gnd) ami06N w=4.2e-06 l=6e-07 as=3.78e-12 ad=6.3e-12 \
      ps=1.8e-06 pd=7.2e-06 m=1 region=sat
_inst58 (gnd EFbar GH gnd) ami06N w=4.2e-06 l=6e-07 as=6.3e-12 ad=3.78e-12 \
      ps=7.2e-06 pd=1.8e-06 m=1 region=sat
_inst59 (EFbar EF gnd gnd) ami06N w=4.2e-06 l=6e-07 as=3.78e-12 ad=6.3e-12 \
      ps=1.8e-06 pd=7.2e-06 m=1 region=sat
_inst60 (gnd CDbar EF gnd) ami06N w=4.2e-06 l=6e-07 as=6.3e-12 ad=3.78e-12 \
      ps=7.2e-06 pd=1.8e-06 m=1 region=sat
_inst61 (CDbar CD gnd gnd) ami06N w=4.2e-06 l=6e-07 as=3.78e-12 ad=6.3e-12 \
      ps=1.8e-06 pd=7.2e-06 m=1 region=sat
_inst62 (gnd ABbar CD gnd) ami06N w=4.2e-06 l=6e-07 as=6.3e-12 ad=3.78e-12 \
      ps=7.2e-06 pd=1.8e-06 m=1 region=sat
_inst63 (ABbar AB gnd gnd) ami06N w=4.2e-06 l=6e-07 as=3.78e-12 ad=6.3e-12 \
      ps=1.8e-06 pd=7.2e-06 m=1 region=sat
_inst64 (gnd GHbar AB gnd) ami06N w=4.2e-06 l=6e-07 as=6.3e-12 ad=3.78e-12 \
      ps=7.2e-06 pd=1.8e-06 m=1 region=sat
```

C.5 Test bench module and netlist to simulate J-SRAM model of 8-port Memory cell operating for a read access time of 1.5nsec

```
//
//
// simulate load cap on bit lines
clbitA (bitA 0) capacitor c=60f //240f
clbitAbar (bitAbar 0) capacitor c=60f //240f
clbitB (bitB 0) capacitor c=60f //240f
clbitBbar (bitBbar 0) capacitor c=60f //240f
clbitC (bitC 0) capacitor c=60f //240f
clbitCbar (bitCbar 0) capacitor c=60f //240f
clbitD (bitD 0) capacitor c=60f //240f
clbitDbar (bitDbar 0) capacitor c=60f //240f
clbitE (bitE 0) capacitor c=60f //240f
clbitEbar (bitEbar 0) capacitor c=60f //240f
clbitF (bitF 0) capacitor c=60f //240f
clbitFbar (bitFbar 0) capacitor c=60f //240f
clbitG (bitG 0) capacitor c=60f //240f
clbitGbar (bitGbar 0) capacitor c=60f //240f
clbitH (bitH 0) capacitor c=60f //240f
clbitHbar (bitHbar 0) capacitor c=60f //240f

clwordA (wordA 0) capacitor  c=50f //198f
clwordAbar (wordAbar 0) capacitor  c=50f //198f
clwordB (wordB 0) capacitor c=50f //198f
clwordBbar (wordBbar 0) capacitor c=50f //198f
clwordC (wordC 0) capacitor c=50f //198f
clwordCbar (wordCbar 0) capacitor c=50f //198f
clwordD (wordD 0) capacitor c=50f //198f
clwordDbar (wordDbar 0) capacitor c=50f //198f
clwordE (wordE 0) capacitor c=50f //198f
clwordEbar (wordEbar 0) capacitor c=50f //198f
clwordF (wordF 0) capacitor c=50f //198f
clwordFbar (wordFbar 0) capacitor c=50f //198f
clwordG (wordG 0) capacitor c=50f //198f
clwordGbar (wordGbar 0) capacitor c=50f //198f
clwordH (wordH 0) capacitor c=50f //198f
clwordHbar (wordHbar 0) capacitor c=50f //198f

// add precharge transistors
mprebitA (bitA prebar vdd vdd) ami06P w=45e-06 l=0.6e-06
```

```
mprebitAbar (bitAbar prebar vdd vdd) ami06P w=45e-06 l=0.6e-06
mprebitB (bitB prebar vdd vdd) ami06P  w=45e-06 l=0.6e-06
mprebitBbar (bitBbar prebar vdd vdd) ami06P w=45e-06 l=0.6e-06
mprebitC (bitC prebar vdd vdd) ami06P  w=45e-06 l=0.6e-06
mprebitCbar (bitCbar prebar vdd vdd) ami06P w=45e-06 l=0.6e-06
mprebitD (bitD prebar vdd vdd) ami06P  w=45e-06 l=0.6e-06
mprebitDbar (bitDbar prebar vdd vdd) ami06P w=45e-06 l=0.6e-06
mprebitE (bitE prebar vdd vdd) ami06P  w=45e-06 l=0.6e-06
mprebitEbar (bitEbar prebar vdd vdd) ami06P w=45e-06 l=0.6e-06
mprebitF (bitF prebar vdd vdd) ami06P  w=45e-06 l=0.6e-06
mprebitFbar (bitFbar prebar vdd vdd) ami06P w=45e-06 l=0.6e-06
mprebitG (bitG prebar vdd vdd) ami06P  w=45e-06 l=0.6e-06
mprebitGbar (bitGbar prebar vdd vdd) ami06P w=45e-06 l=0.6e-06
mprebitH (bitH prebar vdd vdd) ami06P  w=45e-06 l=0.6e-06
mprebitHbar (bitHbar prebar vdd vdd) ami06P w=45e-06 l=0.6e-06

// add write drivers
mwbitA (bitA bitAw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitAbar (bitAbar bitAbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitB (bitB bitBw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitBbar (bitBbar bitBbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitC (bitC bitCw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitCbar (bitCbar bitCbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitD (bitD bitDw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitDbar (bitDbar bitDbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitE (bitE bitEw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitEbar (bitEbar bitEbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitF (bitF bitFw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitFbar (bitFbar bitFbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitG (bitG bitGw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitGbar (bitGbar bitGbarw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitH (bitH bitHw err 0) ami06N w=24.0e-06 l=0.6e-06
mwbitHbar (bitHbar bitHbarw err 0) ami06N w=24.0e-06 l=0.6e-06

//power supplies
VPWR(vdd 0) vsource dc=5.0
VGND(gnd 0) vsource dc=0.0
ERROR(err 0) vsource dc=0.0

//power calc

opts1 options pwr=total save=all
opts2 options currents=all


//
```

```
ic cell=0 cellbar=5
memcell tran stop=30n
save cell cellbar bitA bitAbar bitB bitBbar bitC bitCbar bitD bitDbar bitE bitEbar bitF
bitFbar bitG bitGbar bitH bitHbar


// write 1 on A, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on B, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 1 on C, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on D, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 1 on E, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on F, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 1 on G, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar
// write 0 on H, read from A, Abar, B, Bbar, C ,Cbar,D ,Dbar,E ,Ebar,F ,Fbar,G ,Gbar,H
,Hbar

vpre (prebar 0) vsource type=pwl wave=[0n 0 0.2n 0 0.3n 5 1.2n 5 1.3n 0 2.0n 0 2.1n 5
3n 5 3.1n 0 3.8n 0 3.9n 5 4.8n 5 4.9n 0 5.6n 0 5.7n 5 6.6n 5 6.7n 0 7.4n 0 7.5n 5 8.4n 5
8.5n 0 9.2n 0 9.3n 5 10.2n 5 10.3n 0 11n 0 11.1n 5 12.0n 5 12.1n 0 12.8n 0 12.9n 5 13.8n
5 13.9n 0 14.6n 0 14.7n 5 15.6n 5 15.7n 0 16.4n 0 16.5n 5 17.4n 5 17.5n 0 18.2n 0 18.3n
5 19.2n 5 19.3n 0 20n 0 20.1n 5 21.0n 5 21.1n 0 21.8n 0 21.9n 5 22.8n 5 22.9n 0 23.6n 0
23.7n \
5 24.6n 5 24.7n 0 25.4n 0 25.5n 5 26.4n 5 26.5n 0 27.2n 0 27.3n 5 28.2n 5 28.3n 0 29.0n
0 29.1n 5 30n 5 30.1n 0 ]\
pwlperiod=30.1n

vwordA (wordA 0) vsource type=pwl wave=[0n 0 0.3n 0 0.4n 5 1.1n 5 1.2n 0 2.1n 0 2.2n
5 2.9n 5 3.0n 0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n
5 10.2n 0 11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n 0 16.6n
5 17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n 0 23.7n
0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n 0 30.0n
0]\
pwlperiod=30.1n

vwordAbar (wordAbar 0) vsource type=pwl wave=[0n 0 0.3n 0 0.4n 5 1.1n 5 1.2n 0 2.1n
0 2.2n 5 2.9n 5 3.0n 0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n
5 10.1n 5 10.2n 0 11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n
0 16.6n 5 17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n
```

```
0 23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n
0 30.0n 0]\
pwlperiod=30.1n

vwordB (wordB 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n 5 3.0n
0 3.9n 0 4.0n 5 4.7n 5 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n
5 10.2n 0 11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n 0 16.6n
5 17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n 0 23.7n
0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n 0 30.0n
0]\
pwlperiod=30.1n

vwordBbar (wordBbar 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n
5 3.0n 0 3.9n 0 4.0n 5 4.7n 5 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n
5 10.1n 5 10.2n 0 11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n
0 16.6n 5 17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n
0 23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n
0 30.0n 0]\
pwlperiod=30.1n

vwordC (wordC 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n 5 3.0n
0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 7.6n 5 8.3n 5 8.4n 0 9.3n 0 9.4n 5 10.1n
5 10.2n 0 11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n 0 16.6n
5 17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n 0 23.7n
0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n 0 30.0n
0]\
pwlperiod=30.1n

vwordCbar (wordCbar 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n
5 3.0n 0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 7.6n 5 8.3n 5 8.4n 0 9.3n 0 9.4n
5 10.1n 5 10.2n 0 11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n
0 16.6n 5 17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n
0 23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n
0 30.0n 0]\
pwlperiod=30.1n

vwordD (wordD 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n 5 3.0n
0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n 5 10.2n 0
11.1n 0 11.2n 5 11.9n 5 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n 0
16.6n 5 17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n 0
23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n 0
30.0n 0]\
pwlperiod=30.1n

vwordDbar (wordDbar 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n
5 3.0n 0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n 5
```

```
10.2n 0 11.1n 0 11.2n 5 11.9n 5 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0
16.5n 0 16.6n 5 17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0
22.8n 0 23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0
29.1n 0 30.0n 0]\
pwlperiod=30.1n

vwordE (wordE 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n 5 3.0n
0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n 5 10.2n 0
11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 14.8n 5 15.5n 5 15.6n 0 16.5n 0
16.6n 5 17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n 0
23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n 0
30.0n 0]\
pwlperiod=30.1n

vwordEbar (wordEbar 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n
5 3.0n 0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n 5
10.2n 0 11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 14.8n 5 15.5n 5 15.6n 0
16.5n 0 16.6n 5 17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0
22.8n 0 23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0
29.1n 0 30.0n 0]\
pwlperiod=30.1n

vwordF (wordF 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n 5 3.0n
0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n 5 10.2n 0
11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n 0 16.6n 5 17.3n 5
17.4n 0 18.3n 0 18.4n 5 19.1n 5 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n 0
23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n 0
30.0n 0]\
pwlperiod=30.1n

vwordFbar (wordFbar 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n
5 3.0n 0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n 5
10.2n 0 11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n 0 16.6n 5
17.3n 5 17.4n 0 18.3n 0 18.4n 5 19.1n 5 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0
22.8n 0 23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0
29.1n 0 30.0n 0]\
pwlperiod=30.1n

vwordG (wordG 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n 5 3.0n
0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n 5 10.2n 0
11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n 0 16.6n 5 17.3n 5
17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.0n 5 22.7n 5 22.8n 0
23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n 0
30.0n 0]\
pwlperiod=30.1n
```

```
vwordGbar (wordGbar 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n
5 3.0n 0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n 5
10.2n 0 11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n 0 16.6n 5
17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.0n 5 22.7n 5
22.8n 0 23.7n 0 23.8n 5 24.5n 5 24.6n 0 25.5n 0 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0
29.1n 0 30.0n 0]\
pwlperiod=30.1n

vwordH (wordH 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n 5 3.0n
0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n 5 10.2n 0
11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n 0 16.6n 5 17.3n 5
17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n 0 23.7n 0 23.8n 5
24.5n 5 24.6n 0 25.5n 0 25.6n 5 26.3n 5 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0 29.1n 0
30.0n 0]\
pwlperiod=30.1n

vwordHbar (wordHbar 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 2.2n 5 2.9n
5 3.0n 0 3.9n 0 4.8n 0 5.7n 0 5.8n 5 6.5n 5 6.6n 0 7.5n 0 8.4n 0 9.3n 0 9.4n 5 10.1n 5
10.2n 0 11.1n 0 12.0n 0 12.9n 0 13.0n 5 13.7n 5 13.8n 0 14.7n 0 15.6n 0 16.5n 0 16.6n 5
17.3n 5 17.4n 0 18.3n 0 19.2n 0 20.1n 0 20.2n 5 20.9n 5 21.0n 0 21.9n 0 22.8n 0 23.7n 0
23.8n 5 24.5n 5 24.6n 0 25.5n 0 25.6n 5 26.3n 5 26.4n 0 27.3n 0 27.4n 5 28.1n 5 28.2n 0
29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitAw (bitAw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0 4.8n
0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0 15.6n
0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0 24.6n 0 25.5n 0
26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n



vbitAbarw (bitAbarw 0) vsource type=pwl wave=[0n 0 0.3n 0 0.4n 5 1.1n 5 1.2n 0 2.1n 0
3.0n 0 3.9n 0 4.8n 0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0
13.8n 0 14.7n 0 15.6n 0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0
23.7n 0 24.6n 0 25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitBw (bitBw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0 4.0n 5
4.7n 5 4.8n 0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0
14.7n 0 15.6n 0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0
24.6n 0 25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitBbarw (bitBbarw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0
4.8n 0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0
15.6n 0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0 24.6n 0
25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitCw (bitCw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0 4.8n 0
5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0 15.6n 0
16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0 24.6n 0 25.5n 0
26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitCbarw (bitCbarw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0
4.8n 0 5.7n 0 6.6n 0 7.5n 0 7.6n 5 8.3n 5 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0
13.8n 0 14.7n 0 15.6n 0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0
23.7n 0 24.6n 0 25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitDw (bitDw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0 4.8n
0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 11.2n 5 11.9n 5 12n 0 12.9n 0 13.8n
0 14.7n 0 15.6n 0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0
24.6n 0 25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitDbarw (bitDbarw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0
4.8n 0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0
15.6n 0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0 24.6n 0
25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitEw (bitEw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0 4.8n 0
5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0 15.6n 0
16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0 24.6n 0 25.5n 0
26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitEbarw (bitEbarw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0
4.8n 0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0
14.8n 5 15.5n 5 15.6n 0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0
23.7n 0 24.6n 0 25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitFw (bitFw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0 4.8n 0
5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0 15.6n 0
16.5n 0 17.4n 0 18.3n 0 18.4n 5 19.1n 5 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0
24.6n 0 25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n
```

```
vbitFbarw (bitFbarw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0
4.8n 0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0
15.6n 0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0 24.6n 0
25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitGw (bitGw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0 4.8n
0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0 15.6n
0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0 24.6n 0 25.5n 0
26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitGbarw (bitGbarw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0
4.8n 0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0
15.6n 0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22n 5 22.7n 5 22.8n 0
23.7n 0 24.6n 0 25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitHw (bitHw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0 4.8n
0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0 15.6n
0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0 24.6n 0 25.5n 0
25.6n 5 26.3n 5 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n

vbitHbarw (bitHbarw 0) vsource type=pwl wave=[0n 0 0.3n 0 1.2n 0 2.1n 0 3.0n 0 3.9n 0
4.8n 0 5.7n 0 6.6n 0 7.5n 0 8.4n 0 9.3n 0 10.2n 0 11.1n 0 12n 0 12.9n 0 13.8n 0 14.7n 0
15.6n 0 16.5n 0 17.4n 0 18.3n 0 19.2n 0 20.1n 0 21n 0 21.9n 0 22.8n 0 23.7n 0 24.6n 0
25.5n 0 26.4n 0 27.3n 0 28.2n 0 29.1n 0 30.0n 0]\
pwlperiod=30.1n



//netlist generated from the extracted view of layout.

// Generated for: spectre
// Generated on: Jan 27 20:03:18 2005
// Design library name: ram_lib
// Design cell name: jram
// Design view name: extracted


include "/app1/cadence/ncsu/local/models/spectre/standalone/ami06N.m"
include "/app1/cadence/ncsu/local/models/spectre/standalone/ami06P.m"

// Library name: ram_lib
// Cell name: jram
// View name: extracted
_inst0 (gnd cellbar) capacitor c=4.8042e-15 m=1
_inst1 (wordCbar gnd) capacitor c=8.4192e-15 m=1
_inst2 (wordEbar gnd) capacitor c=8.55456e-15 m=1
_inst3 (wordF gnd) capacitor c=6.74634e-15 m=1
_inst4 (cell vdd) capacitor c=2.19792e-15 m=1
_inst5 (cell gnd) capacitor c=3.6765e-15 m=1
_inst6 (wordD gnd) capacitor c=7.8033e-15 m=1
_inst7 (wordBbar gnd) capacitor c=2.3685e-15 m=1
_inst8 (gnd bitFbar) capacitor c=3.47979e-15 m=1
_inst9 (gnd vdd) capacitor c=9.42597e-15 m=1
_inst10 (gnd bitAbar) capacitor c=3.10683e-15 m=1
_inst11 (bitCbar gnd) capacitor c=2.75025e-15 m=1
_inst12 (bitH gnd) capacitor c=3.24396e-15 m=1
_inst13 (bitG gnd) capacitor c=3.48066e-15 m=1
_inst14 (bitF gnd) capacitor c=3.63557e-15 m=1
_inst15 (bitE gnd) capacitor c=2.45721e-15 m=1
_inst16 (bitD gnd) capacitor c=2.65986e-15 m=1
_inst17 (bitC gnd) capacitor c=3.22839e-15 m=1
_inst18 (bitB gnd) capacitor c=3.42351e-15 m=1
_inst19 (bitA gnd) capacitor c=2.33475e-15 m=1
_inst20 (bitGbar gnd) capacitor c=2.23209e-15 m=1
_inst21 (bitBbar gnd) capacitor c=3.05061e-15 m=1
_inst22 (wordAbar vdd) capacitor c=2.0793e-15 m=1
_inst23 (gnd wordDbar) capacitor c=3.23118e-15 m=1
_inst24 (gnd vdd) capacitor c=2.72292e-15 m=1
_inst25 (gnd wordFbar) capacitor c=3.38826e-15 m=1
_inst26 (gnd wordHbar) capacitor c=2.5707e-15 m=1
_inst27 (gnd wordAbar) capacitor c=2.3328e-15 m=1
_inst28 (wordGbar gnd) capacitor c=2.5107e-15 m=1
_inst29 (wordH gnd) capacitor c=2.33832e-15 m=1
_inst30 (wordC gnd) capacitor c=3.14736e-15 m=1
_inst31 (wordB gnd) capacitor c=2.71248e-15 m=1
_inst32 (wordA gnd) capacitor c=2.06847e-15 m=1
_inst33 (wordBbar wordEbar) capacitor c=2.07792e-15 m=1
_inst34 (vdd cell A vdd) ami06P w=4.2e-06 l=6e-07 as=6.3e-12 ad=6.3e-12 \
      ps=7.2e-06 pd=7.2e-06 m=1 region=sat
_inst35 (C cellbar vdd vdd) ami06P w=4.2e-06 l=6e-07 as=6.3e-12 ad=6.3e-12 \
      ps=7.2e-06 pd=7.2e-06 m=1 region=sat
_inst36 (cellbar B vdd vdd) ami06P w=1.2e-06 l=6e-07 as=1.08e-12 \
      ad=1.8e-12 ps=1.8e-06 pd=4.2e-06 m=1 region=sat
_inst37 (vdd A B vdd) ami06P w=1.2e-06 l=6e-07 as=1.8e-12 ad=1.08e-12 \
      ps=4.2e-06 pd=1.8e-06 m=1 region=sat
_inst38 (D C vdd vdd) ami06P w=1.2e-06 l=6e-07 as=1.08e-12 ad=1.8e-12 \
      ps=1.8e-06 pd=4.2e-06 m=1 region=sat
_inst39 (vdd D cell vdd) ami06P w=1.2e-06 l=6e-07 as=1.8e-12 ad=1.08e-12 \
        ps=4.2e-06 pd=1.8e-06 m=1 region=sat
_inst40 (gnd cell A gnd) ami06N w=1.2e-06 l=6e-07 as=1.8e-12 ad=1.8e-12 \
        ps=4.2e-06 pd=4.2e-06 m=1 region=sat
_inst41 (C cellbar gnd gnd) ami06N w=1.2e-06 l=6e-07 as=1.8e-12 ad=1.8e-12 \
        ps=4.2e-06 pd=4.2e-06 m=1 region=sat
_inst42 (cellbar B gnd gnd) ami06N w=4.2e-06 l=6e-07 as=3.78e-12 \
        ad=6.3e-12 ps=1.8e-06 pd=7.2e-06 m=1 region=sat
_inst43 (gnd A B gnd) ami06N w=4.2e-06 l=6e-07 as=6.3e-12 ad=3.78e-12 \
        ps=7.2e-06 pd=1.8e-06 m=1 region=sat
_inst44 (D C gnd gnd) ami06N w=4.2e-06 l=6e-07 as=3.78e-12 ad=6.3e-12 \
        ps=1.8e-06 pd=7.2e-06 m=1 region=sat
_inst45 (gnd D cell gnd) ami06N w=4.2e-06 l=6e-07 as=6.3e-12 ad=3.78e-12 \
        ps=7.2e-06 pd=1.8e-06 m=1 region=sat
_inst46 (bitG wordG cell gnd) ami06N w=2.1e-06 l=6e-07 as=1.89e-12 \
        ad=3.15e-12 ps=1.8e-06 pd=5.1e-06 m=1 region=sat
_inst47 (cell wordH bitH gnd) ami06N w=2.1e-06 l=6e-07 as=3.15e-12 \
        ad=1.89e-12 ps=5.1e-06 pd=1.8e-06 m=1 region=sat
_inst48 (bitBbar wordBbar cellbar gnd) ami06N w=2.1e-06 l=6e-07 \
        as=1.89e-12 ad=3.15e-12 ps=1.8e-06 pd=5.1e-06 m=1 region=sat
_inst49 (cellbar wordAbar bitAbar gnd) ami06N w=2.1e-06 l=6e-07 \
        as=3.15e-12 ad=1.89e-12 ps=5.1e-06 pd=1.8e-06 m=1 region=sat
_inst50 (bitF wordF cell gnd) ami06N w=2.1e-06 l=6e-07 as=3.15e-12 \
        ad=3.15e-12 ps=5.1e-06 pd=5.1e-06 m=1 region=sat
_inst51 (cellbar wordHbar bitHbar gnd) ami06N w=2.1e-06 l=6e-07 \
        as=3.15e-12 ad=3.15e-12 ps=5.1e-06 pd=5.1e-06 m=1 region=sat
_inst52 (bitE wordE cell gnd) ami06N w=2.1e-06 l=6e-07 as=1.89e-12 \
        ad=3.15e-12 ps=1.8e-06 pd=5.1e-06 m=1 region=sat
_inst53 (cell wordD bitD gnd) ami06N w=2.1e-06 l=6e-07 as=3.15e-12 \
        ad=1.89e-12 ps=5.1e-06 pd=1.8e-06 m=1 region=sat
_inst54 (bitGbar wordGbar cellbar gnd) ami06N w=2.1e-06 l=6e-07 \
        as=3.15e-12 ad=3.15e-12 ps=5.1e-06 pd=5.1e-06 m=1 region=sat
_inst55 (bitC wordC cell gnd) ami06N w=2.1e-06 l=6e-07 as=1.89e-12 \
        ad=3.15e-12 ps=1.8e-06 pd=5.1e-06 m=1 region=sat
_inst56 (cell wordB bitB gnd) ami06N w=2.1e-06 l=6e-07 as=3.15e-12 \
        ad=1.89e-12 ps=5.1e-06 pd=1.8e-06 m=1 region=sat
_inst57 (bitFbar wordFbar cellbar gnd) ami06N w=2.1e-06 l=6e-07 \
        as=1.89e-12 ad=3.15e-12 ps=1.8e-06 pd=5.1e-06 m=1 region=sat
_inst58 (bitA wordA cell gnd) ami06N w=2.1e-06 l=6e-07 as=3.15e-12 \
        ad=3.15e-12 ps=5.1e-06 pd=5.1e-06 m=1 region=sat
_inst59 (cellbar wordEbar bitEbar gnd) ami06N w=2.1e-06 l=6e-07 \
        as=3.15e-12 ad=1.89e-12 ps=5.1e-06 pd=1.8e-06 m=1 region=sat
_inst60 (bitDbar wordDbar cellbar gnd) ami06N w=2.1e-06 l=6e-07 \
        as=1.89e-12 ad=3.15e-12 ps=1.8e-06 pd=5.1e-06 m=1 region=sat
_inst61 (cellbar wordCbar bitCbar gnd) ami06N w=2.1e-06 l=6e-07 \
        as=3.15e-12 ad=1.89e-12 ps=5.1e-06 pd=1.8e-06 m=1 region=sat
```

Appendix D

D.1. Mathematica code for calculation of inverter threshold voltage

```
εox=3.9*8.8542*10^-12;
tox=1.41*10^-10;
Cox=εox/tox

νsatp= 1.3078*10^5;
μp=202.4540*10^-4;
Esatp=2*(νsatp/μp)

RDSWp=2.7468*10^3;
PRWGp=2.3486*10^-3;
Vdd=5.0;
Vtp=Vtho=-0.9179;
Wp=1.2*10^-6;
Rdsp=(RDSWp*(1+PRWGp(Vdd-Vtp)))/(Wp*10^6)

Abulk=1.4;
a1=0;
a2=1;
Lp=0.6*10^-6;
ap=((Abulk^2*Rdsp*Cox*Wp*νsatp))
bp=-((Vdd-Vinv-Vtp)+((Abulk*Esatp*Lp)+(3*Abulk*Rdsp*Cox*Wp*νsatp*(Vdd-
Vinv-Vtp))))
cp=(Esatp*Lp*(Vdd-Vinv-Vtp)+(2*Rdsp*Cox*Wp*νsatp*(Vdd-Vinv-Vtp)^2))
λ=a1(Vdd-Vinv-Vtp)+a2
Vdssatp1=(-bp+(bp^2-4*ap*cp)^0.5)/(2*ap)
Vdssatp2=(-bp-(bp^2-4*ap*cp)^0.5)/(2*ap)

νsatn= 1.3009*10^5;
μn=533.6953*10^-4;
Esatn=2*(νsatn/μn)

RDSWn=2.2683*10^3;
PRWGn=2.0435*10^-3;
Vdd=5.0;
Vtn=Vtho=0.7086;
Wn=9.6*10^-6;
```

```
Rdsn=(RDSWn*(1+PRWGn(-Vtn)))/(Wn*10^6)

Abulk=1.4;
a1=0;
a2=1;
Ln=0.6*10^-6;
an=((Abulk^2*Rdsn*Cox*Wn*νsatn)+((1/λ-1)*Abulk))
bn=-(((Vinv-Vtn)*(2/λ-1))+((Abulk*Esatn*Ln)+(3*Abulk*Rdsn*Cox*Wn*νsatn*(Vinv-
Vtn))))
cn=(Esatn*Ln*(Vinv-Vtn)+(2*Rdsn*Cox*Wn*νsatn*(Vinv-Vtn)^2))
λ=a1(Vinv-Vtn)+a2
Vdssatn1=(-bn+(bn^2-4*an*cn)^0.5)/(2*an)
Vdssatn2=(-bn-(bn^2-4*an*cn)^0.5)/(2*an)

W=Wn/Wp;
Vap=20;
Van=50;
Solve[W*((Vinv-Vtn)-Abulk*Vdssatn1)*(1+((Vinv-Vdssatn1)/Van))  (((Vdd-Vinv-Vtp)-
(Abulk*Vdssatp1))*(1+((Vdd-Vinv-Vdssatp1)/Vap))),Vinv]
```

D.2. Mathematica code for calculation of memory cell voltage

```
Abulk=1.4;
Cox=2.45*10^-7;
Vsat=1.3*10^7;

Esatn=4.07*10^4;
Vthn=0.7086;
Lpass=0.6*10^-4;
Ln=0.6*10^-4;

Vdd=5;
Van=50;
m=4;
vgstpass=(Vdd-Vcell-Vthn)

Wpass=1.5*10^-4;
Wn=8.3*10^-4;

Rdsn=2.2*10^3/(10^4*Wn)
Rdspass=2.2*10^3/(10^4*Wpass)

a=Abulk^2*Rdspass*Cox*Wpass*Vsat
b=-(vgstpass+(Abulk*Esatn*Lpass)+(3*Abulk*Rdspass*Cox*Wpass*Vsat*vgstpass))
c=(Esatn*Lpass*vgstpass)+(2*Rdspass*Cox*Wpass*Vsat*vgstpass^2)
Vdssatpass=(-b-(b^2-4*a*c)^0.5)/(2*a)
Idn0=Vsat*Cox*Wn*((((2*Vdd)-(2*Vthn))*Vcell)-
(Abulk*Vcell^2))/((Esatn*Ln)+Vcell)

Solve[(Vcell*Idn0)/(Vcell+(Idn0*Rdsn))= =m*Wpass*Cox*Vsat*(Vdd-Vcell-Vthn-
Abulk*Vdssatpass)*(Van+Vdd-Vcell-Vdssatpass)/Van,Vcell]
```

VITA

Raghu Koppanathi

Candidate for the Degree of

Master of Science

Thesis: A 1.5ns, 5 V, 8-Port SRAM Memory cell, Capable of 8 writes or 16 reads simultaneously.

Major Field: Electrical Engineering.

Biographical:

Personal Data: Born in Hyderabad, Andhra Pradesh, INDIA, as son to K.R.Vara Prasad and K. Adi lakshmi on 15th September, 1981.

Education:

Received Higher Secondary Education in H.C.L. D.A.V Junior College, Hyderabad, INDIA in May 1998; Graduated from Osmania University, Hyderabad, INDIA in May 2002, earning a Bachelor's Degree in Electrical and Electronics Engineering; completed the requirements for the Master of Sciences degree with a major in Electrical Engineering at Oklahoma State University, May 2005.

Experience:

Graduate Teaching Assistant at School of Electrical and Computer Engineering, fall 2003 – Present.

Abstract

Name: Raghu Koppanathi **Date of Degree:** May, 2005

Institution: Oklahoma State University **Location:** Stillwater, Oklahoma

Title of Study: 8-Port SRAM Memory cell with 8 writes or 16 reads simultaneously.

Pages in study: 133 Candidate for the Degree of Masters of Sciences

Major Field: Electrical Engineering.

Scope and Method of Study: The main purpose of this thesis is to understand the working principle and design of an 8-port SRAM Memory cell with an emphasis to attain maximum possible Noise margin. We have proposed four designs, namely 2-SRAM, 4-SRAM, 8-SRAM, and J-SRAM capable of 8 writes or 16 reads simultaneously. Worst case scenarios in reading from and writing into a memory cell are considered and analyzed. Theoretical analysis is done using a simplified BSIM3v3 model, equations derived for inverter threshold voltage, node and complementary nodes on the memory cell are used to calculate static noise margin of the cell. Layout designs of the models are made in Cadence Virtuoso Layout Editor, and simulations are run for measuring dynamic noise margin in the cell using Cadence Spectre simulator. Designs are made with transistor widths giving maximum noise margin that is possible in with a limited area of 1600 μm^2. The simulated results and the theoretical results are then compared.

Results and Conclusions: The performances of the four designs are compared with respect to noise margin. With read access time (5ns) and silicon area the same for all the designs, noise margin is measured, and it is found that J-SRAM has the maximum noise margin of 0.8251 V for a 5 V supply in an AMI 0.6µ technology. This design is also tested for read access time of 1.5ns, and the noise margin obtained is above 0.5V. If the limitation on silicon area is neglected, then a noise margin of 2V is obtained.

Dr.Louis.G.Johnson

Advisor's Approval