ARCHITECTURAL MACRO-MODELING OF PROCESSOR MEMORY COMPONENTS

by

Ghazanfar Ali Khan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
1995

Doctoral Committee:

Professor Trevor Mudge, Chairman
Professor Ronald Lomax
Associate Professor Pinaki Mazumder
Assisstant Professor Stuart Sechrest

To my late mother who always wanted me to strive for the best. Also to my family for putting up with the long break, and not reminding me of my responsibilities.

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Mudge for putting up with my antics, I know it was not easy to work with me. And for his guidance and timely advice which kept me on track and which helped me in more ways than I care to think.

Many thanks to the members of my dissertation committee, Pinaki Mazumder, Ronald Lomax and Stuart Sechrest for spending time with me and my ideas. Special thanks to Karem Sakallah for his many ideas, that helped to expand my horizons. Thanks to all my roommates starting in 3003 and ending in 2001 for keeping me company and helping me; especially Mike Upton who helped a lot in the beginning of my work, Ayman Kayssi for all his help with modeling concepts, Mike Riepe for helping me with various CAD tools, and Tim Stanley for all his useful suggestions.

Also thanks are due to the Ministry of Science and Technology, Government of Pakistan, who encouraged me to pursue a Ph.D. when I was quite content with my Bachelor's degree, and who financed part of my education here in USA. Special thanks to Mr. Arshad Adil Khawaja of MOST whose kind words of encouragement kept me going in moments of despair.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER I
INTRODUCTION

1.1 Modeling of Processor Components

The continued rapid advances in semiconductor technology translate directly into dramatic increases in computer system performance. Thus design time has become critical – a few months delay can mean that a system's performance falls behind that of its competitors. Indeed it has been remarked that a performance improvement is only worth implementing if it does not significantly increase the design cycle time of a processor [Henne91]. Techniques to quickly explore design alternatives can play an important role in shortening the time to market.

The state-of-the-art in electronic design automation (EDA) tools provides for design entry in a hardware description language (HDL) followed by some form of logic synthesis and finally by further automatic steps to layout. This process is still imperfect and there is much room for improved tools, however the broad outlines of an automatic design process that takes the register transfer description of a target design to chip layout are well defined. In contrast there are few tools in the early design phase to help determine the implementation of the instruction set architecture (ISA). Performance tools can certainly help in this phase by identifying machine organizations that give the best cycles per instruction (CPI). However, determination of the speed, area, and power consumption of design alternatives currently requires that a fairly detailed design of

processor be carried out. It is the lack of early design phase tools that is becoming the bottleneck slowing the design process.

This thesis will demonstrate a macro-modeling technique that is a step in the direction of speeding up this process of early design by reducing the need for a detailed design. Specifically, this thesis develops a method to create models for speed, area, and power for processor memory components such as register files and content addressable memories (CAMs) that can be evaluated quickly. These models let the architect quickly select designs that optimize performance. Before we discuss further the modeling techniques, we briefly review the major trends in processor design, and show the importance of register files to contemporary computer organizations.

1.2 Processor Performance Improvements

The 1980's saw major improvements in processor performance. The rate of improvement in performance was greater than 50% per year [Henne91]. This improvement resulted from a combination of better ISA design, better compilers, better design tools, and improvements in processor implementation technology.

The trend in ISA design has followed the reduced instruction set computer (RISC) philosophy [Henne91, Stone91]. The RISC design philosophy aims to simplify the control circuitry by reducing the overall number of instructions that a processor can handle. A typical RISC machine uses fixed-length symmetric instruction formats. There are exceptions to this rule though, for example the IBM POWER architecture [Bakog90b] has some instructions that can add and multiply operands in the same instruction, and therefore cannot be called simple. This may explain IBM's definition of RISC which is "reduced instruction set cycles."

The simplification of control in RISC machines allows faster instruction decoding, so the clock cycle may be reduced. Further, the instructions are simple enough to pipeline, so an instruction can be issued on every clock cycle. This is achieved by

organizing the ISA around a register file that can be accessed in a single cycle. The register file serves as the source and destination for all data manipulation instructions, and the source and destination for all the data transfers with memory. Most processors introduced in the mid to late 1980's like the R2000, the R3000, the Sun SPARC, and the HP PA-RISC family fall into this category.

In the 1990's high-end implementations of RISC architectures were introduced that can issue multiple instructions per cycles. This class of machines have been termed super-scalar machines [Joupp89a, Joupp89b, Stone93]. The RS/6000 and its close relative the Motorola/IBM PowerPC, the Intel i960CA, and the DEC Alpha microprocessor are all super-scalar RISC machines. Indeed, even the recent version of the Intel architecture, the P6, has a core that is a super-scalar RISC engine. To simplify the issue logic and to keep the cycle time down, all of the above mentioned processors have restrictions in the type of instructions that can be issued together. For example, the RS/6000 can simultaneously issue at most one floating point instruction, at most one branch instruction, at most one condition register instruction, and at most one integer instruction, but not other combinations [Bakog90b]. (A pure super-scalar machine would restrict instruction issue only by data and control dependencies.) To support such instruction level parallelism requires multiple function units and the use of multi-port register files to feed those function units. The impact of multi-porti register files leads to a number of important design issues, which the models we develop allow us to explore (see Chapter 3).

Designing super-scalar systems that exploit instruction level parallelism has been a subject of extensive research. Some of this research is described in [Butle91, Hwang84, Joupp89b, Wall91]. All these techniques lead to a processor requiring large memory bandwidths to keep the instruction pipeline full and to supply all the required instruction operands. Since memory references are slow, most new processors have on-chip caches, with some designs calling for multiple levels of on-chip caches. For example, DEC's

Alpha 21164 has 16K primary and 96K secondary on-chip caches [MPR94]. Indeed, it is the improvements in implementation technology that have allowed designers to put more components such as multi-level caches on the processor chip.

There is no indication that the pace of improvement in processor performance is slowing down during this decade. The processor performance is still a bottleneck in many graphics, virtual reality, and database applications. The barrier of physical limits has so far been highly exaggerated [Patte94]. It was not long ago, that people predicted that it would be impossible to decrease the minimum feature size below the wavelength of visible light [Henne91]. The argument was that transferring patterns using visible light to a wafer with optical lithography would be impossible. However, technical innovations like phase-shift masks have reduced the feature size further than anyone thought possible, using the same optical lithography techniques. Another significant physical limit is the chip clock speed. Some researchers have predicted that 1 GHz may be an absolute limit on the clock speed for CMOS chips using metal as interconnect [Stone91]. However, it should be remembered that only a few years ago, a CMOS processor with a 200 MHz clock was considered impossible. This prediction was proven false when initial samples for a super-scalar processor operating at 300 MHz were announced by DEC [MPR94].

Although we mentioned several contributing factors to the remarkable growth in performance that computers have undergone in the last few decades, the key ingredient appears to have been the phenomenal improvements in process technology. These have shrunk feature sizes and increased reliable chip sizes, for instance the 300 MHz chip mentioned previously has 9 million devices and a die size of 298 sq. mm. The main function of ISA and compiler design has been to expose the potential of these technology developments. Equally important, because of the rapid pace of these developments, has been the improvement and power of EDA tools.

We present a general method for modeling various processor components in this thesis. As stated previously the main aim of the model is to reduce the design time by rapidly narrowing the design space. The model should be fast to compute, so that a large number of instances in design space can be explored, and it should be reasonably accurate so that the instances chosen for more detailed analysis or implementation are the best of the possible choices. The models we present are based on work described in [Kayss93a, Bucki15, Sakur90]. These works developed the concept of a macro-model for small circuit structures such as logic gates. We show in this thesis that the model is indeed applicable to larger circuits consisting of combinations of gates, extending it to macro-models for processor components. The models compute the delay of the cell, the area occupied by the cell and the power dissipated by the component as a function of the technology parameters, device sizes, interconnect capacitances and frequency. A method to optimize the circuit using the derived models is also presented. Finally, by way of illustration the macro-modeling technique is applied to the register files of several proposed super-scalar architectures, and their cycle times are estimated from the delays that are calculated from the models. The impact on system performance of register file organization is then shown.

The thesis is organized as follows. The next chapter presents some previous work in this field. This is not an exhaustive survey but a sampling of the on-going work in this field. Chapter 3 presents details of the modeling methodology used in this thesis. Chapter 4 presents the models developed for processor memory structures and validates the model delays and area to actual designs. It is shown that within the constraints of the domain, macro-models are able to predict areas, delays and power accurately. The chapter concludes with macro-models for content addressable memory (CAM) cells. Chapter 5 concludes with a discussion of the contributions of this work and some suggestions for future research directions.

CHAPTER II
PREVIOUS WORK

2.1 Previous Work

This chapter presents some previous work in the area of macro-modeling. There is a large body of work on modeling the area, delay, power dissipation, maximum current, cost, etc., for VLSI circuits. The work in this area has been presented in a variety of styles, ranging from a very pragmatic experimental style to carefully formalized theoretical style. In the next few sections we briefly review some of the work published in this area.

2.2 Complexity Theory of VLSI

The initial work of developing complexity models for VLSI systems was done by C. D. Thompson [Thomp80]. In his work, Thompson presented lower and upper bounds on the area and delay of VLSI circuits. Circuits are described by a 4-tuple (A, T, P, N) where A is the area occupied by a circuit, T is the time taken to solve a problem P, and N is the number of input variables. The notion of optimality of a circuit that Thompson introduced is characterized by bounding the quantity AT^2 by a function of N. For example, the AT^2 complexity for a VLSI circuit that produces a Fourier Transform is

$$AT^2 = O(N^2 \log^2 N) \tag{2.1}$$

The work assumes that each VLSI circuit solves a particular problem. Several other problems such as sorting are considered in Thompson's work. Each of these problems has its own set of conditions. Thompson derives an AT^2 complexity for each problem.

The circuit that solves a particular problem P is decomposed into a set of nodes and nets. The nodes are simple processors that route the signals coming from various nets. Nets are the means by which different nodes are connected and communicate with each other. The problem is reduced to embedding a graph of nodes and nets in a plane. The 'graph' is the circuit diagram for the processing elements and their interconnects, and the 'plane' is the substrate of the VLSI circuit on which the structure is to be laid out. By noting some properties of the resulting graphs, Thompson was able to specify the upper and lower bounds on the area and time delays of some VLSI structures that solve several common problems.

The main problem with Thompson's approach is that it assumes that each chip is specialized to solve only one problem. That assumption may have been valid in 1980 but is no longer true today. Also, the assumptions that I/O delays are not important and the area for power distribution is negligible compared to the area occupied by the rest of the chip, have been largely invalidated by recent developments. In the last half dozen years a significant part of the performance gains in processor performance have come from the use of large on-chip caches, which among other things are a means to reduce the I/O delays. Another assumption made in this work is that signal propagation delay between nodes is zero. This is also no longer true, since, much of the delay in modern VLSI chips occurs in the interconnect rather than in processing elements [Bakog90a].

2.3 SUSPENS

SUSPENS by Bakoglu [Bakog90a] is a generic model that can be applied at various levels of the processor implementation hierarchy. These levels can be the chip itself, the module level where chips are packaged on multi-chip modules or printed circuit

boards, and finally the groups of modules that form complete systems. SUSPENS predicts the clock speed, power dissipation, die size and integration level of a processor as a function of the implementation technologies of these various levels. More specifically, the input to the SUSPENS model consists of the material type (Silicon, GaAs, etc.), device parameters (transconductance, input impedance, etc.), logic depth, package characteristics (number of metal layers, density, interconnect impedance, etc.), and architecture parameters.

The basis of the SUSPENS model prediction is Rent's rule [Bakog90a] which is an empirical relationship between the number of input and output connections of a processing element and the number of gates in that element. It is given by

$$n = Kg^{\beta} \tag{2.2}$$

where, n is the number of pins on a logic block, g is the number of gates in the same logic block, β is the Rent's constant and K is the proportionality constant. β and K depend upon the chip type (e.g., static memory, microprocessor, gate array, etc.) and the implementation technology (CMOS, GaAs, etc.)

Initially Rent's rule is applied at the chip level to determine the number of output pins for the chip. The following calculations are carried out to determine various characteristics of the chip (see Section 9.7 [Bakog90a]):

- The average length of the interconnect on the chip is determined by hierarchically dividing the chip into blocks of logic and the repeatedly applying Rent's rule.
- The gate and chip dimensions are predicted using the results in the above step. These dimensions are then used to calculate interconnect resistance and capacitance, which in turn are used to calculate interconnect delays.
- The maximum delay of the chip is determined by adding the delays due to latches, gates, the effect of clock skews and the interconnect delays computed above. The chip delay is the product of the average gate delay and the number of gates on the longest path from input to output. This also gives the maximum clock frequency of the chip.
- The pin count and the power dissipation of the chip is calculated in the final step.

The process is repeated at the package level using chip level parameters as inputs to the model. The SUSPENS model is then used to calculate package level parameters instead of the chip level parameters. Finally, another iteration of the same process with the chip replaced by package level parameters produces the system level performance predictors.

SUSPENS is a very general model and can predict system performance in a wide variety of technologies by replacing the technology dependent parameters. However, since it is mostly based on the Rent's rule, which was derived based on a specific vendor's (IBM) architecture and packaging techniques, it has some limitations. For example, the model does not take multiplexed outputs into account. Several vendors market their processors in a low cost version which have a reduced number of output pins. These usually employ some form of multiplexing at the pinout, thus for example a 64 bit processor may only use 32 pins for its data bus. This is done for a variety of reasons, for example, to make a current processor chip compatible with some previous processor (e.g., the Intel Pentium, and the earlier 486 DX2). The SUSPENS model would need modification to take this multiplexing into account.

2.4 TELE2.0 and LAST

Ramachandran and Kurdahi's work on developing the macro-models for the area and delay estimates for VLSI circuits is presented in [Rama90, Rama92]. Their approach is a combination of symbolic and constructive modeling. The models are built by accumulating a library of VLSI circuits and their relevant parameters such as the aspect ratio, timing delay, power dissipation, etc. Once a detailed library is compiled, a constructive model for a component describes it in terms of sub-components from the pre-compiled library. The area, dimensions, and delay for the component being modeled is calculated from the structure of the component and values of the sub-components in the library. The library of pre-compiled components is part of a larger tool that is used

primarily for automatic layout generation. The area and dimensions are used in this process to generate an optimal layout for the structure.

2.5 The Alpha Power Law

Sakurai et al. present the Alpha Power Law in [Sakur90]. This law is based on revised analytical equations for MOSFETs. The need to revise existing MOSFET models arose as the minimum layout features shrunk to the sub-micrometer range. At these small feature sizes the velocity saturation effects of carriers becomes pronounced. Since these effects were ignored in Shockley's model, it failed to apply to MOSFET devices in more recent integrated circuits. Sakurai's MOSFET model takes these effects into account.

The new device models were applied to a MOSFET inverter and its delay was calculated. Application of a one transistor model to an inverter is possible, since, in static operation, only one device is active at any give time in an inverter. The inverter delay in its simple form is the given by

$$t_{pHL} = 0.1T_i + 0.5\frac{C_l V_{dd}}{I_{DO}} \tag{2.3}$$

where, t_{pHL} is the time delay between supply voltage V_{dd} and the output voltage measured at the 50% voltage levels as the output falls from high to low, T_i is the input voltage transition time and is the interval between the initial value of input voltage and its final value, C_l is the load capacitance at the output, and I_{DO} is the drain current when gate-to-source voltage is the same as supply voltage. Note that the first term in (1.3) is due to the input wave form delay and the second part is the delay to discharge the capacitor at the output of the inverter.

A power model for a CMOS inverter is also presented in [Sakur90]. The authors have presented a model for static short circuit power, which is the power dissipated in the circuit when the input voltage is half of the difference of the supply voltage and ground. The short-circuit static power is given by

$$P_s = V_{dd} T_i I_{DO} \frac{1}{\alpha+1} \frac{1}{2^{\alpha-1}} \frac{(1-2\frac{V_{th}}{V_{dd}})^{\alpha+1}}{(1-\frac{V_{th}}{V_{dd}})^{\alpha}} \quad (2.4)$$

where, P_s is the short-circuit static power, V_{th} is the threshold voltage and α is a factor that depends on the transistor type and has the value of 1.2 for NMOS and 1.5 for PMOS.

2.6 The Area Model of Mulder et al.

Mulder et al. [Mulde91] presents models for determining cache and register file area. The register area models take as input the basic parameters of the registers like the number of registers, the number of bits in the register, and the type of the memory cell used in the register. Three different types of memory cells are considered. They are a register cell, a static cell, and a dynamic cell. The register cell is a six-transistor memory cell with high bandwidth, the static cell is a six-transistor memory cell with medium bandwidth and the dynamic cell is a three-transistor memory cell. The bandwidth refers to the number of output ports that the memory cell drives. Higher bandwidth requires larger area for a memory cell. The additional area accommodates the extra data and control lines and the larger drivers for data lines, to achieve a specified delay. The area of the register cell is taken as a unit and is termed register bit equivalent, *rbe,* by the authors. The static cell is empirically determined to be 0.6 rbe and the dynamic cell to be 0.3 rbe. The register file area is calculated to be

$$area_{register} = (register_w + L_{sa})(datawidth_b + W_{driver}) \quad (2.11)$$

where, $area_{register}$ is the area of the register file with register memory cells, $register_w$ is the number of registers in the file, L_{sa} is the length of sense amplifiers (assumed to be constant at 6 rbe), $datawidth_b$ is the width of data path in bits and W_{driver} is the width of the decoders and word line drivers (also assumed to be a constant 6 rbe). The area is calculated in rbe units.

The area of a cache is calculated as an aggregate of the components of a cache and is given by,

$$area_{sac} = pla + data + tags + status \tag{2.12}$$

where, $area_{sac}$ is the area of a set-associative cache, *pla* is the control PLA area, *data* the memory array area, *tags* is area of tags for the cache lines, and *status* is the status bit area. The authors present the models for each of these components. The paper also presents the area for direct-mapped and fully-associative caches, which are similar to (2.12).

2.7 Timing Macro-Models using Dimensional Analysis

Kayssi's work on developing accurate timing models is presented in [Kayss92, Kayss93a]. The experimental data is analyzed and function fitting is used to devise equations that represent data. An innovation in this process is the application of the dimensional analysis technique to devise the functions. Dimensional analysis was first presented by Buckingham [Bucki15] in the context of developing mechanical models. Kayssi has successfully demonstrated its applicability to electronic circuits. Buckhingam's techniques are useful in reducing the number of dependent variables in an equation, resulting in simpler macro-models that are faster to evaluate. This technique is discussed more fully in Chapter 3.

Kayssi's approach consists of three steps. First the experimental data is gathered for the circuit to be modeled. Then the independent variables are determined in the function for the circuit using dimensional analysis. Finally, the variables are multiplied by suitable parameters to fit the observed data. At each step the process may be started over. For example, if the fit is not good enough then the dimensional analysis may be repeated to find some other independent variables. The process is repeated until a good fit is obtained. As an example of this analysis, the following macro-model is derived for a CMOS inverter in [Kayss92]:

$$\Delta = T_i\left(c_0 + \frac{c_1}{T_i\sigma} + \frac{c_2}{(T_i\sigma)^2} + \frac{c_3}{\sqrt{T_i\sigma}}\right) \tag{2.5}$$

where, c_0, c_1, c_2, c_3 are constants, and $\sigma = \frac{K V_{dd}}{C_l}$. The quantity $K = \beta\frac{W}{L}$ is the transistor gain factor. The terms W, L are the width and length of P transistor in the inverter, and $\beta = \frac{\mu_n \varepsilon}{t_{ox}}$ where μ_n is the electron mobility through Silicon, ε is the permittivity of Silicon, and t_{ox} is the oxide thickness. The first two terms in (2.5) $c_0 T_i$ and $\frac{c_1}{\sigma}$ are similar to the two terms in (2.3). This is because the second term can be expanded as follows.

$$\frac{c_1}{\sigma} = \frac{c_1 C_l}{K V_{dd}} \tag{2.6}$$

For the condition when drain current $I_D = I_{DO}$, the gate voltage $V_{GS} = V_{dd}$ and the N device is on. If we assume that the N device length is > 1 micron then using Shockley's model

$$I_{DO} = 0.5K(V_{dd} - V_{in})^2 \tag{2.7}$$

where, V_{in} is the threshold voltage for N device. Substituting in (2.6)

$$\frac{c_1 C_l (V_{dd} - V_{in})^2}{I_{DO} V_{dd}} \approx \frac{c_1 C_l V_{dd}}{I_{DO}} \tag{2.8}$$

This is same as the second term in (2.3), and, therefore work of Kayssi may be considered a continuation of work of Sakurai et al.

2.8 The Models of Wada et al.

Wada et al. present their access time models for memories and caches in [Wada92]. Their model for caches takes into account the organization parameters of cache memories, e.g., the number of lines, associativity, and the number of sets. In addition, it also relies on the physical organization parameters of caches. These are the number of segments per word line and the number of segments per bit line for both data

and addresses. In effect, these parameters measure the load on each data and bit line driver. Thus, for a direct-mapped cache that has 16 sets of 16 words per line and 1 segment per word line, the load on the word line driver is 16 words. If the same cache has 2 segments per word line then the load on each word line driver is halved to 8 words. A similar relationship holds for the number of segments in the case of bit lines. This is graphically shown in Figure 2.1.

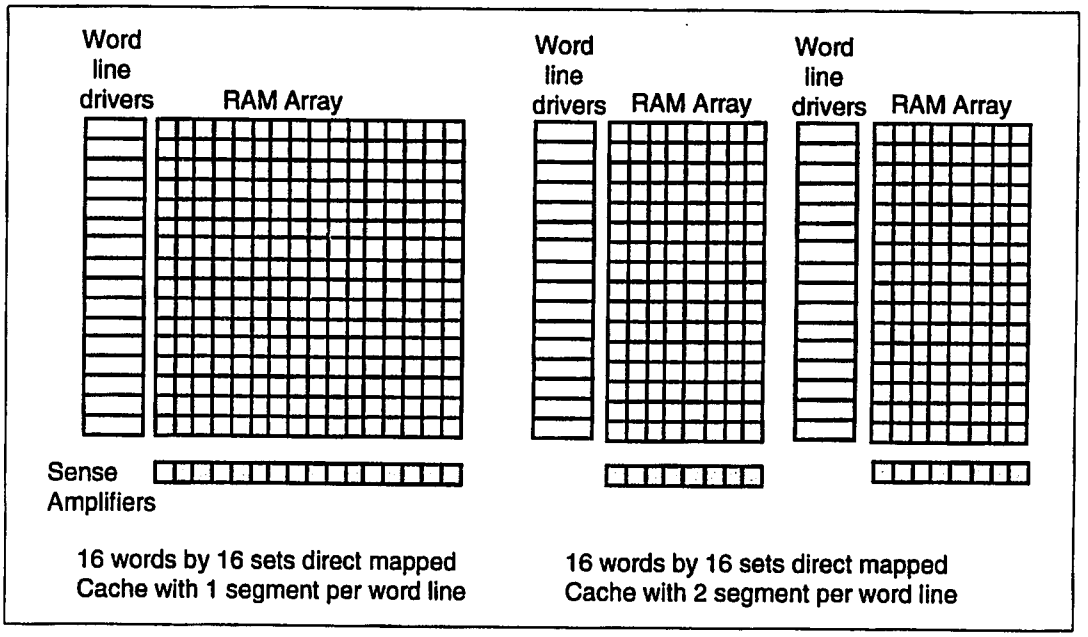


Figure 2.1 Cache Organizations.

The analytical models presented in [Wada92] use the above parameters to calculate the access time of the cache memory. The total access time is sum of decoder delay, word line delay, bit line/sense amplifier delay and the data bus delay. Each of these are analyzed separately and then summed to provide the total access delay of the cache. The total delay is given by,

$$T_{access} = t_0 \bullet D^{\frac{1}{\ln D}} \bullet \ln D + \left(\frac{4C_{wcell} \bullet V_{dd}}{I_{dn}}\right) \times (B \bullet A) \frac{1}{Ndwl} \tag{2.9}$$
$$+ \frac{1}{2G_m} \sqrt{\frac{V_{dd}}{R_l I_c}} (3C_{cm} + C_{dbd}) + \left(-\ln\left(0.5\left(1 - \frac{V_{sense}}{R_l I_c}\right)\right)\right)$$
$$\bullet R_l \bullet C_{bcell} \times (S) \times \left(\frac{1}{Ndbl}\right) + \frac{V_{dd}}{2}\left(\frac{C_{dod}}{I_{dbd}} + \frac{C_{out}}{I_{dod}}\right)$$
$$+ \frac{V_{dd} C_{sout}}{2 I_{dbd}} \times (A \bullet Ndbl) + \frac{4 V_{dd} C_{metal}}{I_{dbd}} \times (B \bullet A \bullet Ndbl)$$

where, t_0 is the unit delay time corresponding to the delay of an inverter with fan-out of one, D is a complex term explained later, C_{wcell} is the word line capacitance per memory cell, I_{dn} is the average saturation current of an NMOSFET word-line driver, B is the cache block size, A is the cache associativity, $Ndwl$ is the number of segments per word line for cache data, G_m is the transconductance of the NMOS transistors in the sense amplifiers, R_l is the load resistance on the pre-charge line, I_c is the cell current on the bit lines, C_{cm} is the input capacitance of the second stage amplifier, C_{dbd} is the input capacitance of the data bus driver, V_{sense} is the minimum voltage difference that can be sensed by the differential sense amplifier, C_{bcell} is the drain junction capacitance of the pass transistors, S is the number of sets in the cache, $Ndbl$ is the number of segments per bit line for cache data, C_{dod} is the input capacitance of a data-bus driver, C_{out} is the output capacitance of the data output buffer, I_{dbd} and I_{dod} are the discharging current through the data-bus driver and the data output driver, C_{sout} is the output capacitance of a data-bus driver, and C_{metal} is the metal line capacitance. The term D is given by

$$D = \left(\frac{C_{Nstage+1}}{C_i}\right) \times (S \bullet \ln S) \times (Ndwl + Ntwl) \tag{2.10}$$

where, $C_{Nstage+1}$ is the input capacitance of each of the word-line drivers, C_i is the input capacitance of the first stage in the decoder and $Ntwl$ is the number of segments per bit line in the tag circuitry.

This analytic model is based on circuit parameters. It does not use any device geometry terms, such as gate length and width, rather, P and N device sizes are assumed to be fixed. The fact that device geometry is not explicit in the formula means that the effect of changing the driver size, wire length, etc. cannot be readily predicted. Such changes can only be accounted for indirectly through changes in capacitance and resistance.

2.9 Summary

We have presented some existing methodologies for building models in the order in which they were published. Generally, these models fall into two broad categories. The first category uses architectural parameters, without taking into consideration technological parameters. Such models are easy to compute for large circuits but are not general enough to be applied to a wide variety of circuits. The second category uses the low level devices as the basis of modeling and considers technological parameters. These models are more accurate and applicable to every circuit that uses the basic devices on which these models are based, but they become very complex and costly in terms of computation time for a component that has more than a few devices.

Thompson's complexity theory of VLSI falls into the first category with the number of inputs and algorithm being the architectural parameters. The work of Bakoglu and Ramachandran are examples of high level modeling of first category. The Alpha Power Law is an example of second category, since it models a single CMOS device using only technology dependent parameters. Kayssi's models can be used to model devices using only technology parameters, and thus can be classified into second category. However, Kayssi's techniques can also be used to model multi-device circuits which may depend on the number of inputs or other architectural features. The resulting models are neither overly complicated nor expensive in terms of computation time. They seem to represent a hybrid type that use technological parameters, but are still simple

enough to compute quickly. Finally, the models of Wada et al. and the area models of Mulder et al. are examples of first category of modeling.

CHAPTER III
BUILDING MACRO-MODELS

3.1 Introduction

In this chapter we briefly discuss the theory of macro-modeling and the methodology we use to build macro-models for area, delay and power. Section 3.2 discusses various issues involved in the modeling process. Section 3.3 discusses various types of models. Section 3.4 discusses the methods for modeling a set of data. It also discusses the methods used to choose various variables in the function and the use of dimensional analysis to reduce the number of variables required in the model. Section 3.5 illustrates two methods to arrive at macro-models for a circuit. Finally, Section 3.6 presents the experiment designs that are used for deriving various models presented in Chapter 4.

3.2 Issues in Modeling

In essence, the problem is that of fitting a function to a set of data, with the condition that the function should have the important variables of interest in it. The purpose of modeling is to arrive at a general rule applicable to all the data under consideration and possibly to other instances of the same problem lying outside the range of data. The main issues in modeling are the accuracy, precision, range of applicability, cost of developing the model, and the cost of evaluating the model.

The accuracy of a model is its ability to predict the actual behavior of a process, given certain operating conditions. The range, or domain, of applicability of a model needs to be carefully determined and explicitly stated. Outside the domain, the model may behave erratically.

The cost of developing experimental behavior models consists mainly of gathering the experimental data for the given process over the domain of interest. If the model is based on a few data points, then the cost of developing the model is low, since the number of experiments required to gather the data is low. Of course, this model is unlikely to be as accurate as a model that uses more data points, and has consequently more development costs. The model developer has to make a trade off between the desired accuracy and the cost of developing the model. Also, the cost of such a model includes the time spent in searching for a mathematical function that fits the experimental data. In the case of a mathematical behavior model, the cost consists of the work done to derive the physical equations for a circuit.

The cost of evaluating a macro-model depends on the complexity of the model. A model consisting of a transcendental functional is more costly to evaluate than a linear function. Since one of the reasons for developing macro-models is to speed up the evaluation, a very complex model may not be desirable. The fastest evaluation may be possible using a table lookup method. However, discontinuous functions cannot be evaluated this way, because interpolation cannot be used to calculate intermediate values in the region of discontinuity.

3.3 Model Types

There are several different types of models that can be built to explain a scientific phenomenon. The usual purpose of a model is to concisely represent certain behaviors of an entity. In our case the entity is a complex electronic circuit, but a wide range of entities are candidates including mechanical machines, chemical reactions or physical bodies.

The behaviors of certain phenomena are very well understood and can easily be represented in terms of mathematical equations with well defined parameters. These are called mechanistic models [Box87]. An example of such a model is Ohm's Law, which states that, given appropriate units, the voltage across an impedance is proportional to the current flowing through that impedance. Impedance is the constant of proportionality in this relationship. On the other hand, there are some phenomena that are not sufficiently well understood to develop a mechanistic model. In addition, there are mechanistic models that are extremely complex and unsuitable for fast evaluation. In either case, it may be beneficial to devise a function that estimates the behavior of a phenomenon over a limited range of input conditions in terms of some parameters of interest. Such models are referred to as empirical models or macro-models. Empirical models may be devised in the beginning of an investigation into an observed phenomenon [Box87] or they may be used to simplify the task of computation [Kayss93a] for a particular function. In the later case the underlying assumption is that by giving up some of the details in the complex or mechanistic model, the resulting model would preserve all the effects of the parameters of interest but be easier and faster to compute. Various types of empirical models are described in literature [Box87, Danie80, Focke53, Kayss93a, Schic68, Shara94, Stone93, Taylo74, Wong94]. We can classify empirical models based on the method of derivation and the way the data is represented. Some major types of empirical models based on these criteria for electrical circuits are presented in the next few sections.

3.3.1 Circuit Reduction Model

The circuit reduction model replaces the original circuit with a simplified circuit having the same behavior as the original circuit, at those ports of interest. The simplification may be in the reduced number of circuit components or the replacement of components with ones that are easier to analyze. The most common form of circuit reduction technique, the Thévenin equivalent circuit, replaces a circuit with multiple

voltage sources and impedances with one equivalent voltage source and one equivalent impedance in series with the voltage source. Alternatively, the circuit may be replaced by an equivalent current source and an equivalent impedance in parallel with the current source. This is the familiar Norton equivalent circuit. Both of these techniques reduce the number of components in the circuit under consideration.

3.3.2 Circuit Behavior Models

Circuit behavior models describe the behavior of a circuit between two ports within a given range of input parameters, without fully describing the actual mechanisms involved. These can be further classified into three types based on the method of derivation and way the model data is stored.

3.3.2.1 Experimental Behavior Model

In this approach, the circuit to be modeled is subjected to various experiments. The input and corresponding outputs are plotted and a curve fitting approach is used to describe the behavior of the output. This is sometimes described as a black box approach to modeling, because it does not require any knowledge of the circuit structure (within the black box). This model may not be the same as a model for the same circuit derived using circuit analysis, although the values they produce should closely agree. Typical circuit attributes that are modeled this way include signal propagation delay between two ports of a circuit, power dissipation, input or output impedance, and area.

3.3.2.2 Mathematical Behavior Model

The mathematical behavior model is another type of macro-model which is a simplified form of the typically complicated physical equations of all the circuit components. These may be polynomial or piecewise linear equations. A well known example of a mathematical model is the quadratic FET model of Schichman and Hodges [Schic68] which is used in the SPICE circuit simulator. The major difference between

this and the experimental model is the fact that in this type of model, mathematical equations are derived using the physical models of various components, whereas the experimental model uses experimental data to relate outputs to inputs.

Generally, mathematical behavior models are simpler than the mechanistic models, but they are still complex compared to experimental behavior models. Typically they are too complicated to derive for circuits with more than a few devices.

3.3.2.3 Symbolic Model

The symbolic macro-model is implemented as a pre-compiled function or subroutine. Whenever analysis of a circuit of this type is required, its associated function or subroutine is called to return the quantity of the interest for this analysis. Usually, these quantities are stored as multi-dimensional tables with each dimension representing an independent variable of the model.

3.4 Building a Macro-Model

The macro-models presented in this thesis are of two types. The first type of macro-model predicts the area of a component and the second type of the macro-models predicts the delay and power of the cell.

The area macro-models are derived by observing the layout geometry, number of inputs, number of outputs, and the buffer size of the circuit. Thus, these macro-models take cell dimensions, number of ports, increment in cell size as the buffer size increases, and process dependent rules as parameters. These rules are the familiar design rules which include the minimum feature widths of poly and first metal layer lines, minimum overlap of poly over active, minimum spacing between first metal layer lines, etc. The macro-models can be applied to another process by substituting the design rules for the new process.

The process of macro-modeling for delays consists of the five steps shown in Figure 3.1. First, the variables to be used in the equations are chosen. The variables can be chosen arbitrarily or by examining the circuit equations using the circuit analysis techniques. Choosing the variables arbitrarily has the problem that some important variables may be ignored, and is useful for very simple systems only. The circuit analysis technique on the other hand does not have this disadvantage. The circuit equations are derived based on Kirchoff's voltage law, Kirchoff's current law and the basic circuit equations. In deriving these equations the circuit itself may be simplified to remove any components that have a negligible effect on the quantities under consideration. A rule of thumb is to ignore any component whose effect is less than 10% on the quantity under investigation.

In the second step the domain is specified by choosing suitable ranges for the variables selected in step 1, based on their physical meaning.

The third step consists of designing an experiment to measure the dependent and independent variables. The collected data confirms the choice of variables in step 1. However, if the data does not correspond to the choice of variables, for example, the chosen dependent variables are not dependent on the independent variables, we have to backtrack to step 1 and choose another set of variables. If the variables are chosen based on physical equations, this backtracking can be avoided. The experiments to collect data may be actual circuit measurements based on a prototype circuit or they may be simulated on a computer. All of the experiments in this thesis were carried out on computer using circuit simulators.

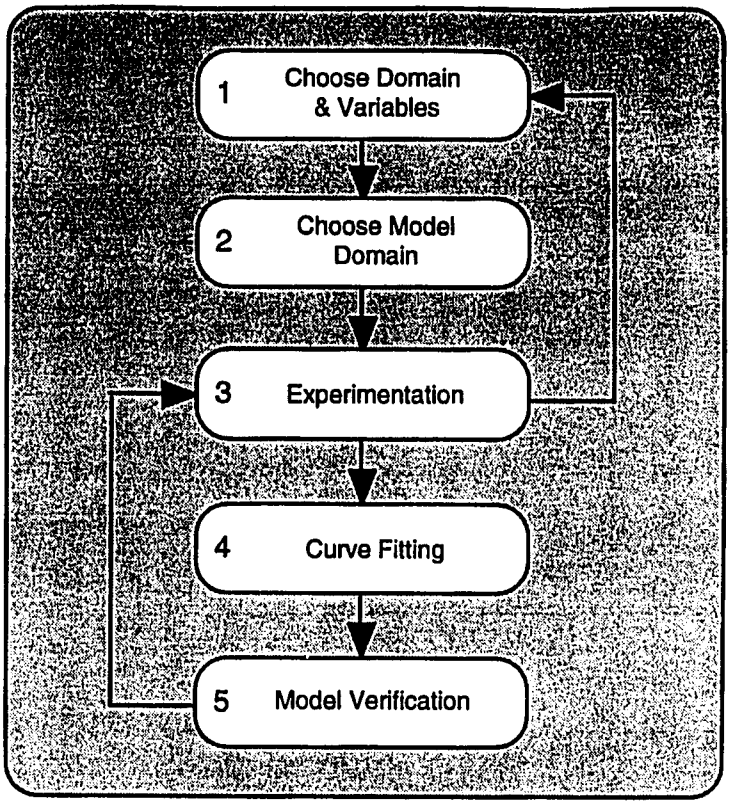


Figure 3.1: The Modeling Process.

The fourth step consists of using mathematical techniques to fit a suitable function to the data collected in step 3. All the variables chosen in step 1 must be included in this function. The function may be a linear or non-linear function. There are a wide variety of techniques for determining the coefficients of the functions, common examples include least squares fit, singular value decomposition, and the Levenberg-Marquardt method. The last method is used to fit non-linear coefficients and was used in our work when the other methods failed. The Levenberg-Marquardt method is implemented in C and described in [Press92]. Most of the time the curve fitting algorithms in various statistical analysis packages (Systat, Deltagraph) are quite adequate to arrive at a suitable curves for the given set of experimental data. However, if two coefficients of a variable fit a function equally well (or equally badly), they lead to singular matrix results that

interrupts convergence to a solution. In such cases we fall back on the Levenberg-Marquardt method.

Given sets of observed values (x_i, y_i) with the y_i depending on the x_i, the goal of curve fitting is to find a functional relationship between the underlying variables x and y. The usual approach is to select a function $y = f(x)$ such that $[y_i - f(x_i)]$ is minimized for all i. For example, in our later experiments we develop the following function

$$y = a_0 + \frac{a_1}{\sqrt{x}} + \frac{a_2}{x} + \frac{a_3}{x^2} \tag{3.1}$$

and then select values for the constants $a_0,\ldots,a_3$ so that the error

$$\left[y_i - \left(a_0 + \frac{a_1}{\sqrt{x_i}} + \frac{a_2}{x_i} + \frac{a_3}{x_i^2} \right) \right]$$

is minimal. In fact, a least squares fit is used to obtain the constants, as noted above.

The fifth and final step consists of verifying the accuracy of the derived macro-model. This is again done by comparing the model predictions against the actual circuit measurements or simulations. If the macro-model's accuracy is within the given bounds it may be accepted or it may be necessary to go back to step 3 and derive the model again.

The basic process of modeling in Figure 3.1 can be augmented to simplify the equations and to reduce the number of variables by dimensional analysis. This modified process in show in Figure 3.2. It adds one more step to the process of modeling. Dimensional analysis is based on the Buckingham's Pi theorem. The theory of dimensional analysis is briefly described in the following sub-sections. The use of dimensional analysis is not without pitfalls. It may lead to elimination of some variables that are in fact important to the model being derived. If this is the case we may have to start over and choose different fundamental variables.

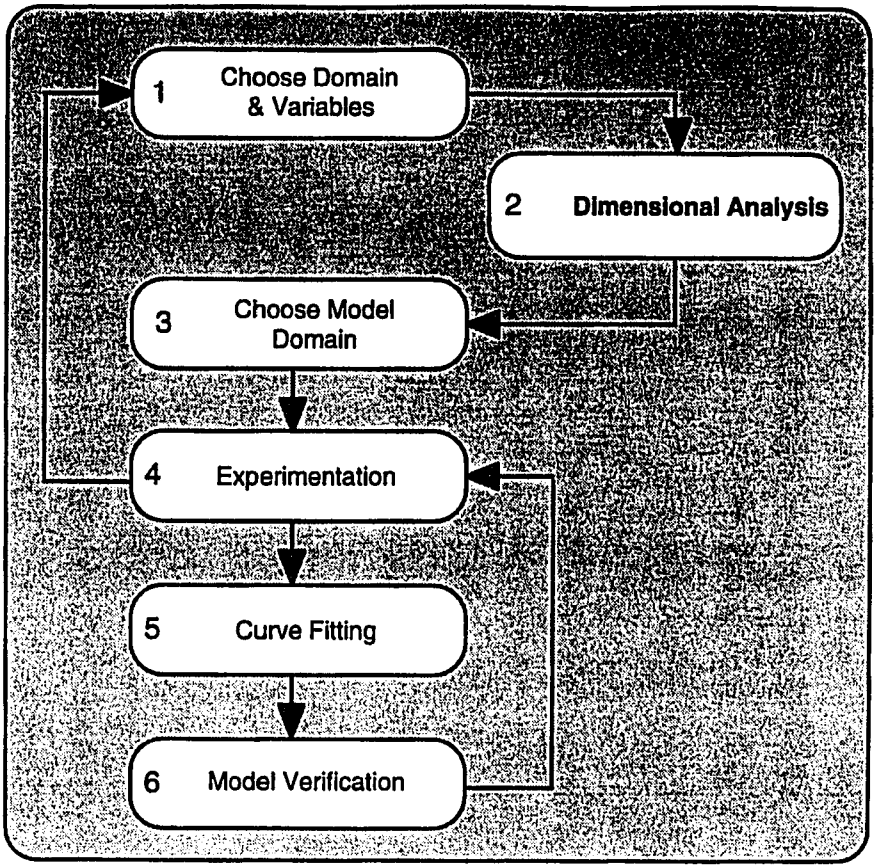


Figure 3.2: The Modified Modeling Process with Dimensional Analysis.

3.4.1 Dimensional Analysis

Dimensional analysis is an analytic method which investigates the relation between the physical magnitudes associated with various phenomena [Focke53, Hammi62, Kayss93a, Taylo74]. This analysis can simplify functional relationships and thus reduces their computational requirements. The disadvantage of the method is that the resulting information may be incomplete compared to that achieved using a detailed mathematical analysis.

Dimensional analysis is widely used in mechanical engineering problems like fluid dynamics, thermodynamics, etc. It has also been used in chemical engineering and to a lesser extent in electrical engineering [Kayss93a, Hammi62].

3.4.2 The Theory of Dimensional Analysis

In order to discuss the theory of dimensional analysis the following definitions are useful. A *physical quantity* is a concept which can be expressed numerically in terms of one or more standards [Taylo74]. The physical quantity is measurable by some experimental method. Different experiments to measure the same physical quantity may yield different results. The result of a specific measurement is known as the *magnitude* of the physical quantity. The magnitude consists of a *numerical measure* and a concept of *unit*. The measurement of a physical quantity is in fact a comparison between two quantities, one of which is taken as standard and called a unit. The magnitude of a physical quantity Q is represented as $Q = q[\overline{Q}]$, where q is a numeric and $[\overline{Q}]$ is the unit of the quantity. Magnitudes are of two kinds: fundamental and derived. The fundamental magnitudes are ones which cannot be expressed in terms of other fundamental magnitudes or quantities, while the derived magnitudes can be expressed in terms of fundamental units. The quantities $Q_1, Q_2, Q_3, \ldots, Q_n$ are said to be dimensionally dependent if there exist numbers $a_1, a_2, a_3, \ldots, a_m \in R$, not all equal to zero, such that

$$\prod_{i=1}^{m} Q_i^{a_i} = \alpha \qquad \alpha \in R \tag{3.2}$$

If this is not satisfied then all the quantities are independent. For example, for a 2 kg mass that is $1 \times 2 \times 4$ m in volume, i.e., 8 m^3, the following quantities are involved

Q_1 = length = 1 m
Q_2 = height = 2 m
Q_3 = depth = 4 m
Q_4 = mass = 2 kg
Q_5 = density = 0.25 kg/m^3

thus $Q_1^{-1} Q_2^{-1} Q_3^{-1} Q_4 Q_5^{-1} = 1$ (constant), hence the quantities $Q_1, \ldots, Q_5$ are dimensionally dependent.

However, if we try to express a quantity of electrical power in this system of length, height, depth, and mass, we get $a_1 = 0$, $a_2 = 0$, $a_3 = 0$, and $a_4 = 0$. Since all the numbers are 0, power is not dimensionally dependent on length, height, depth, and mass.

There is considerable freedom in the selection and number of fundamental magnitudes in a given situation, however the set chosen must be sufficient to form a basis. The analyst can choose some of the magnitudes as fundamental, and the rest as derived, although the fundamental magnitudes in some physical problems are well defined. For example, in electrical problems potential difference (voltage), current and time are usually chosen as fundamental magnitudes while power is usually a derived unit.

A derived or dependent unit can be expressed in one and only one way in terms of the fundamental units. When every magnitude in an equation is expressed in terms of some fundamental magnitudes, the equation becomes homogenous, i.e., each term in equation consists of the same magnitudes raised to the same powers. This is known as the *principle of dimensional homogeneity.* For example, consider the equation

$$v + s = at + \frac{1}{2}at^2$$

where, v is velocity, s is distance, a is acceleration, and t is time. This equation is not homogenous since velocity has units of meters/seconds and distance has units of meters, at has units of meters/seconds and at^2 has units of meters. On the other hand the equation

$$s = vt + \frac{1}{2}at^2$$

is homogenous, since, each term has the units of meters. The power to which each fundamental magnitude is raised is called the *dimension* of that fundamental magnitude.

3.4.3 Buckingham's Pi Theorem

With the help of principle of dimensional homogeneity, Buckingham [Bucki15] showed that any equation relating n quantities $Q_1,\ldots,Q_n$

$$F(Q_1,Q_2,\ldots,Q_n)=0 \tag{3.3}$$

is always reducible to the form

$$f(\Pi_1,\Pi_2,\ldots,\Pi_{n-k})=0 \tag{3.4}$$

in which each of the variables Π (known as a Pi number) represents a dimensionless quantity of the form

$$\Pi_x = Q_1^{\alpha_1} Q_2^{\alpha_2} \cdots Q_n^{\alpha_n} \tag{3.5}$$

where, $\alpha_1,\alpha_2,\ldots,\alpha_n$ are the dimensions to which each of the fundamental units are raised. The number of independent fundamental units, k, required to specify the units of n kinds of quantities and f is the unknown function (the desired macro-model) to be found. This is known as the Pi theorem.

Dimensionless quantities have numeric values that do not change when the fundamental units of measurement change, e.g., the ratio of two voltages stays the same whether the measurement is in millivolts or volts.

3.4.3.1 Calculating Pi numbers

Let $Q_1,\ldots,Q_k$ be the fundamental quantities from a set of quantities $Q_1,\ldots,Q_n$ and let $\alpha_1,\alpha_2,\ldots,\alpha_k$ be the dimensions of Q_1 and $\beta_1,\beta_2,\ldots,\beta_k$ be the dimensions of Q_2 and so on then

$$\begin{aligned} Q_1 &= Q_1^{\alpha_1} Q_2^{\alpha_2} \cdots Q_k^{\alpha_k} \\ Q_2 &= Q_1^{\beta_1} Q_2^{\beta_2} \cdots Q_k^{\beta_k} \\ &\vdots \\ Q_n &= Q_1^{\sigma_1} Q_2^{\sigma_2} \cdots Q_k^{\sigma_k} \end{aligned} \tag{3.6}$$

From (3.5), a typical Pi term has the form $Q_1^a Q_2^b \cdots Q_n^s$ where there are n exponents $a,b,\ldots,s$. Each of these terms is dimensionless provided that $\left[Q_1^{a_1} Q_2^{a_2} \cdots Q_k^{a_k}\right]^a \left[Q_1^{b_1} Q_2^{b_2} \cdots Q_k^{b_k}\right]^b \cdots \left[Q_1^{\sigma_1} Q_2^{\sigma_2} \cdots Q_k^{\sigma_k}\right]^s$ is dimensionless. This is true only if the following linear equations are satisfied

$$\begin{aligned} &\alpha_1 a + \beta_1 b + \cdots + \sigma_1 s = 0 \\ &\alpha_2 a + \beta_2 b + \cdots + \sigma_2 s = 0 \\ &\vdots \\ &\alpha_k a + \beta_k b + \cdots + \sigma_k s = 0 \end{aligned} \tag{3.7}$$

There are k equations and n unknowns $a,b,\ldots,s$.. It has been shown in [Focke53] that if $k < n$, which is generally the case there are n - k solutions to the system of linear equations (3.7). Once the unknowns $a,b,\ldots,s$ are determined each of the Pi terms can be calculated.

3.4.3.2 Choosing Primary Quantities

As mentioned earlier the choice of the primary quantities is arbitrary as long as they remain independent. The minimum requirement is that there should be at least one primary quantity or fundamental unit in each problem under discussion. There is no upper limit to the number of primary quantities. Theoretically, all the quantities can be taken as the primary quantities except the quantity to be related in the macro-model. However, the following general considerations should be followed where possible to select a possible set of the primary quantities.

1. The quantity of interest should always be chosen as a secondary quantity, to make sure that it appears only once and in the numerator of a Pi number, in case Pi number is a ratio of fundamental quantities.

2. The quantities chosen as primary should make the subsequent work of deriving the macro-model simpler without compromising accuracy.

3. Enough fundamental quantities should be chosen to prevent two or more physical magnitudes having the same dimensions. On the other hand the fundamental quantities chosen should not be so many as to complicate the resulting equation with too many experimental constants.

4. There are traditional choices for fundamental quantities in different fields of study, e.g., length, mass and time in mechanical systems or voltage, current and time in electrical systems. In most of the cases, these choices work very well. However, if the nature of the problem dictates an alternate choice, there is no reason to adhere to these quantities.

In addition, Kayssi has cited some other guidelines for choosing the primary quantities in [Kayss93a]. These are briefly described here. The primary quantities chosen should result in Pi numbers that are (i) constant, e.g., ratio of two voltages, (ii) Pi numbers that can be set to constant, (iii) Pi numbers which can be ignored, (iv) Pi numbers that can be recognized, e.g., Poisson's ratio in mechanics problems.

3.4.4 Tables: A Variation on Macro-Modeling

Sometime it is not possible to get a good curve fit for a set of data using the methods given above. This might happen for example when the error in the fitted equation is too high from the observed values. In this case a look-up table may be used to store various values for the different variables in the system. The table has to be n-dimensioned in order to capture information about n variables. A linear interpolation scheme may be used to get the values that are not explicitly stored in the table but are within the range of the stored values.

3.5 An Example of Macro-Modeling

This section describes the modeling methodology outlined above to derive a macro-model for the delay of an n input NOR gate. A 2 input static CMOS NOR gate is shown in Figure 3.3.

The circuit output is *low* (0 volts) when any or both of the inputs are *high* (V_{dd} volts). When both inputs are low output goes to high. To measure the worst case rise delay of the circuit we have to measure the following delays:

1. Rise time of the output when in_1 goes low (in_2 already low)
2. Rise time of the output when in_2 goes low (in_1 already low)
3. Rise time of the output when both inputs go low.

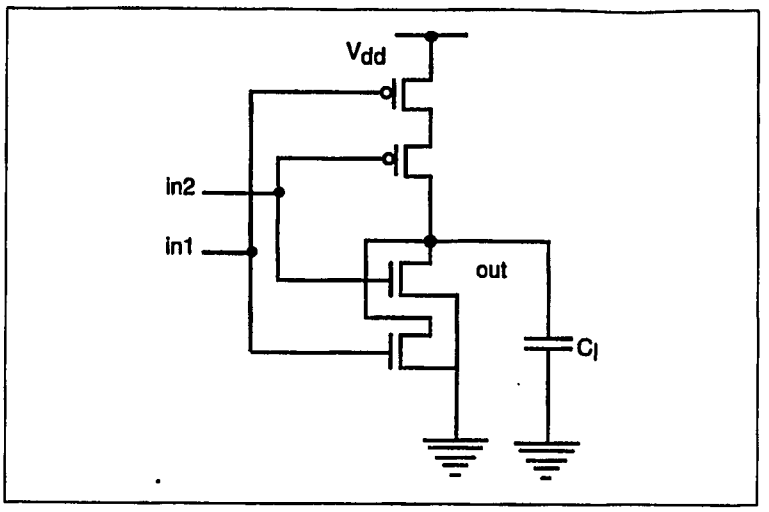


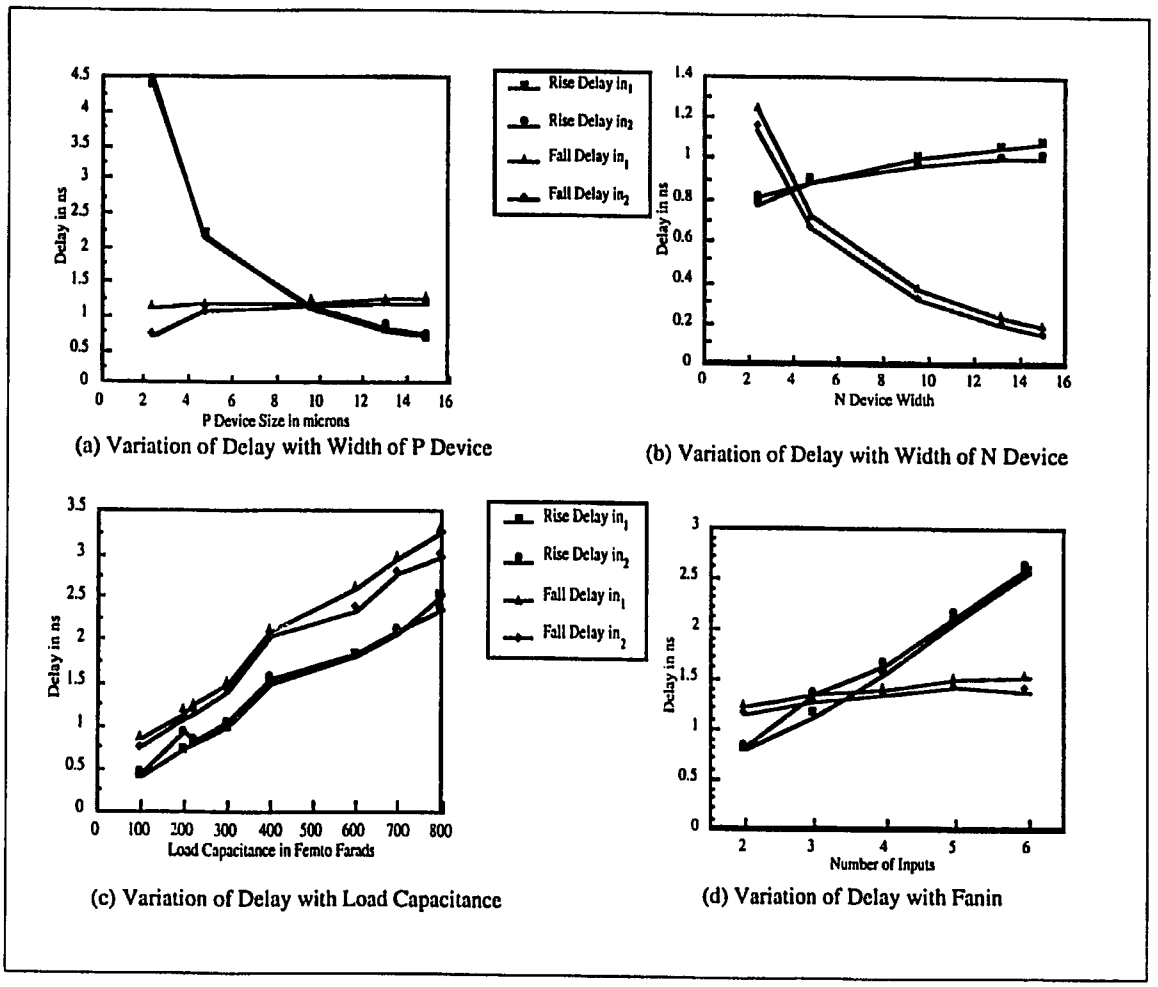
Figure 3.3: A 2-Input Static CMOS NOR Gate.

(A similar list of delays exists for the fall delay of the circuit.) We can eliminate the last case in the above list if we note that in practice both inputs changing at the same time is very rare. Thus if we know the delay between the two inputs separately, we can choose the longest of these as the worst case delay.

Once the voltage levels (usually 50% of V_{dd}) for measuring delays are selected, a set of circuit and process variables that are sufficient for a particular macro-model must be determined. In the case of a NOR gate it is the supply voltage, size of the P devices (for fall time it would be N devices), the transistor gain for the P device, and the load capacitor. Figure 3.4 shows dependence of circuit delays as different parameters vary.

Each of these relationships could be modeled individually. For example, the following four equations represent the curve fits for the relationships shown in Figure 3.4(d).

$$\begin{aligned}
\Delta_{r1}(n) &= -0.00783n^3 + 0.09521n^2 + 0.11305n + 0.58040 \\
\Delta_{r2}(n) &= \ 0.01700n^3 - 0.14685n^2 + 0.80514n + 0.14460 \\
\Delta_{f1}(n) &= -0.00033n^3 - 0.01000n^2 + 0.14626n + 1.07720 \\
\Delta_{f2}(n) &= -0.00592n^3 + 0.03225n^2 + 0.04716n + 1.07640
\end{aligned} \tag{3.8}$$



(a) Variation of Delay with Width of P Device

(b) Variation of Delay with Width of N Device

(c) Variation of Delay with Load Capacitance

(d) Variation of Delay with Fanin

Figure 3.4: Variation of Rise and Fall Delays with Various Parameters.

where Δ_{r1} is the rise delay for input in_1, n is the fanin count, Δ_{r2} is the rise delay for input in_2, Δ_{f1} is the fall delay for the input in_1, and Δ_{f2} is the fall delay for input in_2. The number of equations can be reduced to two if we choose only the worst case rise and fall delays. In the absence of any more information the number cannot be reduced any further. However, if we know, for example, that output is going to be pre-charged high, we can reduce the number of equations to 1 by just taking the worst case fall delay equation.

The problem with (3.8) is that only the relationship of fanin count is considered in deriving the equations and if another parameter like the size of P device in the gate changes, (3.8) become invalid. A better solution is to make all the important parameters variables in the relationship (if one exists), or to have all the relationships represented in one table for easy lookup.

One such relation can be found by applying the circuit analysis techniques to the circuit of Figure 3.3. Assume that in_1 is low and in_2 is changing from high to low. In this condition the upper P device of the circuit is on. The voltage at the source of the second P device is V_{dd}. When in_2 goes from high to low the N device connected to it turns off and the P device turns on. Using first order MOS model [Weste88] the current through the 'on' P device is given by

$$I_{ds} = 0.5 K_p (V_{gs} - V_{tp})^2 \tag{3.9}$$

where, V_{gs} is the gate-to-source voltage, V_{tp} is the P device threshold voltage, and K_p is the MOS transistor gain for the P device. K_p is given by

$$K_p = \frac{\mu_p \varepsilon}{t_{ox}} \left(\frac{W}{L} \right) \tag{3.10}$$

where, μ_p is the hole mobility in the gate channel of P device, ε is the permittivity of the gate insulator, t_{ox} is the gate oxide thickness, W is the width of the device and L is the length of the device. The time to charge the load capacitor is given by

$$\Delta = \int C_l \frac{dV_{gs}}{I_{ds}} \tag{3.11}$$

where, C_l is the load capacitance.

(3.11) gives the time to charge the capacitor high when in_2 changes thus (3.11) is used to model the rise time delay when in_2 changes. If we replace the current and voltages with those across the N device a similar equation would describe the fall time of the circuit.

Substituting value of V_{gs} and I_{ds} from (3.9) into (3.11) yields the following relationship

$$\Delta_{r2} = \int C_l \frac{d(V_{dd} - V_{in2})}{K_p (V_{dd} - V_{tp} - V_{in2})^2} \tag{3.12}$$

which leads to the following

$$\Delta_{r2} = F(V_{dd}, V_{tp}, K_p, C_l, T_i) \tag{3.13}$$

where, T_i is the time constant of voltage input wave form. T_i is a variable in (3.13) to take care of the different kinds of input wave forms that can be applied at the input. (3.13) has 6 variables. As discussed in Section 3.4.2 it is customary to use voltage, ampere and seconds as the primary units for electrical systems. However, these three are not the only choices and as shown in [Kayss93a] any set of mutually orthogonal units could be used as primary units. Using voltage, ampere and seconds as primary units, we can deduce by using Buckingham's Pi theorem that (3.12) can be expressed in terms of 3 dimensionless quantities. One possible form of the relation is

$$\frac{\Delta_{r2}}{T_i} = F_m\left(\frac{V_{dd}}{V_{tp}}, \frac{C_l}{K_p V_{dd} T_i}\right) \tag{3.14}$$

Note, that every variable is divided by a primary quantity that makes it dimensionless, thus C_l which by definition is $i\frac{dt}{dV}$ (units As/V) is divided by K_p (units A/V2) V_{dd} (units V) and T_i (units s), further, $\frac{V_{dd}}{V_{tp}}$ is constant for a given process and technology, for example for MOSIS 1.2 micron process V_{dd}=5V and V_{tp}=0.7V , thus[1]

$$\frac{\Delta_{r2}}{T_i} = F_m\left(\frac{C_l}{K_p V_{dd} T_i}\right) \tag{3.15}$$

---

[1] Strictly speaking F_m in (3.14) and that in (3.15) are different. However, for notational convenience we do not distinguish them– it being understood that the second is the same as first but with some variables evaluated at constant points. We keep this convention in all our later discussions.

This equation is significant because we are able to express the relationship between delay and various independent variables in one analytical relationship (except the fanin count, which will be incorporated into the model later). This is shown graphically in Figure 3.5. Also shown are curve fits using various techniques to model the data in the graphs. Assuming $x = \frac{C_l}{K_p V_{dd} T_i}$ these models are given below.

The polynomial model:

$$\frac{\Delta_{r2}}{T_i} = -0.0027395x^3 + 0.077121x^2 - 0.6685502x + 1.999429 \qquad (3.16)$$
$$R_3^2 = 0.23,\ R_2^2 = 0.32,\ R_1^2 = 0.49,\ R_0^2 = 0.81$$

The exponential model:

$$\frac{\Delta_{r2}}{T_i} = 1.1e^{-.15x} \qquad (3.17)$$
$$R^2 = 0.81$$

The power model:

$$\frac{\Delta_{r2}}{T_i} = 1.42x^{-0.82} \qquad (3.18)$$
$$R^2 = 0.95$$

Kayssi's model:

$$\frac{\Delta_{r2}}{T_i} = 0.03164 + \frac{0.236}{x} + \frac{0.7814}{x^2} + \frac{0.5543}{\sqrt{x}} \qquad (3.19)$$
$$R^2 = 0.99$$

The R^2 terms are the regression coefficient in these equations. A higher value of R denotes a better fit. The models are valid for a load capacitance from 100 fF to 800 fF, input time constant from 1 ns to 8 ns, and width of P device from 2.4 micron to 15 microns. Together these specify the domain of the model. Figure 3.5(c) and (d) show the importance of domain of model since data tends to infinity near $\frac{C_l}{K_p V_{dd} T_i} = 0$ while the curve is close to 0 at that value.

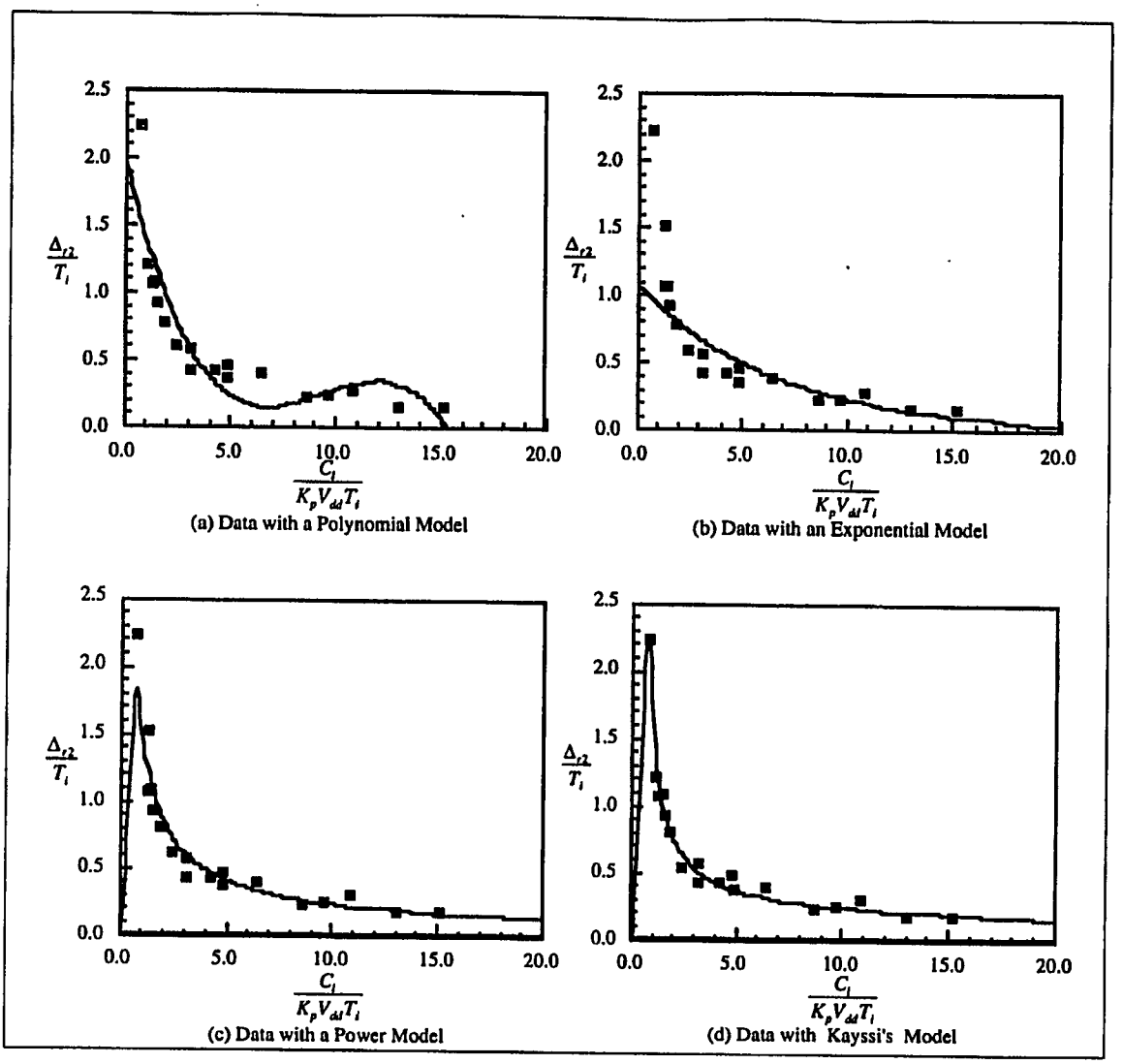


Figure 3.5: Various Curve Fits for the NOR Gate.

For an n input gate (3.14) can be modified to (3.20)

$$\frac{\Delta_{r2}}{T_i} = F_m\left(\frac{V_{dd}}{V_{tp}}, \frac{C_l}{K_p V_{dd} T_i}, n\right) \tag{3.20}$$

where, n is the number of inputs to the NOR gate. As explained previously $\frac{V_{dd}}{V_{tp}}$ can be eliminated. Finally one form of analytic function that satisfies (3.20) is given by (3.21).

$$\frac{\Delta_{r2}}{T_i}(x) = -0.421 + \frac{9.005}{x} - \frac{3.649}{x^2} - \frac{3.478}{\sqrt{x}} + 0.332n - 0.145\sqrt{n} \qquad (3.21)$$

$$R^2 = 0.802$$

Figure 3.6 shows the model prediction and the delay for the 3 input NOR gate ($n = 3$). The domain of (3.21) is the same as that for (3.16)-(3.19) with the added condition that $n \geq 2$. A better prediction can be obtained if we use the Table 3.1 for calculating the delays and interpolate between the entries when necessary. The domain for this table is the same as for the analytic model above.

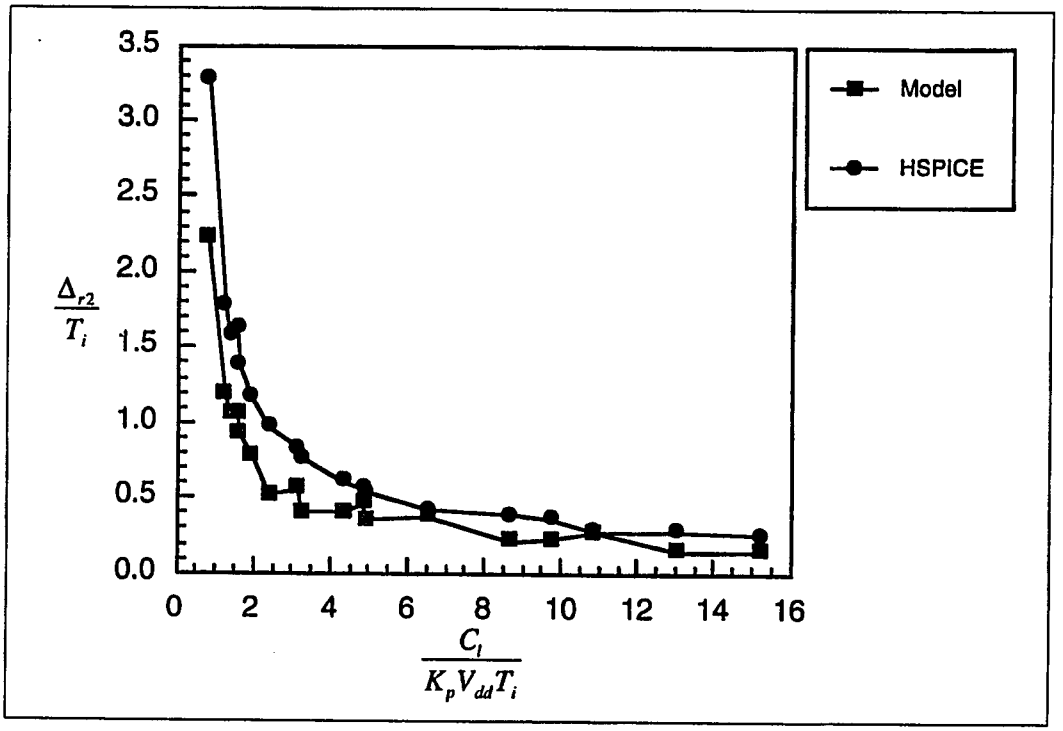


Figure 3.6: Model vs. the Curve Obtained from HSPICE Simulation for 3-Input NOR.

$\frac{C_l}{K_p V_{dd}}$	Delay in nano-seconds				
	n=2	*n=3*	*n=4*	*n=5*	*n=6*
0.790	2.219	3.273	4.428	5.875	7.105
1.217	1.194	1.778	2.365	3.022	3.867
1.391	1.053	1.573	2.091	2.700	3.450
1.580	1.060	1.625	2.188	2.896	3.504
1.623	0.916	1.379	1.832	2.376	2.835
1.947	0.779	1.162	1.547	2.197	2.425
2.434	0.516	0.971	1.281	1.660	2.020
3.161	0.555	0.829	1.103	1.463	1.777
3.245	0.406	0.761	1.019	1.326	1.614
4.346	0.406	0.608	0.809	1.071	1.309
4.868	0.463	0.561	0.748	0.995	1.214
4.939	0.350	0.537	0.715	0.949	1.158
6.519	0.380	0.419	0.548	0.712	1.012
8.692	0.215	0.388	0.418	0.540	0.648
9.735	0.224	0.362	0.487	0.665	0.814
10.865	0.275	0.280	0.379	0.475	0.561
13.038	0.147	0.280	0.336	0.416	0.519
15.211	0.147	0.254	0.300	0.385	0.430

Table 3.1: Table for NOR Delay Calculation.

A similar procedure leads to the power macro-model for the NOR gate. From Figure 3.3, we can deduce that the power of the NOR gate depends on the supply voltage, frequency of the switching input, the device sizes for each of the transistors, the output capacitance, the input voltages at the inputs, etc. Note that the CMOS circuits only dissipate power when the output switches, otherwise the power dissipation is negligible and is ignored in the following analysis. Also, the gate current in a typical CMOS device is negligible, so power dissipation that takes place as a result of the input voltages driving

the NOR gate can be ignored. From these assumptions we derive the following relationship.

$$P_{nor} = F(V_{dd}, \upsilon, K_{n1}, K_{n2}, K_{p1}, K_{p2}, V_{tn1}, V_{tn2}, V_{tp1}, V_{tp2}, C_l) \tag{3.22}$$

where V_{dd} is the supply voltage, υ is the frequency of the switching input, K_{n1} and K_{n2} are the transistor gains for the N type devices in the circuit, K_{p1} and K_{p2} are the transistor gains for the P type devices in the circuit, V_{tn1} and V_{tn2} are the N device threshold voltages, V_{tp1} and V_{tp2} are threshold voltages for P type transistors, and C_l is the load capacitance at the output of the NOR gate. The transistor gain of N device is given by

$$K_n = \frac{\mu_n \varepsilon}{t_{ox}} \left(\frac{W}{L} \right) \tag{3.23}$$

where μ_n is the hole mobility in the gate channel of N device, ε is the permittivity of the gate insulator, t_{ox} is the gate oxide thickness, W is the width of the device, and L is the length of the device. If we assume that both N devices are the same size then $K_{n1} = K_{n2} = K_n$ and $V_{tn1} = V_{tn2} = V_{tn}$. A similar assumption about the P devices leads to $K_{p1} = K_{p2} = K_p$ and $V_{tp1} = V_{tp2} = V_{tp}$. This leads to a simplified form of function that is given by (3.24).

$$P_{nor} = F(V_{dd}, \upsilon, K_n, K_p, V_{tn}, V_{tp}, C_l) \tag{3.24}$$

The dependence of power on various circuit parameters is shown in Figure 3.7. Experimental estimation of power in these graphs were made with the assumption that the power dissipation due to the input voltage is negligible (as CMOS gate current is negligible). The power is measured for 2 clock cycles and then averaged. The power is calculated as the product of current and the power supply voltage. The switching frequency is fixed at 10 MHz. An interesting point to note about the graphs of Figure 3.7 is that the power dissipation decreases as we increase the fanin count of the circuit. This is due to the increased impedance offered by the series connected P devices.

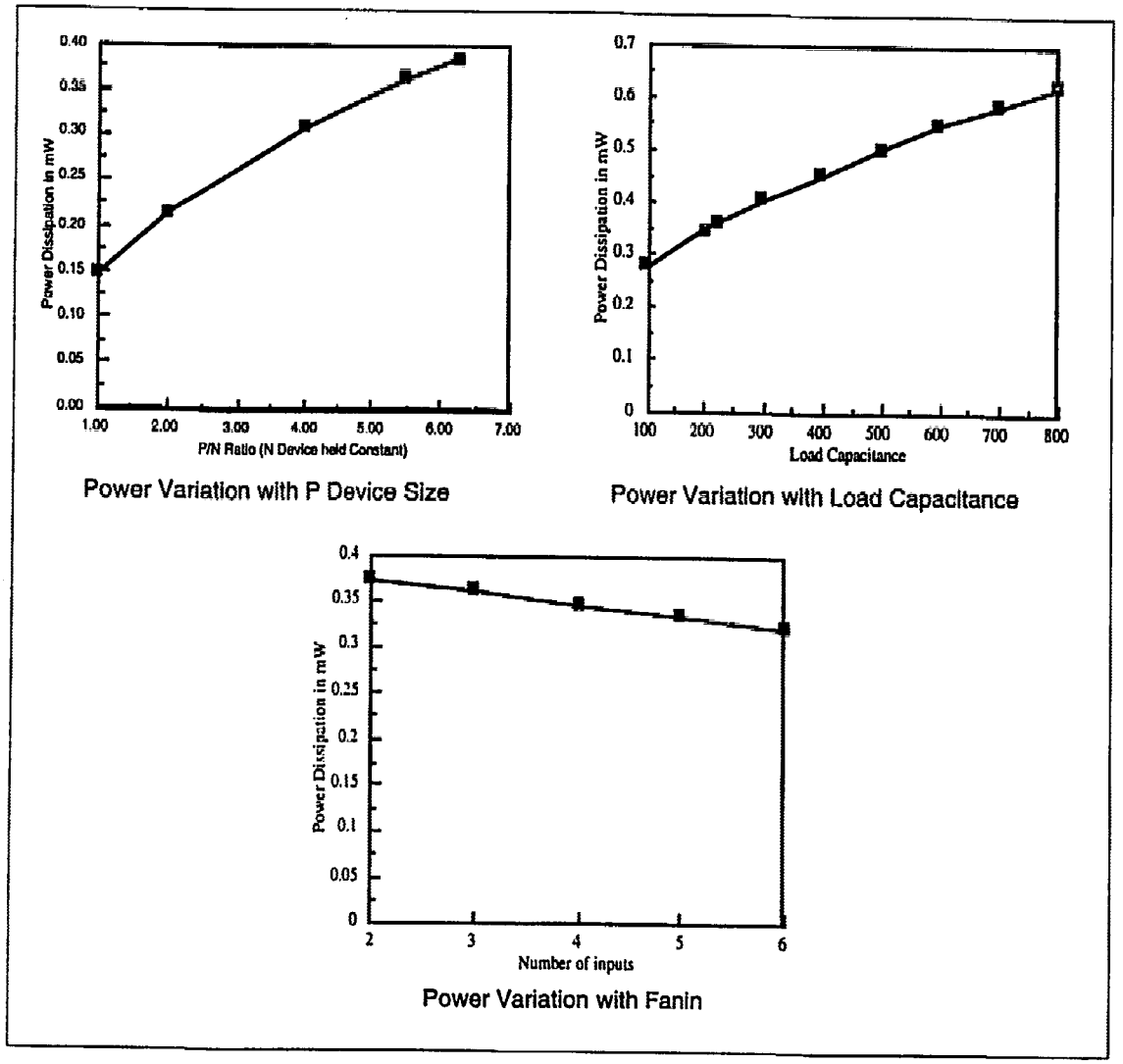


Figure 3.7: Power Dissipated by a NOR Gate vs. Various Parameters.

By applying dimensional analysis and a reasoning similar to that used in deriving (3.14) we get

$$\frac{P_{nor}}{K_p V_{dd}^3} = F\left(\frac{K_n}{K_p}, \frac{V_{tn}}{V_{dd}}, \frac{V_{tp}}{V_{dd}}, \frac{C_l \upsilon}{K_p V_{dd}}\right) \tag{3.25}$$

Of the three independent variables, the voltage ratios can be eliminated if we are only concerned with one particular technology. Further simplification can be made in (3.25) if we are concerned with only rise or only fall delay, since if we are only concerned with

rise delay then N device size can be of fixed size and the ratio $\frac{K_n}{K_p}$ can be eliminated, because it is constant for a given process and is dimensionless. Similarly, if we are optimizing for fall delay then P device size may be kept constant, and therefore eliminated from (3.25). With the assumption that we are dealing with MOSIS 1.2 micron process and optimizing the gate delay for rise time, a possible macro is given below:

$$\frac{P_{nor}}{K_p V_{dd}^3} = 2.767\times10^{-4} \times \left(\frac{C_l \upsilon}{K_p V_{dd}}\right)^{0.43} \tag{3.26}$$

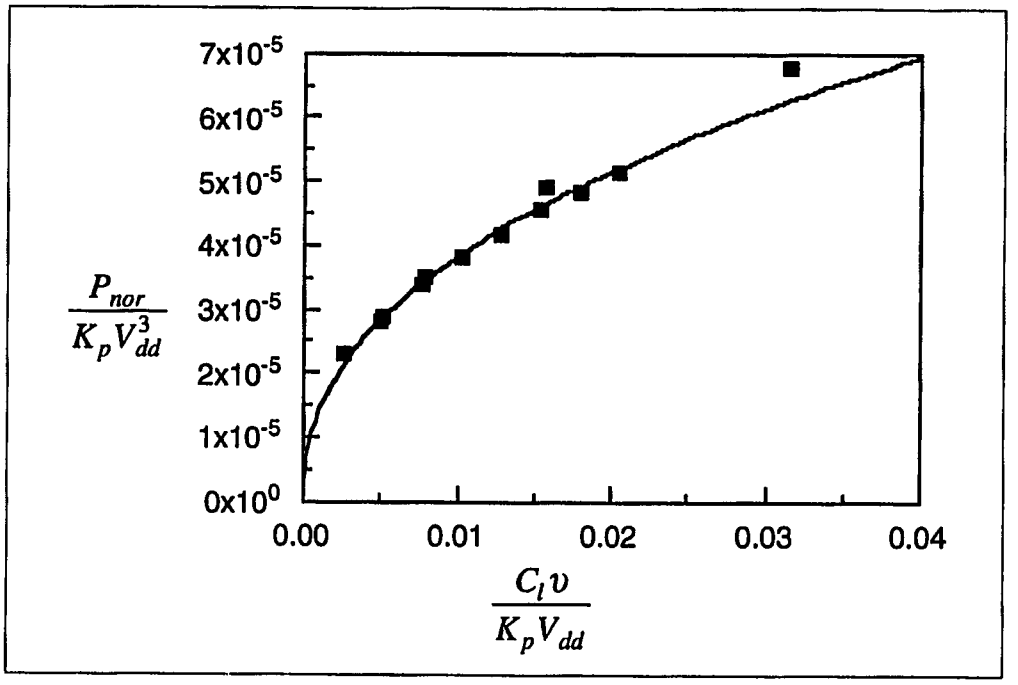


Figure 3.8: Curve Fit for 3-Input NOR Gate.

The curve fit for the function is shown in Figure 3.8. The square points represent the observed values while the solid line represents (3.26). The domain of (3.26) is given by a capacitance range from 100 fF to 800 fF, and a P device size range from 2.4 micron to 15 microns.

3.6 Methodology

The experiment design (step 4 of Figure 3.2) of the modeling process is described in the next few paragraphs. The first substep in experimentation is to layout the target cell being modeled. The cells were laid out using Mentor Graphics IC Graph (IC) package. The cells are laid out according to the design rules of MOSIS SCMOS 1.2 micron process. The layout was verified using the layout design rule checker and the electric rule checker. Once the leaf cells layout was complete and verified, the corresponding electric network structure was created in Mentor Graphics Design Architect (DA) package and the structure was compared with layout using layout vs. schematic checks in IC. The layouts for various cells are attached to this document in Appendix A. The leaf cells were then placed side by side to form arrays of cells. These were again checked for any design rule violations. If there were any errors then the design of the original leaf cell was modified to eliminate these errors. Some of the circuits that were constructed for experimentation are enumerated in Table 3.2. Then these arrays were compared to the corresponding logic arrays created in DA. The arrays were successively incremented. This enabled us to determine the relationship between the number of cells and the capacitances. After that the layout parasitic capacitances were extracted using the tools provided in IC module of Mentor Graphics. The layout capacitances are generated in the form of a netlist in HSPICE format.

Once the Spice netlist is generated it can be directly read by HSPICE. This is helpful if one is interested in only the delay of the circuit being simulated. We also wanted to know the capacitive loading on each of the signal lines[2]. Therefore, we used the translated netlist to determine the capacitive loading on each of the line. This was done with a small awk script. The whole process is presented in Figure 3.9.

---

[2] The capacitances were needed as parameters in macro-models.

Circuit Name	Number of Devices	Number of Nets	Area
5 × 32 Decoder	330	172	1282.8 × 109.2
1 Read Port Register File	8,948	3,515	1383.6 × 1150.2
2 Read Port Register File	10,398	3,749	1651.8 × 1315.2
3 Read Port Register File	11,848	3,981	1780.2 × 1720.8
4 Port Data Path	14,834	5,200	2539.2 × 1971.0
5 Read Port Register File	14,556	4,385	2265.8 × 2605.8
6 Read Port Register File	15,868	4,458	2656.8 × 2605.8
12 Read Port Memory Array	20,032	4,264	4389.0 × 2364.0

Table 3.2: Details of Some of the Circuits Constructed for Experiments.

Notes: All register files have one write port. The data path circuit includes a register file with a CLA adder. The memory array circuit is different from register file in that it does not have the address decoder and bit line sense amplifiers.

Another problem in simulating the HSPICE netlist generated by the IC module is that it is very large and has thousands of elements. Simulating such a large file requires extraordinary resources, for instance we were unable to simulate the file on a 32 Mbyte machine with 150M of virtual memory (the new version of HSPICE has the capability to dynamically assign memory [Hspic92], otherwise this would have been another stumbling block in our investigation). Also, it takes a long time just for one iteration of simulation. To simulate a register file with multiple configurations, each with various parameters indeed requires a lot of time. Therefore, we simplified the task, by simulating one row and one column of the register only. This required pruning the HSPICE netlist generated by the expert IC module. This extra step resulted in large savings in time. For

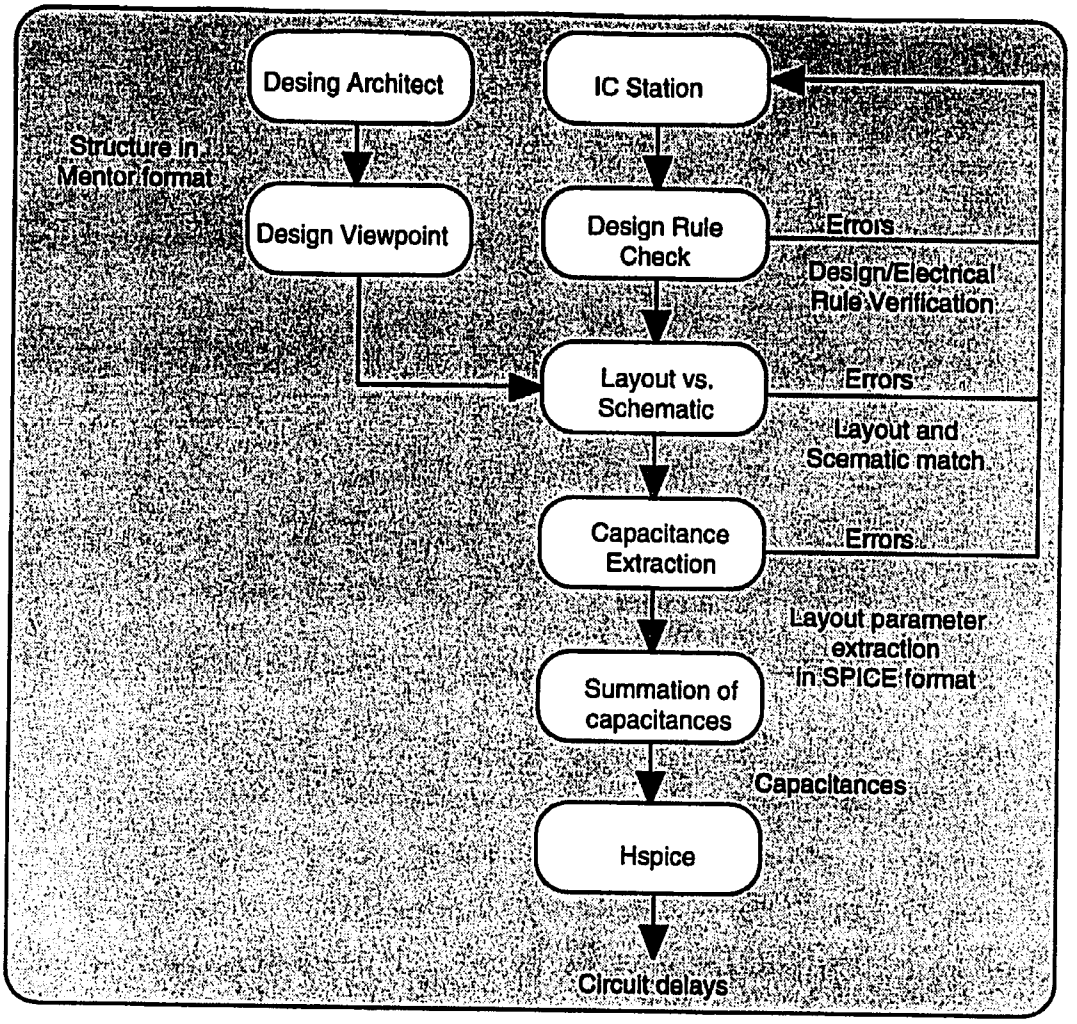


Figure 3.9: The Experimental Design.

example, the average run of a 1 port register file was reduced to 1 minute from 3.5 hours previously. This enabled us to do a lot more experiments than would have been otherwise possible.

3.7 Conclusion

In this Chapter we have presented the theory of modeling and dimensional analysis. The process for modeling used in this dissertation is described in detail. Two slightly different techniques of building the macro-models are illustrated. The first method starts with the circuit to be modeled and simplifies it by analyzing the circuit

operating conditions. The simplified circuit has fewer components than the original. This circuit is then analyzed using circuit analysis techniques, to determine the relevant variables for the model. Dimensional analysis then reduces the number of variables. The circuit is then simulated for a range of values of the variables. The results are tabulated and fed to a curve fitting program. This leads to a general macro-model for the given circuit. This model is verified over the specified domain. These techniques were applied to a CMOS NOR gate and a model for its rise time delay is presented. The model is different from any that have been presented so far as it makes the fanin count a parameter of the model. This allows for a more complete model that has more applications.

The second method starts by enumerating all the parameters of the circuit to be modeled. It then eliminates some of the parameters by making valid assumptions, thus reducing the number of variables. Then, dimensional analysis further reduces the number of variables. Finally, the curve fitting techniques are used to fit a function to the experimental data. This technique was used to build the power macro-model for the NOR gate.

CHAPTER IV

MACRO-MODELS FOR PROCESSOR MEMORY STRUCTURES

4.1 Introduction

In this Chapter we discuss the design of processor memory structures, specifically register files and CAMs. Section 4.2 presents the basic components of a multi-ported register file. Section 4.3 discusses the circuit design of these components. Section 4.4 presents the design approach used to combine the components into a multi-port register file. Section 4.5 develops macro-models for multi-ported register files. These include area, access time, and power models. Section 4.6 discusses the use of replication as a means to improving area, access time, or power. Sections 4.7 and 4.8 illustrate the application of the macro-models developed in Section 4.5 to some published architectures that have register files as a central part of their architectures. Section 4.9 presents CAM designs and the macro-models for one CAM structure. Section 4.10 summarizes the chapter.

4.2 Register Files

The register file design used in out study is very simple. The address bus is connected to decoders which select the appropriate register. There is one driver per register, at each of the decoder outputs. The selected register puts its data on the bit lines. The bit lines are read by sense amplifiers. The output of the sense amplifiers is connected to the data bus.

As explained in Chapter 1 current high performance architectures call for multi-port register files to increase the data bandwidth to and from the functional units. For example, a super-scalar architecture requires multi-port register files so that multiple function units can access operands simultaneously. Generally each of these function units may require reading of up to two operands from the register file and the writing of up to one operand to the register file per instruction. Thus, in a super-scalar machine with n functional units, the upper limit on the number of read ports in a register file is $2n$ and write ports is n. Not all of the these functional units may be needed, however. The machine may require less read ports for the following reasons:

- Some units may not require 2 read ports or 1 write port, e.g., load units require only one write port, and store units only require one read port.
- In the presence of a dedicated floating point register file, the floating point units do not require access to the integer register file and vice versa.

A block diagram of a two read port register file is shown in Figure 4.1. Each of the ports requires its own address bus and its own set of decoders for selecting the appropriate register port. The output of these decoders are connected to separate read ports on each register, so that, for example, the same address applied to each of the address busses will select the same registers to be output on both read ports. Apart from the duplication, the decoding scheme is exactly the same as that of a one read port register file.

The main design trade-offs in a multi-port register file are among the area of the register file, its access time, and its power dissipation. A register file has to be fast so that the processor cycle time is not limited by the register file read access time. It should be small enough to satisfy the area constraints and should have a power rating that satisfies the power dissipation budget. These requirements conflict with each other. To make a register file fast we need to have large drivers in the memory cells which increase both the area and the power dissipation of the register file. The circuit designer has to achieve a balance between these conflicting parameters depending on the requirements of a

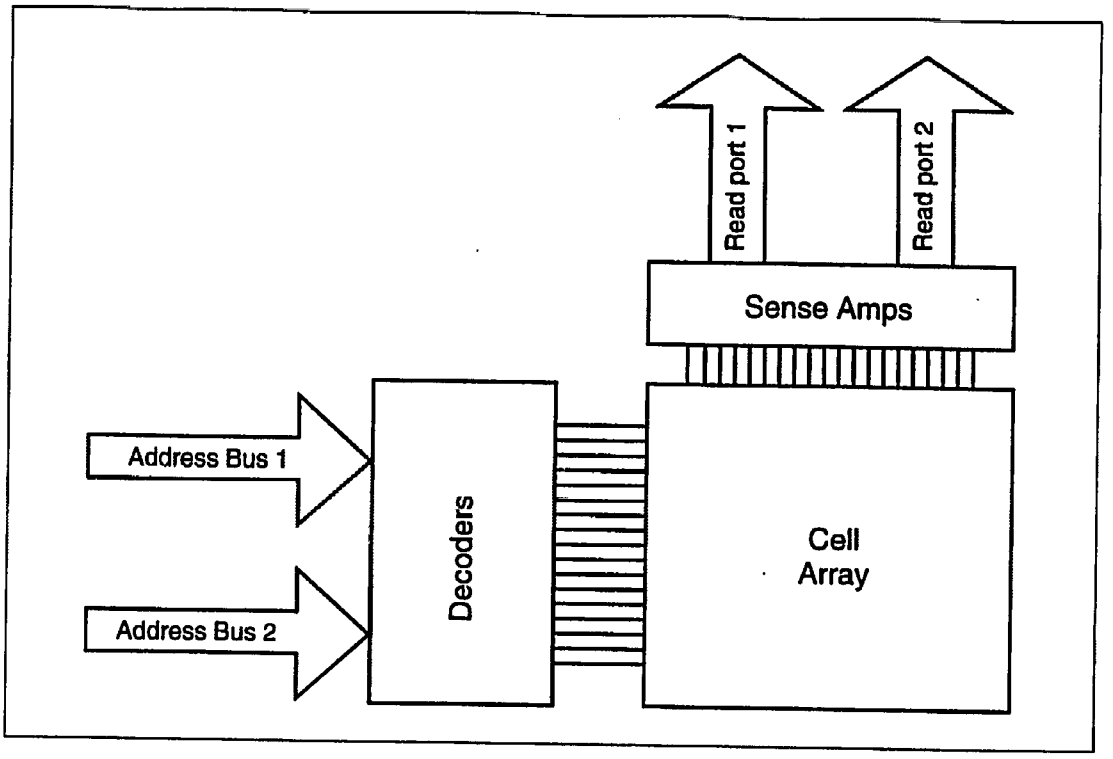


Figure 4.1: A Register File with Two Read Ports.

particular design. For example, if the design calls for a fast access time then the designer may use large drivers in the circuit. This is a trade-off between chip area and power to achieve fast access time. However, if the area of the chip is the main concern then the speed of the register file may be sacrificed by using smaller drivers.

4.3 Register File Components

The following sections briefly discuss various register file components and layout choices with respect to their operation and designs.

4.3.1 Memory Cells

The basic building block of a register file is a memory cell, a D-latch which stores data. The memory cells can either be static or dynamic. The dynamic memory cells store the data in a capacitance and require periodic refreshing. The static memory cells use two

cross coupled inverters and do not require refreshing. The dynamic memory cells are small and are well suited for dense VLSI; however, the periodic refreshing and lack of an active driver makes them much slower than static cells. Register files are accessed by the processor functional units directly. Each of the functional units may have a latency of the order of a few ns. Hence the register file needs to be fast to keep up with the data demands of the functional units. Static memory cells have more transistors than dynamic cells[1], consequently they occupy a larger area. But as the register files are usually small, of the order of 1 Kbit, the larger layout size of static cells is tolerable. In spite of their larger size, the advantage of static memory cells is the fast access time. We focus on static designs.

The most basic static memory cell is shown in the Figure 4.2. This uses six transistors. It has been used extensively in the design of fast memory structures [Hinds91, Holli78], for example, primary caches and register files. The cells uses the same port for the read and write operations. The cell read operation starts by pre-charging both the *bit* and $\overline{bit}$ line. Then the word select goes high and depending on the value stored in the cell either the *bit* or the $\overline{bit}$ line is pulled low. The write operation starts by putting the value to be written on *bit* and its complement on $\overline{bit}$ and then taking the word select high. The pre-charge is turned off during write cycle.

[1] A typical dynamic memory cell has 1 transistor and 1 capacitor while a typical static memory cell has at least 6 transistors.

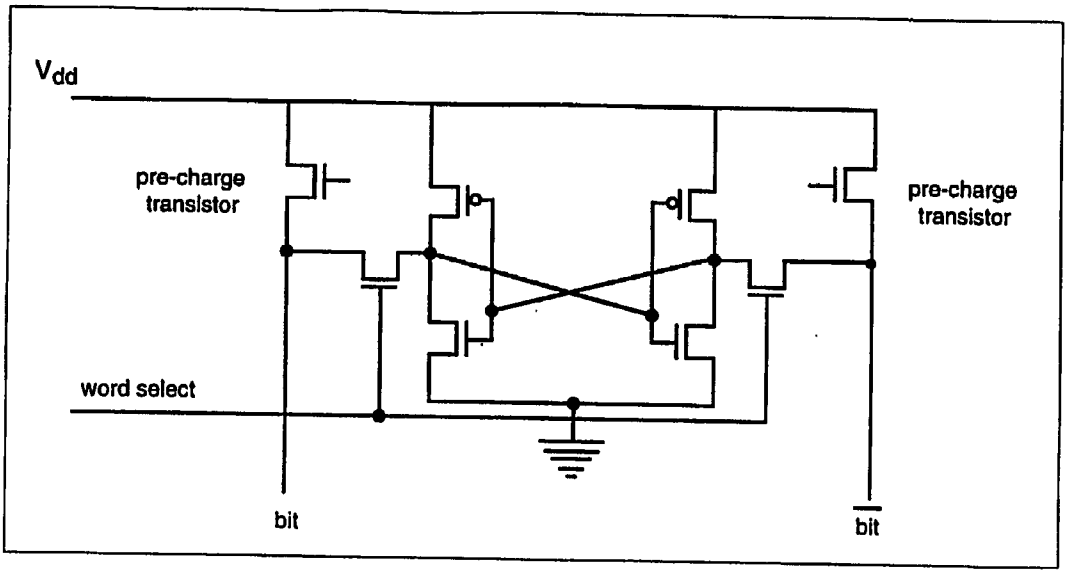


Figure 4.2: Static Cell Schematic.

The structure of the basic cell, requires 2 bit lines per port. This means that additional lines need to be routed on the chip, which increases chip area. The advantage of this cell is that fast, differential type, sense amplifiers like the clamped bit line sense amplifier [Blalo91] can be used at the outputs. They are among the fastest amplifiers presently available to circuit designers.

A minor modification to the basic cell of Figure 4.2 is to remove the $\overline{bit}$ line to obtain a more compact five transistor cell, but the fast differential type amplifiers can no longer be used with this cell. This results in a slower but more compact cell.

Faster cells are possible in other technologies. The fastest cells have been reported in ECL, for example, the 1 Kbit static RAM array in [Kawar78] and the 1 Kbit register file with an the access times of 0.85 ns in [Maly92]. However, the high power dissipation of the ECL requires special cooling techniques to be employed and makes its use in general purpose microprocessors impractical. Another fast technology is GaAs, and [Hinds91] presents the design of a 1 Kbit array implemented in GaAs with an access time of 2.5 ns. However, GaAs has limited device density and it may be a while before it is available in bulk for commercial applications. Another way to achieve the low power

consumption of CMOS and fast access times of bipolar technology is to combine the two. For example, [Tran88] presents a design where storage occurs in a CMOS latch and the sensing stages are built with bipolar devices. Another approach is to combine ECL level word-line voltage swings and emitter follower bit coupling with a static CMOS latch. An example of this technology is the CSEA (CMOS storage, emitter access) cell reported in [Yang88a] and [Yang88b]. Again, this requires special fabrication techniques and may not be suitable for general purpose low-cost processors.

Most of these exotic technologies have not realized their potential. In contrast CMOS improves at a surprising rate. The continued success of CMOS may be due, in no small part, to overwhelming investment that comes from dominant market share. Whatever the cause, it seems likely that CMOS will be the principal technology for implementing computers for the next five years, therefore our work will only consider CMOS designs. Our experiments are performed with a 1.2 micron CMOS process having two metal layers and one poly layer.

The cell shown in Figure 4.3 was used for the succeeding experiments. It was first used successfully used for the implementation of multi-port files reported in [Shin91, Maly92]. It is a modified six transistor CMOS structure with an additional buffer added at the read port. This modified cell offers three main advantages over the basic cell of Figure 4.2.

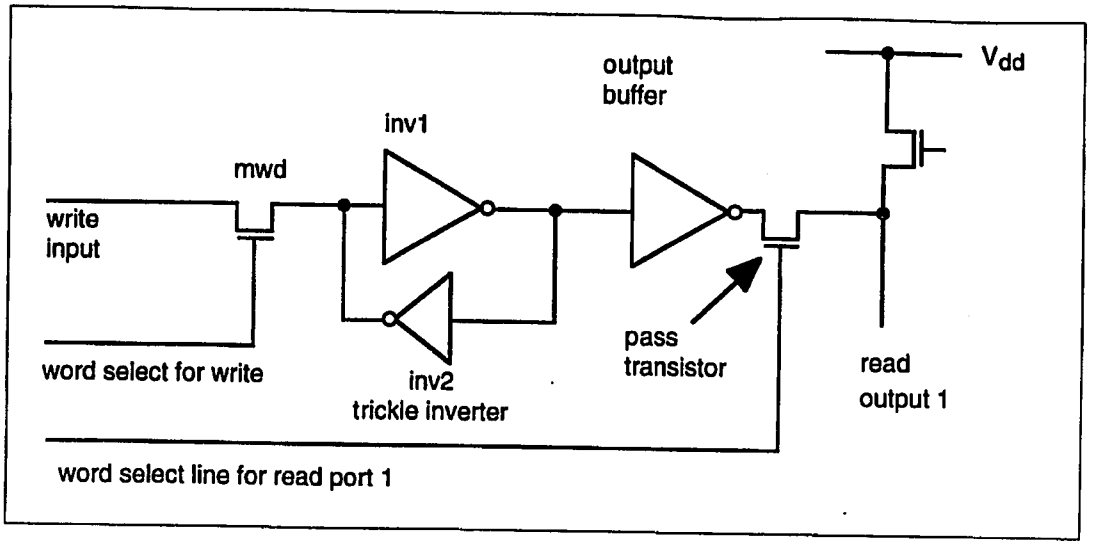


Figure 4.3: Memory Cell Suitable for Multi-Port Registers.

First, the number of read and write ports are independent of each other. Thus, we can have arbitrary number of read and/or write ports. A direct result of this read and write port independence is that read access time is independent of the write access time which depends on the ratio of the write enable transistor (mwd) to the N devices in the trickle inverter.

Second, the modified cell results in a smaller cell structure particularly in the case of a large number of read ports. The number of transistors in the basic cell of Figure 4.2 is $2 * \max(n, w) + 4$ for n read and w write ports, because the structure of the basic cell dictates that there be as many pairs of bit lines as the greater of the number of read or write ports. For the modified cell of Figure 4.3 the number of transistors is $n+w+6$ for n read and w write ports. The number of transistors for a basic cell with 3 read and 1 write ports for the basic cell is 10 and for the modified cell is 10. The addition of further read ports leads to the basic cell using more transistors than the modified cell. For example, for a basic cell with 4 read and 1 write port the basic cell has 12 transistors, while the modified cell uses 11 cells. If read port outnumber write ports by a ratio of 2:1 – a common situation – then the modified cell is more compact. In addition, register files

constructed from modified cells are smaller because only one signal line per port needs to be routed through the rest of the register file.

Third, in the basic cell of Figure 4.2 storage inverters are directly connected to all the read/write ports. These could pull down the voltage at the input of one of the cross-coupled inverter to a level which may turn off the N device (assuming that a '1' was stored in the cell) thus causing an erroneous value in the cell. The isolation buffer between the storage inverters and the read ports in Figure 4.3 eliminates potential erroneous operation.

The modified cell operation proceeds as follows. The write port operation is the same as in the basic cell. The only difference being, that this cell does not require both *bit* and $\overline{bit}$ for writing. Practically this means that the voltage level in the storage cell swings between 0 and about 4 volts, instead of 0 to 5 volts. (The N type transistor, mwd in Figure 4.3, is good at transmitting a logical '0' but not a logical '1' due to the voltage drop of approximately 1 volt across its source and drain). The write cycle is the same as the write cycle of the basic cell. The read cycle starts by pre-charging the output high and then selecting the port and register.

Finally, it should be noted that data transfer from the write input to the flip-flop is a ratioed logic operation. Data is written only if the drive of the write enable transistor is strong enough to overcome the drive of the N transistor in the trickle inverter. To make sure that the write operation is successful the beta ratio of the write enable transistor to the N transistor in trickle inverter was fixed at 10. However, the main inverter has a larger device for faster write operation. This makes the cross-coupled inverters of the basic cell asymmetrical in the modified cell. This is not the case in the basic cell as inputs to both inverters of the basic cell, bit and $\overline{bit}$, change simultaneously.

4.3.2 Decoders

Decoders are required for selecting one particular register for reading or writing. Decoders can be designed as a single level structure with either NAND or NOR gates each with n inputs for a register file containing 2^n registers. Or they can be designed as a tree level structure with 2 or 3 input gates at each level. A third possibility is to design a tree like structure with a fixed number of gate levels. Further they can be designed in an NMOS style with a P type load device, or static CMOS with n P devices (see Figure 4.4), or they can use any one of the various dynamic CMOS styles like domino logic.

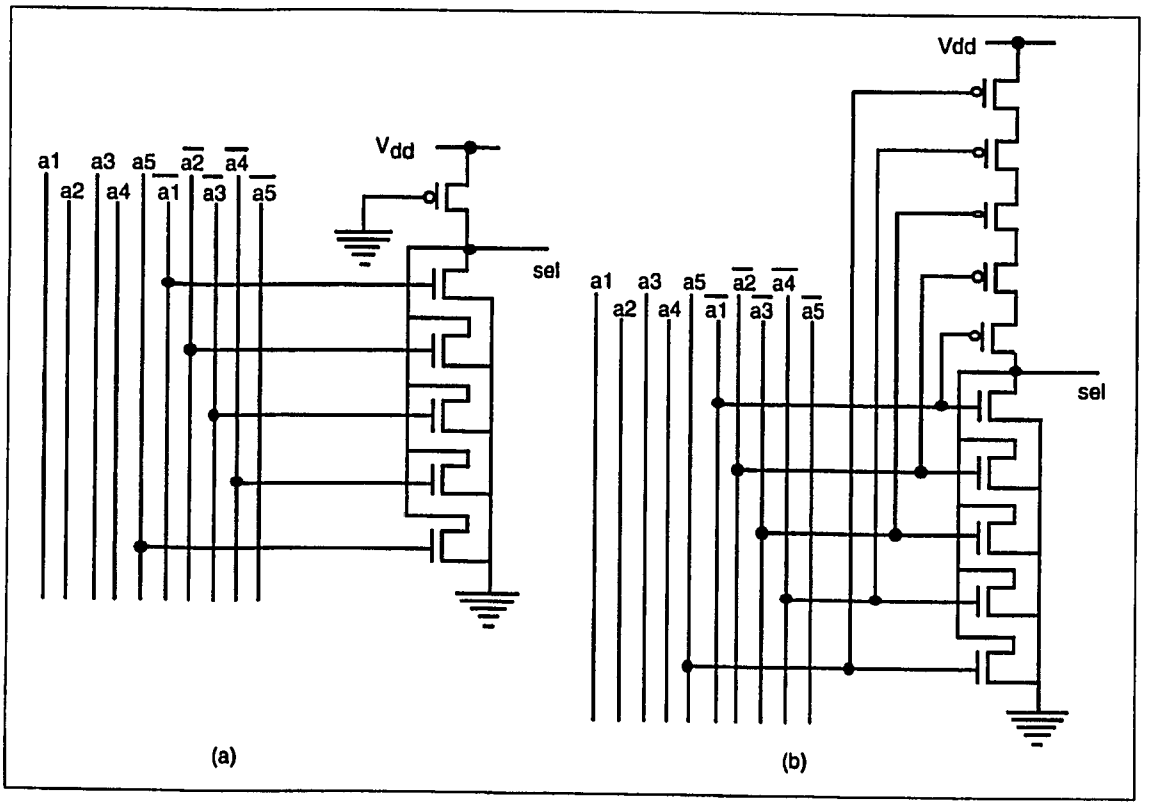


Figure 4.4 : A NOR Based Decoder (a) Pseudo-NMOS style (b) Static CMOS style The connection show a decoder for register 16.

The single level structures have a small footprint. They are well suited for dense VLSI. However, the large fanin required may cause large switching delays, and are thus

not recommended for use in large memory structures. However, a register file has a relatively small number of addresses, therefore the single level structure is quite suitable.

Pseudo-NMOS style layouts only use one P device and are thus compact. The negative side of using NMOS style layouts is static power dissipation. This is avoided in static CMOS style layouts but with some area penalty. The dynamic style CMOS is not discussed here because its complexity makes it unattractive for use in register files.

In our register file we used a static CMOS 5-input NOR gate for decoding purposes. This avoids the static power dissipation of the NMOS style layouts. The area penalty was not critical because the size of a static CMOS 5-input NOR gate is smaller than a single read port memory cell.

4.3.3 Sense Amplifiers

Sense amplifiers can be simple inverters with devices sized to have a fast rise time or they can be differential type which sense the difference between two signals (*bit* and $\overline{bit}$).

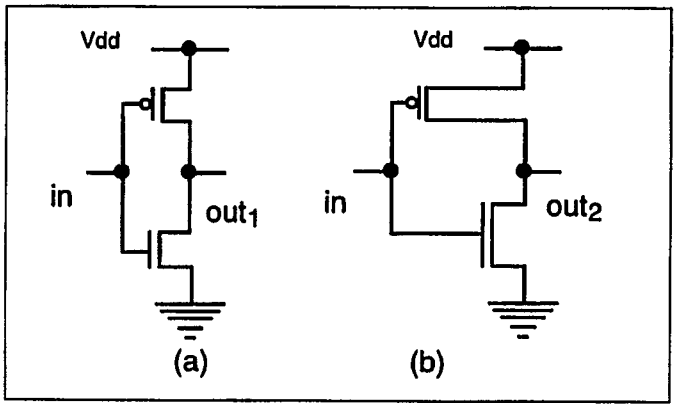


Figure 4.5: Sense Amplifiers (a) A Balanced Inverter (b) An Unbalanced Inverter.

The differential type sense amplifiers are fast and have a higher immunity to signal noise. However, as mentioned previously, because the memory cell of Figure 4.3

has only a *bit* line and no $\overline{bit}$ line, differential type sense amplifiers cannot be used. Therefore only unbalanced inverter type sense amplifiers are discussed here.

The sense amplifiers are connected to the bit lines. Because the bit lines are pre-charged, reading out a '1' from the cell is a fast operation since the output does not switch. Thus, reading out a '0' on the bit line is the determining factor in the delay since the bit line has to fall to '0' from the pre-charged value. A pre-charge to 5 Volts is assumed. This condition gives us a clue as to the design of the sense amplifier. It has to have a fast rise time at the output. The fall time is not important due to pre-charging. Thus, an inverter with a wide P device and long N device is called for (see Figure 4.5). Figure 4.6 contrasts the characteristics of a balanced inverter (out_1) and an inverter with wide P device and long N device (out_2). As can be seen the unbalanced inverter has faster rise time. The P and N device sizes are 1.2×25.2 microns and 1.2×12.6 microns respectively for the balanced inverter, and 1.2×106.8 and 1.2×12.6 microns respectively for the unbalanced inverter.

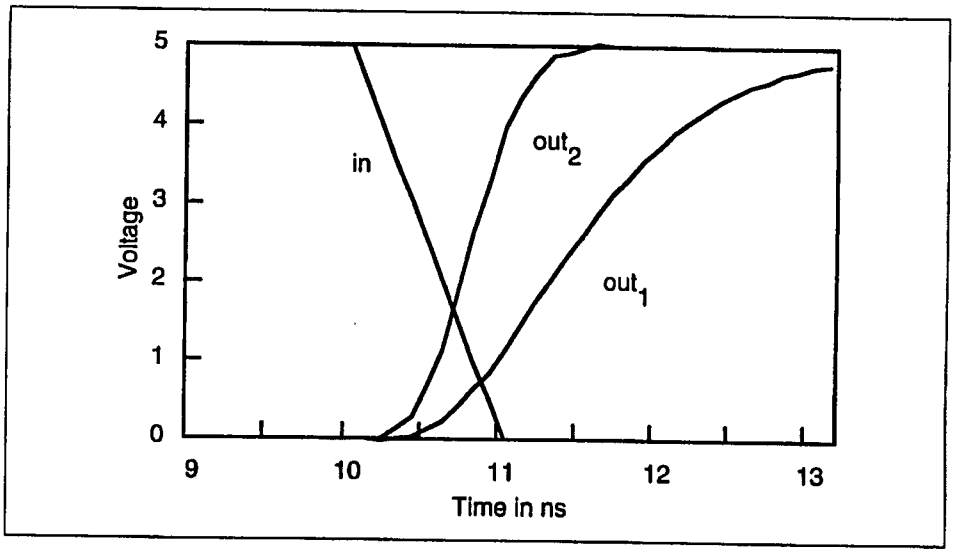


Figure 4.6: Response of a CMOS Balanced Inverter (out_1) and Inverter with a Wide P Device and Long N Device (out_2).

4.4 Design Considerations for the Memory Cell

Initially, the memory cell was designed with a fixed device size for decoders, pass transistors, output buffers, and sense amplifiers. All these components interact with each other and this initial design needs tuning to obtain the optimum overall access time for the cell. To find the optimum combination of these components for the cell we ran some experiments. These involved fixing the device sizes in all but one of the components and then varying the device size(s) in it alone. Ideally, all the device sizes should be varied simultaneously to find an optimum, but our approach was necessary to limit the search space. The results of these experiments are summarized in the next few paragraphs.

The access delay in the following paragraphs refers to the delay incurred from the time an address is provided to the decoder to the time the output of the register is available at the output of the sense amplifiers. More specifically, the access delay of a register file consists of the delays in decoding the address of a particular register (refer to Figure 4.1), enabling that particular register, sensing the output of the register and charging the output capacitance of the data bus.

Experiment 1 consisted of finding an optimum size for the decoder devices. Initially, the decoder was designed with the smallest possible devices (1.2 microns by 2.4 microns). Once the functionality of the decoder was validated the device sizes in the decoder were varied and the decoder delay was measured for each device size. We are interested in optimizing the decoder for fast rise time, since the memory cells are selected when decoder output is high. The P devices are responsible for pulling the output high, therefore, in experiment 1 only the widths of the P devices of the decoder were varied. The results of these measurements are shown in Figure 4.7. After experimentation we picked a P device width of 14.4 microns. This choice is based on the graph of Figure 4.7, which shows that the point of diminishing returns is at P device width of 14 microns.

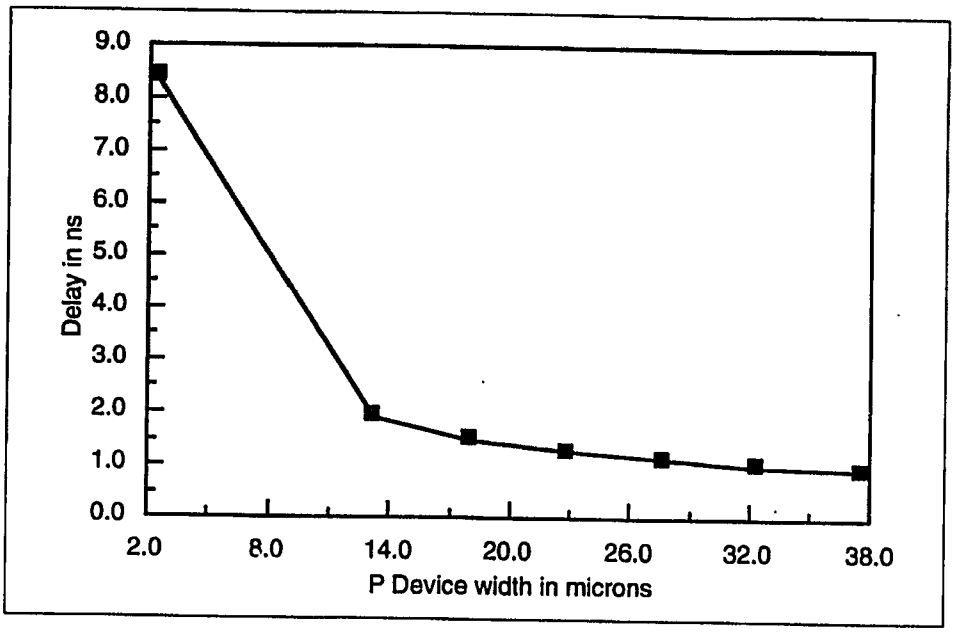


Figure 4.7: Variation of the Decoder Access Time Delay with P Device Width.

Sense Amp. Design	N device size W × L	P device size W × L	Sense Amp. Area
1	3.0 × 1.2	28.2 × 1.2	16.4 × 26.0 = 436.8
2	4.8 × 1.2	43.2 × 1.2	19.2 × 31.0 = 595.2
3	6.0 × 1.2	57.6 × 1.2	21.6 × 36.0 = 777.6
4	7.8 × 1.2	72.0 × 1.2	24.0 × 41.0 = 984.0

Table 4.1: Table Describing Various Sense Amplifiers (all dimensions in microns).

Experiment 2 involved studying the relationship between the size of sense amplifier devices and the access time of the register file. The pass transistor width was fixed at 2.4 microns, and the P device width in output buffer was fixed at 9 microns. Various device sizes of the sense amplifier transistors were tried, given in Table 4.1. The resulting variation in the access time is shown in Figure 4.8.

The graph of Figure 4.8 shows the access time vs. sense amplifier size for 6 configurations of register file. The number of read ports ranges from 1 to 6, and the number of write ports is fixed at 1 (denoted by RxW1 with x =1,...,6). The graph shows that increasing the size of the sense amplifier improves the access time but the improvement is only weakly related to the increase in the area of the sense amplifier. Sense amplifier 4 is 2.25 times the area of sense amplifier 1, but the access time increases only by about 30%. Increasing the size of the sense amplifier further should decrease the access time and a final choice for the size of sense amplifier device may depend on the chip area available at the time of fabrication. We use sense amplifier 4 for the design of register files in later sections.

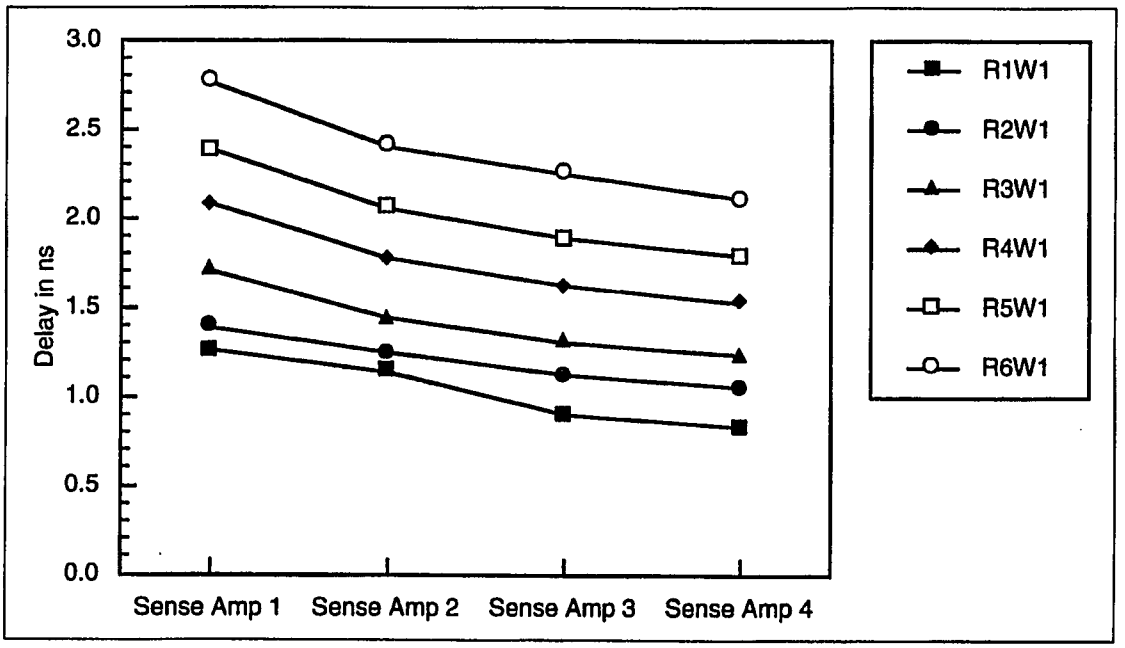


Figure 4.8: The Effect of Increasing Sense Amplifier Device Size on Access Time Delay.

Experiment 3 studied the effect of varying the size of the pass transistor and its effect on the access time of the register file. Sense amplifier 4 of experiment 2 was used for this experiment. The P device width in output buffer was fixed at 9 microns. Also, the

experiment was done only for one register file configuration, a single port memory. This should not effect the validity of the results for other configurations, because each of the output ports in a multi-port cell has its own pass transistor. The effect of varying the width of the pass transistor is shown in Figure 4.9.

The graph of Figure 4.9 shows that changing the width of the pass transistor does not have much effect on the access time (it varies between 1.13 to 0.91 ns). After a small initial drop the access time becomes almost constant. This is due to the fact that increased drive of the larger transistor is offset by the larger input and output capacitances of the device. We chose a 14.4 micron width pass transistor in our experiments, as this is a convenient point of diminishing returns. For example, increasing the pass transistor width from 9 to 14.4 causes a reduction in access time of 11% while increasing the size from 14.4 to 21 causes a reduction of only 3.3%. The 14.4 micron device also fits well with other devices in the memory cell layout.

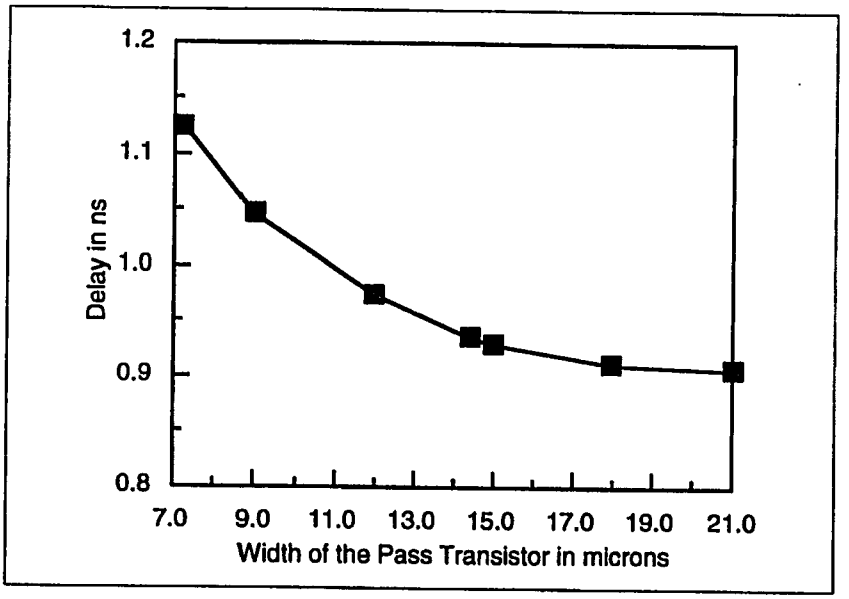


Figure 4.9: The Effect of Increasing Pass Transistor Size on Access Time Delay.

Experiment 4 studied the effect of the output buffer size on the access time. The pass transistors and the sense amplifier sizes were fixed. The pass transistor size was

fixed at 14.4 microns as determined in the experiment 3, and sense amplifier 4 of experiment 2 was used for this experiment.

As can be seen in the graph of Figure 4.10, the increase in the width of the P device in output buffer has a dramatic effect on the access time. This is especially true for the register files with the larger number of read ports. For instance, the decrease in delay for a 6 read port file is 1.35 ns as the width of the output buffer P device increases from 9 microns to 24 microns (a factor of 0.55 drop in delay). The corresponding decrease for a 2 port file is 0.35 ns (a factor of 0.85 drop). The figure also shows that the decrease in delay drops off after the width of P device has increased beyond a certain width. For example, in Figure 4.10 the decrease in delay of a 2 port register file after the size of 18 microns is minimal. The points of diminishing

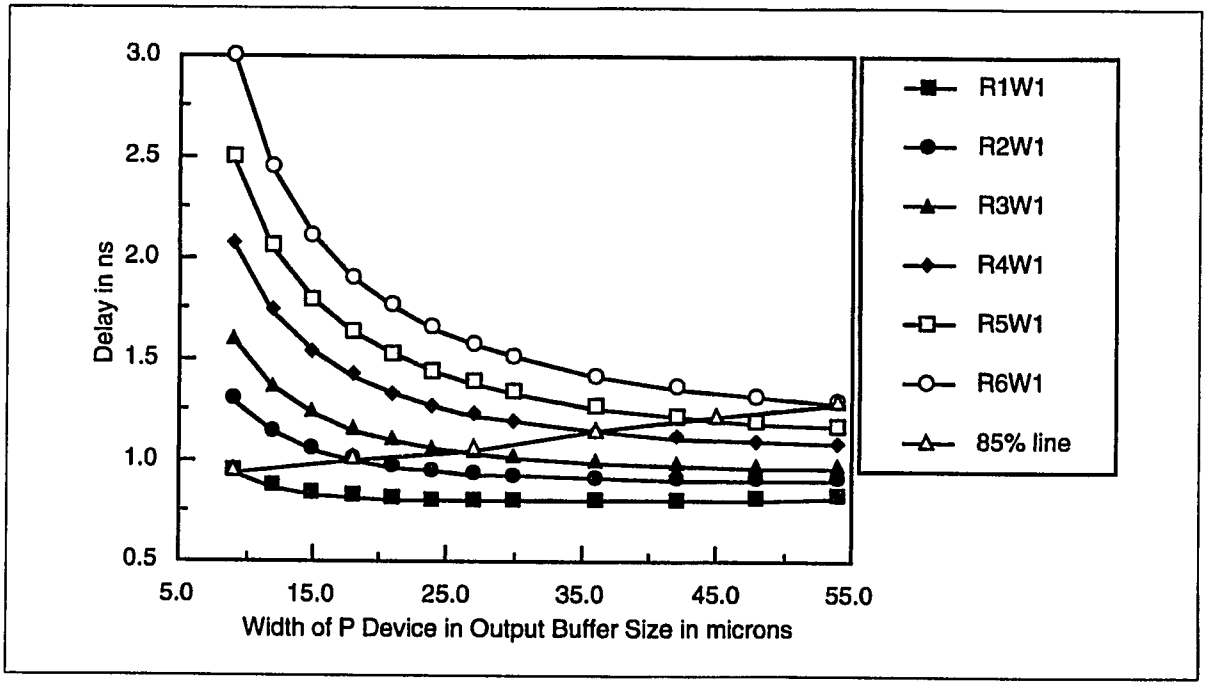


Figure 4.10: The Effect of Increasing Output Buffer Size on Access Time Delay.

returns are represented on the graph by the 85% line. It connects the point on each curve (i.e., each configuration) where the delay has fallen to approximately 85% of the way to its minimal value. It is calculated by the following formula

$$85\%line = M - 0.85(M - m) \tag{4.1}$$

where, M is the delay with a P device width of 9 microns and m is the minimum delay for each configuration, or, the delay with a P device width of 54 microns, whichever is lower.

Increasing the size of the buffer, increases the current drive of the buffer. This causes the capacitive loading at the output of the buffer to be charged quickly, thus reducing the delay. As we increase the size of the output buffer, we expect a decrease in the cell delay. However, the graph in Figure 4.10 shows that the cell delay levels off. This is due to the fact that increasing the buffer size, is not free, and the cost is the increased size of the cell. This increase in cell size causes overall register file dimensions to grow. Thus, the metal lines carrying the output of the cells tend to grow longer. The capacitive loading of a metal line is directly proportional to its length, therefore, increasing the size of the metal line causes an increase in the capacitive loading at the output of the buffer, which negates in part the increased drive of the buffer.

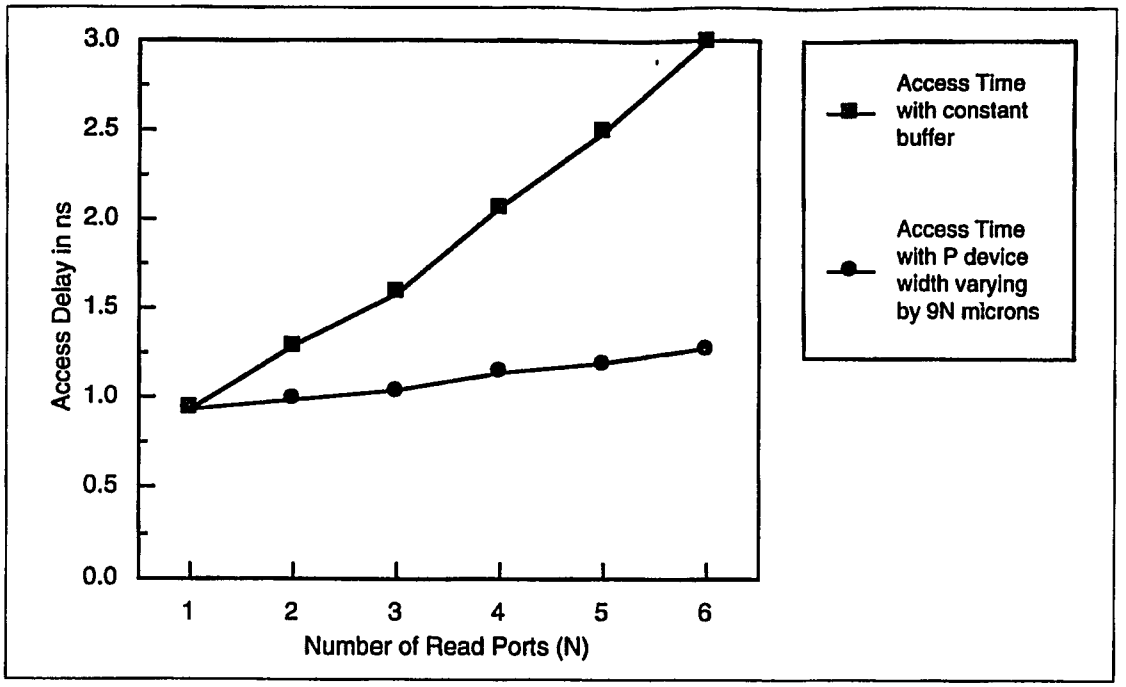


Figure 4.11: Access Time Delay vs. the Number of Read Ports.

Figure 4.10 shows that there is an advantage in increasing the size of the buffer as we increase the number of the read ports on a register file. It also shows that delay of a N+1 read ports file will never be lower than an N read ports register file for equal buffer sizes. Also, this figure suggests that the buffer size should be a function of number of ports, increasing with the number of ports to offset increased delay. In order to find a suitable function relating number of ports to the output buffer size we selected the buffer size for each of the configurations at the point beyond which the reduction of delay is minimal. The 85% line was used to define these buffers. The data suggests that a buffer size of about $9N$ microns, where N is the number of read ports, is the appropriate function. This function slightly underestimates for the 1 port register file.

We note that the buffer sizes predicted by the function, $9N$, are quite large for the registers with large number of ports, but if the buffer size is not scaled according to the number of ports then the resulting register file will have a much larger access time. Thus,

our model may be taken as a lower bound on register file access time. Figure 4.11 shows the effects of scaling vs. holding the buffer size fixed. The upper curve in this figure shows the increase in the access time as the number of the read ports is increased from 1 to 6 for a constant buffer size of 9 microns. The lower curve shows the same relationship but with the output buffer size increasing with the increase in the number of read ports. The figure shows that scaling provides considerable improvement in the access time, e.g., for a 4 port register file the improvement is about 40%. However, the access time still increases 28% as the number of ports goes from 2 to 6. Thus, scaling cannot completely counteract the slowing caused by adding ports to a register file.

Figure 4.11 makes an important point central to this thesis, that micro-architectural features which may improve performance by allowing greater parallelism may also slow the operation of the system by increasing critical delays. The top curve in Figure 4.11 shows that in the case of multi-ported register files this can be dramatic. The bottom curve shows that no matter how much we try to counteract the adverse effects of multi-porting by scaling, significant slowing will occur. Thus, we argue, it is in some sense a necessary consequence.

Figure 4.12 shows the area penalty that results from scaling the buffers. As can be seen, the area penalty is minimal. The height of the cell is such that only a small increase in width of the cell provides enough additional area to increase the widths of the buffer devices sufficiently.

4.5 The Development of Register File Macro-Models

As we have seen the access time for the register file slows down as the number of read ports is increased, while its area increases. The above experiments were performed for a limited number of read ports (1 to 6). To determine the effects of increasing number of ports on area and delay, for a larger number of read ports than 6, macro-models relating area and access time to number of ports were derived.

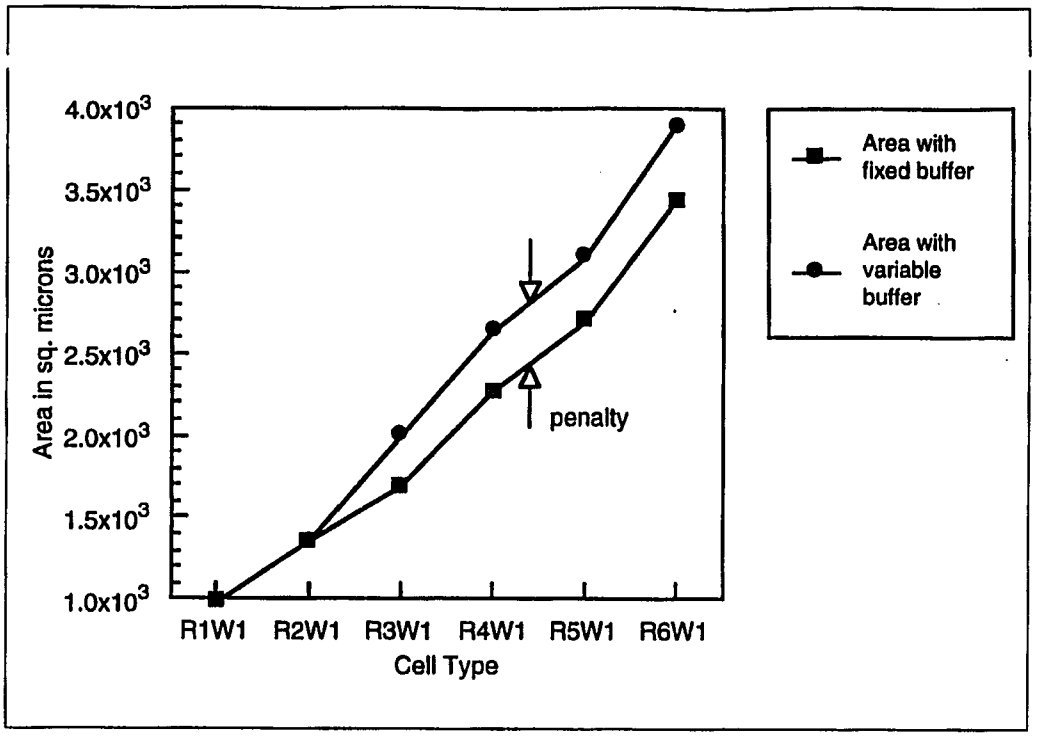


Figure 4.12: Area Penalty due to Buffer Scaling.

4.5.1 Macro-Model for Cell Area

Total area of a register file is the sum of the area for the cell, decoders and the sense amplifiers. The general model for the area of a register file is therefore,

$$Area = decoder\ area + cell\ area + sense\ amplifier\ area \tag{4.2}$$

4.5.1.1 Decoder Area

The decoder area is the sum of all the gates that form the decoder plus the area occupied by the address lines. In our designs the address lines ran along the length of the decoder. The number of gates in the decoder equals the number of registers. The number of the address lines on the input to gates equals twice the $\log_2$ of the number of registers (twice because we need to route both the address lines and their complements). The area occupied by decoder is then given by

$$A_d = R * A_g + 2 * \log_2(R) * M_p \tag{4.3}$$

where, A_d is the area of decoder, R is the number of registers in the register file, A_g is area of one gate in decoder with $\log_2(R)$ inputs, M_p is the pitch of the first metal layer in the process. The model of (4.3) gives the area of one decoder. For an N port register file the total area is $N*A_d$.

The area is derived from the cell geometry with the size of the P device as a parameter in the equation. The area of one gate in the decoder is given by

$$A_g = xy + y_\Delta x(P_{gw} - P_{\min w}) / \Delta P \tag{4.4a}$$

where xy is the area of the decoder gate with a minimum P device, y_Δ is the increment in the cell height per ΔP increase in P device width, P_{gw} is the width of P device in the decoder being modeled, $P_{\min w}$ is the minimum P device width, and ΔP denotes the fixed incremental change in P device width; it is set to 6 microns. The value of y_Δ was determined to be 1.8 microns. The area is in sq. microns. The reason for a larger increase in the P device width compared to the increase in cell height is that the poly gate is snaked in the layout and a small increase in the cell height causes a large increase in the P device width.

The shortcoming with (4.4a) is that it does not take into account the number of inputs to the gate (it assumes 5 inputs, for decoding addresses of 32 registers). To make the model more general the following relation was derived which includes the number of inputs to the decoder as a parameter in the model. For a gate with $2+n$ inputs the area is given by:

$$A_g = (x + C_1 n\lambda) * (y + C_2 n\lambda) + y_\Delta x\left(P_{gw} - P_{\min w}\right) / \Delta P \tag{4.4b}$$

where this time x is the width and y is the height of a 2-input decoder gate with a minimum P device width $P_{\min w}$, n is the number of additional inputs to the gate, λ is the

minimum feature width of the layout process, and ΔP is 6 as before. The constants C_1 and C_2 in (4.4a) and (4.4b) are process dependent and were derived from the following process design rules.

i) min. poly to metal 1 spacing (= 4λ)
ii) min. poly to poly spacing (= 2λ)
iii) min. overlap of metal to active region around a contact to active (= 4λ)
iv) min. metal width (= 3λ), and
v) min. metal to metal spacing (= 3λ)

Based on these design rules C_1 = (i) + (ii) + (iii) and C_2 = (iv) + (v). Similarly, in (4.3), the constant M_p = (iv) + (v).

The model of (4.4b) is compared to the actual layout area of the decoder gate for various P device widths in the graph of Figure 4.13. It shows that the model predicts the area of a decoder quite accurately.

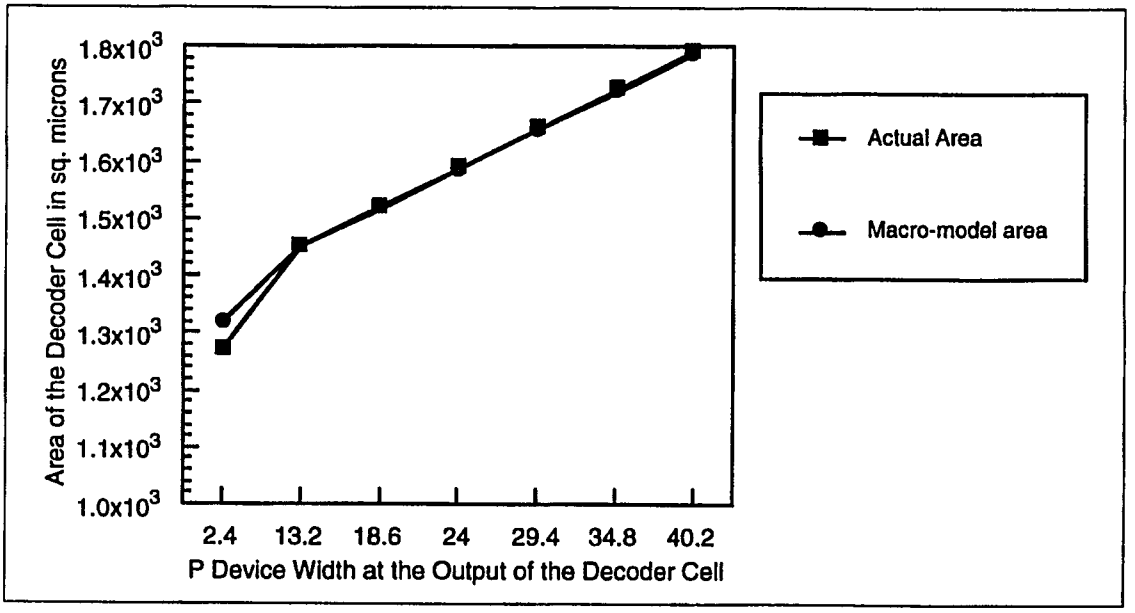


Figure 4.13: Comparison of the Macro-Model to the Actual Area for Decoder Cell.

4.5.1.2 Cell Area

The macro-model for the cell area was derived based on the geometric scheme used to expand the cells as the number of read ports increases. The expression for the macro-model is given in (4.5).

$$Area = x_1 y_1 + 27.72(n-1)^2 + 4.2 y_1 (n-1) \tag{4.5}$$

where x_1 is width of cell with 1 port (24 microns), y_1 is height of cell with 1 port (41 microns), n is the number of ports, 27.72 (4.2 × 6.6) is a constant depending on the design rules, the value 4.2 is the minimum spacing between first metal layer (1.8) + minimum feature width of first metal layer (2.4), and the value 6.6 is 2 * minimum overlap of the active region over the poly (1.8) + minimum poly width (1.2) + minimum spacing between active regions (1.8). The area is in square microns. The design parameters are shown on a cell layout in Figure 4.14.

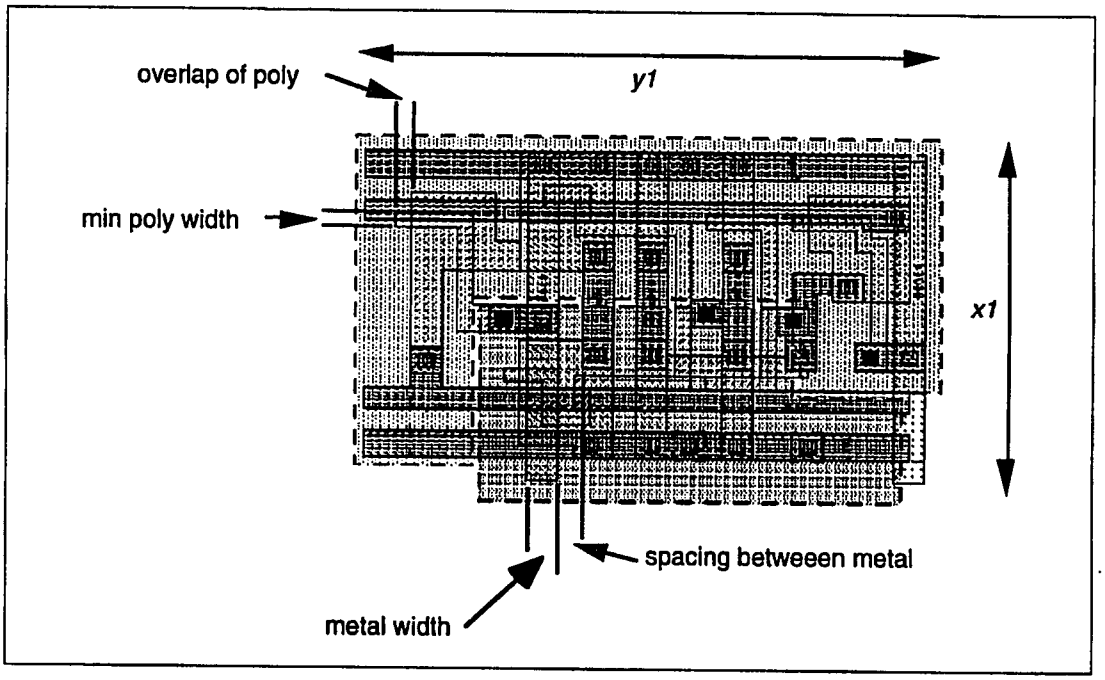


Figure 4.14: Cell Showing Various Parameters Used in the Macro-Model for Area.

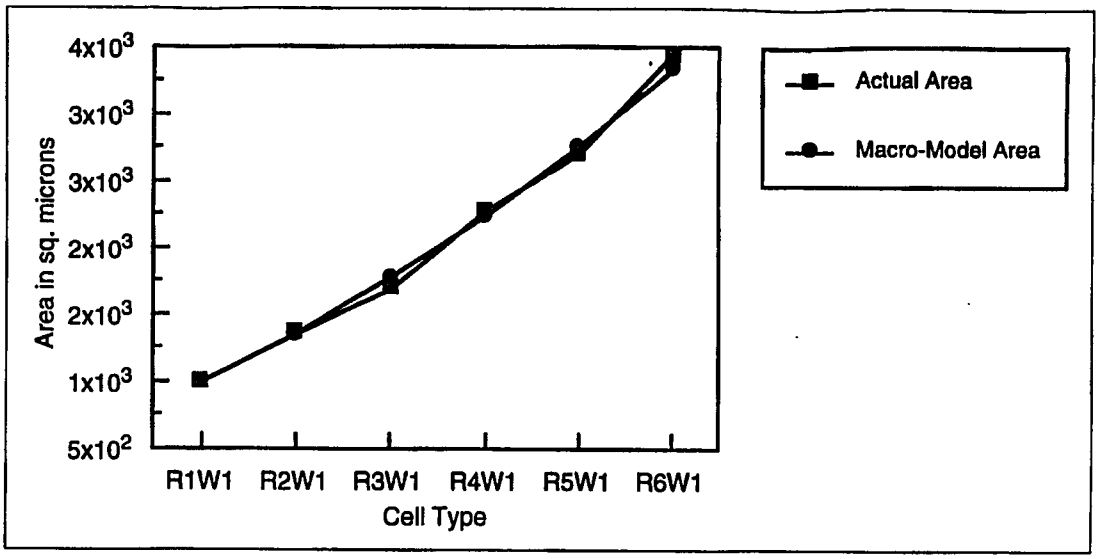


Figure 4.15: Macro-Model Predictions vs. the Actual Cell Area.

The area predicted by this model is compared in Figure 4.15 to the actual areas of various layouts of the cell. It can be seen in Figure 4.15 that the predicted area line moves back and forth across the actual area line. This is because it takes less area to add a port when the cell already has an even number of ports. Addition of an even numbered port requires the addition of another active area which has to be 3 lambda apart from the existing active area. This is not required for the addition of odd numbered port.

Finally, the graph of Figure 4.15 shows that there is a considerable penalty in adding ports in terms of area. For example the register file size almost quadruples for an increase in number of read ports from 1 to 6. The area for a 16 port file is 12 times the area of a single port register file.

4.5.1.3 Sense Amplifier Area

The sense amplifier area depends on the particular type of sense amplifier used. We used a simple unbalanced inverter (shown in Figure 4.5) as the sense amplifier. This sense amplifier occupies an area of 21 × 36 microns.

4.5.1.4 Verification of Area Model

The area of the register file was calculated using (4.2). It was verified by comparing the predictions of the model to the actual area occupied by various register files. The results are presented in Table 4.2. The model predictions are within 5% of the actual area of the register file. The model does not account for the area wasted between the cells which accounts for the some of the negative errors in the table.

In addition, various published register file designs were compared to the model predictions. The data is presented in Table 4.3. In computing the model area for these files a scaling factor was used to account for the difference in the minimum feature size of different technologies. This scaling factor is $(\delta/1.2)^2$ where, δ is the smallest possible feature size in the technology in which the register file was constructed and 1.2 is the feature size of our process. In calculating the model predictions for each of these cells, we used the size of the memory cell (24×41) that we constructed as the basis for all the calculations. The agreement in this table is not as good as in Table 4.2 but considering the lack of published details about the files it is still quite good.

Register File Configuration	Actual Area	Predicted Area	Error
R1W1	1,162,268	1,137,930	-2.09%
R2W1	1,689,860	1,635,164	-3.24%
R3W1	2,180,895	2,189,169	0.38%
R4W1	2,923,712	2,799,944	-4.23%
R5W1	3,516,861	3,467,489	-1.40%
R6W1	4,422,616	4,191,805	-5.22%

Table 4.2: Actual Area vs. the Predicted Area (in square microns).

Register File	Number of Ports	Number of Registers	Actual Area	Model Area	Error
MIPS-X [Horow87]	2	32	3,330	3,933	+18.0%
DEC 3 [Joupp89c]	2	48	3,534	3,322	-5.0%
HP 1 [Yette87]	4	31	3,450	3,511	+2.0%
GE 1 [Molne89]	4	8	4,760	4,583	-3.7%

Table 4.3: Various Published Register Files and the Area Macro.

4.5.2 Macro-Model for Cell Delay

This macro-model relates the number of read ports to the delay of the cell. Later, the number of registers is also made a parameter of the model. The macro-model for the decoder was derived from the circuit of Figure 4.15. The memory cell delay for a read operation was derived from the structure shown in Figure 4.16. The equivalent circuit of Figure 4.16 is derived with the assumption that the voltage at the input to pass transistor is constant. This follows from the fact that the value stored in the cell (which is connected to the pass transistor) has to be a constant value during the whole read cycle to ensure a reliable output (for both a logic '1' or logic '0'). The resulting network structure has one gate of a decoder (5-input NOR), one inverter and a pass transistor between the output and the input. The total delay in the cell is the sum of delays as the signal propagates through these three components, i.e.,

$$Delay = \Delta_1 + \Delta_2 + \Delta_3 \tag{4.6}$$

where, Δ_1 is the delay in the sense amplifier, Δ_2 is the delay in pass transistor, and Δ_3 is the delay in decoder.

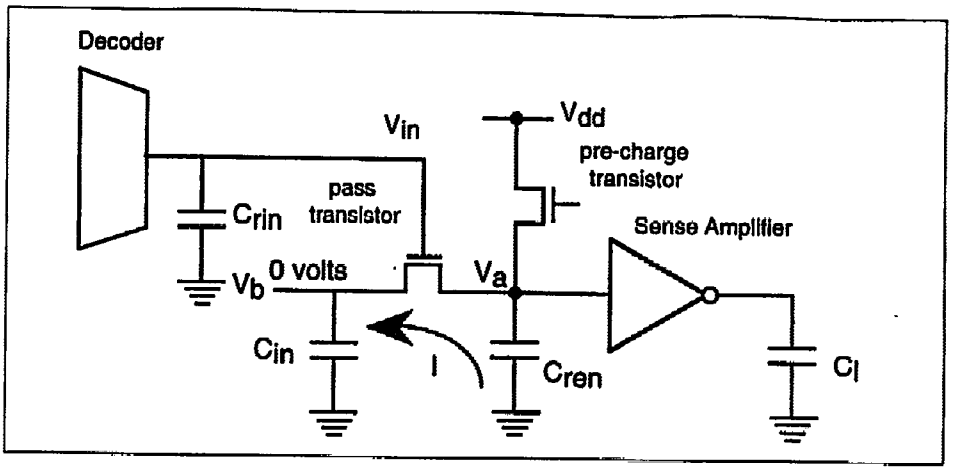


Figure 4.16: Structure for Macro-Model.

An accurate macro-model of a CMOS inverter is presented in [Kayss92] and given by

$$\Delta_i = T_i\left(a_o + \frac{a_1}{T_i\sigma} + \frac{a_2}{(T_i\sigma)^2} + \frac{a_3}{\sqrt{T_i\sigma}}\right) \tag{4.7}$$

where Δ_i is the inverter delay, a_o, a_1, a_2, a_3 are constants with values -0.873, 278.895, 5341.5 and -30.129 respectively, T_i is the input time constant, and $\sigma = \frac{K_1 V_{dd}}{C_L}$ is the capability of the inverter to drive a capacitive load. Further, $K_1 = \beta_p \frac{W_p}{L_p}$ where, W_p and L_p are the width and length of P device in the inverter, $\beta_p = \frac{\mu_p \varepsilon}{t_{ox}}$, t_{ox} is the oxide thickness, and μ_p is the hole mobility in the P device. We focus on the P device of the inverter because it is responsible for pulling the output high when the input goes from the pre-charged high to low. The C_L in the macro-model varies from 60 fF to 500 fF, which corresponds to the load on the data bus. The input time constant varies from 2 to 10 ns. The curve fit is shown in Figure 4.17. The fitting curve is shown as a line while the data is represented by the square dots.

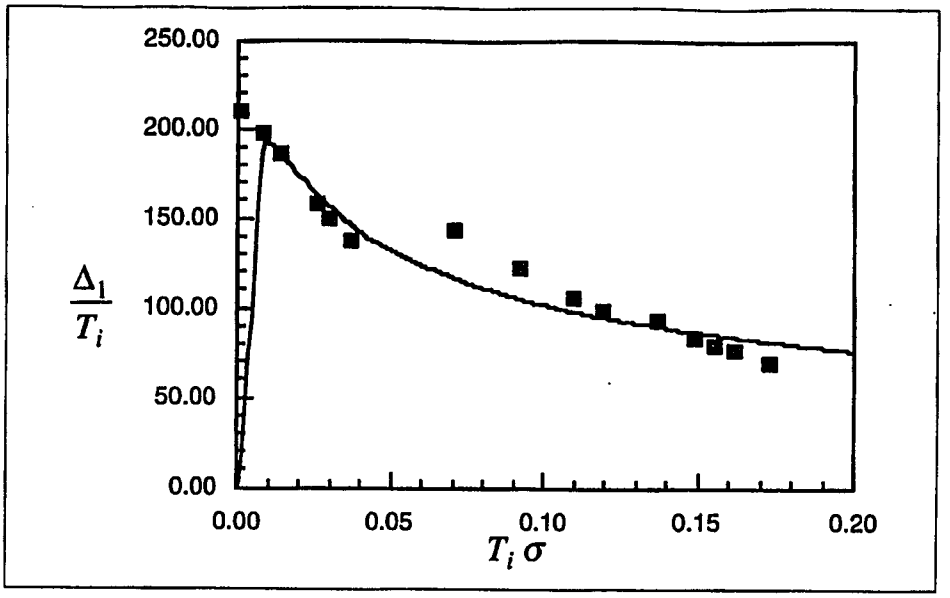


Figure 4.17: Inverter Curve Fit – Equation 4.7.

The delay for the pass transistor is calculated next. This also follows the analysis techniques applied in [Kayss92]. Initially, in the circuit of Figure 4.16, when the pre-charge transistor is on and pass transistor is off, V_a is given by

$$V_a = V_{dd} - V_{tn} \tag{4.8}$$

As V_{in} begins to change, the pass transistor starts to discharge the capacitance C_{ren} and the current through the pass transistor is given by

$$I = C_{ren}\frac{d(V_a)}{dt} = \begin{cases} 0 & V_{in} - V_{tn} \le 0 \\ 0.5K(V_{in} - V_{tn})^2 & V_{in} - V_{tn} \ge 0, \\ & V_{in} - V_a - V_{tn} \ge 0 \\ 0.5K(V_{in} - V_{tn})^2 - (V_{in} - V_a - V_{tn})^2 & V_{in} - V_b \ge 0, \\ & V_{in} - V_a - V_{tn} \le 0 \end{cases} \tag{4.9}$$

V_b is the voltage at output of the memory cell output buffer. In Figure 4.16, it is assumed that a logic '1' is stored in the cell, therefore V_b is at 0 volts. From (4.8) and (4.9) the delay in the pass transistor can be expressed as

$$\Delta = F(V_{dd}, V_{tn}, C_{ren}, K, T_{in}) \tag{4.10}$$

This functions has six variables and 3 units (volts, amperes, and seconds). By applying Buckingham's Pi-Theorem [Bucki15], (4.10) can be expressed in terms of 3 (= 6 - 3) dimensionless quantities. A possible form of the reduced equation is

$$\frac{\Delta}{T_{in}} = F\left(\frac{V_{tn}}{V_{dd}}, \frac{C_{ren}}{KV_{dd}T_i}\right) \tag{4.11}$$

This can be further simplified if we note that $\frac{V_{tn}}{V_{dd}}$ is constant for a given process, therefore the equation can be written as

$$\frac{\Delta}{T_{in}} = F\left(\frac{C_{ren}}{KV_{dd}T_{in}}\right) \tag{4.12}$$

The capacitance values determined experimentally for the various register files were plotted vs. the number of read ports and a relationship was derived by fitting a curve. The relationship is a straight line and the capacitance equation is of the form

$$C_{ren} = e'N + g' \tag{4.13}$$

where e' and g' are constants and N is the number of the read ports. When the number of registers in the register file is added as a parameter then the equation becomes

$$C_{ren} = eNR + gR \tag{4.14}$$

where e and g are new constants, and R is the number of registers. We found the values of e to be 1.49 and g to be 12.28 from our experiments. The input capacitance was related to the number of ports by

$$C_{in} = hN \tag{4.15}$$

The value of h was calculated to be 3.5, and the value of C_{in} is in fF. The load capacitance was assumed to be fixed at 1 pF. In an actual circuit this would depend on the length of the data bus.

Finally, after experimenting with different fitting functions, the following relation gives a good fit to the simulation data

$$\Delta_2 = T_i\left(b_o + \frac{b_1}{T_i\sigma} + \frac{b_2}{(T_i\sigma)^2} + \frac{b_3}{\sqrt{T_i\sigma}}\right) \tag{4.16}$$

where, Δ_2 is the pass transistor delay, b_o, b_1, b_2, b_3 are constants with values 0.729, 5.735, -1.515 and -2.889 respectively, T_i is the input time constant, and, $\sigma = \frac{K_2(V_a - V_b)}{C_{ren}}$ is the capability of pass transistor to drive a capacitive load. Further $K_2 = \beta\frac{W}{L}$ similar to before, and $\beta = \frac{\mu_n \varepsilon}{t_{ox}}$. The C_{ren} in the macro-model varies from 100 fF to 1000 fF which corresponds approximately to a range of 7 to 81 registers of 2 ports each. The input time constant T_i varies from 2 to 10 ns, and the pass transistor width varies from 2.4 micron to 37.5 microns. The curve fit is shown in Figure 4.18.

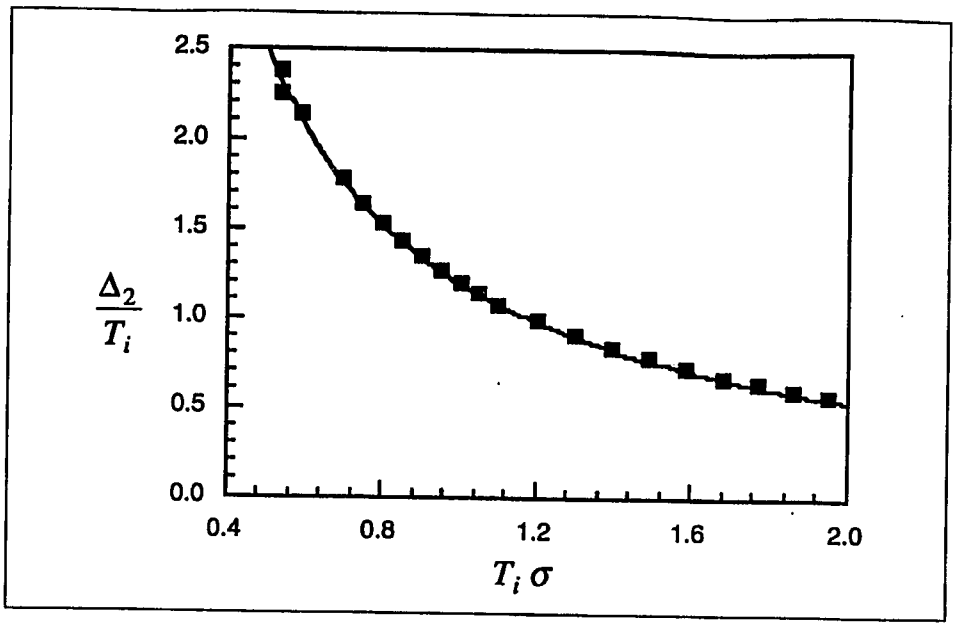


Figure 4.18: Curve Fit for the Pass Transistor.

Using a similar procedure the decoder macro-model was derived as follows:

$$\Delta_3 = T_i\left(c_o + \frac{c_1}{T_i\sigma} + \frac{c_2}{(T_i\sigma)^2} + \frac{c_3}{\sqrt{T_i\sigma}}\right) \tag{4.17}$$

where $\Delta_3$is the decoder delay, c_o, c_1, c_2, c_3 are constants with values 2.135, 37.545, -24.376 and -9.144 respectively, T_i is the input time constant, and $\sigma = \frac{K_3 V_{dd}}{C_{rin}}$ is again capability of pass transistor to drive a capacitive load, etc. The C_{rin} is the load capacitance on the decoder output. It depends on the number of ports and number of bits in each of the registers in the file. The relationship was found to be

$$C_{rin} = jB + kNB \tag{4.18}$$

where j is 20.72, B is the number of bits in the register, k is 5.43 and N is number of ports in the register file. (4.17) is valid for a C_{rin} range of 200 fF to 600 fF. This corresponds to approximately 18 bits to 57 bits for a 2 port register file with 32 registers. The input time

constant T_i in (4.17) varies from 2 to 10 ns and the pass transistor width varies from 2.4 micron to 37.5 microns.

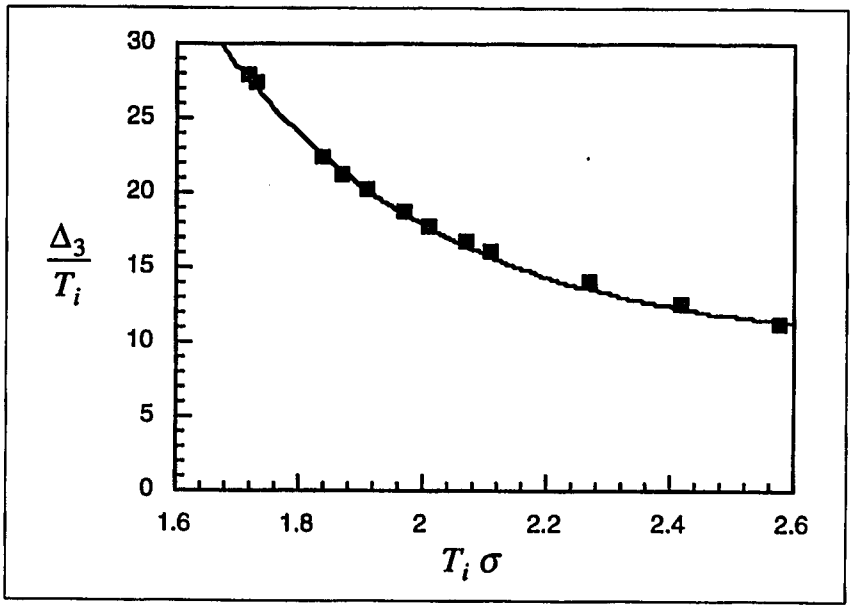


Figure 4.19: Curve Fit for the Decoder.

The graph of Figure 4.20 compares the predicted delay of macro-model vs. the actual-delay as determined from the SPICE simulations of the register file cells. It shows that the macro-model predicts the cell delay with less than ±10% variation from the observed values. Based on this macro-model the delay of a register file with 16 read ports is calculated to be 9.55 ns which is a twofold increase over a two port register file.

Finally, we note that our register file design compares well with other published register files, as far as speed is concerned [Maly92]. If we assume that register file access time is the critical path that sets the cycle time of a processor, then a super-scalar processor that accesses (both reads and writes) a 4R2W version of our register file in one pipeline stage, can be operated at 90 MHz. This assumes a register file with 32 registers of 32 bits each. The 4R2W design will support a 2 function unit super-scalar machine. For a 16 read port register file (i.e., a machine with 8 function units) the maximum achievable frequency is 70 MHz. This compares well with the state-of-the-art in

processor design at present (1995) considering the uncompetitive processes we used (2 layers of metal and minimum feature size of 1.2 microns). The machine configuration and design used to compute these frequencies is explained in greater detail in Section 4.7.4.

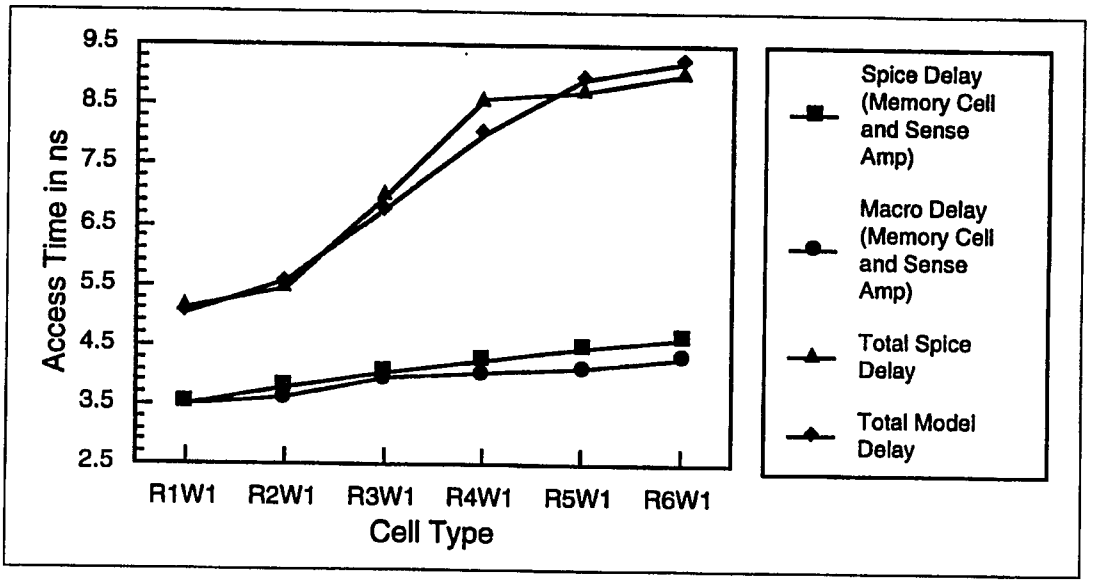


Figure 4.20: Actual Delays vs. the Predicted Delays.

4.5.3 Macro-Model for Power

While several delay and area models are described in literature, very few power models are available to the designer of a register file, in spite of the growing importance of power considerations.

Similar to our area and delay models we model the power for a register file by adding together the power dissipated in decoders, memory cells and the sense amplifiers. We assume that CMOS circuits dissipate zero static power. However, pre-charging forces some power dissipation during every clock cycle. The power dissipation is frequency dependent and is centered on the particular register selected by the decoder.

4.5.3.1 Power Model for Decoder

The power dissipation in the decoder depends on the supply voltage V_{dd}, the frequency of the input wave form υ, threshold voltages for N and P transistor V_{tn} and V_{tp}, the transistor gains for N and P transistors K_n and K_p, and the load capacitance on the output of the decoder C_{rin}. Mathematically,

$$P_{dec} = f(V_{dd}, \upsilon, V_{tn}, V_{tp}, K_n, K_p, C_{rin}) \tag{4.19}$$

where, P_{dec} is the power dissipation in decoder. There are 8 variables and 3 fundamental units (volts, amperes and seconds) in (4.19), therefore we can represent (4.19) with 5 variables. A possible form of the reduced relation is,

$$\frac{P_{dec}}{K_n V_{dd}^3} = f\left(\frac{V_{tn}}{V_{dd}}, \frac{V_{tp}}{V_{dd}}, \frac{K_p}{K_n}, \frac{C_{rin}\upsilon}{K_n V_{dd}}\right) \tag{4.20}$$

This equation can be further simplified by noting that ratio of threshold voltages and supply voltage is a constant for a given process. This observation leads to the following dimensionless relation:

$$\frac{P_{dec}}{K_n V_{dd}^3} = f\left(\frac{K_p}{K_n}, \frac{C_{rin}\upsilon}{K_n V_{dd}}\right) \tag{4.21}$$

Further simplification of (4.21) is possible, if we know the specifics of the circuit, e.g., if the ratio of P and N type device gains is fixed then $\frac{K_p}{K_n}$ can be eliminated. The model in (4.22) is based on these assumptions.

To fit a curve to (4.21) the simulations were performed for a 5 to 32 decoder. The assumption was that the decoder output changes on every cycle for one output, which is how a decoder would operate in a multi-port 32 register, register file with an independent decoder for each port. Only one gate is assumed to be switching on each cycle, the other 31 gates do not switch. The power was measured for 3 cycles of the input. The

experimental data and the curve fit are shown in Figure 4.21. The fitted curve is represented by

$$\frac{P_{dec}}{K_n V_{dd}^3} = 34.39\left(\frac{C_{rin}\upsilon}{K_n V_{dd}}\right)^{.013} \tag{4.22}$$

and is valid for C_{rin} between 300 fF and 1 pF.

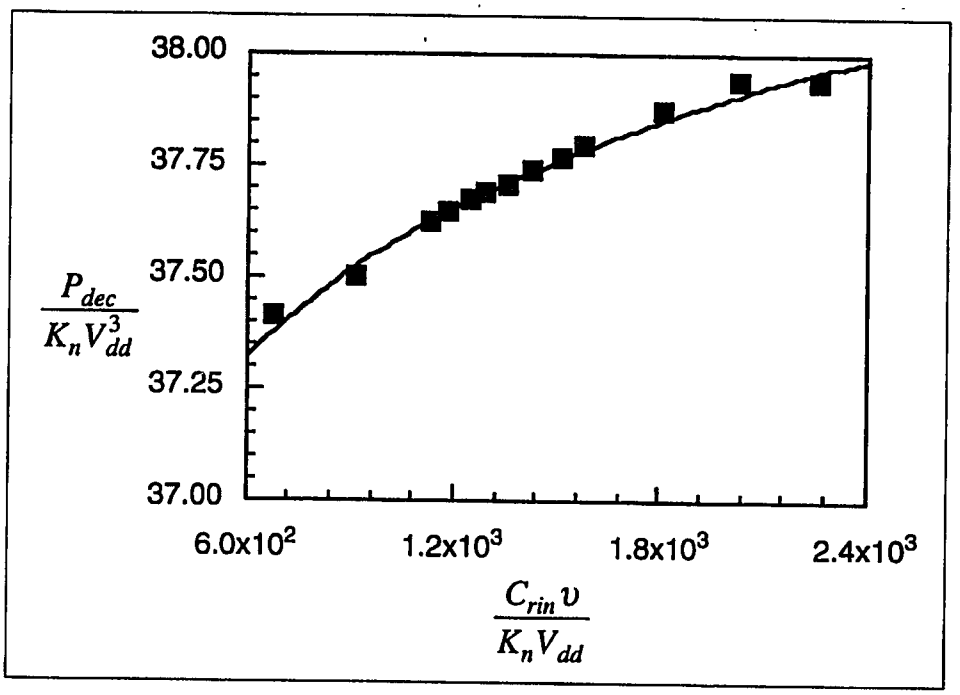


Figure 4.21: Decoder Power Variation with Load Capacitance.

4.5.3.2 Power Model for the Memory Cell

The power dissipation in the memory cell depends (refer to Figure 4.16) on the supply voltage V_{dd}, frequency of the input wave form υ, the threshold voltage of the pass transistor V_{tnpass}, the threshold voltages for N and P transistors in buffer V_{tnbuff} and V_{tpbuff}, the threshold voltage of the pre-charge transistor V_{tnpre}, the pass transistor gain K_{npass}, the transistor gains for N and P transistors in the buffer K_{nbuff} and K_{pbuff}, the pre-charge transistor gain K_{npre}, and the capacitances C_{ren} and C_{in}. Mathematically,

$$P = f(V_{dd}, \upsilon, V_{tnpass}, V_{tnbuff}, V_{tpbuff}, V_{tnpre}, K_{npass}, K_{nbuff}, K_{pbuff}, K_{npre}, C_{ren}, C_{in}) \tag{4.23}$$

There are 13 variables and 3 fundamental units (volts, amperes and seconds), therefore we can represent (4.23) with 10 variables. A possible reduced form of the equation is shown in (4.24).

$$\frac{P}{K_{npass}V_{dd}^3} = f\left(\frac{V_{tnpass}}{V_{dd}}, \frac{V_{tnbuff}}{V_{dd}}, \frac{V_{tpbuff}}{V_{dd}}, \frac{V_{tnpre}}{V_{dd}}, \frac{K_{nbuff}}{K_{npass}}, \frac{K_{pbuff}}{K_{npass}}, \frac{K_{npre}}{K_{npass}}, \frac{C_{ren}\upsilon}{K_{npass}V_{dd}}, \frac{C_{in}\upsilon}{K_{npass}V_{dd}}\right) \tag{4.24}$$

Eliminating constants from (4.24) gives

$$\frac{P}{K_{npass}V_{dd}^3} = f\left(\frac{K_{nbuff}}{K_{npass}}, \frac{K_{pbuff}}{K_{npass}}, \frac{K_{npre}}{K_{npass}}, \frac{C_{ren}\upsilon}{K_{npass}V_{dd}}, \frac{C_{in}\upsilon}{K_{npass}V_{dd}}\right) \tag{4.25}$$

Again, (4.25) has several parameters that are constant for a given configuration, for example, buffer width is constant for a given configuration. Also, it was experimentally determined that the power dissipation does not depend on buffer device sizes. This is shown in Figure 4.22. Consequently, $\frac{K_{nbuff}}{K_{npass}}$ and $\frac{K_{pbuff}}{K_{npass}}$ can be removed from (4.25).

To curve fit a model to the memory cell power, we consider two different cases. In one case the output is not switching, which occurs when the stored value is a logic '0' and the only power dissipation is due to the lines being pre-charged. We denote this power with P_{ns}. In the second case the output switches, which occurs when the stored value is logic '1' and the output discharges once the line is pre-charged. This case causes some additional power loss over the P_{ns} case and is denoted by P_s.

The power is measured over 2 cycles for both P_{ns} and P_s, when all possible read ports are active. The measurement is on a per register basis, with each register consisting of 32 memory cells. The power can be averaged per cell, allowing the user to choose both the number of bits in a register and the total number of registers to arrive at the final power dissipation of a particular design.

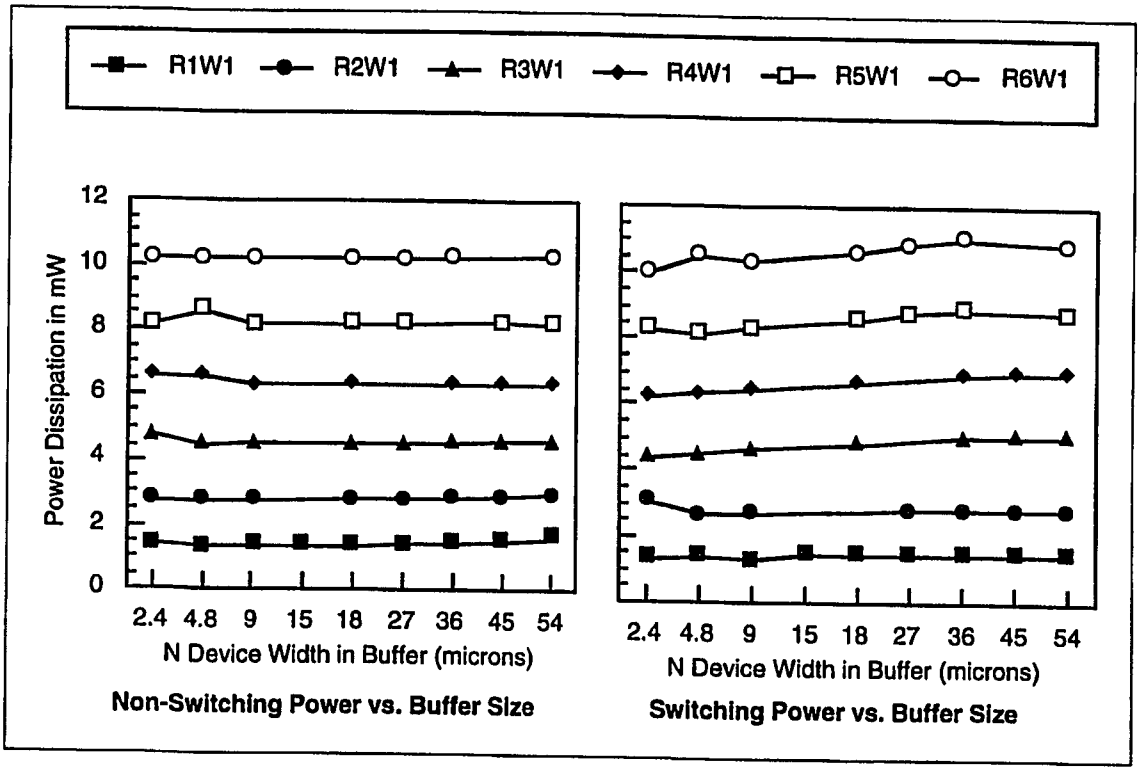


Figure 4.22: Variation of Power vs. Buffer Sizes.

The following functions provide good fits to the power dissipation characteristics of the memory cell; it is valid for a range of C_{ren} from 100 fF to 1 pF.

$$\frac{P_{ns}}{K_{npass}V_{dd}^3} = 8.514\times10^{-16}\left(\frac{C_{ren}\upsilon}{K_{npass}V_{dd}}\right)^{3.46}$$

$$\frac{P_{s}}{K_{npass}V_{dd}^3} = 6.817\times10^{-16}\left(\frac{C_{ren}\upsilon}{K_{npass}V_{dd}}\right)^{3.487} \qquad (4.26)$$

If we are strictly looking for the maximum power dissipation for a register file, which is often the case in circuit design, then, only the P_s is required since it includes P_{ns} and, as such, bounds the worst case power dissipation in the register file.

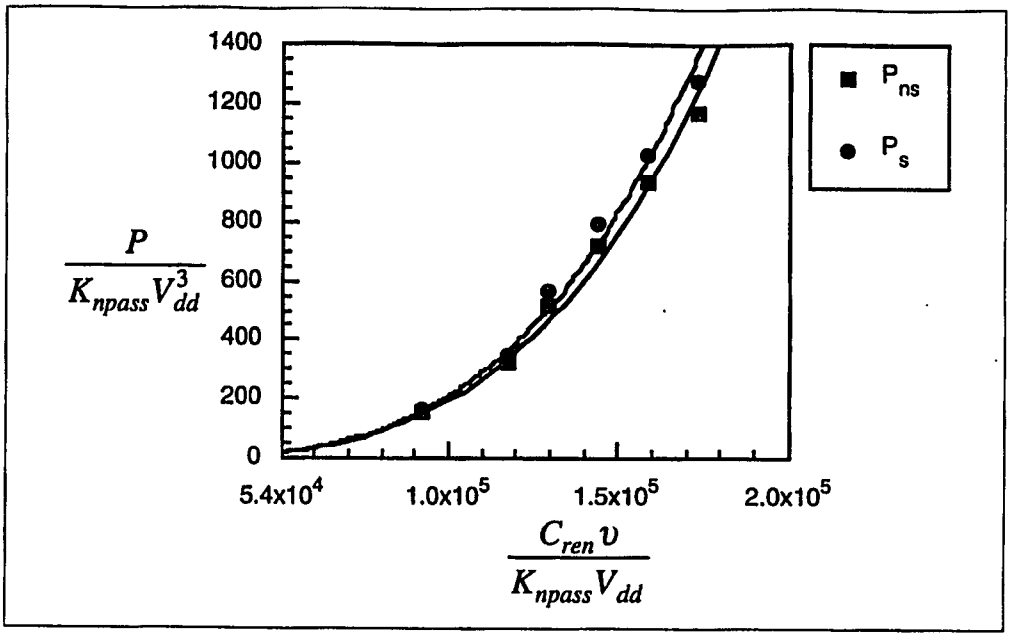


Figure 4.23: Curve Fit for the Multi-Port Memory Cell.

4.5.3.3 Power Model for Sense Amp

Our sense amplifier is an unbalanced inverter. Its power dissipation can be modeled easily if we consider that inverter is a special case of a NOR gate with only one input (each decoder cell is a 5-input NOR gate). The power model for the sense amplifier is therefore almost the same as (4.22). The difference is the values of the $\frac{K_p}{K_n}$ which depend on the widths of the P and N devices of the sense amplifier rather than on the devices in the NOR gate. Thus,

$$\frac{P_{samp}}{K_n V_{dd}^3} = f\left(\frac{K_p}{K_n}, \frac{C_l \upsilon}{K_n V_{dd}}\right) \tag{4.27}$$

where, C_l is the load capacitance.

Making the same assumption that the ratio of widths of P and N devices is constant (0.113 in the case of the sense amplifier) yields the following function

$$f\left(\frac{P_{samp}}{K_n V_{dd}^3}\right) = -5.778\times10^{-12}\left(\frac{C_l \upsilon}{K_n V_{dd}}\right)^3 + 2.763\times10^{-7}\left(\frac{C_l \upsilon}{K_n V_{dd}}\right)^2 - 3.110\times10^{-3}\left(\frac{C_l \upsilon}{K_n V_{dd}}\right) + 87.406 \quad (4.28)$$

The macro-model of (4.28) is valid for C_l from 50 fF to 230 fF. The frequency of the input wave forms is varied from 1 MHz to 100 MHz. The curve fit is shown graphically in Figure 4.24. The data is for one sense amplifier. Each register file has a bank of 32 sense amp-lifiers per read port. The macro-model of (4.28) shows that the power dissipation is roughly constant for a sense amplifier since the constant term in (4.28) dominates all the other terms.

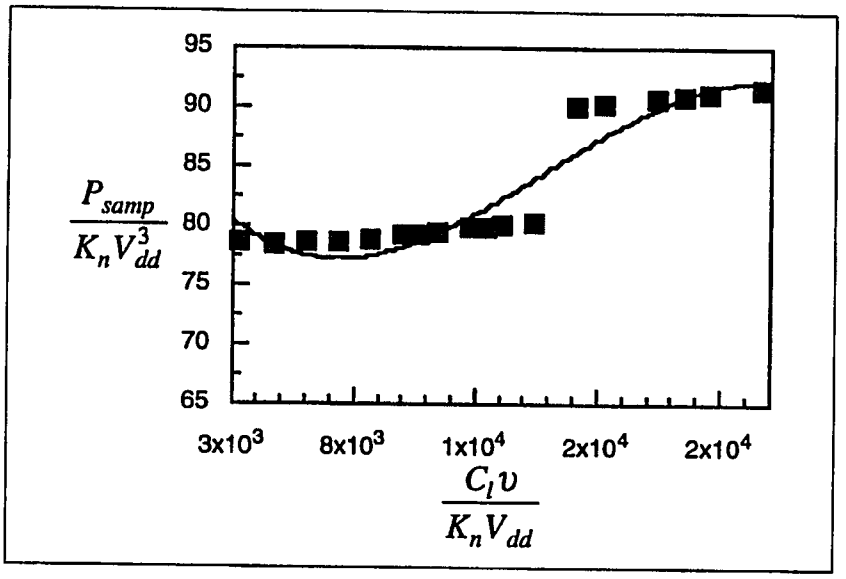


Figure 4.24: Curve Fit for the Sense Amplifier Power.

4.5.3.4 Power Model for Register File

Aggregating the three components we arrive at the macro-model for the whole register file. The total power is derived using the following relation.

$$Total\ \ Power = (N+w) * P_{dec} + (B-n) * P_s + n * P_{ns} + B * N * P_{samp} \quad (4.29)$$

where, N is the number of read ports, w is the number of write ports, n is the number of bits that are not switching, and B is the total number of bits in each register. The first term in (4.29) is the power dissipated by the decoders. It is assumed that each of the read and write ports has its own decoder, thus there are $N+w$ decoders. The next two terms are due to the power dissipation in the memory cells. The term $n * P_{ns}$ is due to the memory cells whose output is the same as the pre-charge value on the bit lines and $(B-n) * P_s$ is the power for the rest of the memory cells. The last term is for the power dissipation in the sense amplifiers. The total number of sense amplifiers is the product of the number of read ports and the number of bits in the register file. The power dissipation in the cells that are not selected is ignored, because of the negligible power loss in static CMOS. The graph of Figure 4.25 is drawn with N varying from 1 to 6 and w and B fixed at 1 and 32 respectively. The worst case power dissipation occurs when we assume $n = 0$, i.e., all the bit lines are switching. Figure 4.25 is drawn with this assumption.

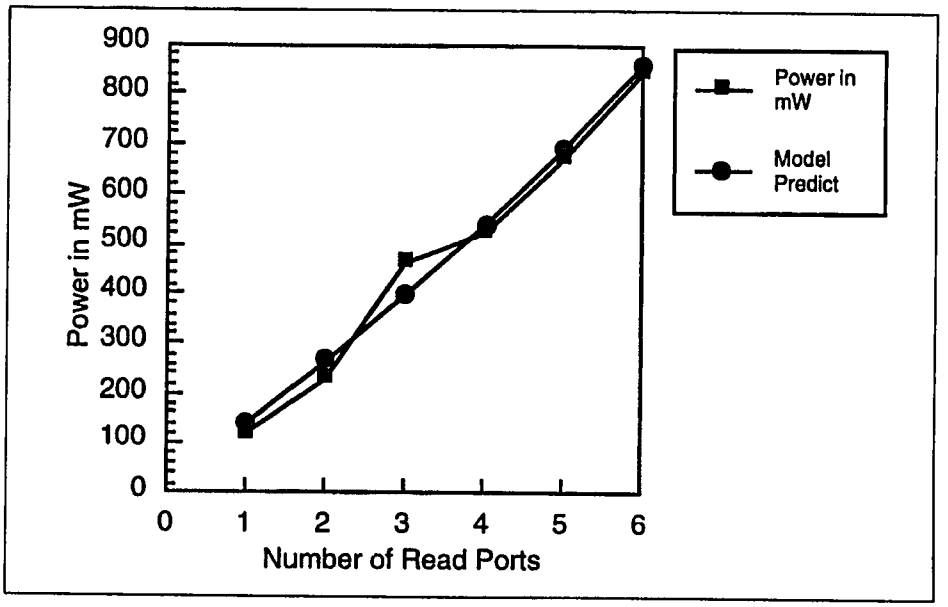


Figure 4.25: Total Register File Power vs. the Model Prediction.

It is clear from Figure 4.25 that the model and experimental data follow the same trend. However, as shown in Table 4.4, the percent error between the actual and model is

quite high (as much as 17% in one case). Nevertheless, it can be used as a first indicator of power dissipation in the circuit.

One reason for error in the model is that it is based on individual models for the components and certain items are duplicated in the individual components. For example, the output capacitance of decoder, which is also the input capacitance of the memory cell is duplicated. The reason for duplication is that we want to simulate the conditions that exist in the actual register file. This means that the output of the decoder must be terminated with the same capacitance as the input capacitance of the memory cells. This duplication however, leads to overestimate of power dissipation. This explanation is supported by the fact that the percent error decreases to less than 5% as the number of read ports becomes more than 3. For higher number of ports the effect of duplicated components becomes insignificant as compared to the power dissipation in other components.

File Configuration	Actual Power Dissipation (SPICE) mW	Model Power Dissipation mW	Percent Error %
R1W1	119.288	140.017	17.38
R2W1	228.549	262.906	15.03
R3W1	462.397	394.217	-14.75
R4W1	523.577	535.654	2.30
R5W1	670.712	689.020	2.72
R6W1	843.063	856.211	1.56

Table 4.4: Total Register File Power and Model Power.

4.5.4 Extrapolating the Macros

As we have stated in Chapter 3 the macro-models are valid within a given range of variables. Outside this range, the prediction error may be large. In the previous sections we have developed and verified the delay and power macros for up to 6 read ports register files. There is need for macros that can predict delay and power for larger number of read ports. This need arose because more and more designers are calling for register files with 12 or more ports in their new designs.

In order to determine the accuracy of our macros for predicting the performance parameters for register files with more than 6 read ports, we built a register file with 12 read ports and measured its delay and power. We then used the macros to calculate the delay and power for the 12 read port configuration. A comparison of the two set of numbers provided a measure of validity of the macros. For delays the macro model predicted 8.24 ns delay while the actual delay was 9.25 ns which represents an error[2] of 11%. The power dissipation of a 12 port register file was found to be within 4.4% of the actual power measured by SPICE simulations. These experiments show that the macro-models may be extrapolated without dramatic loss in accuracy.

In this work we have considered register files with multiple read ports but with only a single write port. We made this simplifying assumption because most of the writes from multiple functional units can be buffered so that the writes are not on the critical path for determining the delay. In a multiple functional unit processor, we expect to have the same number of write ports to a register file as the number of the functional units. However, the number of write ports can be less, because, many of the high performance pipelined architectures employ techniques for reducing the traffic back to register file.

---

[2] $Percent\ \ Error = \frac{Model - Actual}{Actual} \times 100$

Two techniques that can reduce this need for write ports are:

1. Forwarding results that have been computed by functional units in a previous cycle directly to the function units that require them as input in the current cycle. The best example is the forwarding algorithm of Tomasulo [Ander67].

2. Buffering writes by inserting buffers between function units and the register file. In this scenario, functional units write to the buffer. The control circuitry then waits to write the contents of the buffer to register file until a write port becomes free. The buffers can be accessed directly by the functional units, if results from previous computations are required which are not yet written to register file. Reorder buffers are perhaps the best examples of using buffers.

4.6 Effect of Register File Organization on Access Time

So far we have shown the dependence of the register file access time on its number of ports. In this section we explore the possibility of organizing the register file so as to reduce its access time. The previous sections assumed a very straightforward register file design as shown in Figure 4.26. We can arrange this in several different ways to reduce its access time. More specifically we present two organizational ideas that can reduce the access delay of a register file; each has its disadvantages too. These organizations are presented and discussed in the next few sections.

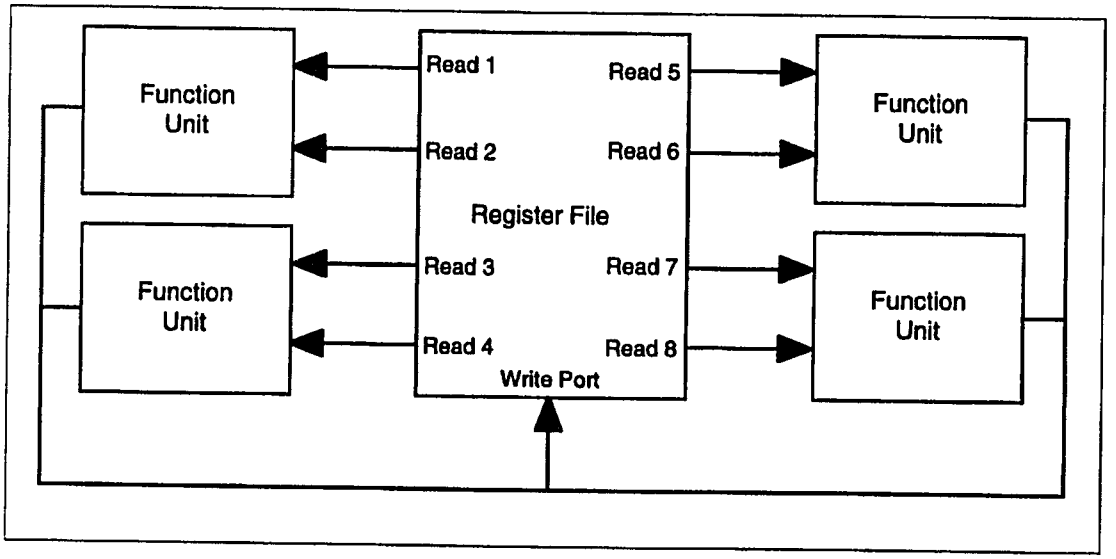


Figure 4.26 : A Typical Register File Organization.

4.6.1 Replicating the Register File

This is a simple reorganization of an *N* port register file. The register file is replicated *S* number of times with each register file having *N/S* ports each. The access time is reduced since the access time of a register file with larger number of ports has been shown to be larger than a register file with smaller number of read ports. Since *N* and *S* are integers the number *N/S* is always less than *N*, hence the access time of the replicated register file is always less than an equivalent non-replicated register file.

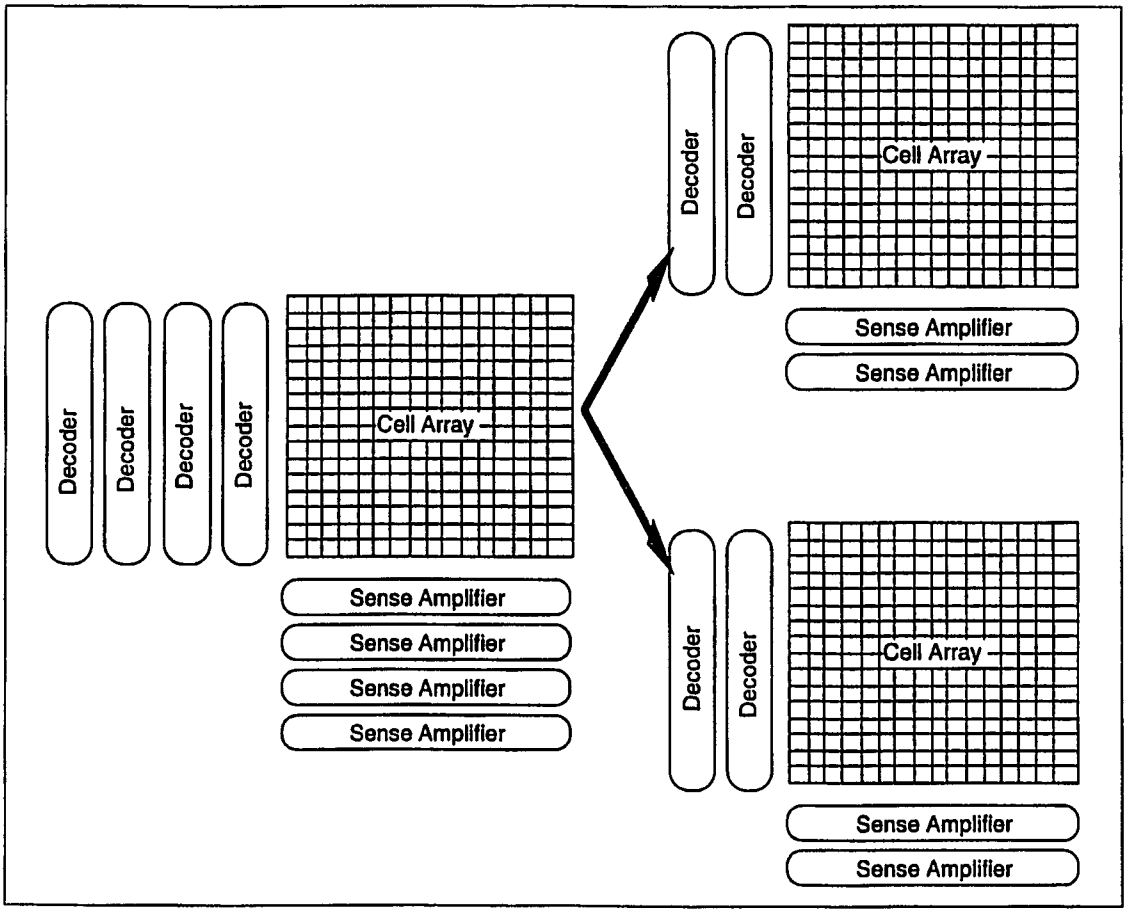


Figure 4.27: A 4 Port Register File Realized by Two 2 Port Files.

The scheme is illustrated in Figure 4.27 where a 4 read port register file is realized by 2 files each with 2 read ports. Each port has its own bank of sense amplifiers and is connected to a separate data bus for maximum throughput to the ALU's. However, if the

output from sense amplifiers has to be connected to the same data bus then some sort of multiplexing scheme may be needed which is not shown in Figure 4.27. This may also increase the access delay in the register file.

The interconnection to the functional units in a processor is shown in Figure 4.28. Here 4 functional units are connected to an 8 read port register file, which is realized as 2 files each with 4 read ports. The dashed line shows the connection required in case some data needs to be transferred from one register file to the other.

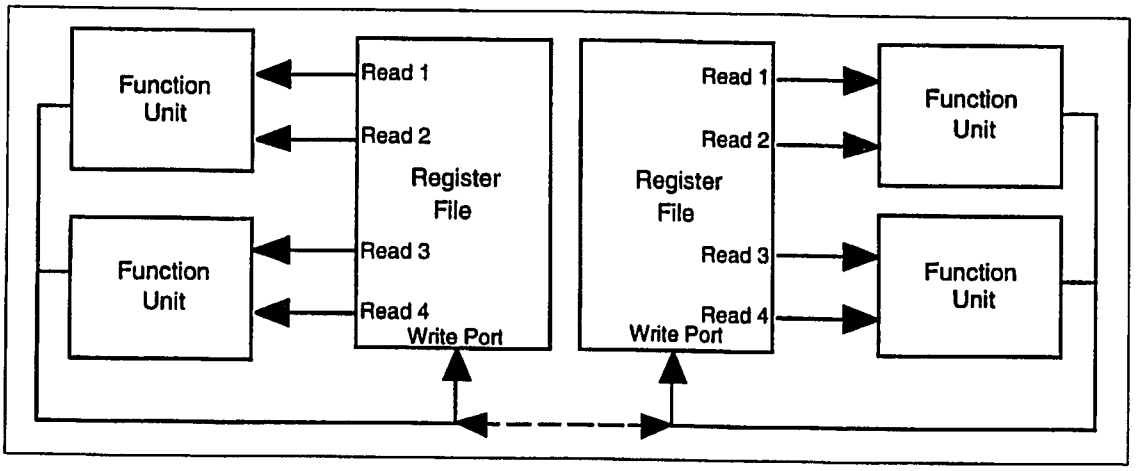


Figure 4.28: A Replicated Register File.

There are two main disadvantages of replicating register files. First, the replicated register file occupies a larger area on the chip die than the original register file. The increase in area wastes precious space on the chip and increases circuit complexity. Clearly, this presents a design trade-off. Second, is the problem of synchronizing the contents of two register files. Since the files are replicated, the same registers in the two files may contain different values at the same time. This problem can be avoided by writing to the both register files simultaneously.

4.6.2 Distributed Register Files

In this scheme the registers are divided into two or more banks such that each bank contains distinct registers. Therefore, the data coherency problem of the replicated registers is avoided. Each bank of registers is connected to the functional units most likely to require the values stored in these registers A super-scalar mode of computation is assumed here, as most of the time multi-port register files are needed to satisfy the large data bandwidths of super-scalar processors. The banks are also connected with one another in case one functional unit needs to pass on its output to other functional unit. This adds an extra R/W port to each of the segments of the distributed register file. The divided register files in effect provide a small local storage space for each of the functional units.

The distributed scheme is shown in Figure 4.29 where a single 8 read port register file with 32 registers has been distributed into 4 segments of 8 registers each. Each segment of the file has 3 read and 2 write ports.

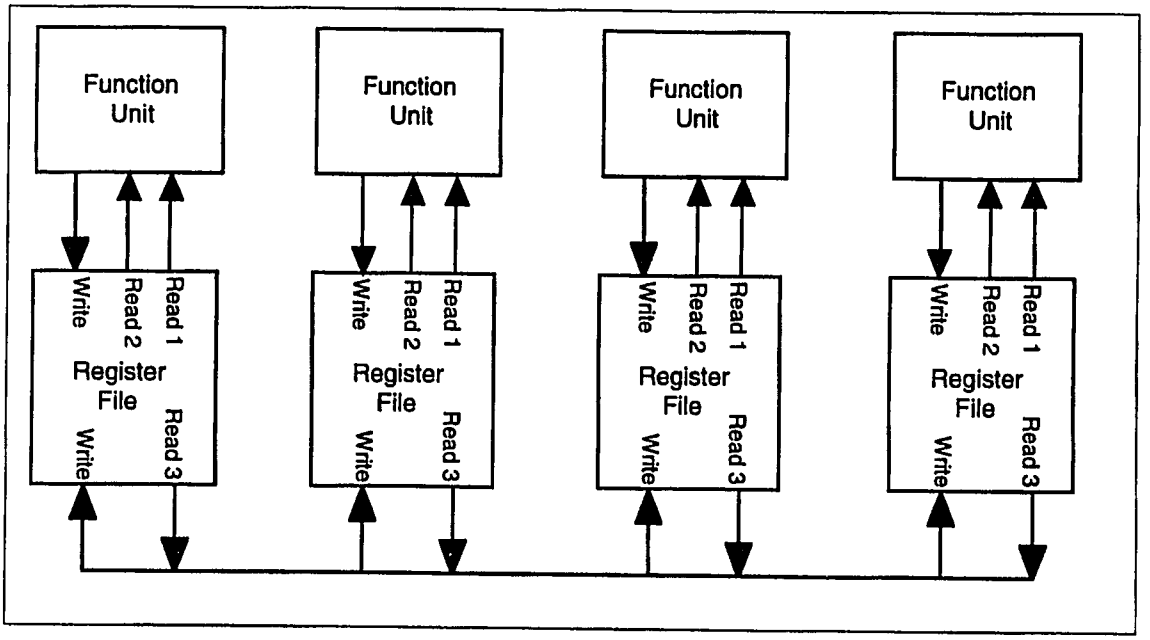


Figure 4.29: A Distributed Register File.

The disadvantages of this scheme are that each functional unit has a limited amount of local storage. Furthermore, this number goes down if there are more functional units, since for a given number of register R and number of functional units F, registers in each segment of the distributed register file are R/F. Thus, there is a large latency if the variable required by the functional unit is not in the segment of the register file connected to the functional unit. In this case, the variable has to be fetched from the appropriate register file segment and written into the register file segment connected to the functional unit. Alternately, if the variable is not an intermediate result it can be directly fetched from the primary data cache. Yet another variation is to directly connect the fetched variable to the functional unit thus bypassing the local register file segment. All these imply some extra logic to be built with the access mechanism, thus making the system more complex and probably slowing the operation.

4.6.3 Comparison of Various Register File Configurations

Table 4.5 compares various register file configurations discussed in previous sections in terms of area occupied and the access delay. All figures were computed by using the macro-models of the previous section. The configuration is presented as an ordered triplet *(n,p,r)*, where, n is the number of the register files, p is the number of read ports in each register file, and r is the number of registers in the register file. In case of the single register file n is always 1. In case of a replicated file $n > 1$ but all the files have same number of registers r which is equal to the total number of registers in the file. In the case of distributed files $n*r$ is the total number of registers in the file. (All the calculations are for a register file with a P device width in the decoder buffer = 37.5 microns).

From Table 4.5 it can be seen that the distributed register file configurations are always better than the other two as they occupy less space and are faster. However, these calculations ignore the effect of the extra time and logic required to pass variables among

No. of ports	Single Register File			Replicated Register File			Distributed Register File		
	Configuration	Size (mm^2)	Delay (ns)	Configuration	Size (mm^2)	Delay (ns)	Configuration	Size[3] (mm^2)	Delay (ns)
2	(1,2,32)	1.52	4.88	(2,1,32)	2.26	4.68	(1,2,32)	1.52	4.88
4	(1,4,32)	2.48	5.44	(4,1,32)	4.52	4.68	(2,2,16)	2.06	4.01
				(2,2,32)	3.04	4.88			
6	(1,6,32)	3.66	5.76	(6,1,32)	6.78	4.68	(3,3,10)[4]	2.07	3.88
				(3,2,32)	4.56	4.88			
				(2,3,32)	3.94	5.14			
8	(1,8,32)	5.06	6.46	(8,1,32)	9.04	4.68	(4,3,8)	2.24	3.88
				(4,2,32)	6.09	4.88			
				(2,4,32)	4.95	5.76			
16	(1,16,32)	1.30	8.25	(16,1,32)	18.07	4.68	(8,3,4)	2.55	3.88
				(8,2,32)	12.17	4.88			
				(4,4,32)	9.90	5.76			
				(2,8,32)	10.13	6.46			

Table 4.5: Comparison of Various Register File Configurations using the Area and Delay Models.

[3] The size ignores the extra logic area required as discussed in Section 4.6.2.
[4] The total number of registers is 30 for this special case.

different functional units. Also the number of local registers for each functional unit is reduced. A final point to note is that for a large number of ports (> 8) the area occupied by a single file of *N* ports is larger than the area occupied by 2 register files of *N/2* ports.

4.7 Application of Delay Macro-Models – Examples

In this section we present some examples of the applications of the macro-models just developed. We apply the delay macro-models to some of the published super-scalar architectures and show the effects of the multi-porting register files that are ignored in the original performance calculations.

As discussed in Chapter 1, super-scalar machines can issue multiple instructions per machine cycle. The machine cycle in a super-scalar machine is the maximum of the latencies of the functional units in the machine. Super-pipelined machines on the other hand only issue one instruction per cycle but the cycle time is much less than the latency of any of the functional units. (The maximum of the latencies of the functional units in such machines is termed as a major cycle, as opposed to the machine cycle which is usually called a minor cycle). Thus, effectively a super-pipelined machine can issue instructions while previously issued instructions are still under execution. It has been shown that both of these techniques are equivalent ways of exploiting instruction level parallelism [Joupp89a]. In many advanced architectures multiple-issue and pipelining are combined.

If multiple-issue is not possible in a super-scalar machine due to lack of resources then the instructions are said to have *structural dependencies*. This could occur when for example, there are not enough function units to execute all possible instructions. Sometimes it is not possible to execute instructions together, in spite of available resources. This is because the instructions require a specific order of execution for correct results. Such instructions are said to have *data dependencies*.

The earliest example of a super-pipelined machine is Control Data's CDC 6600 which had 16 functional units. It can issue one instruction per minor cycle. There were 3 minor cycles in one machine cycle. This was followed by IBM 360/91 which used out of order issue techniques based on Tomasulo's algorithm to speed-up its floating point units. The algorithm is, however, equally applicable to integer and load/store units.

4.7.1 The Smith and Johnson Machine

Smith and Johnson presented a super-scalar machine in [Smith89]. Their machine has multiple, independent, pipelined functional units. The machine allows for instructions with different latencies. The instruction issue unit is capable of issuing instructions out of order, and can issue multiple instructions per cycle. The machine employs branch prediction to reduce the branch penalty.

The instruction scheduler pre-fetches the instructions from the instruction stream and places them in an *instruction window*. These instructions are then issued to the available functional units, after checking for data hazards, where the latencies of these instructions are recorded. On each machine cycle the recorded latencies are reduced by one until they are zero, at which time they are retired from the *instruction window*. The instruction window behaves a lot like the scoreboard mechanism of the CDC 6600 [Thorn70].

The machine configurations studied by Smith and Johnson are presented in Table 4.6. Different machine configurations were simulated with different assumptions to estimate a measure of the concurrency. These ranged from assuming no structural and control hazards to a more realistic assumption of considering all the hazards and assuming a more realistic 85% accurate branch prediction. The reported speed-ups given the later assumptions are summarized in Table 4.7 on next page. The authors also experimented with the *instruction window* size, but found out that an instruction window

size larger than 32 is not justified, as above this limit instruction level parallelism is practically non-existent.

Functional Unit	Machine 1	Machine 2	Machine 3	Machine 4
Integer ALU	1	1	2	2
Barrel Shifter	1	1	1	1
Load Pipe	1	2	1	2
Store Pipe	1	1	1	1
Branch Unit	1	1	1	1
FP Adder	1	1	1	1
FP Multiplier	1	1	1	1
FP Divider	1	1	1	1
FP Convert	1	1	1	1
Number of Read ports	5	6	7	8

Table 4.6: Configurations of S&J Machines.

Machine Type	Number of Read Ports	Speed-Up
Machine 1	5	1.90
Machine 2	6	1.95
Machine 3	7	2.00
Machine 4	8	2.10

Table 4.7: Number of Read Ports vs. Reported Speed-Ups for the S&J Machines.

4.7.2 Jouppi and Wall Machine

Jouppi and Wall presented their MultiTitan machine in [Joupp89b]. The machine has a simple look ahead instruction issue unit that looks ahead up to 8 instructions in the instruction stream. The issue unit detects any hazards associated with an instruction and issues it to the relevant function unit. If there are hazards or there is no function unit available, the machine stalls. MultiTitan has function units with different latencies, one cycle for integer ALU operations, two for load/store and branches, and three for floating point operations. The machine relies heavily on the integrated compiler for code optimization, register allocation and instruction scheduling. Thus, this system first schedules the code statically and then tries to schedule it dynamically, to extract maximum instruction level parallelism. The machine has a divided register file with one part reserved for temporary result storage and the other part reserved for variables.

As part of a design study, simulations were performed with 2 to 8 function units. The reported speed-up ignores the effects of cache misses and other system effects like interrupts. Table 4.8 summarizes the results. The speed-up is over a base machine that can issue one instruction per cycle. However, it should be noted that the static code scheduling allows even the base machine to exploit some instruction level parallelism by moving code to execute integer instructions in parallel with floating point instructions and other high latency instructions, or by filling branch delay slots. As such the reported speed-ups may be less than the speed-ups if the super-scalar configurations were compared with a purely scalar machine, without static code scheduling.

Table 4.8 also summarizes the number of required read ports, which is calculated based on the assumption that each of functional units requires two read ports[5] which, as mentioned previously, is an upper limit on the number of read ports.

---

[5] [Joupp89b] does not specify the functional unit types.

Number of Functional Units	Number of Read Ports	Speed-Up
1	2	1.00
2	4	1.70
3	6	2.00
4	8	2.10
5	10	2.15
6	12	2.18
7	14	2.20
8	16	2.20

Table 4.8: Number of Read Ports Required with Reported Speed-Ups for each of the MultiTitan Configurations.

4.7.3 Restricted Data Flow Machine

Butler, et al. present the restricted data flow machine in [Butle91]. The Restricted Data Flow (RDF) machine tries to exploit all the available instruction level parallelism in an instruction stream. A particular implementation is characterized by a window size, issue rate and the latencies of the function units. The RDF studies start by assuming perfect branch prediction. They also assume an infinite number of functional units so that instructions are never held up due to structural hazards. The instructions are issued in order they appear in the instruction stream.

The RDF looks at the instruction stream and constructs the dynamic data flow graph. It tries to execute an instruction as soon as its dependencies are resolved. The word "restricted" in RDF refers to the restrictions imposed on the window size (number of instruction buffers), and issue rate (the instruction bandwidth). If these restrictions are removed, then the machine is simply a pure data flow computer.

Clearly perfect branch prediction and an infinite number of functional units are not physically possible. Accordingly, the authors study 6 more realistic configurations in [Butle91] which limit the number of functional units and branch prediction accuracy of the RDF machine. The speed-ups and the register port requirements for different machines are summarized in Table 4.9. The authors have reported speed-ups of up to 5.8 for floating point benchmarks but our study deals with only the integer register files, thus we will focus only on the integer speed-ups. With multiple floating point units and a single floating point register file similar arguments also apply to the floating point speed-ups.

Machine Name	Number of Read Ports	Ideal Speed-Up
4F2M	8	< 2.9
4F2M.B85	8	< 2.2
6F3M	12	< 3.6
8F3M	16	N.A
8F3M.B85	16	< 2.9
8F3M RB	16	< 3.4

Table 4.9: Number of Read Ports Required with Reported Speed-Ups for Each of RDF Machine Configurations.

Each of the machine configurations of Table 4.9 are described in more detail in Table 4.10. There are no integer functional units. This is because each of the units is capable of executing integer instructions, and as such, requires 2 read ports into the integer register file. All of the machine configurations assume a perfect branch prediction except the machines with the B85 suffix, which assume an 85% success rate for branch

prediction and the machine with an RB suffix which assumes a real branch prediction mechanism that uses the previous branching history of the executing program.

Functional Units	4F2M	4F2M.B85	6F3M	8F3M	8F3M.B85	8F3M.RB
FP Add/Mul/Div	1	1				
FP Add			1	1	1	1
FP Mul/Div			1			
FP Mul				1	1	1
FP Div				1	1	1
Branch	1	1	1	1	1	1
Bit				1	1	1
Load/Store & Bit	2	2	3	3	3	3
Number of Read Ports	8	8	12	16	16	16

Table 4.10: Details of the Configurations for the RDF Machines.

4.7.4. Impact of Multi-Port register Files

The following analysis is based on a pipelined machine capable of issuing more than one instructions per cycle. Thus, this machine could be called a super-scalar pipelined machine. We assume the classic pipeline consisting of 5 stages. Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory access (M) and register Write Back (WB). The WB and ID both require access to the register file in the same cycle for writing the results and reading the operands, respectively. It is assumed that all

the writes complete in the first half of the cycle and all the reads in the later half. This limits to 2 the stages to be bypassed, for data forwarding purposes. Thus, the cycle time c of this machine must be greater than or equal to the sum of read and write access times of the register file.

$$c \geq c_r + c_w \tag{4.30}$$

where, c_r and c_w are the read access time and write access time of the register file. If we assume that the cycle time is totally bounded by the register file access times than the inequality may be replaced by equality.

Let the scalar machine cycle time be c_1 and the super-scalar machine cycle be c_2. Thus, from (4.30),

$$c_1 = c_{r1} + c_{w1} \tag{4.31}$$

Similarly,

$$c_2 = c_{r2} + c_{w2} \tag{4.32}$$

where, c_{r1} is the read time of the register file and c_{w1} is the write time to the register file for a scalar machine and c_{r2} and c_{w2} are the read and write times for a super-scalar machine.

If the average number of instructions executed per cycle by the super-scalar machine is n then the speed-up of the super-scalar machine over the scalar machine is given by the relation

$$Speed\text{-}up = n\frac{c_1}{c_2} = n\frac{c_{r1} + c_{w1}}{c_{r2} + _{w2}} \tag{4.33}$$

if the difference in cycle times is taken into account. Ideally $c_1 = c_2$ and the speed-up is n. However, most of the reported speed-ups do not consider the longer cycle time that may be required when a multi-port register file is used. When we consider the macro-models

of Section 4.5 and apply the delays, and then use (4.33), we get a speed-up which is less than the reported speed-up. Effects of (4.33) on these machines discussed earlier are tabulated in the three tables 4.11, 4.12, and 4.13.

Table 4.11 presents the effects of (4.33) on Smith and Johnson's machines described in Section 4.7.1. The actual speed-up is practically constant from machine 1 to 4 at 1.7, whereas, if we ignore the effects of register file slow down, the execution speed increases as the number of the functional units is increased. Machines 2 and 3 actually have a lower execution rate than machine 1, which shows that adding extra load store units may not be as beneficial as adding more ALUs (see Table 4.6).

Machine Type	Number of Read Ports	Reported Speed-Up	Speed-Up Using (4.33)
Machine 1	5	1.90	1.744
Machine 2	6	1.95	1.734
Machine 3	7	2.00	1.724
Machine 4	8	2.10	1.755

Table 4.11: Effects of Multi-Porting on S&J Machines.

Table 4.12 presents the effects of (4.33) on the MultiTitan machine. The speed-up increases until the number of functional units is 4 after which it starts declining. This shows that super-scalar machines can realize speed-up but exploiting too much of the instruction level parallelism may actually slow down the machine. Another interesting point that is noticeable from Table 4.12 is that even without the register file penalty the speed-up becomes constant for last two lines of the table.

Number of Functional Units	Number of Read Ports	Reported Speed-Up	Speed-Up Using (4.33)
1	2	1.00	1.00
2	4	1.70	1.61
3	6	2.00	1.78
4	8	2.10	1.76
5	10	2.15	1.69
6	12	2.18	1.63
7	14	2.20	1.56
8	16	2.20	1.49

Table 4.12: Effects of Multi-Porting on MultiTitan Machine.

Table 4.13 presents the effects of (4.33) on the RDFs described in Section 4.7.3. Again exploiting more instruction level parallelism seems to slow down the whole of the machine rather than speeding it up. From all the machines presented 6F3M seems to be the best choice as it provides the most speed-up. The 8 function unit machines are not justified at all – their performance is lower than a 4 function unit machine. This is contrary to what we would expect if we look at the problem just from "exploiting instruction level parallelism" point of view as machines 8F3M are exploiting more instruction level parallelism than 6F3M. The numbers cited in the paper are derived looking at the problem from just this angle. The more realistic assumption of 85% branch prediction does allow the 8 function unit machine 8F3M.B85 to be faster than the 4F3M.B85 machine but not by much.

Generalizing on the concept of (4.33), Table 4.14 presents a performance penalty factor that can be applied to super-scalar architecures to arrive at a more accurate performance estimate. This estimate includes the effect of the multi-port register files that are required for super-scalar machines.

Machine Name	Number of Read Ports	Reported Speed-Up	Speed-Up Using (4.33)
4F2M	8	< 2.9	2.50
4F2M.B85	8	< 2.2	1.90
6F3M	12	< 3.6	2.76
8F3M.B85	16	< 2.9	1.97
8F3M RB	16	< 3.4	2.31

Table 4.13: Effects of Multi-Porting on RDF Machines.

Number of Required Read Ports	Performance Penalty Factor Using (4.33)
1	1.000
2	0.980
3	0.974
4	0.946
5	0.918
6	0.889
7	0.862
8	0.836
9	0.811
10	0.788
11	0.766
12	0.746
13	0.727
14	0.709
15	0.694
16	0.679

Table 4.14: Performance Penalty for Multi-Port Register Files.

4.8 Application of Power Macros

In this section we illustrate the use of the power macros. From (4.22), (4.26), and (4.28) we can determine the power characteristics of the various register file components as the input frequency varies. Then using (4.29) the total power of the register files can be plotted at different frequencies. Figure 4.30 shows the variation of power for the decoder cells. The power is practically constant for a particular configuration. A small power increase is observed as the frequency is increased. The plot represents the first term in (4.29). Again the assumption is that each port has its own decoder.

The corresponding graphs for memory cell and sense amplifier are shown in Figures 4.31 and 4.32. Figure 4.31 shows an exponential increase in the power dissipation of the memory cell, this corresponds with the second and third terms in (4.29) with the assumption that all bit lines are switching every cycle. This is when the ratio $\left(\frac{C_{rin}}{K_n V_{dd} t}\right)$ becomes greater than one. The sense amplifier power dissipation (4.28) is a function of output capacitance and therefore, does not vary with the changes in memory cell configurations, since a constant output capacitance of 224 fF is assumed for the purposes of this experiment. This is the value generally used in commercial CAD packages, for example [EPOCH93], and is considered equivalent to input impedance of 10 gates. This graph represents the fourth term in (4.29). This graph is also constant and does not vary with frequency due to the same reasons as cited for the decoder graph. However, this is a major part of the total power dissipation, due to the large size of the output drivers and the number of sense amplifiers in each of the configurations (each addition of ports adds 32 more sense amplifiers to the register file).

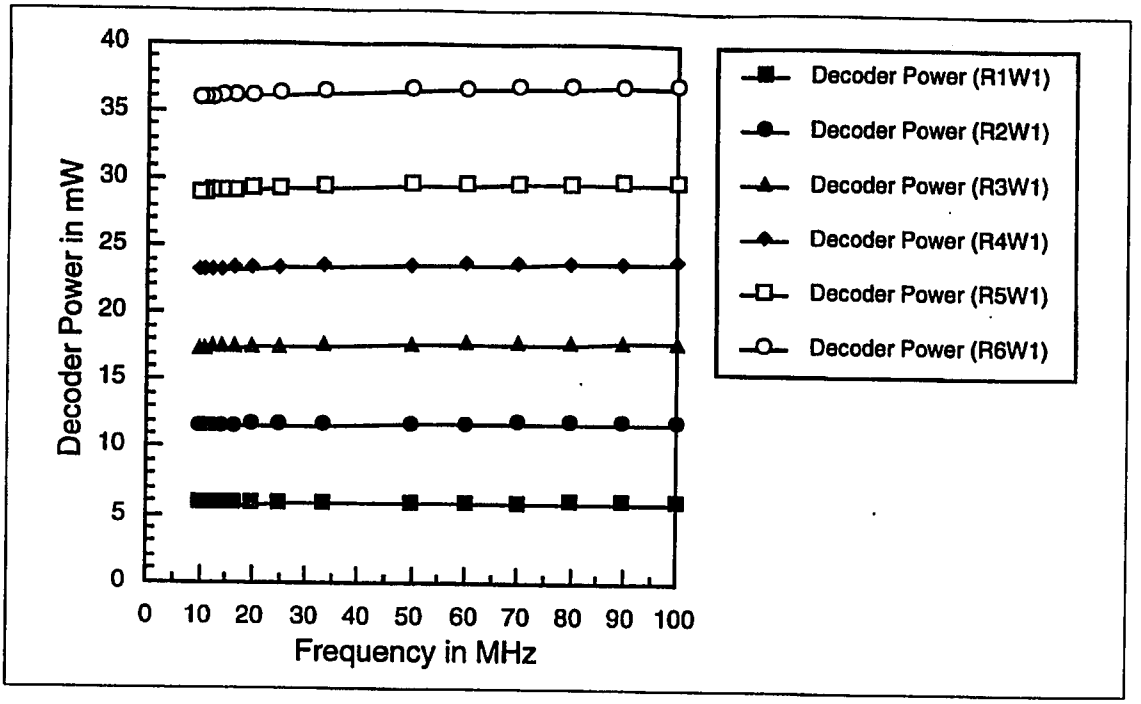


Figure 4.30: Effect of Frequency on Decoder Power Dissipation.

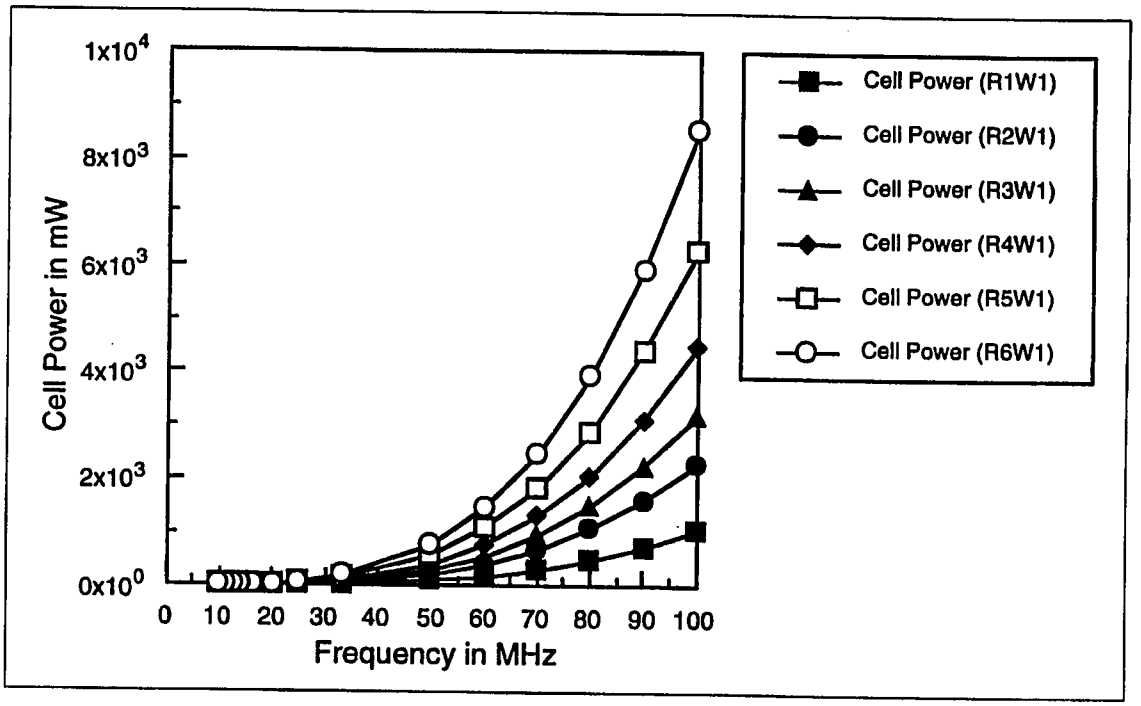


Figure 4.31: Effect of Frequency on Memory Cell Power Dissipation.

Finally, the effect of (4.23), (4.26) and (4.28) is summed to arrive at the total power dissipation of the register file. This is shown in Figure 4.33. The power is seen to rise rapidly with frequency. This is a combined effect of the addition of the sense amplifiers and the exponential rise in power for the memory cell. The exponential nature of Figure 4.32 means that total power dissipation will eventually be dominated by the power dissipation in the memory cell after a certain frequency, because the power dissipation in decoders and sense amplifiers is practically constant with frequency. The power dissipation for the 8 port register file is almost 9W at 100 MHz. This is a lot of power to be dissipated and may require special heat removal techniques if this is to be implemented. In fact, new sub-micron technologies with their much lower power requirements will be needed before register files with large numbers of ports are practical.

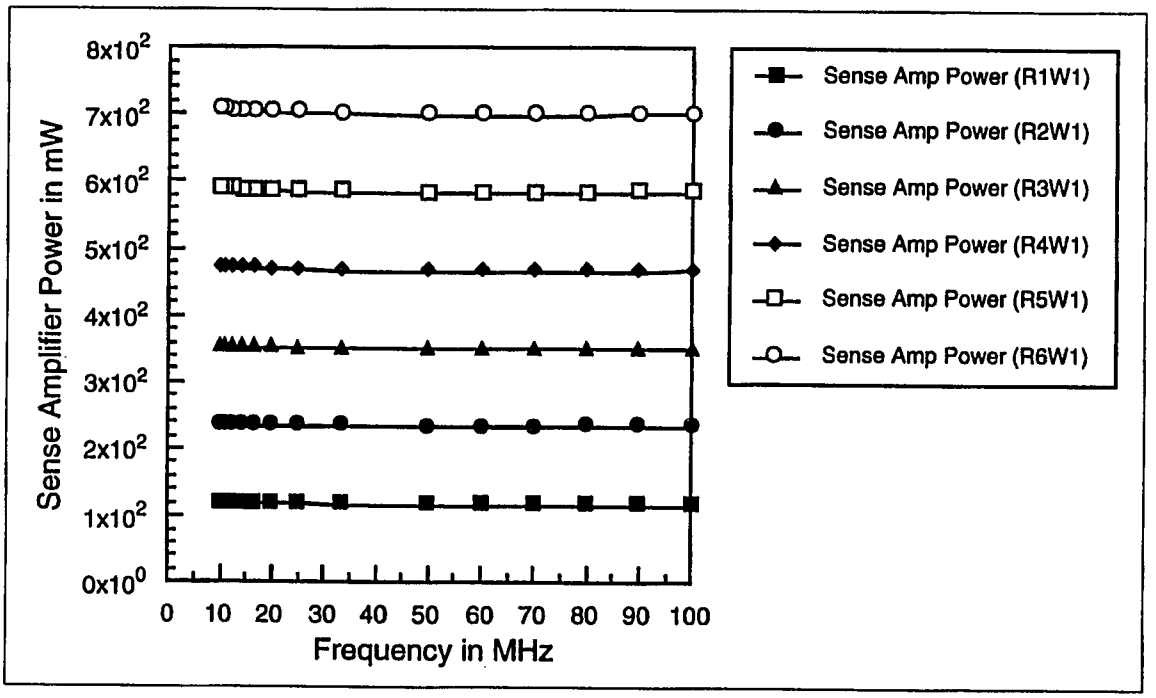


Figure 4.32: Effect of Frequency on Sense Amplifier Power Dissipation.

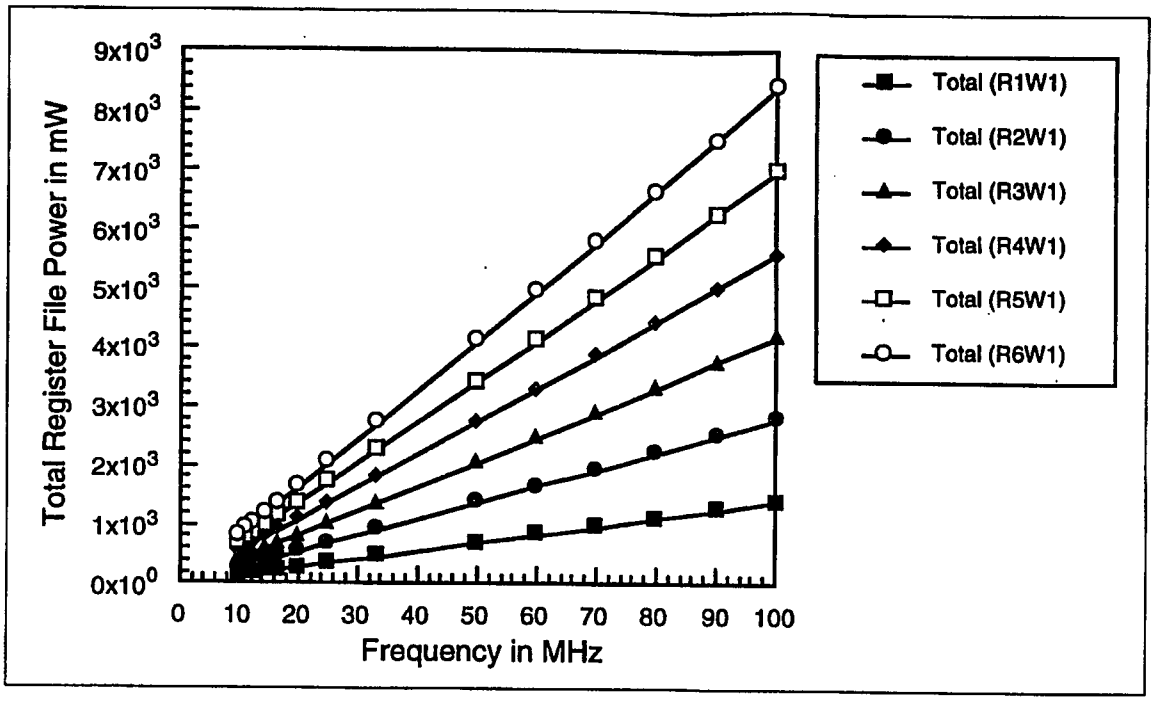


Figure 4.33: Effect of Frequency on Total Register File Power Dissipation.

4.9 CAM Macro-Models

Content Addressable Memory (CAM) cells are a special kind of memory device that access cells based on their content rather than their address. These cells are commonly used in caches for storing the tags associated with the data lines (or blocks), and in translation lookaside buffers [Sawad89, Uneka94].

A common CAM circuit [Weste88] is shown in Figure 4.34. Data can be written and read from the circuit of Figure 4.34 in a manner similar to the circuit of Figure 4.2 as explained in Section 4.3.1. The additional devices in the circuit of Figure 4.34 allow us to access the data based on a match or mismatch of the contents of memory cell to the data. This mode of operation works as follows. Memory contents can be accessed by placing the data on the $\overline{bit}$ and its complement on the *bit* line. The match operation proceeds as follows. The *match* line is pre-charged high. If the stored value is '1' (node A is '1' and node B is '0') and the values placed on $\overline{bit}$ line is '0' and *bit* is '1' then the match

transistor turns on and pulls the pre-charged *match* line low, indicating a mismatch. On the other hand if the $\overline{bit}$ line is '1' and *bit* is '0' then the match transistor stays off and the *match* line stays high, indicating a match. Thus cells whose contents matches a datum can be identified and retrieved.

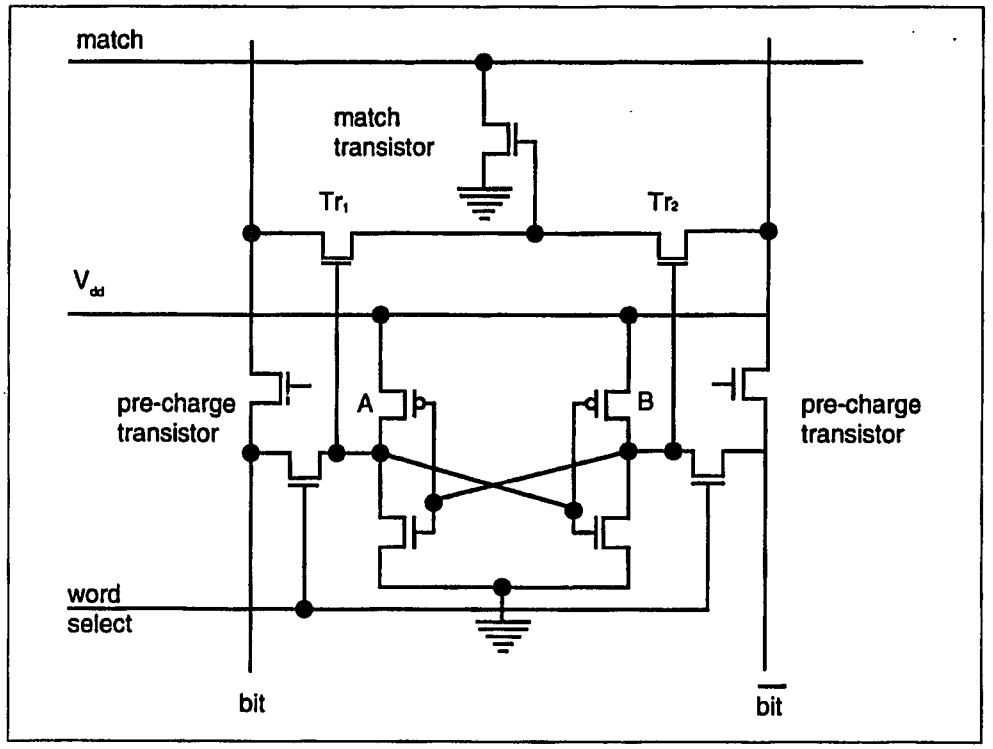


Figure 4.34: A CAM Cell.

4.9.1 Delay Macro-Model for CAM

There are three delays associated with a CAM cell. The read delay, the write delay and the match delay. The read and write delays are identical to the delays of a memory cell as described in the Section 4.3. The match delay is the time difference between the time data is placed on the *bit* and $\overline{bit}$ lines and the time at which the *match* line goes low (indicating a match). Again we do not need to consider the mismatch condition because the *match* line is pre-charged.

The model for the match delay is derived in the same way as the models for the memory cell in Section 4.5. The equivalent circuit that is used for the derivation is given in Figure 4.35. The delay occurs in determining if there is a mismatch between the data placed on data bus and the value stored in the memory. Figure 4.35 is annotated to show the voltage levels which would exist in the circuit in case of a mismatch. The match line is pre-charged and therefore V_a is at 5 volts. The gate of Tr_1 is high (a logic '1' is stored in the cell) and the *bit* line is at 5 volts (the complement of the data is applied at the $\overline{bit}$ line). Tr_1 is on and the gate of the match transistor is high, causing it to turn on.

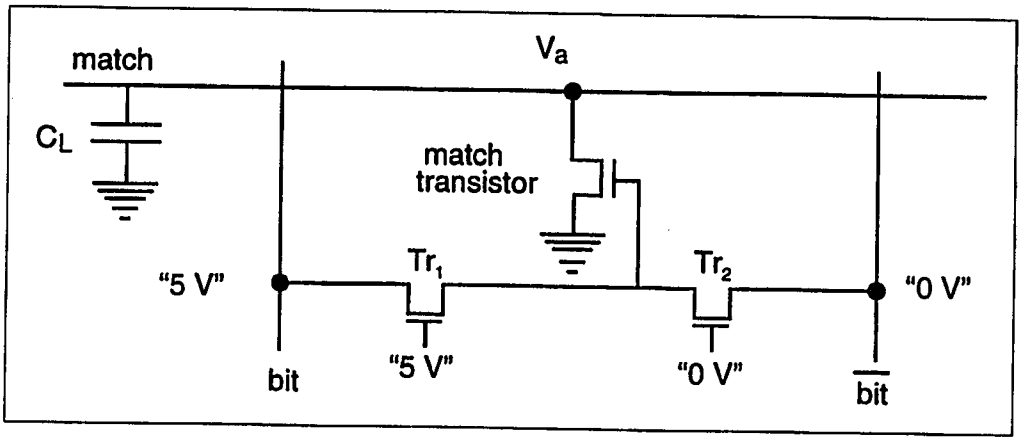


Figure 4.35: The Circuit Used to Determine Match Delay.

Initially, the match line voltage is the same as V_{dd}. As the match transistor turns on, it starts discharging the load capacitance. The current through the match transistor is given by

$$I = -C_L \frac{dV_a}{dt} = \begin{cases} 0 & V_{bit} - V_{tn} \le 0 \\ 0.5K(V_{bit} - 2V_{tn})^2 & V_{bit} - V_{tn} \ge 0, \\ & V_{bit} - V_{tn} \le V_a \\ 0.5K(V_{bit} - 2V_{tn})^2 - (V_{bit} - 2V_{tn} - V_a)^2 & V_{bit} - V_{tn} \ge 0, \\ & V_{bit} - V_{tn} \ge V_a \end{cases} \quad (4.34)$$

where, I is the current through the match transistor, C_L is the load on the match line and V_a is the voltage on the match line, K is the N transistor gain, V_m is the N device threshold voltage, and V_{bit} is the voltage on the bit line. From (4.34) it follows that

$$\Delta = F(V_{dd}, V_m, C_L, K, T_{in}) \tag{4.35}$$

The function of (4.35) has 6 variables and 3 units (volts, amperes and seconds). By applying Buckingham's Pi-Theorem [Bucki15], (4.35) can be expressed in terms of 3 (= 6 - 3) dimensionless quantities. A possible form of the reduced equation is

$$\frac{\Delta}{T_{in}} = F\left(\frac{V_{tn}}{V_{dd}}, \frac{C_L}{KV_{dd}T_{in}}\right) \tag{4.36}$$

the above relation can be further simplified if we note that $\frac{V_{tn}}{V_{dd}}$ is a constant for a given process, therefore the equation can be written as

$$\frac{\Delta}{T_{in}} = F\left(\frac{C_L}{K\ V_{dd}\ T_{in}}\right) \tag{4.37}$$

After curve fitting the final model is as give below:

$$\Delta = T_{in}\left(c_0 + \frac{c_1}{T_{in}\sigma} + \frac{c_2}{(T_{in}\sigma)^2} + \frac{c_3}{\sqrt{T_{in}\sigma}}\right) \tag{4.38}$$

where, Δ is the match delay, c_0, c_1, c_2, c_3 are constants with values 0.661, -0.058, -0.02577 and 0.3700, respectively, T_i is the input time constant, and $\sigma = \frac{K\ V_{dd}}{C_L}$ is the capability of match transistor to drive a capacitive load. Further, $K = \beta \frac{W}{L}$ where, W, L are the width and length of the match transistor, $\beta = \frac{\mu\ \varepsilon}{t_{ox}}$, t_{ox} is the oxide thickness and μ is the hole mobility in the match transistor. The C_L in the macro-model varies from 100 fF to 1 pF. This corresponds to the load on the match bus. The curve fit is shown in Figure 4.36.

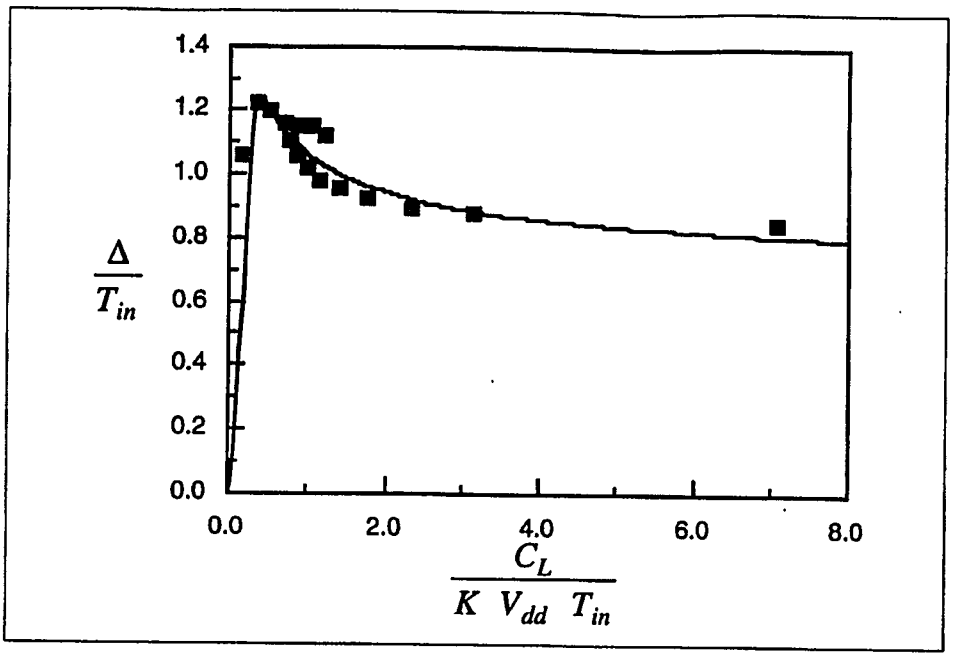


Figure 4.36: Curve Fit for Delay in the CAM.

4.9.2 Power Macro-Model for CAM

For the CAM circuit of Figure 4.34 power in a CAM cell can be expressed as

$$P_{cam} = f(V_{dd}, V_{tn}, V_{tp}, C_L, K_n, K_p, \upsilon) \tag{4.39}$$

where, V_{dd} is the supply voltage, V_{tn} is the threshold voltage for N device (we are assuming that all N devices are the same size and as such have the same threshold voltage), V_{tp} is the threshold voltage for P device, C_L is the load capacitance on the match line, K_n and K_p are the transistor gains for N and P devices respectively, and υ is the frequency of the input waveform on the *bit* and $\overline{bit}$ lines. Applying dimensional analysis and after removing other constants, we get

$$\frac{P_{cam}}{K_n V_{dd}} = f\left(\frac{C_L \upsilon}{K_n V_{dd}}, \frac{K_p}{K_n}\right) \tag{4.40}$$

Using, the assumption that the output is switching on every cycle and that there is a data match on every cycle (worst case scenario), the following function provides a good

fit for the power dissipation of a CAM. The function is valid for the load capacitance varying from 100 fF to 1 pF and input frequency varying from 1 MHz to 20 MHz.

$$\frac{P_{cam}}{K_n V_{dd}} = 9.54\times10^{-14}\left(\frac{C_L \upsilon}{K_n V_{dd}}\right)^3 - 2.23\times10^{-8}\left(\frac{C_L \upsilon}{K_n V_{dd}}\right)^2 + 1.31\times10^{-3}\left(\frac{C_L \upsilon}{K_n V_{dd}}\right) + 1016.5 \quad (4.41)$$

The curve and the experimental data are shown in Figure 4.37. It is interesting to compare this to Figure 4.23. It is a similar function.

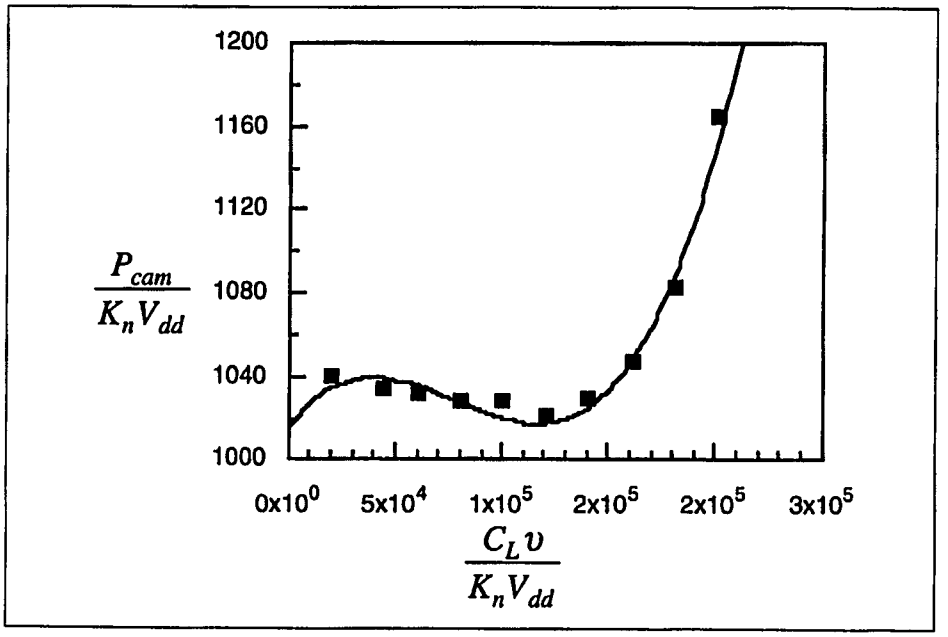


Figure 4.37: Curve Fit for the Power Dissipation in CAM.

4.10 Conclusion

In this chapter area, delay and power macro-models for the register files were developed based on an optimized design of a multi-port memory cell. A general form of the macro-model was derived based on the network equations. Then a specific macro-model is presented using the design parameters obtained in the actual design of a register file. The models developed are then compared to actual layouts of the register files and were shown to be fairly close. An immediate application of the delay model was shown

in the case of various register file configurations where designers can easily compute the trade-offs of area and delay using the models. A specific example was to show the effect of large numbers of ports on some proposed super-scalar architectures. It illustrates how computer architects can use these models to more accurately predict the performance of their machines. An application of the power macros is also shown where the macros are used to compute the power vs. frequency characteristics of the individual components of register files.

Finally, the delay and power macro-models for the content addressable memories are presented which can be used in conjunction with the macros for memory cells to derive models for caches.

CHAPTER V
CONTRIBUTIONS AND FUTURE DIRECTIONS

5.1 Contributions

There exist a wide range of delay macro-models for various components of a processor, most of which, as we have noted, can be divided into two broad categories. One kind is more architecturally oriented and ignores the effect of technology, and the other kind uses technological parameters and ignores the architectural effects. We have developed an approach to macro-modeling that combines both architectural as well as technological parameters that is still fairly simple and easy to evaluate. It is a general method for building macro-models for multi-gate circuits and we have shown that these techniques are applicable to register files and other memory components. As an illustration, we have developed expressions for the area, delay and power models for memory components (and register files in particular) for one particular technology (MOSIS CMOS 1.2 micron process) in this dissertation. The obvious applications of these macro-models are to enable designers of computers to calculate the performance of their designs quickly and accurately. However, these can be applied in a number of related ways: they can be used to determine the area/speed trade-offs as shown in Section 4.6.3 for a register file; they can be used to determine the performance of a new or a published architecture taking into account a more realistic register file delay as in Section 4.7; or they can be applied to derive the frequency characteristics of various components, as shown in Section 4.8.

5.2 Comparison with other Published Work

In this section we briefly compare our models with other published models. Some of this published work on models is described in Chapter 2. As noted in Chapter 4 not many models for calculating power of register files are published. The only instance that we found of power models that can be applied to register files has been in [Kayss93a]. In that work the model was presented as a general form for smaller gates.

The area models for the register file presented by us are more flexible than the register file models presented by Mulder [Mulde91]. The models presented by Mulder, assume that the decoders occupy a constant area, equal to 6 rbes (register bit equivalents). Our decoder model takes the size of the decoder driver as a parameter, resulting in more accurate area prediction. Also, our models take into account the pitch of the metal line in CMOS process while Mulder's model does not. This makes our models more applicable to CMOS processes in particular. Another difference between our approach and Mulder's is the fact that he uses a technology factor for applying his models to processes where the minimum feature width is not the same as one used in his studies him (2 microns). This implies that all the geometries in the cell scale by the same factor. This assumption is not generally valid, especially for the newer sub-micron processes. Our models are much more flexible, allowing for variation of different layer widths independently, thus the poly width can be different from the minimum width of the first metal layer and so on, and each can be individually varied.

Wada's [Wada92] macros are derived for on-chip caches, but with a minor modification these can be applied to register files. The major difference between Wada's approach and ours is that our models take into account the process dependent parameters and the geometry dependent parameters. The process dependent parameters are oxide thickness, electron and hole mobility. Geometry dependent parameters are the width and length of devices. Wada's models use the cache size, line size, associativity, the physical

organization parameters (number of segments per word line and number of segments per bit line) and various input and output capacitances in the circuit. The capacitances depend on the device sizes and the oxide thickness, and therefore indirectly capture the process and geometry dependent parameters. However, these capacitances are not readily available for a given circuit. The only way to determine these capacitances is to construct a layout and then extract them. Only then these can be used in calculating the delay of the circuit. There is no quick way of computing the delay when for example, the width of the decoder driver is doubled, or the number of stages in the decoder is reduced. Our models also use the capacitances, but we have presented rules for calculating the capacitances for register files from the number of read ports and the number of registers.

Ramachandran's [Rama90, Rama92] work in modeling can also be applied to register files. However, we first have to collect all the possible design variations for a memory cell, decoder, sense amplifier, etc., in a library. Once this is done, calculating the delay or power for a register file still requires laying out the register file using the appropriate components from the library and determining its critical path. However, collecting all possible variations of various register file components is time consuming and tedious, further, the last step of laying out the whole register file is also slow. Our models use mathematical interpolation for components, and consequently construction of every instance of each component is not required.

The Alpha power law [Sakur90] is a low level model of a transistor. As such, it can be applied to each transistor of a register file. However, there are approximately 9,000 transistors in a 1 read port register file, and 3,800 nets that connect them. Using a transistor level model for such a circuit is nearly impossible.

5.3 Future Directions

We started this work hoping that all of the microprocessor components can be macro-modeled. We have succeeded in building macro-models for decoders, sense

amplifiers, multi-port RAM's, and CAM's. However, there were some unforeseen problems in modeling other structures such as the ALU's. The different types of ALU's are not easily described by one encompassing model, because of the large number of possibilities e.g., the kind of adder used in the ALU, the number of functions performed by the ALU, the number of pipeline stages, etc. All these variations are in addition to the usual network parameters of fanout, type of driver at the output, size of the output buffer, number of gates between input and output, fanin, and the usual technological parameters such as oxide thickness, threshold voltages, substrate capacitances, etc.

The effects of long metallic lines (interconnects) on the delays in a processor needs to be further investigated. Some models for these exist in the literature, based on transmission line theory, but they are much too detailed to be applied to large circuits like a microprocessor. We built a sample data path with two different kinds of adders, a simple ripple carry adder and a 4 bit CLA adder with propagate and generate circuitry (we assumed that most of the processing in an ALU takes place in an adder, hence an adder is a close approximation of ALU behavior). However, we ran into difficulties after investing a substantial time designing data paths, as it turns out that in the absence of other constraints, it is possible to place the ALU and the RF side by side on a substrate without much routing. Thus there are no long interconnect lines between a register file and the processing elements whose effect could be studied. The effects of the long interconnect lines within the register file are taken care of in the register file models. If we try to artificially place the register file and the ALU apart, the interconnect lines run on substrate. These kind of capacitive and resistive effects are already specified by the process specification documents and no new information is gained. Hence, we could not get any meaningful results. However, the effect of interconnect delay needs to be further investigated, possibly by building a more complete layout of the processor that imposes some of the constraints that we missed in our design. Examples of constraints are the presence of pipeline buffers in between different stages, memory and register bypassing circuitry, TLB's, etc.

APPENDICES

APPENDIX A

LAYOUT AND SCHEMATICS OF THE CELLS

Figure A.1: Layout Layers and Corresponding Patterns.

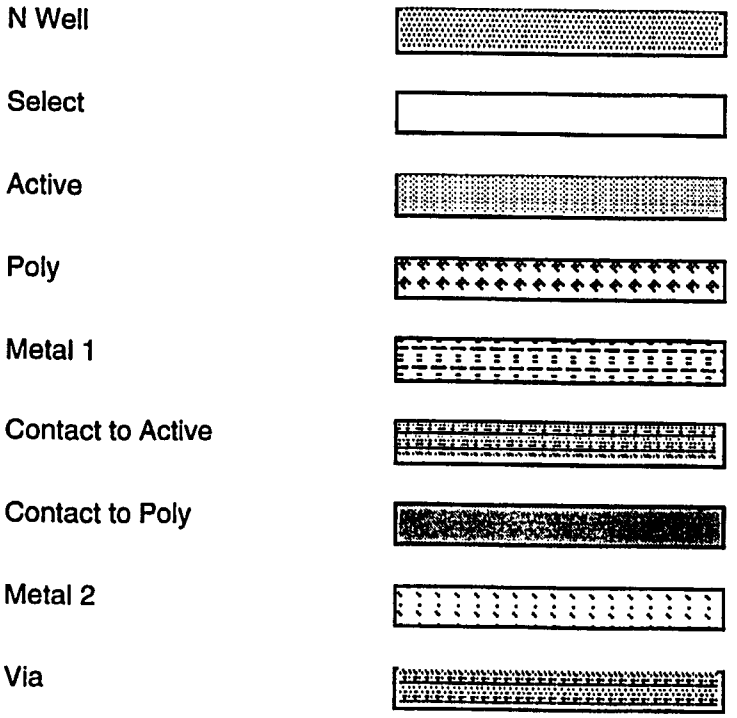


Figure A.2: Layout and Schematic of Memory Cell with 1 Read Port.

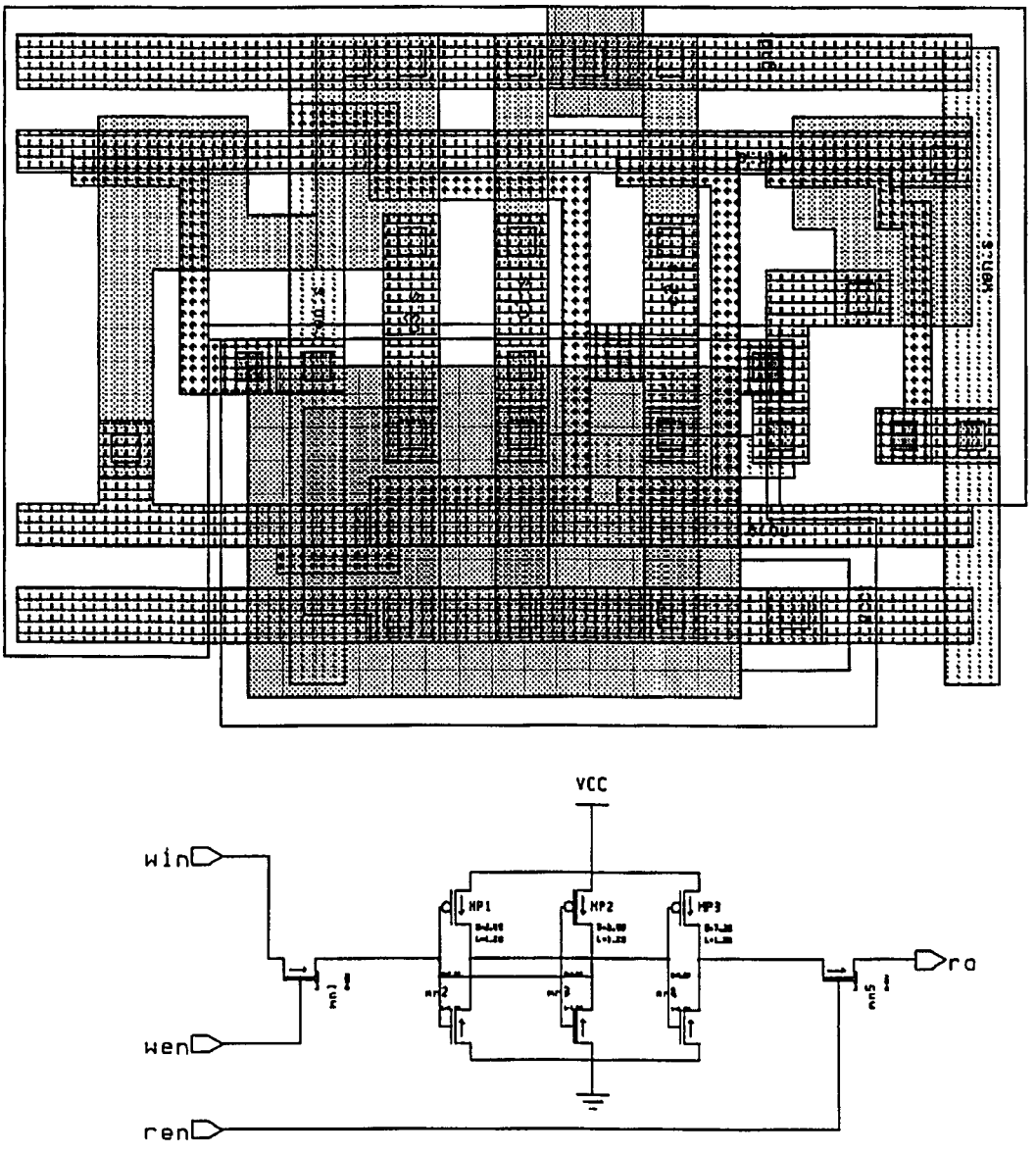


Figure A.3: Layout and Schematic of Memory Cell with 2 Read Ports.

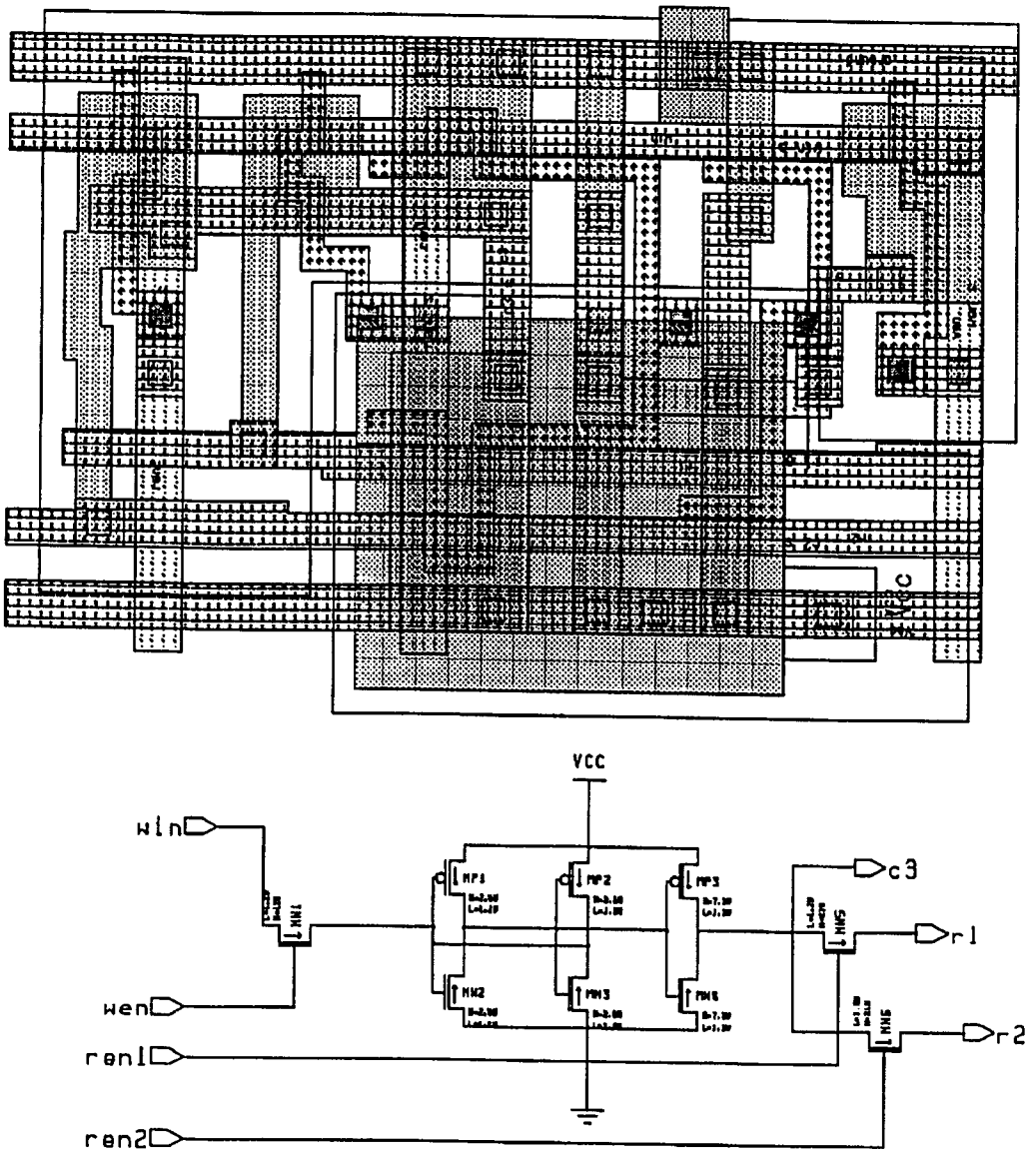


Figure A.4: Layout and Schematic of Memory Cell with 3 Read Ports.

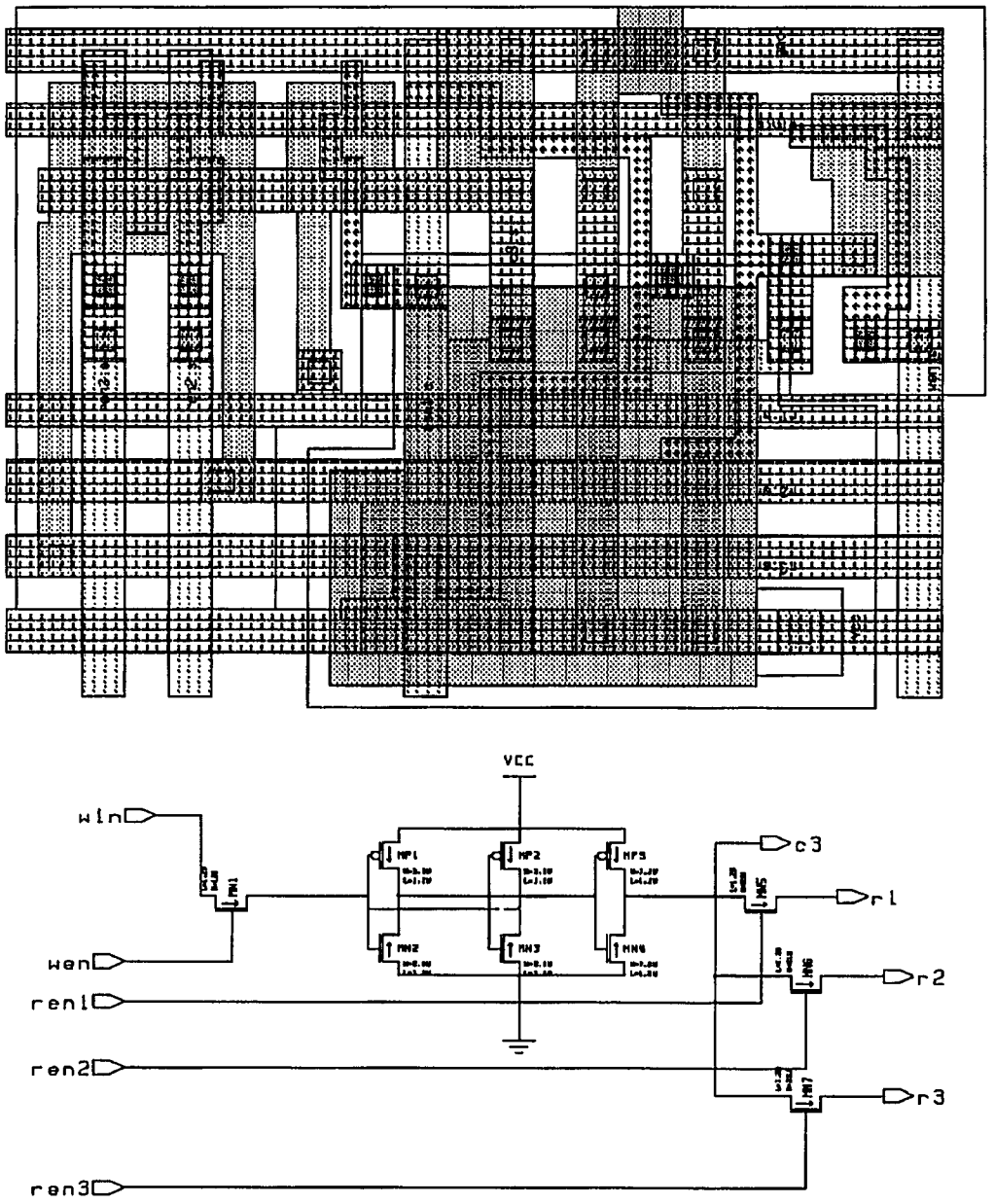
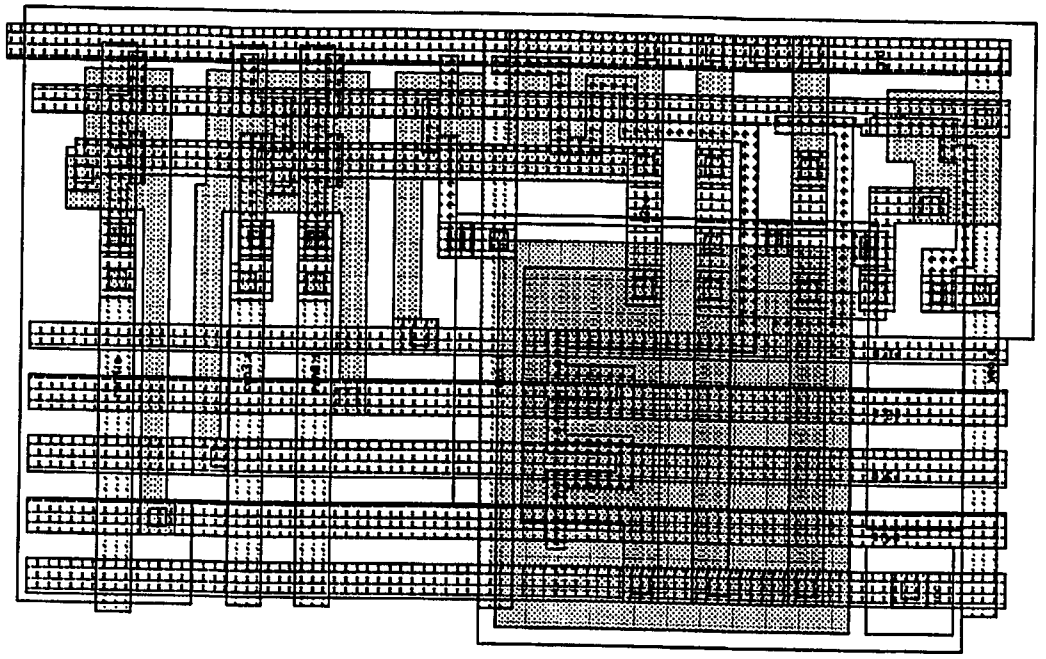


Figure A.5: Layout and Schematic of Memory Cell with 4 Read Ports.



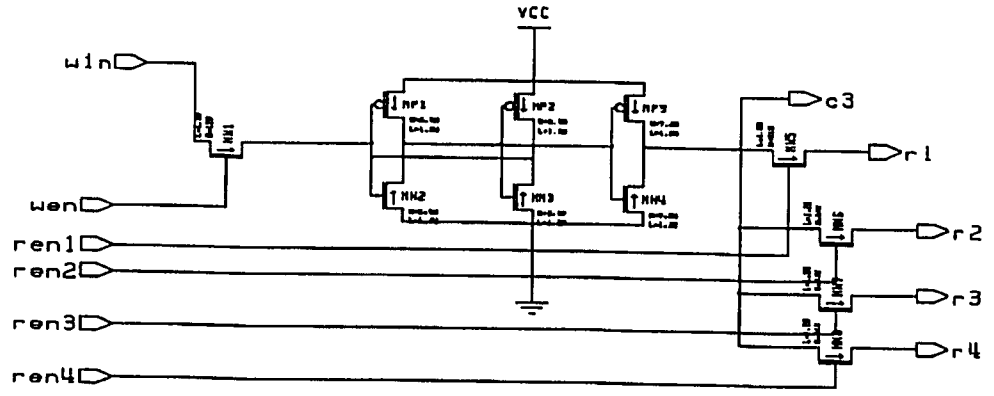
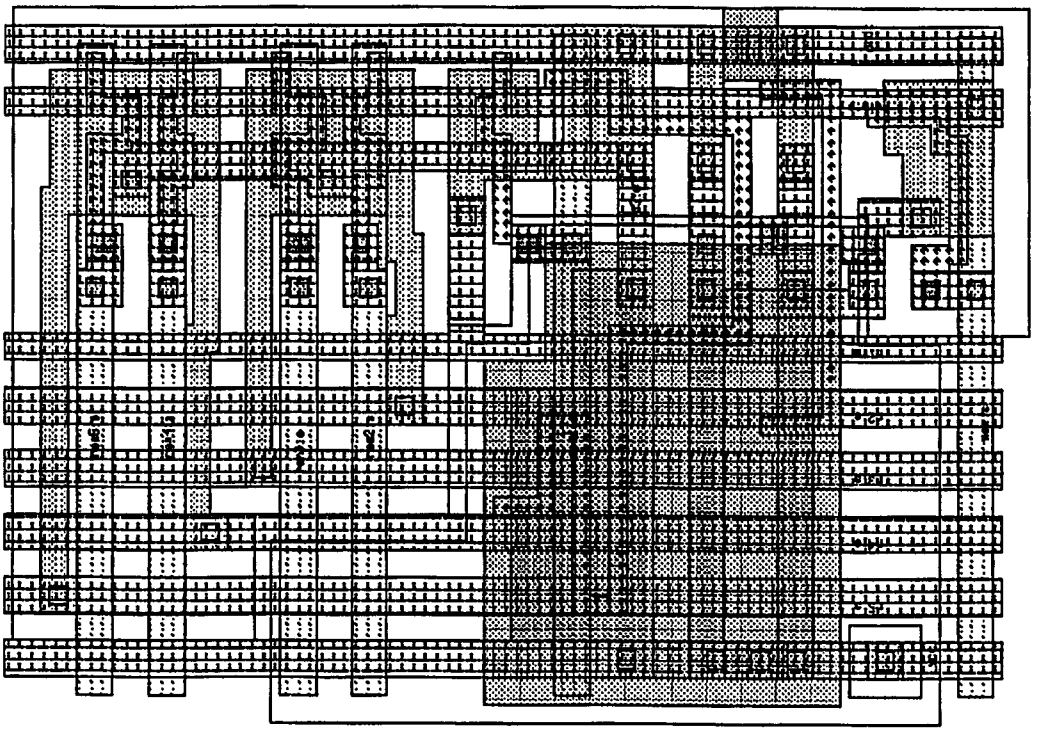


Figure A.6: Layout and Schematic of Memory Cell with 5 Read Ports.



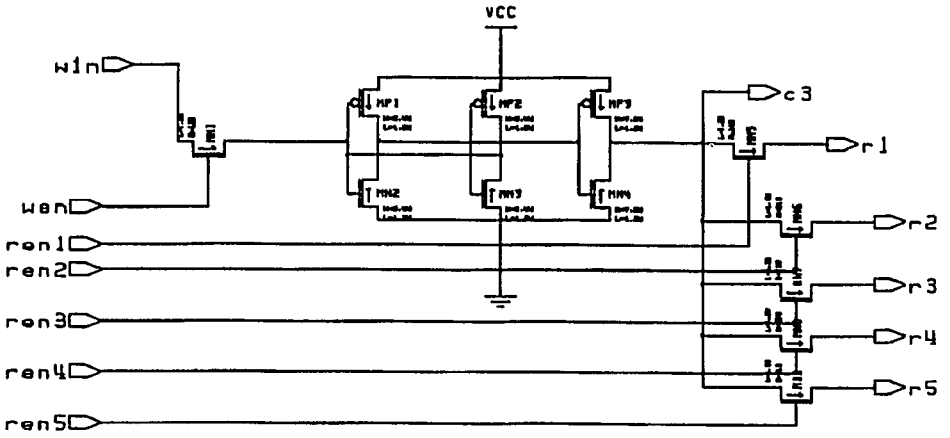
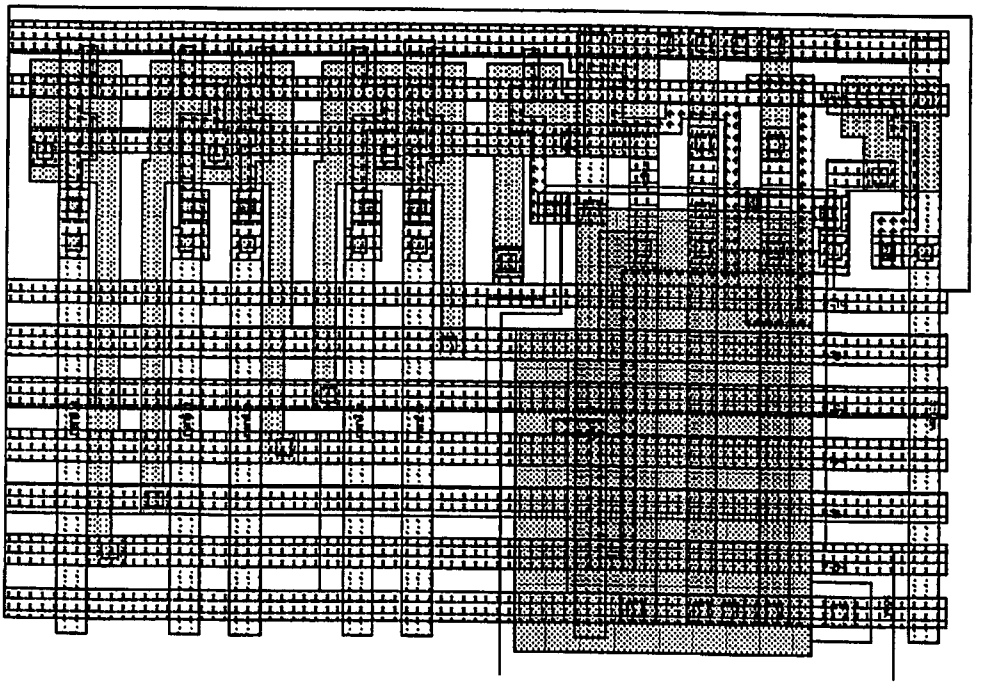


Figure A.7: Layout and Schematic of Memory Cell with 6 Read Ports



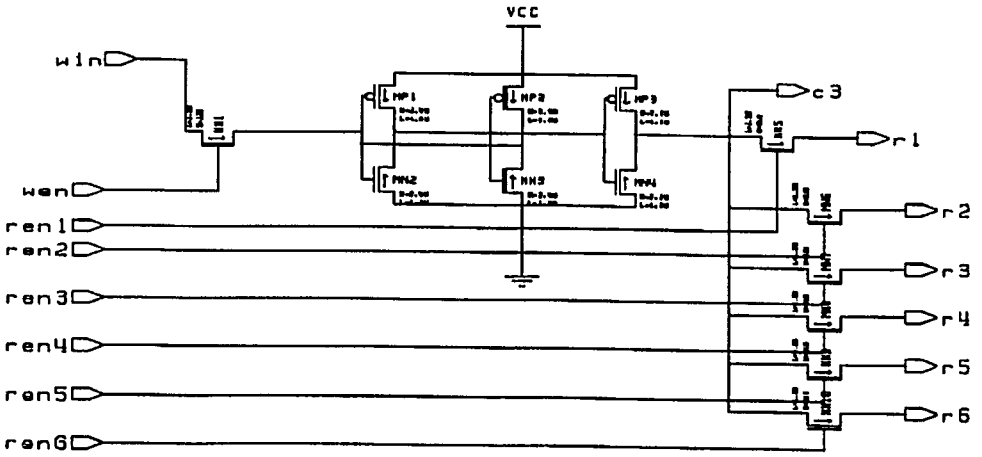


Figure A.8: Layout of Memory Cell with 12 Read Ports.

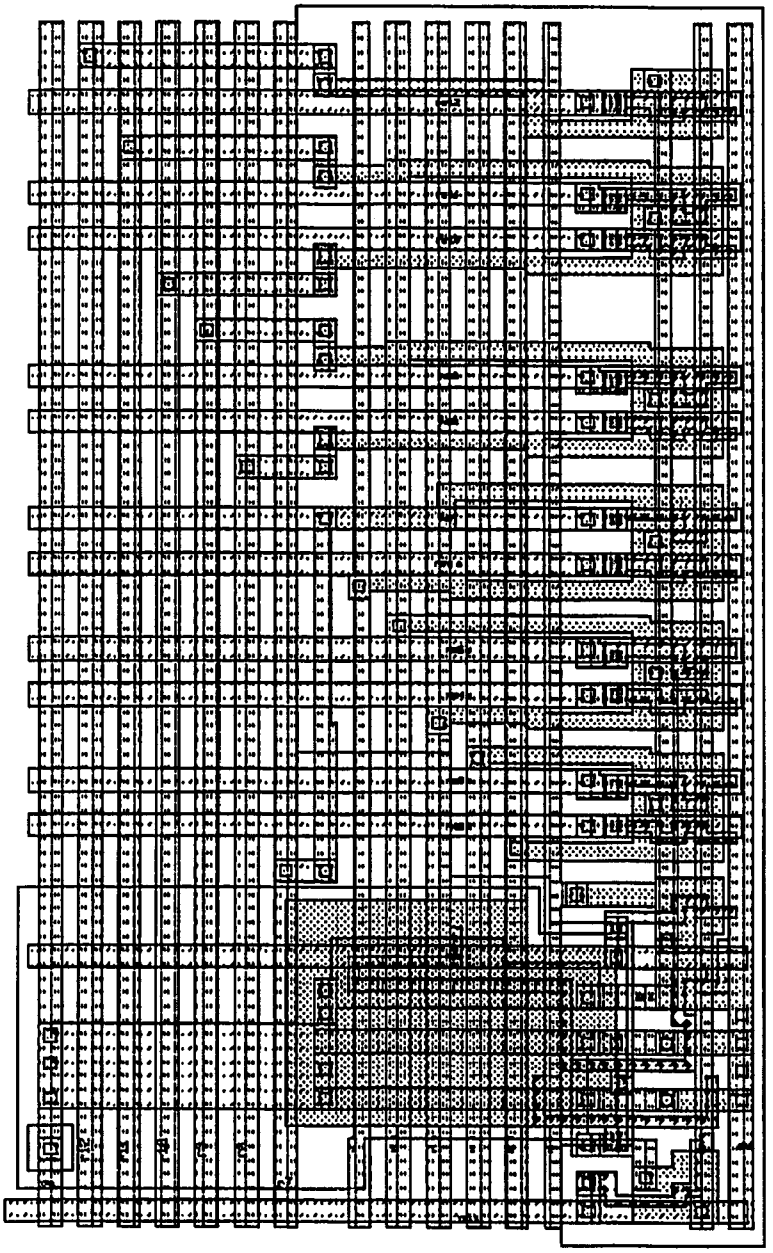


Figure A.9: Schematic of Memory Cell with 12 Read Ports.

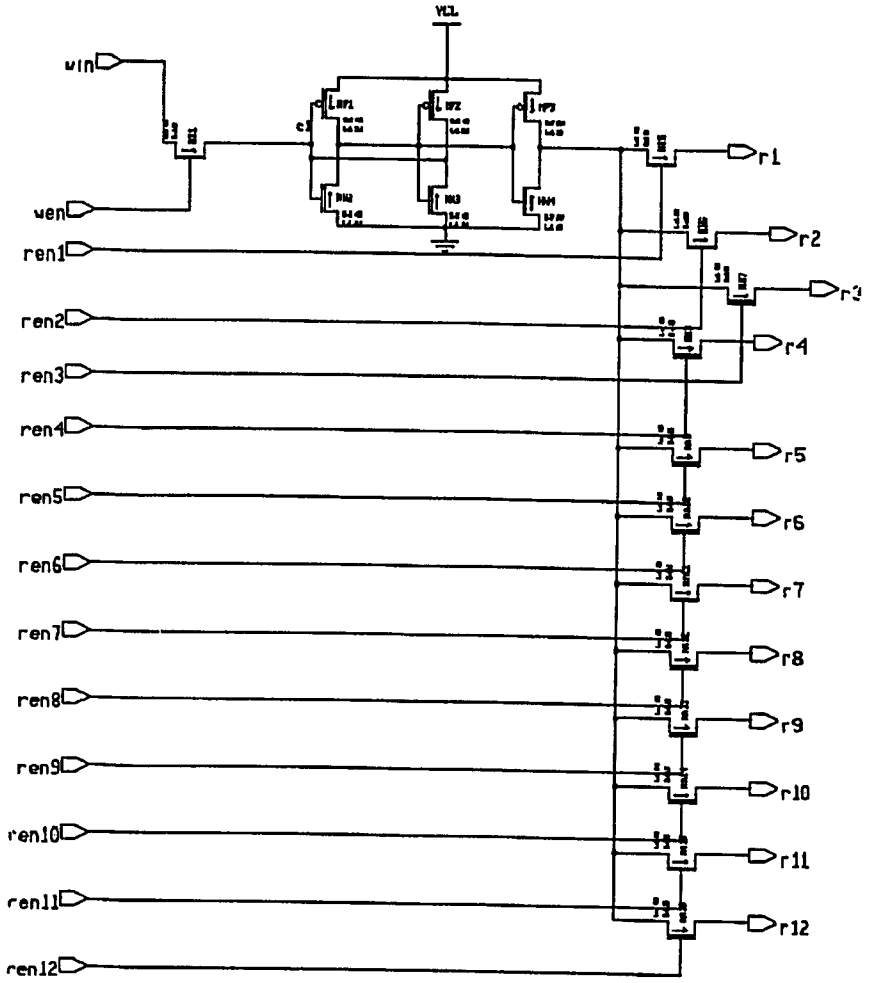


Figure A.10: Layout and Schematic of Decoder Cell.

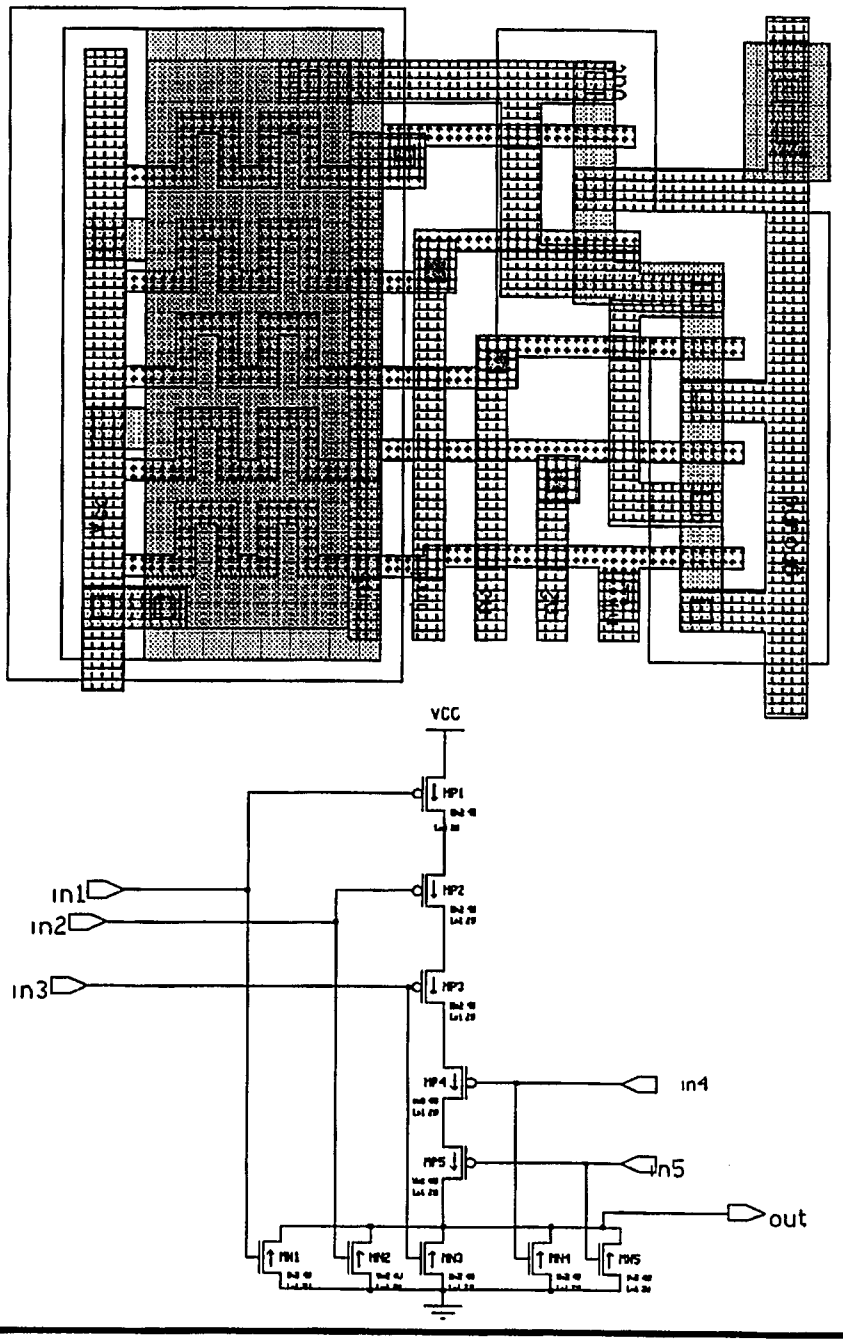
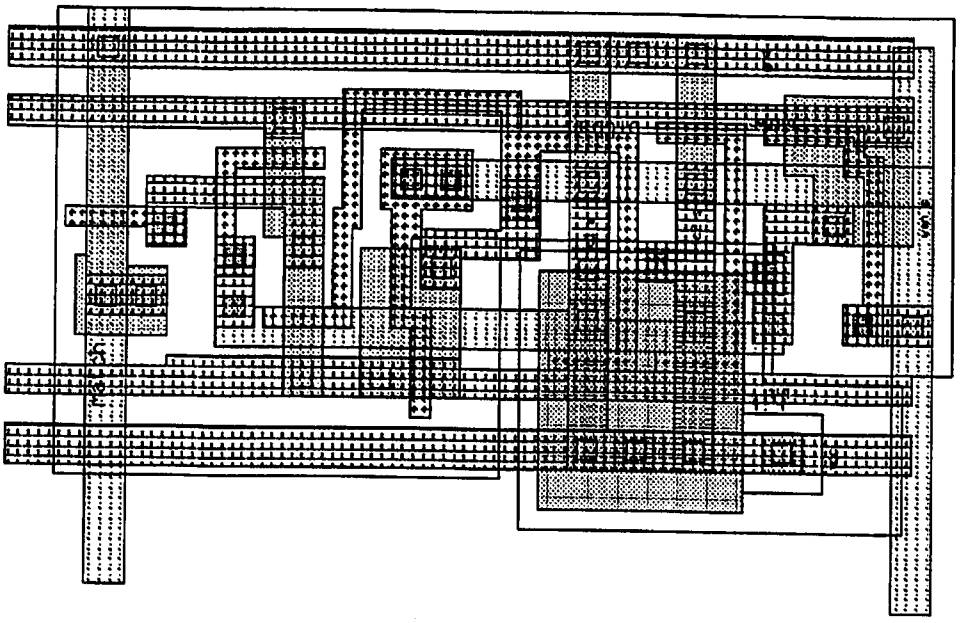


Figure A.11: Layout and Schematic of CAM Cell.



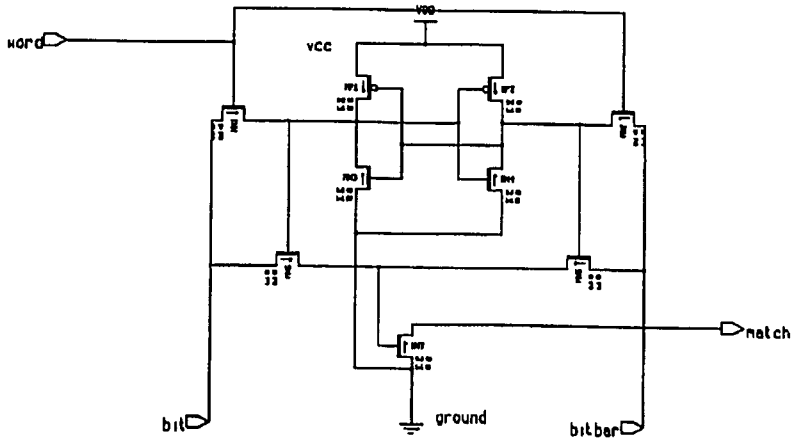
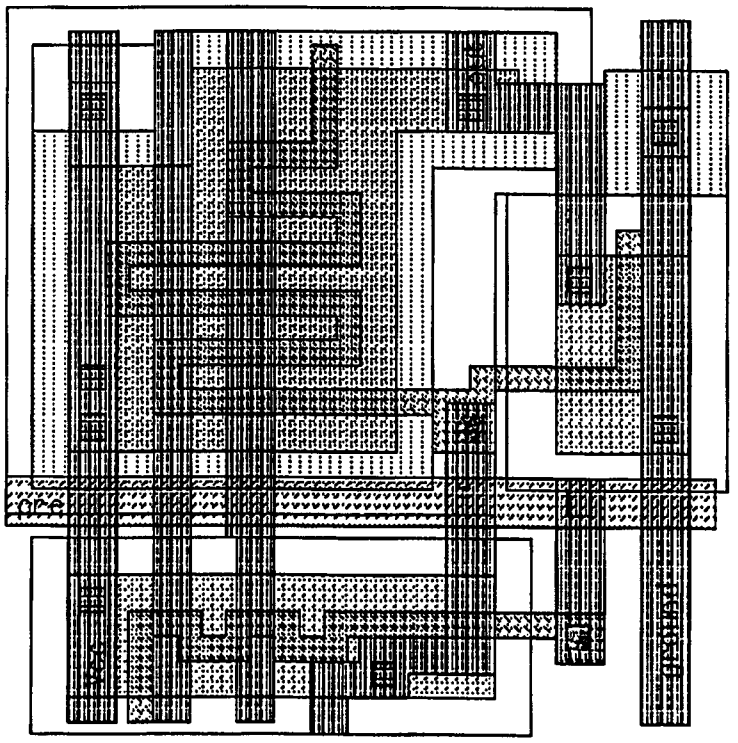
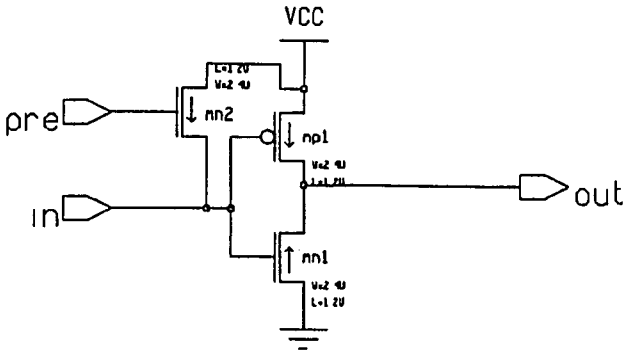


Figure A.12: Layout and Schematic of Sense Amplifier Cell with Pre-Charge Circuitry.





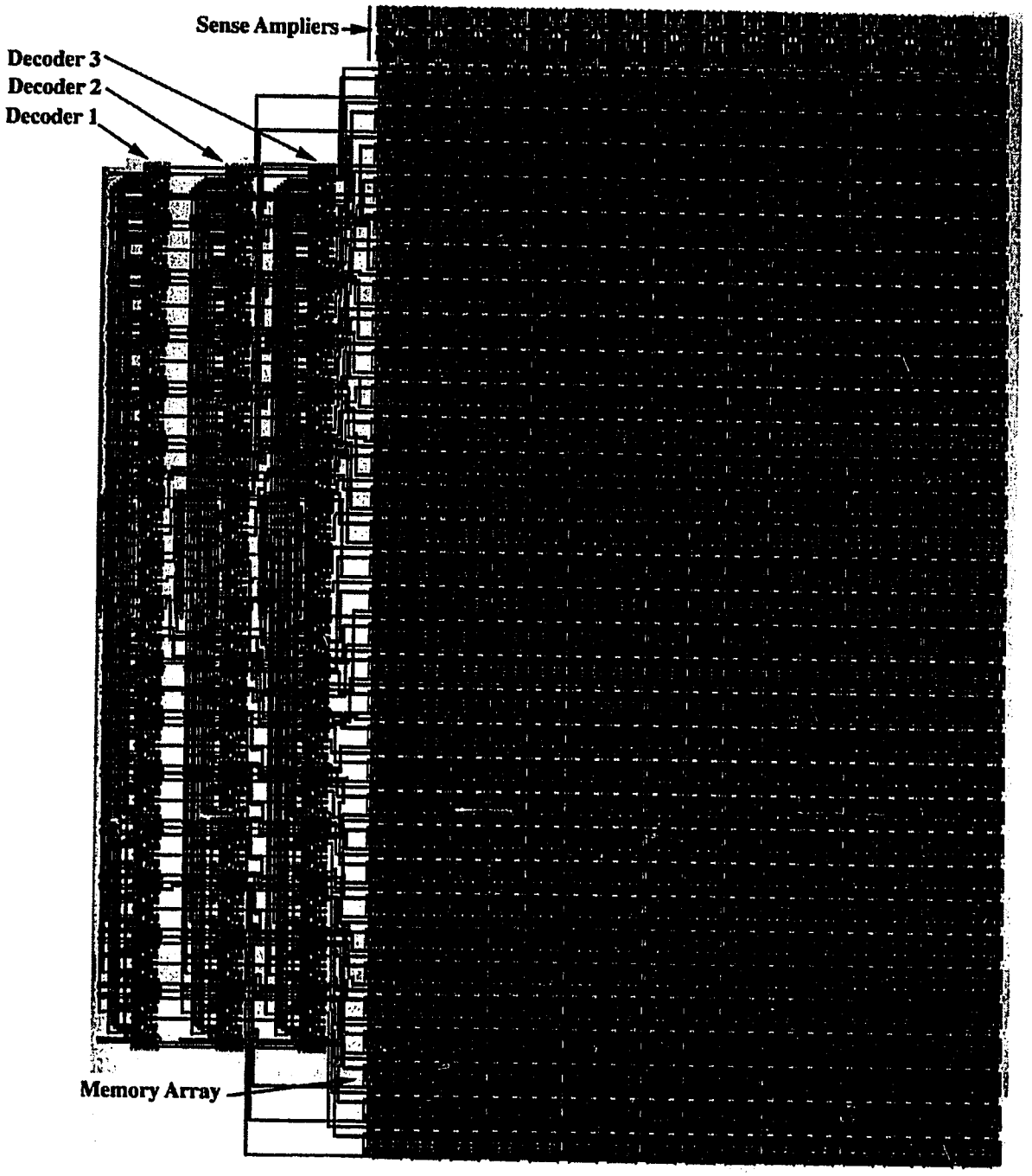


Figure A.13: A Picture of a 2 Read Port Register File Layout.

APPENDIX B

PHYSICAL CONSTANTS

B.1 Constants

Following are values of the physical constants used in Chapters 3 and 4.

ε	Permittivity of material	$= \varepsilon_p \times \varepsilon_0$
ε_0	Permittivity of free space	$= 8.854 \times 10^{-14}$ F/cm
ε_p	Relative permittivity of SiO_2	$= 3.9$
μ_p	Bulk mobility of holes in Si	$= 480$ cm^2 /V.s
μ_n	Bulk mobility of electron in Si	$= 1350$ cm^2 /V.s
t_{ox}	Oxide thickness over substrate	$= 0.5 \times 10^{-6}$ m

BIBLIOGRAPHY

BIBLIOGRAPHY

[Ander67] D. W. Anderson, F. J. Sparacio, and F. M. Tomasulo, "The IBM System/360 Model 91: Machine Philosophy and Instruction-Handling," *IBM J.*, Vol. 11, January, 1967.

[Annar86] M. Annaratone, *Digital CMOS Circuit Design*, Kluwer Academic Publishers, 1986.

[Bakog90a] H. B. Bakoglu and T. Whiteside, "RISC System/6000 Hardware Review," *IBM RISC System/6000 Technology*, pp. 8-15, 1990.

[Bakog90b] H. B. Bakoglu, "Circuits, Interconnections and Packaging of VLSI," 1990, Addison-Wesley Publishing Company.

[Blalo91] T. N. Blalock and R. C. Jaeger, "A High-Speed Clamped Bit-Line Current Mode Sense Amplifier," *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 4, pp. 542-548, April, 1991.

[Box87] G. E. Box and N. R. Draper, *Empirical Model Building and Response Surfaces*, John Wiley and Sons, 1987.

[Brocc88] L. M. Brocco, S. P. McCormick and J. Allen, "Macromodeling CMOS Circuits for Timing Simulation," *IEEE Transactions on Computer-Aided Design*, Vol. 7, No. 12, pp. 1237-1249, December, 1988.

[Bucki15] E. Buckingham, "Model Experiments and Forms of Empirical Equations," *Transactions of ASME*, Vol. 37, pp. 263-296, 1915.

[Butle91] M. Butler, T. Yeh, Y. Patt, M. Alsup, H. Scales and M. Shebanow, "Single Instruction Stream Parallelism is Greater than two," *Computer Architecture News*, Vol. 19, No. 3, pp. 276-286, March, 1991.

[Casin91] G. Casinovi and A. Sangiovanni-Vincentelli, "A Macromodeling Algorithm for Analog Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 10, No. 2, pp. 150-160, February, 1991.

[Conte93a] T. M. Conte, K. N. P. Menezes and S. W. Sathaye, "The Impact of Power and Area Efficiency on Superscalar Processor Design," Technical Report Computer Architecture Research Laboratory, University of South Carolina, Columbia, SC 29208.

[Conte93b] T. M. Conte and B. A. Patel, "Auto-Aligning Decoder Mechanisms for High Issue Rate Superscalar Processors," Computer Architecture Research Laboratory, University of South Carolina, Columbia, SC 29208.

[DA93] *Design Architect User's Manual*, Software version 8.2, Vol. 1-2, Mentor Graphics Corporation, 1993.

[Delta91] *DeltaGraph Professional for Macintosh: The Art and Science of Charting, Graphics and Presentations*, Deltapoint Inc., 1991.

[Embab91] S. H. K. Embabi, A. Bellaouar and M. I. Elmasry, "Analysis and Optimizations of BiCMOS Digital Circuit Structures," *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 4, pp. 676-679, April, 1991.

[EPOCH93] *EPOCH Designer's Handbook*, Cascade Design Automation, 1993.

[Focke53] C. M. Focken, *Dimensional Methods and their Applications*, Edward Arnold and Co, 1953.

[Hammi62] Hamming, *Numerical Methods for Scientists and Engineers*, McGraw Hill Book Company, 1962.

[Hayes88] J. P. Hayes, *Computer Architecture and Organization*, McGraw-Hill Book Company, 1988.

[Heden87] N. Hedenstierna and K. O. Jeppson, "CMOS Circuit Speed and Buffer Optimization," *IEEE Transactions on Computer-Aided Design*, Vol. CAD-6, No. 2, pp. 270-281, March, 1987.

[Henne91] J. L. Hennessy and N. P. Jouppi, "Computer Technology and Architecture: An Evolving Interaction," *Computer*, pp. 18-29, September, 1991.

[Hinds91] R. S. Hinds, S. R. Canaga, G. M. Lee and A. K. Choudhury, "A 20K GaAs Array with 10K of Embedded SRAM," *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 3, pp. 245-256, March, 1991.

[Holli78] R. J. Hollingsworth, A. C. Ipri and C. S. Kim, "A CMOS/SOS 4K Static RAM," *IEEE Journal of Solid-State Circuits*, Vol. SC-13, No. 5, pp. 664-674, October, 1978.

[Horow87] M. Horowitz and J. L. Hennessy, "A 32-bit Microprocessor with On-Chip 2K byte Instruction Cache," *IEEE International Solid-State Conference*, pp. 30-31, 1987.

[Howes94] R. Howes, Redman-White, K. G. Nichols, P. J. Mole, M. J. Robinson and S. Bird, "An SOS MOSFET Model Based on Calculation of the Surface Potential," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 4, pp. 494-505, April, 1994.

[Hspic89] *HSPICE User's manual*, Meta-Soft Corporation, 1989.

[Hwang84] K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*, McGraw Hill Book Company, 1984.

[ICGra93] *IC-Station Reference Manual*, Software version 8.2, Vol. 1-3, Mentor Graphics Corporation, 1993.

[ICLin93] *IC Link User's Manual*, Mentor Graphics Corporation, 1993.

[Intel93] *The Intel Pentium Processor: A Technical Overview*, Intel Corporation, 1993.

[Jha93] P. K. Jha and N. D. Dutt, "Rapid Estimation for Parametrized Components in High-Level Synthesis," *IEEE Transactions on Very Large Scale Integration,* Vol. 1, No. 3, pp. 296-303, September, 1993.

[Joupp89a] N. P. Jouppi, "Super-Scalar vs. Super-Pipelined Machines," Digital Equipment Corporation Western Research Labs, May, 1989.

[Joupp89b] N. P. Jouppi and D. W. Wall, "Available Instruction-Level Parallelism for Super-scalar and Super-pipelined Machines," *ASPLOS-III Proceedings,* pp. 272-282, 1989.

[Joupp89c] N. P. Jouppi, J. Y. F Tang and J. Dion, "A 20 MIPS sustained 32b microprocessor with 64b data bus," in *IEEE International Solid-State Conference Digest of Technical papers,* pp. 84-85, 1989.

[Kadot87] H. Kadota, J. Miyake, I. Okabayashi, et al., "A CMOS 32b Microprocessor with On-Chip Cache and Transmission Lookahead Buffer," in *IEEE International Solid-State Circuits Conference,* 1987.

[Kawar78] K. Kawarda, M. Suzuki, H. Mukai, et al., "A Fast 7.5 ns Access 1 Kbit RAM for Cache Memory Systems," *IEEE Journal of Solid-State Circuits,* Vol. SC-13, No. 5, pp. 656-663, October, 1978.

[Kayss92] A. I. Kayssi, K. Sakallah and T. M. Burks, "Analytical Transient Response of CMOS Inverters," *IEEE Transactions on Circuits and Systems,* Vol. 39 No. 1, pp. 42-45, January, 1992.

[Kayss93a] A. I. Kayssi, *A Methodology for the Construction of Accurate Timing Macro-Models for Digital Circuits*, Ph. D. Thesis, University of Michigan, 1993.

[Kayss93b] A. I. Kayssi, K. A. Sakallah and T. N. Mudge, "The Impact of Signal Transition Time on Path Delay Computation," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing,* Vol. 40, No. 5, pp. 302-309, May, 1993.

[Keyes82] R. W. Keyes, "The Wire Limited Logic Chip," *IEEE Journal of Solid State Circuits,* Vol. SC-17 No. 6, pp. 1232-1233, 1982.

[Lilja94] D. J. Lilja, "Exploiting the Parallelism Available in Loops," *Computer,* Vol. 27 No. 2, pp. 13-26, February, 1994.

[Maly92] W. Maly, M. Patrya, A. Primatic, V. Raghavan and A. W. T. Storey, "Memory Chip for 24-Port Global Register File," Personal Communication.

[McGea90] S. McGeady, "The i960CA Superscalar Implementation of the 80960 Architecture," *IEEE COMPCON Digest of Papers,* pp. 232-239, 1990.

[Miya86] H. Miyanagi, S. Konaka, Y. Kobayashi, et al., "A 0.85-ns 1-kbit ECL RAM," *IEEE Journal of Solid-State Circuits,* Vol. 21, No. 5, pp. 501-504, August, 1986.

[Molne89] K. Molner, C. -Y. Ho, D. Staver, B. Davis, et al., "A 40 MHz 64-bit floating point processor," in *IEEE International Solid-State Conference Digest of Technical papers,* pp. 48-49, 1989.

[Motor93] *PowerPC 601 RISC Microprocessor User's Manual*, Motorola Inc., 1993.

[MPR93] L. Gwennap, "Digital Leads the Pack with 21164, "*Microprocessor Report,*. pp. 1, 6-10, September, 1993.

[MPR94] L. Gwennap, "Cyrix Describes Pentium Competitor," *Microprocessor Report,* pp. 1, 6-10, October, 1994.

[Mulde91] J. M. Mulder, N. T. Quach and M. J. Flynn, "An Area Model for On-Chip Memories and its Application," *IEEE Journal of Solid-State Circuits,* Vol. 26, No. 2, pp. 98-106, February, 1991.

[Patte94] D. A. Patterson and J. L. Hennessy, "Computer Organization and Design: The Hardware/Software Interface," 1994, Morgan Kaufman Inc., San Mateo, California.

[Pohm83] A. V. Pohm and O. P. Agarwal, "High Speed Memory Systems," 1983, Reston Publishing Company, Inc., Reston, Virginia.

[Press92] W. H. Press, S. A. Tevkolsky, W. T. Vettering and B. P. Flannery, *Numerical Recipes in C "The Art of Scientific Computing,"* 2nd ed., New York: Cambridge University Press, 1992.

[Rama90] C. Ramachandran and F. J. Kurdahi, "LAST: A Layout Area and Shape Function Estimator for High Level Applications," Technical Report ECE-90-23, VLSI Synthesis and Design Automation Lab, Dept. of ECE, University of California, Irvine, CA 92717, October 1990.

[Rama92] C. Ramachandran and F. J. Kurdahi, "TELE2.0: A Combined Topological and Functionality-Based Delay Estimation Tool Using Layout Driven Approach for High Level Applications," Technical Report ECE-92-04, University of California, Irvine, CA 92717, February 1992.

[Rande82] B. Randell and P. C. Treleaven, *VLSI Architecture*, Prentice Hall International, 1982.

[Rao93] D. S. Rao and F. J. Kurdahi, "Hierarchical Design Space Exploration for a Class of Digital Systems," *IEEE Transactions on Very Large Scale Integration,* Vol. 1, No. 3, pp. 282-295, September, 1993.

[Sakur90] T. Sakurai and A. R. Newton, "Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas," *IEEE Journal of Solid-State Circuits,* Vol. 25, No. 2, pp. 584-593, April, 1990.

[Sawad89] K. Sawada, T. Sakurai, K. Nogami, et al., "A 32-kbyte Integrated Cache Memory," *IEEE Journal of Solid-State Circuits,* Vol. 24, No. 4, pp. 881-887, August, 1989.

[Schae85] T. J. Schaefer, "A Transistor-Level Logic-With-Timing Simulator for MOS Circuits," in *22nd Design Automation Coference,* 1985.

[Schic68] H. Schichman and D. A. Hodges, "Modeling and Simulation of Insulated-Gate Field Effect Transistor Switching Circuit," *IEEE Journal of Solid State Circuits,* Vol. SC-3, pp. 285, March, 1968.

[Shara94] K. M. Sharaf and M. I. Elmasry, "An Accurate Analytical Propagation Delay Model for High-Speed CML Bipolar Circuits," *IEEE Journal of Solid State Circuits,* Vol. 29, No. 1, pp. 31-45, January, 1994.

[Shin91] H. Shinohara, N. Matsumuto, et al., "A Flexible Multiport RAM Compiler for Data Path," *IEEE Journal of Solid-State Circuits,* Vol. 26, No. 3, pp. 343-348, March, 1991.

[Smith89] M. D. Smith, M. Johnson and M. A. Horowitz, "Limits on Multiple Instruction Issue," in *Third International Conference on Architectural Support for Programming Languages and Operating Systems,* 1989.

[Stone91] H. S. Stone and J. Cocke, "Computer Architecture in 1990's," *Computer,* pp. 30-37, September, 1991.

[Stone93] H. S. Stone, *High Performance Computer Architecture*, 3rd ed., Addison-Wesley Publishing Company, 1993.

[Taylo74] E. S. Taylor, *Dimensional Analysis for Engineers*, Clarendon Press Oxford, 1974.

[Thomp80] C. D. Thompson, *A Complexity Theory for VLSI*, Ph. D. Thesis, Carnegie-Mellon University, 1980.

[Thorn70] J. E. Thornton, *Design of a Computer: The Control Data 6600*, Scott Foresman and Company, 1970.

[Tran88] H. V. Tran, D. B. Scott, P. K. Fung, et al., "An 8-ns 256K ECL SRAM with CMOS Memory Array and Battery Backup Capability," *IEEE Journal of Solid-State Circuits,* Vol. 23, No. 5, pp. 1041-1046, October, 1988.

[Uneka94] Y. Unekawa, T. Kobayashi, T. Shirotori, et al., "A 110-MHz/1-Mb Synchronous TagRAM," *IEEE Journal of Solid-State Circuits,* Vol. 29, No. 4, pp. 403-408, April, 1994.

[Wada92] T. Wada, S. Rajan and S. A. Przybylski, "An Analytical Access Time Model for On-Chip Cache Memories," *IEEE Journal of Solid-State Circuits,* Vol. 27, No. 8, pp. 1147-1156, August, 1992.

[Wall91] D. W. Wall, "Limits on Instruction Level Parallelism," in *Architectural Support for Programming Languages and Operating Systems,* Santa Clara, California, ACM, 1991.

[Weste88] N. Weste and K. Eshraghian, *Principles of VLSI Design: A Systems Perspective*, Addison-Wesley Publishing Company, 1988.

[Wilki91] B. Wilkinson, *Computer Architecture*, Prentice Hall International, 1991.

[Wilto94] S. J. E. Wilton and N. P. Jouppi, "An Enhanced Access and Cycle Time Model for On-Chip Caches," WRL Research Report 93/5, Western Research Laboratory, Digital Equipment Corporation, June 1994.

[Wong94] W. W. Wong and J. J. Liou, "JFET Circuit Simulation Using SPICE Implemented with an Improved Model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 13, No. 1, pp. 105-109, January, 1994.

[Yang88a] T. S. Yang, M. A. Horowitz and B. A. Wooley, "A 4 nsec. 4K X 1 bit Two-Port BiCMOS SRAM," in *IEEE Custom Integrated Circuits Conference,* 1988.

[Yang88b] T.-S. Yang, M. A. Horowitz and B. A. Wooley, "A 4ns 4K X 1 bit Two-Port BiCMOS SRAM," *IEEE Journal of Solid-State Circuits,* Vol. 23, No. 5, pp. 1030-1040, October, 1988.

[Yette87] J. Yetter, M. Forsyth, W. Jaffe, et al., "A 15 MIPS 32b Microporcessor," in *IEEE International Solid-State Circuits Conference,* 1987.