To the Graduate Council:

I am submitting herewith a dissertation written by P. R. Mukund entitled "OPTIMAL CLOCK DISTRIBUTION IN VLSI SYSTEMS." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Electrical Engineering.

Donald W. Bouldin

D. W. Bouldin, Major Professor

We have read this dissertation
and recommend its acceptance:

James M. Rochelle

E J Kennedy

Robert B Bodenheimer

D J Strojk

Accepted for the Council:

Edward Winkel

Vice Provost
and Dean of The Graduate School

OPTIMAL CLOCK DISTRIBUTION IN VLSI SYSTEMS

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

P. R. Mukund

December 1990

Dedicated to

my mother, Smt. Tara Bai

and

the memory of my father,

late Sri. P. Raghavendra Rao

ACKNOWLEDGEMENTS

It is a pleasure to acknowledge my gratitude to Professor D. W. Bouldin. Apart from being my dissertation advisor, he has been a true friend. I sincerely appreciate his continued interest in my professional advancement. Prof. E. J. Kennedy has not only been on my doctoral committee, but has taught me a great deal in the past. The advice of my other committee members, Professors J. M. Rochelle, R. E. Bodenheimer and D. W. Straight, is sincerely appreciated.

The following people read an early draft proposal of this research and provided many valuable comments: Dr. H. B. Bakoglu of International Business Machines, Prof. Jack McDonald of Rensselaer Polytechnic Institute, Dr. Jack Raffel of Lincoln Labs (MIT) and Major Rich Linderman of Rome Air Development Center. I sincerely thank all of them for sparing me their valuable time.

During the last four years, several professors in the ECE department at the University of Tennessee have helped me in one way or another. They include Professors J. M. Googe, W. L. Green, T. V. Blalock, J. W. Waller, R. C. Gonzalez, M. M. Trivedi and M. A. Abidi. I am grateful to all of them. I thank Ms. Jenny Daniel for the help in preparing the figures and Ms. Janet Smith for help in making the tables in this dissertation.

The support of my family has always been an essential ingredient in my quest for education. In particular, I want to gratefully acknowledge the constant support of my wife, Vanditha. The many technical discussions I have had with

her on optimization techniques have been educational. My daughters, Kavitha and Anupama, make everything worth while.

My fellow graduate students, too many to mention individually, have provided an intellectually stimulating atmosphere. In particular, Mr. Hrishikesh Gadagkar gave many helpful hints on data structures. The enthusiasm of many of my undergraduate students has been instrumental in my choosing an academic career.

In the last three years, my research activities have been supported by a fellowship from the Perceptics Corporation, an assistantship from the U.S. Department of Energy, and the ECE department at the University of Tennessee.

ABSTRACT

Global electrical design issues such as power distribution and clock distribution in VLSI design have assumed added dimensions as feature sizes below one micron and device counts of more than a million are becoming feasible. This dissertation presents a methodology for optimally distributing the clock signal in synchronous digital VLSI circuits. Optimality is measured in terms of minimizing delay in each critical path as well as the differences in delays between different paths. The inputs to the system consist of the location of each cell, the input capacitance of each cell and the locations of all possible buffer sites. The outputs consist of the optimal number and location of buffers as well as a netlist for routing the clock signal. This system provides a formal application specific methodology for optimal clock distribution whose utility increases with the size of the circuit.

TABLE OF CONTENTS

CHAPTER PAGE

CHAPTER PAGE

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

Introduction

1.1 General

Constraints in digital microelectronic systems can be broadly classified into those related to architectural issues and those related to electrical design issues. Architectural issues dictate efficient mapping of algorithms to hardware, and often tend to be application specific. On the other hand, electrical design issues are more general in nature and dictate the very success of a given technology. As technology advances, some of the electrical design constraints become severe since device dimensions reduce and device densities increase. Two of the most important electrical design issues are the distribution of power (and ground) and assurance of proper timing. This work deals with the issue of timing in digital VLSI systems.

The design of any digital system encompasses a major effort in the temporal domain. This is due to the fact that signals have to be stored and processed at fairly precise intervals. Yet, these signals have to travel over interconnects whose delays and characteristics cannot always be accurately predicted. To compound the complexity of the problem, the chosen technology and the size of the system

have direct bearings on the delay in signal propagation. For example, as device characteristics have an effect on the delay between two devices, the problems associated with CMOS technology are different from the problems associated with bipolar ECL technology. Further, the dimensions of the interconnects are different in different systems. These variations have often forced designers to deal with the timing problem on a case to case basis. In very large systems, such an exercise is often futile and results in costly mistakes. This dissertation provides a methodology for the optimal distribution of the clock signal with respect to minimal clock skew in synchronous digital VLSI systems.

1.2 Timing Methodologies

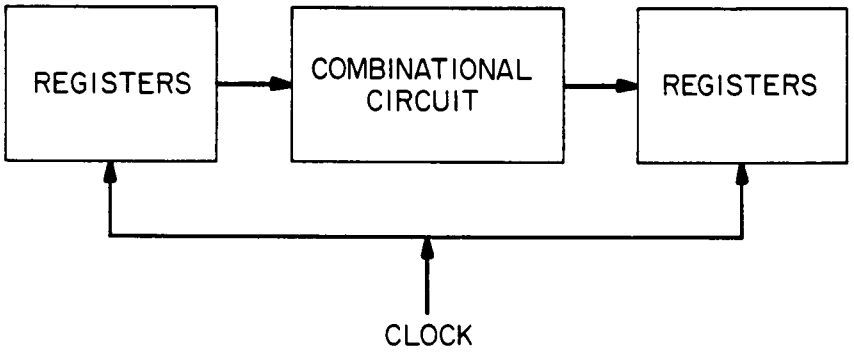
At present, two broad timing methodologies exist: synchronous and asynchronous. Though there are many variations, synchronous systems are clock driven whereas asynchronous systems are event driven. There are advantages and disadvantages in both methodologies. The synchronous system has the major advantage of having been the most popular methodology for several decades. As such, various clocking schemes and circuits exist. The main disadvantage is its sensitivity to delays in buffers and interconnects resulting in the clock skewing. On the other hand, asynchronous system design has been practiced by a very small number of designers due to its radical divergence from tested theories in synchronous design [1,2,3]. As such, many of the asynchronous methodolo-

gies remain untested for robustness in large systems. Asynchronous systems are also generally considered area inefficient [4]. However, being event driven, asynchronous systems are not hampered by some of the problems associated with clock distribution such as skewing.

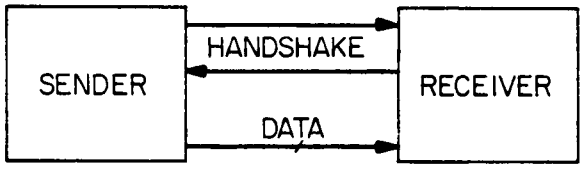
1.2.1 Synchronous Systems

In general, digital signals are manipulated in gates and stored in registers, as depicted in figure 1-1(a). The need for synchronization arises due to varying propagation delays in the path of various signals. The clock signal ensures that valid data is interpreted and stored by holding all signals occasionally until all the signals arrive [4].

The simplest means of achieving synchronization is the single-phase clock. Here, the entire system is driven by a single clock signal. In spite of being the most area efficient, a single-phase clock has one major disadvantage. The maximum delay in any combinational logic block has to be less than one period of the clock. On the other hand, all combinational logic delays must be greater than the ON period of the clock. This poses a severe constraint on the design by defeating the original purpose of the clock, which is to overcome the unpredictable nature of the delays. This problem is generally overcome with the use of two non-overlapping phases of the clock. The use of a two-phased clock removes one of the restrictions in the single-phased clock. Signals may pass as fast as possible, and yet not generate a race or hazard condition. However, routing



(a)



(b)

Figure 1.1: (a) Synchronous System (b) Self-Timed System

two phases of the clock in the entire system requires more area. Still, in large systems, the use of multiple phases of a clock, and even multiple clocks are not uncommon in order to minimize the complexity of the design.

Clock signals are either from an external source or internally generated. If the clock is external and the generator has infinite fan-out, then there is no need for buffers within the system. However, this is seldom the case. Clocks are often internally generated, or derived from a master clock, and buffers are needed to distribute the clock throughout the system. Re-synchronization is sometimes performed with the use of a phase locked loop [5,6,7,8]. The use of buffers to occasionally increase the drive capability introduces delays. Further, the delays in the interconnects (R-C time constant of the metallization) add to the delay of the clock. Differences in the delays result in the skewing of the clock signal. Elimination of clock skew is possible only if the delay in all the paths are exactly the same. In practice, this is an impossible situation. For example, cells in a semi-custom VLSI system are placed based on communication and other architectural constraints. It is nearly impossible to balance not only the loads but also the physical distances. Hence, it is essential that a methodology be derived for the distribution of the clock signal such that it arrives at various modules at precisely desired time intervals. Such a methodology would provide a means of not only distributing the clock but also reducing the area required for the distribution.

1.2.2 Self-Timed Systems

The conceptual framework in which self-timed systems are designed is entirely different from the clocked synchronous system design [3]. Self-timed systems are also referred to as asynchronous systems [2], delay-insensitive circuits and micro-pipelines. In a sense, a combination of these phrases adequately describes self-timed systems. According to Seitz [9], a self-timed system could be either a self-timed element or a legal interconnection of self-timed elements. Timing and sequencing are critical only inside an element since intra-element data transfer is accomplished with message passing, as shown in figure 1-1(b). If a self-timed element is to perform a specific function, this activity is started with a signal on one of the input lines and the element signals the completion of the task on one of its output lines. Since the initiation of the task does not occur until the signal arrives on the input line, delays in the interconnect do not affect the system. However, it is essential that circuitry for communication between elements be provided.

Even though the system designer need consider only the sequencing of intra-element communication, the element itself is generally clocked and synchronous. Hence, asynchronous or self-timed systems are essentially globally-asynchronous-locally-synchronous systems. One variation of such an element is the *Q Module* developed at the Washington University [10]. In the Q module, a two-phase, single-wire clock and a single-wire clock-acknowledge are used for proper sequencing. In such a system, a single delay element is required in the module

with the condition that its value be greater than the delay in the combinational circuit.

Sutherland [3] has explained the philosophy of treating a self-timed system as a *micro-pipeline*. Examples of both FIFOs and general purpose computing are provided. Even though some relatively simple modules have been designed using the transition signaling framework, its widespread use is not presently feasible. When fault diagnosis and and a better understanding of redundancy in micro-pipelines is available, the transition signaling framework might partially replace the clock framework in all digital system design.

1.3 Overview of the Dissertation

The main contribution of this dissertation is a methodology for distributing the clock signal in a synchronous digital VLSI system. Since the delays in all the critical paths are evaluated as a part of the methodology, determination of the largest delay is not difficult. Therefore, a secondary benefit would be the calculation of the upper frequency limit in the designed system. Specific goals of this work as well as the organization of the dissertation are in the following subsections.

1.3.1 Technical Objectives

The specific technical objectives of this work have been:

- Evaluate the constraints that properly reflect the problems associated with clock distribution.
- Derive an objective function for optimization utilizing the derived constraints.
- Design an algorithm for optimal distribution of the clock signal.
- Test the utility of the algorithm on realistic problems.

1.3.2 Organization of the Dissertation

The remainder of the dissertation is organized as follows: Chapter 2 provides the essential background for the work by means of a literature search of related previous work. This includes clocking schemes, distribution methodologies, interconnect and buffer modeling, deficiencies in existing distribution methodologies, and the scope and significance of this work. Chapter 3 is a major contribution of this research since it contains the formulation of the constraints and the objective function for optimization along with a formal problem formulation. Chapter 4 forms the crux of the dissertation as the developed algorithm and its various components are explained. Chapter 5 contains the experimental results. The examples that were used to test the algorithm as well as the results at various levels of implementation are described. Chapter 6 has conclusions and ideas for future research. The Appendix contains the program codes for the various components of the algorithm.

CHAPTER 2

Background

2.1 Clocking Strategies

The strategy used for clocking in a given system is perhaps one of the most important decisions taken at the beginning of the system design [11]. The complexity of the clocking strategy could vary from a relatively simple single-phase clock to an eight-phase non-overlapping clock. The issues that influence the choice of a particular strategy could be many, such as the speed of operation, available area for clock distribution and personal preferences of the designer. A detailed treatment of clocking strategies can be found in most standard textbooks on VLSI design [11,4,12]. Since this work deals more with distribution strategies, only a brief summary of the single-phase clock, the two-phase non-overlapping clock, and some important issues related to these two strategies are presented in this section.

The single-phase clocking strategy consists of a single pulse that triggers all the latches in the system. This type of clocking is conceptually simple and easy to implement. Figure 2.1(a) shows a simplified block diagram of a clocked sequential circuit. In principle, the feedback loop functions in an uncomplicated

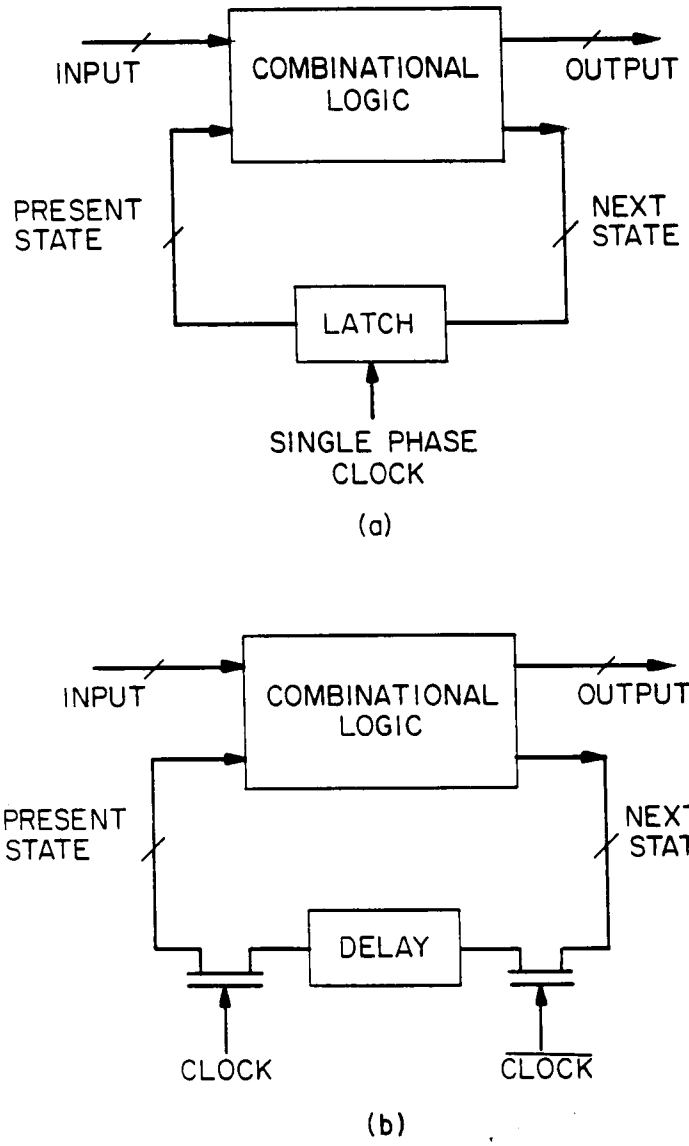


Figure 2.1: (a) Single-Phase Strategy (b) Single-Phase Complementary Strategy

manner. Every clock pulse triggers the latches and thereby the output at the combinational logic that represents the next state is latched into the input as the present state. However, in a large circuit, such a clocking strategy is prone to problems. For instance, the width of the clock pulse might be larger than the set-up time of the combinational circuit and less than the minimum delay in the network. In such a case, *racing* can occur and result in the system stabilizing in an erroneous state. This is due to the fact that the present state values can change more than once during a single clock pulse with data racing through the feedback. Even if the final state is correct and the race is non-critical, the output can possibly have glitches. The problem is even worse in the case of a *critical race* when the final state is not the desired state. Some of these problems can be avoided if the complement of the clock is available and can be utilized in the circuit as shown in figure 2.1(b). This is still a single-phase clock, but requires two interconnects to connect the *clock* and $\overline{clock}$ to all the feedback loops. Here, races are avoided completely since the $\overline{clock}$ signal latches the next state information and the clock signal generates the present state data. Therefore, the clock pulse width still needs to be larger than the set-up time of the combinational circuit, but need not be smaller than the minimum delay.

Even though the use of the complement of the single-phase clock seems to solve a major problem, the edges of the *clock* and $\overline{clock}$ signals need to be aligned completely for the system to avoid glitches completely. This may not always be possible due to process-dependent skewing of the clock signal [13].

In large systems, the skew can never be totally eliminated, and hence better clocking strategies need to be examined.

The necessity of having to align the *clock* and $\overline{clock}$ signals in the complementary single-phase clocking strategy can be totally eliminated by utilizing the two-phase non-overlapping clocking strategy as shown in figure 2.2(a). This is based on the *master-slave* concept of static latches. Here, there are essentially two latches: the master and the slave. In a typical data path, combinational logic is inserted between the two registers. Since the two registers are clocked by alternating the two clock signals, the problems associated with the single-phase clock are mostly eliminated. The additional area requirements for generating and distributing two clock signals are justified in many cases if races and hazards are totally removed.

The problem associated with skew has prompted the use of a phase-locked-loop for synchronization [6,7,8]. The basic configuration of a phase-locked-loop is shown in figure 2.2(b) and consists of a phase comparator, a low-pass filter and a voltage-controlled oscillator. By using the delay in the circuit in the feedback loop, the output of the oscillator is automatically adjusted to overcome process-dependent delays in the system. However, such circuits consume considerable area and should be utilized only when delays in the system are excessive.

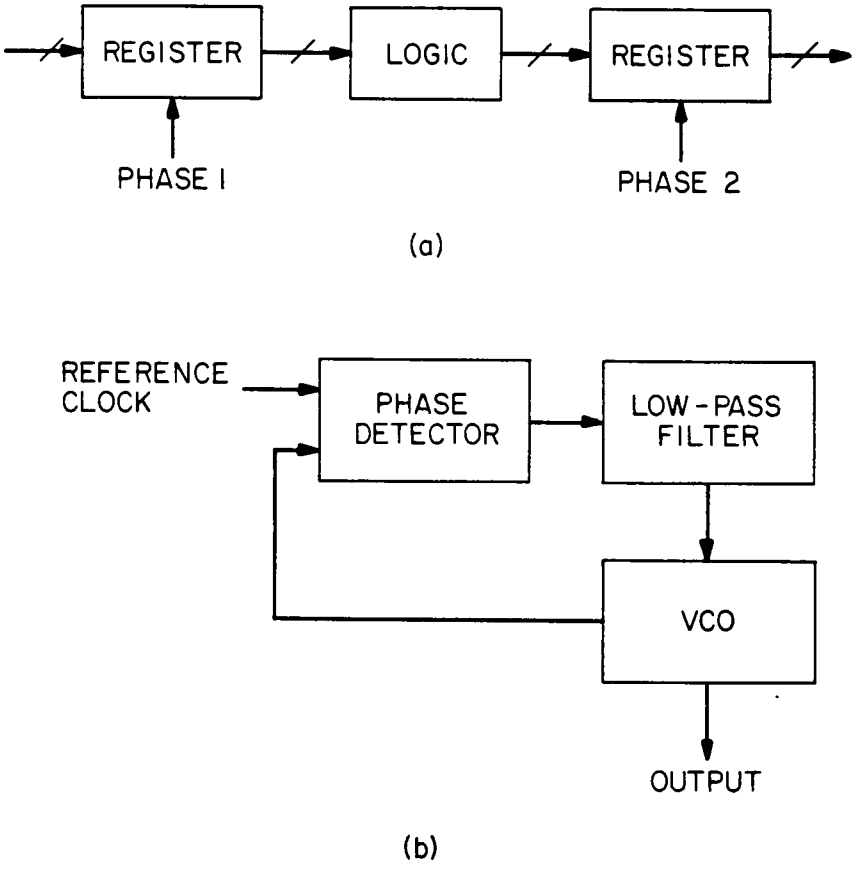


Figure 2.2: (a) Two-Phase Strategy (b) The Basic Phase-Locked-Loop

2.2 Modeling Interconnects

With decreasing feature sizes and increasing chip dimensions, the study of both inter-chip and intra-chip interconnections have become fundamental [14,15,16,17,18,19]. In inter-chip interconnections, the delays are not critical if the resistance of the source driver and the input capacitance of the load totally dominate the delay in signal propagation. However, in CMOS technology, as device dimensions have reached less than $1\mu M$, the delay in the interconnects is at least comparable to the delay in the devices. Unfortunately, the parasitic effects of the interconnects are not easy to model exactly and the level of abstraction adopted generally depends on the frequency of operation as well as the interconnect medium. Interconnects have been treated as RC trees and more recently as transmission lines at very high frequencies. In this section, the parasitic resistance, the parasitic capacitance, the RC model and the transmission line model are all examined. Bakoglu [20] has a more detailed treatment on this subject.

Generally, aluminium is used for interconnection even though gold and some superconductors are under investigation [18]. The D.C. resistance of the interconnect is given by

$$R = \frac{\rho}{WH} \tag{2.1}$$

where R is the resistance per unit length, ρ is the resistivity, W and H are the width and the height of the interconnect, respectively, as shown in figure 2-3.

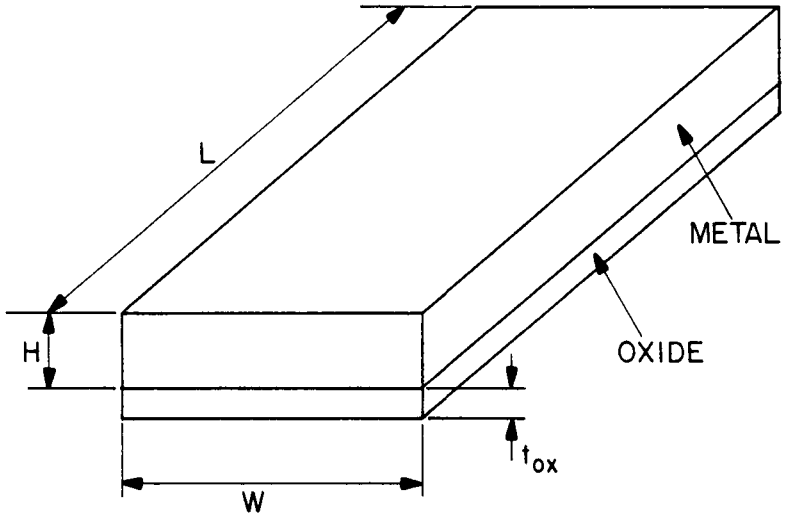


Figure 2.3: Interconnect Model

In reality, the cross-section of the interconnect is not likely to be exactly rectangular, but is a good enough approximation to empirical results [20]. Also, with use, the dimensions may not remain constant since the electrons bombarding the metal particles cause metal migration. Unfortunately these variations are process-dependent and cannot be modeled very precisely. If all the dimensions of the interconnect are reduced by a factor of S, then the resistance of the interconnect also increases by a factor of S [20]. The value of the interconnect resistance not only affects the delay in signal propagation, but also the voltage drop along the line.

The capacitance of the interconnect also has a bearing on the power consumption and the delay. As a matter of fact, it is the RC time constant that is important in the delay evaluation. The capacitance of the interconnect, ignoring adjacent lines and edge effects, is given by

$$C = \epsilon_{ox} \frac{W}{t_{ox}} \tag{2.2}$$

where C is the capacitance per unit length, ϵ_{ox} is the dielectric constant of the oxide layer, and W is width of the interconnect as before. Bakoglu [20] has shown how interconnect capacitance can become significant and comparable to device input capacitance. In a $0.7\mu M$ feature size CMOS technology, a minimum size transistor has an input capacitance of $2fF$. Therefore, a typical size transistor with a W/L ratio of 10 would have an input capacitance of $20fF$. This is equivalent to the capacitance of an interconnect of length $l = 100\mu M$ which would be fairly common in a $10,000\mu m \times 10,000\mu m$ size chip.

Traditionally, interconnects have been modeled as RC meshes, with all the capacitors connected to ground and no resistor connected to ground [21,22,23,24,25]. This assumes that at the frequency of operation, $\omega L << R$, where L and R are the inductance and resistance of the interconnect, respectively. If this condition is true, then the effects of LC interaction such as ringing and reflection almost disappear. The delay will then be a purely exponential delay. Figure 2-4(a) shows a typical driver, interconnect and load. For the purpose of delay calculations, the transistor output can be represented, as shown in figure 2-4(b), by a resistance. This resistance, which is represented by R_S in figure 2-4(b), is actually the r_{out} of the transistor. In CMOS technology, the buffer would have relatively constant output resistance. Therefore,

$$R_S \approx \frac{L/W}{\mu C_{gox}(V_{DD} - V_T)} \tag{2.3}$$

where L and W are the effective length and the width of the device, C_{gox} is the gate capacitance per unit area, μ is the mobility, V_{DD} is the power supply voltage and V_T is the threshold voltage [20]. The justification for this approximation is as follows. In CMOS, the drain current is given by

$$I_D = \frac{1}{2}\mu C_{gox}(W/L)\left(2\left(V_{GS} - V_T\right)V_{DS} - V_{DS}^2\right) \tag{2.4}$$

$$r_{out} = \begin{cases} \frac{V_D}{I_D} & \text{static} \\ \frac{\delta V_D}{\delta I_D} & \text{dynamic} \end{cases} \tag{2.5}$$

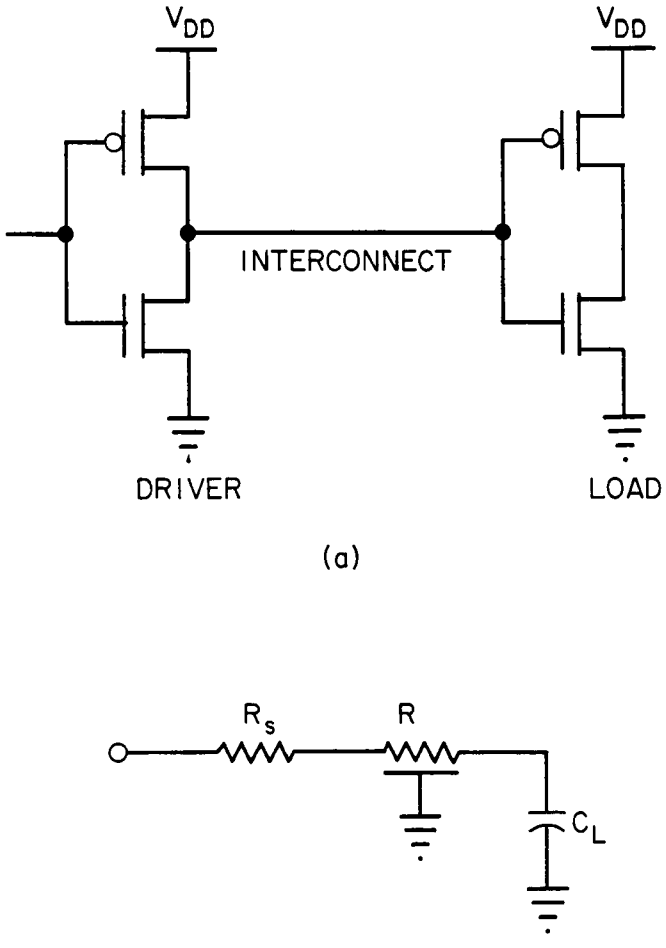


Figure 2.4: (a) Typical CMOS Network (b) RC Model

Now, suppose that the NMOS is ON and PMOS is OFF. Then

$$g_{out} = \frac{1}{r_{out}} = \frac{1}{2}\mu C_{gox}(W/L)\{2(V_{GS} - V_T) - 2V_{DS}\} \quad (2.6)$$

$$= \mu C_{gox}(W/L)\{(V_{GS} - V_T) - V_{DS}\}. \quad (2.7)$$

If $V_{DS} << (V_{GS} - V_T)$, then

$$r_{out} \approx \frac{(L/W)}{\mu C_{gox}\{V_{GS} - V_T\}}. \quad (2.8)$$

Iff $V_{GS} = V_{DD} = V_{in}$, then

$$r_{out} \approx \frac{(L/W)}{\mu C_{gox}\{V_{DD} - V_T\}}. \quad (2.9)$$

It should be noted that the above values of (L/W) and μ are for the NMOS device. A similar derivation can be performed for the PMOS device when $V_{in} = 0$. If

$$\bar{\mu_n}(W/L)_{NMOS} = \bar{\mu_p}(W/L)_{PMOS} \quad (2.10)$$

then, the approximation for R_S will be valid.

The interconnect itself is represented by a distributed RC network and the load by its input capacitance. The interconnect is generally analyzed using the lumped parameter technique. Here, the resistance and the capacitance of the interconnect are approximated by lumped components. Several circuit configurations, such as the T and the π, have been proposed for the approximation in the literature. The three-section π network is generally accurate enough and has an error, which is the difference between estimated and measured delay, of less than 3 % [20]. Such a network is shown in figure 2-5(a).

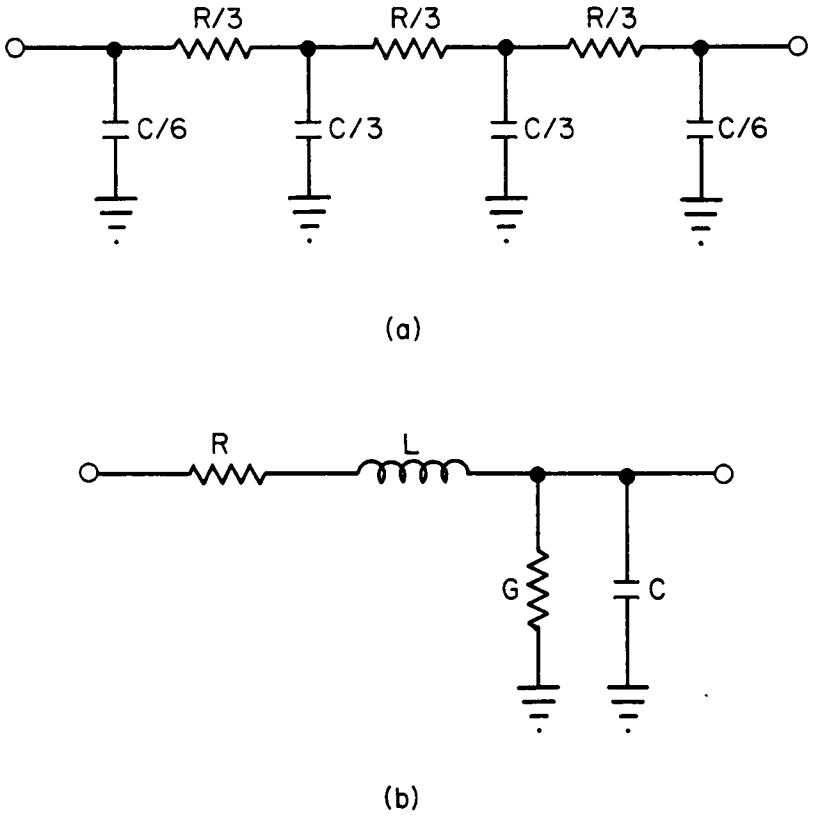


Figure 2.5: (a) 3 Section π Network (b) Transmission Line Model

2.3 Buffer and Related Issues

In CMOS technology, most on-chip wires can be modeled accurately as RC meshes as discussed in the previous paragraph. However, with emerging technologies such as GaAs with typical rise time in the range of $100pS$, it has been recognized that the inductive effect in the interconnect could play a significant role in proper modeling [26,27,15,28,20]. The emergence of the significant inductive effect results in ringing, overshoot and problems with reflections at improperly terminated lines. Therefore, in treating interconnects as transmission lines, it is essential to include a conductance to ground as well as an inductor in series as shown in figure 2-5(b). This somewhat simplified model could represent per unit length of the microstrip. If the interconnect exhibits transmission line phenomenon, special drivers have to be designed to handle reflections [29]. Usually, transmission line modeling needs to be resorted to when the time-of-flight of the signal is comparable to the rise time of the signal [20]. Another case where transmission line effect is dominant is in very thick lines used in wafer scale integration where the lines are intentionally designed to be LC lines in order to achieve very high bandwidths [29].

The sizing and placement of buffers are important factors in the distribution of signals in VLSI. Buffers are necessary to drive the loads consisting of both interconnects and the input impedances of individual cells. In addition, the placement of a buffer along an interconnect essentially converts RC loads to

dominantly capacitive loads. However, buffers consume area and power, and have built-in delays. In this section, previous work accomplished in the design of buffers and some of the important related issues are presented.

In CMOS technology, the simplest method of buffering is to use a single inverter to drive the load as shown in figure 2-6(a). Here, the sizing of the inverter is critical and cannot be made indiscriminately large. At the same time, for a given load, it is only by increasing the size of the device that the speed can be increased. Even though the inverter buffer does not consume any static power, during a transient, both the PMOS and the NMOS devices will be ON for a short duration and a short circuit current will flow from V_{DD} to ground [30].

Lin and Linholm [31] were the earliest to address the issue of optimizing the buffer size for minimum delay. They determined that using a cascade of inverters whose size gradually increase provides the optimal area and drive capability. Here, the signal input looks at a relatively small input capacitance of the first inverter stage and the overall delay in the buffer is reduced since it is dominated by the last stage. The figure of merit used in this work was

$$F = \left(\frac{A_N}{A_0}\right)\left(\frac{t_{ps}}{t_{po}}\right)^k \tag{2.11}$$

where K is the weight factor, A_0 is the area of the first inverter stage, A_N is the area of the last inverter stage, t_{ps} is the average propagation delay per stage and t_{po} is the smallest propagation delay per stage. Therefore, to use this design methodology for the buffer design, one needs to choose between area and

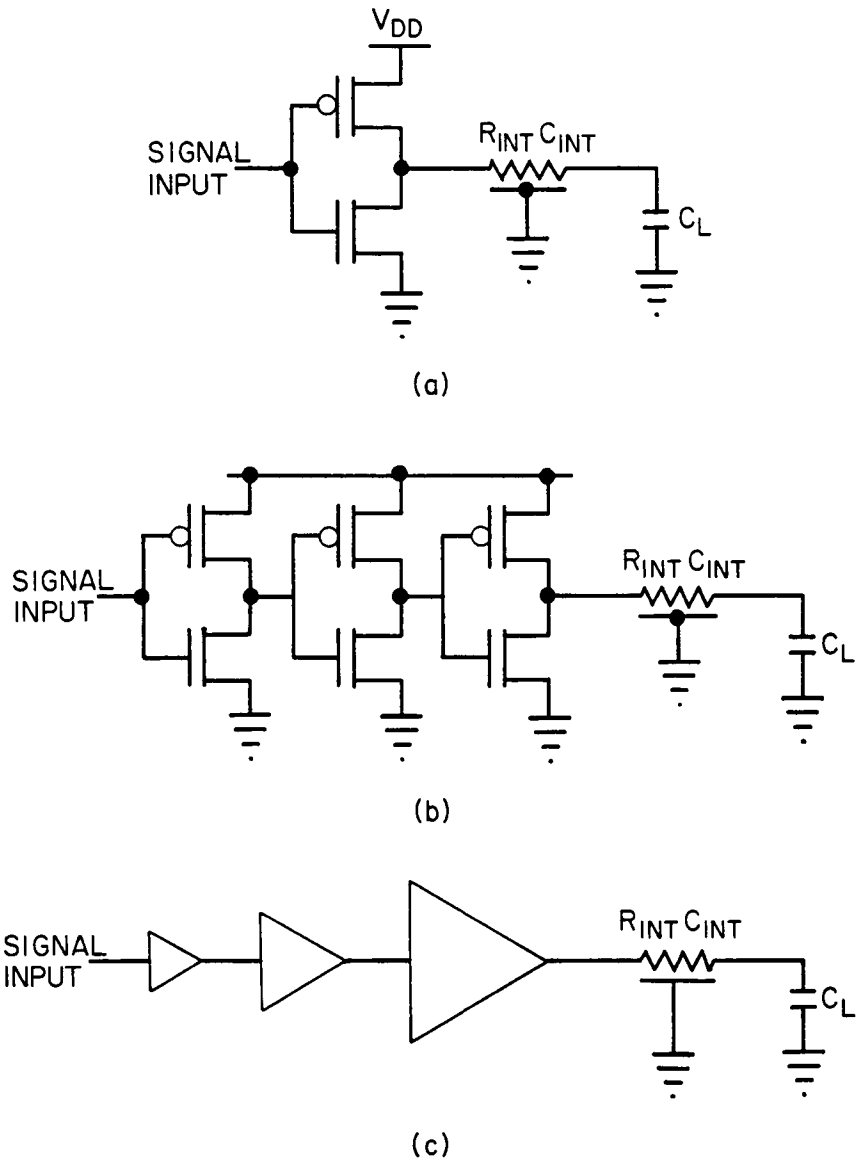


Figure 2.6: (a) Single Inverter Buffer (b) Cascaded Inverter Buffer (c) Sizing of the Cascaded Buffer

delay optimization. True optimization for minimal delay will often result in an impractical size of the last stage [32].

The work of Lin and Linholm [31] has been improved upon by Lee and Soukup [32]. They have developed an algorithm and a CAD tool that investigates the technology parameters and the load capacitance, and evaluates the minimum achievable delay. Then, based on the acceptable delay in the design application (which should be greater than the minimum achievable delay), the program calculates the optimal number of stages and the (W/L) ratios of all the transistors.

One major problem in the aforementioned methodologies is that irrespective of the size and number of stages in the buffer, the final stage has to charge the interconnect capacitance and the load capacitance to V_{DD} when the PMOS transistor is turned ON. The delay in doing so can be reduced by pre-charging the interconnect either to V_{DD} or $V_{DD}/2$. Two methods, as described by Bakoglu [20], are summarized here. The first method, shown in figure 2-7(a), pre-charges the interconnect to V_{DD}. This circuit utilizes a two-phase non-overlapping clock. When the clock phase ϕ_1 is HIGH, the PMOS transistor is turned ON and the interconnect gets charged to V_{DD}. At the same time, since ϕ_2 is LOW, the NMOS device cannot turn ON and discharge the interconnect. When the second phase of the clock ϕ_2 is HIGH, the value of v_{in} determines whether the interconnect should be discharged or not. Of course, for proper operation of this circuit, it is essential that v_{in} has reached its correct value before ϕ_2 is activated. In this

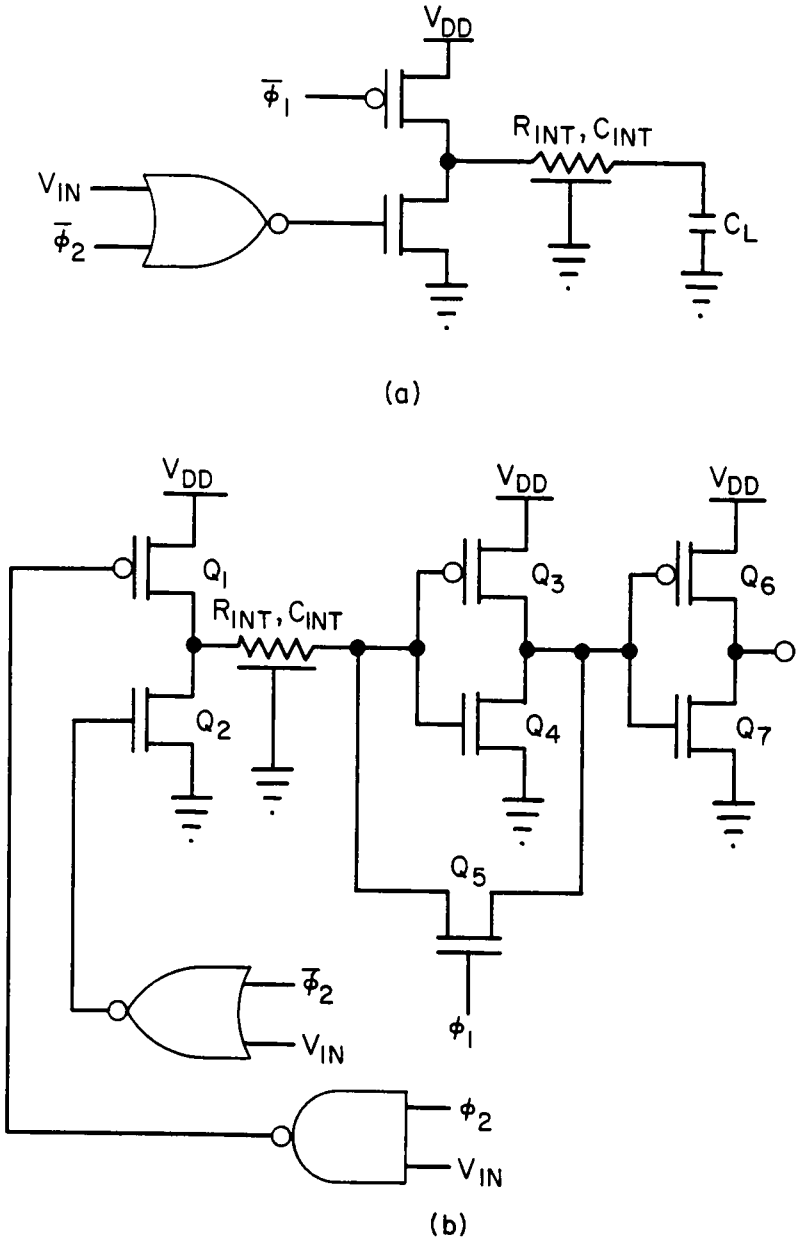


Figure 2.7: (a) Pre-charging to V_{DD} (b) Pre-charging to $V_{DD}/2$

method, it is also possible to use a single pre-charge PMOS device and several NMOS pull-down devices if more than one interconnect have to be driven in parallel. In the second method, depicted in figure 2-7(b), the interconnect is pre-charged to $V_{DD}/2$. When ϕ_1 is active, both the input transistors Q_1 and Q_2 are turned OFF. Since the output of Q_3 and Q_4 are fed back to their inputs by means of Q_5, the interconnect gets charged to $V_{DD}/2$. When ϕ_2 is activated, depending on the value of v_{in}, the interconnect either gets discharged to ground or gets charged up to V_{DD}. Since the interconnect is pre-charged to only half of the maximum voltage, the ability to go either way with less potential difference renders the second methodology superior to the first.

2.4 Clock Distribution

In a large system, signals have to be distributed over backplanes, printed circuit boards and integrated circuits. The variation in impedances and the delays associated with the interconnects make signal distribution a formidable task. Even within a chip, the nature of the problem varies depending on the signal and the average wire length. For example, in distributing power and ground, voltage drops across the interconnects is a special concern. In general, global signal distribution is more problematic than local signal distribution due to the long wire lengths.

Distribution of the clock signal causes additional problems since not only does

it have to be distributed globally, but also has to meet rigid time constraints. The complexity of the problem has prompted the attempt of drastic measures such as self timed systems [3] and optical clock distribution [33,34]. Such radical solutions cannot be incorporated in a majority of the systems today either due to the need of a unique time reference in the system or technological problems. In this section, we ignore all such non-traditional solutions to the clock distribution problem and examine the traditional methodologies that have been adopted in the past.

Anceau [5] was one of the earliest to address the clock distribution problem in synchronous VLSI. The basic configuration in Anceau's methodology is shown in figure 2-8(a). The basic clock from the clock generator is distributed directly to each functional block. Buffering and local synchronization are performed inside the functional module. A phase-locked-loop is incorporated in each functional block. Since clock skew greatly affects inter-module communication, a slower clock is used for driving the communication interface circuitry in each module. The communication clock is derived from the master reference clock by using a divider. Since both the clocks are derived from a single reference, the phase-locked-loop within each module maintains proper phase relationship between the two clocks to avoid metastability problems. Also, the use of the common reference clock makes the system completely synchronous as opposed to locally-synchronous globally-asynchronous systems.

A different approach has been taken by Friedman and Powell [35,36]. A

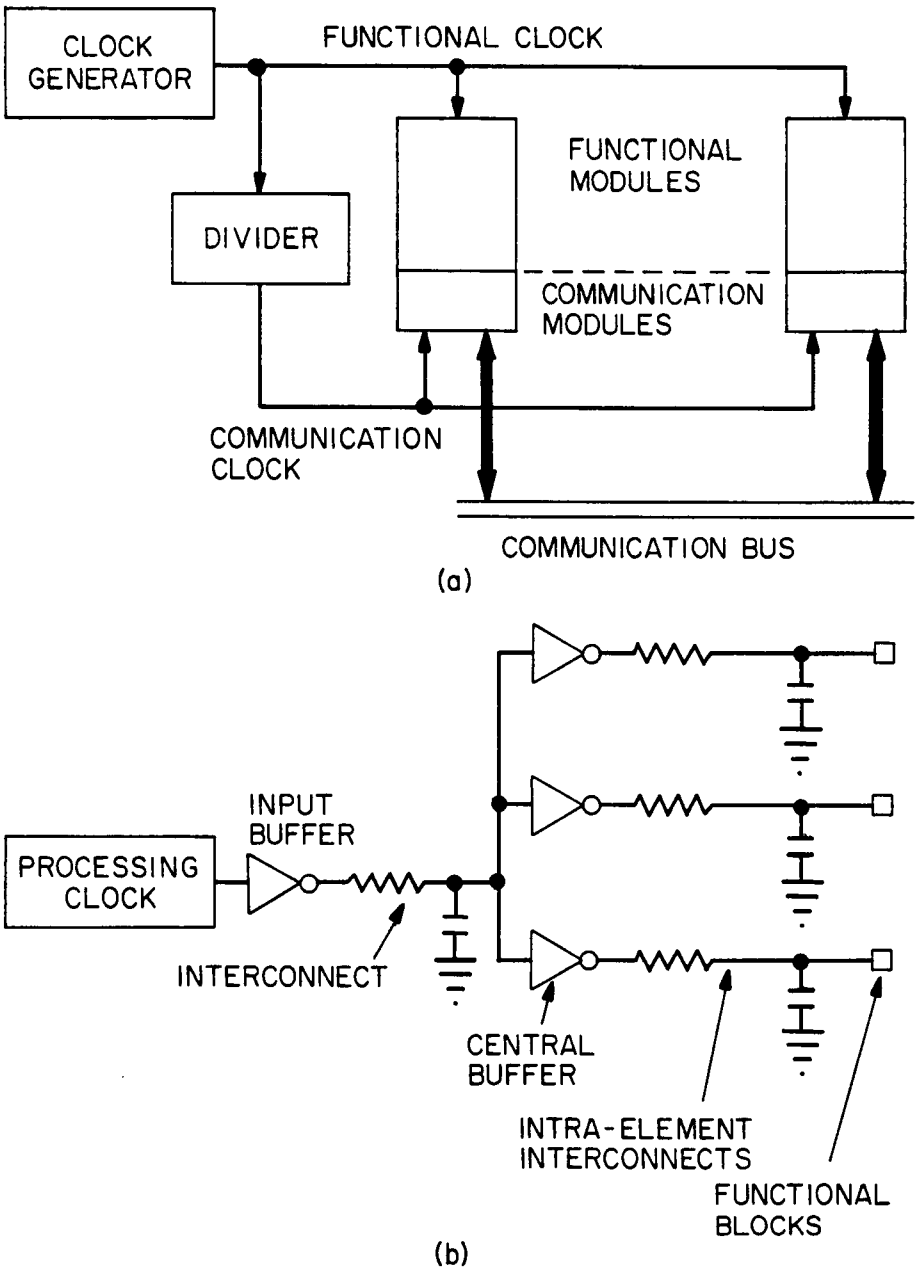


Figure 2.8: (a) Anceau's Technique (b) Friedman's Technique

block diagram of their approach is shown in figure 2-8(b). Their objective in the design methodology is to reduce the clock skew. The processing clock is buffered at the input to drive the interconnect that connects to a bank of buffers located centrally. No clock skew is introduced at this stage since all the buffers are located in close physical proximity. Each one of these buffers drives a functional element which could be a standard cell in VLSI or a module in WSI. At this stage, skewing of the signal would be introduced since the distance from the central buffer bank to each functional element would probably be different. Further, the input impedance of each element could be different. To compensate for these variations, the size of the buffers are fine-tuned such that the delays in the buffers reduce, if not totally eliminate, the clock skew. This methodology could be used hierarchically in the sense that the distribution of the clock within each functional element could be performed using the same technique iteratively.

Bakoglu, Walker and Meindl [37] have used a symmetric H-tree, as shown in figure 2-9(a), for clock distribution in VLSI and WSI. Again, the objective is to reduce the clock skew which is achieved by delaying the clock equally in the symmetrical tree. However, the wide lines in the tree are to be treated as LC lines and proper termination of the lines become critical. At each fan-out point, the characteristic impedance must be twice that of the incoming line so that, in parallel, the impedance will be matched. Properly matching the impedances at the branch points will assure the absence of reflections. The response of a uniformly lossy line has two components: a step function and a slow rising

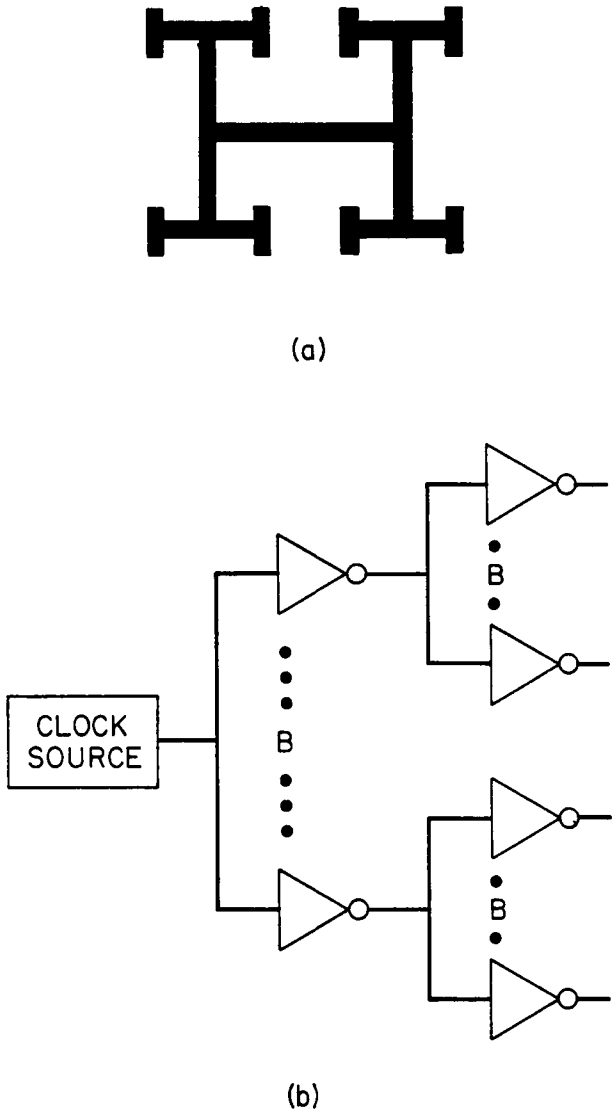


Figure 2.9: (a) Bakoglu's Technique (b) Mijuskovic's Technique

RC-like response. If $L_{int}C_{int} >> 2Z$, where L_{int} and C_{int} are the inductance and capacitance of the interconnect, respectively, and Z is the characteristic impedance, then the RC-like response dominates. Therefore, to increase the speed, the vertical dimensions of the metal and the oxide layers will have to be scaled properly.

In designing application-specific-integrated-circuits (ASICs), several approaches could be taken. One approach is to use only standard cells. The distribution of the clock signal in such an approach has been studied by Mijuskovic [38]. In this methodology, it is assumed that the buffer is always the largest inverter available in the standard cell library. Further, it is assumed that the interconnect does not introduce any delay. Therefore, every buffer is assumed to drive only the capacitance of the next stage. As shown in figure 2-9(b), the distribution network is a *k-ary* tree of inverters. Mijuskovic has derived the optimal number of inverters in this tree for a given set of standard cells whose input capacitances are known. Since the impedances in the interconnects are ignored and each inverter drives the same number of gates, all the buffers drive the same amount of capacitance. If the frequency of operation is not very high (order of $20 - 30MHz$) and the feature size is not very small (order of at least $2\mu M$), then this distribution methodology could be useful in semi-custom standard cell design.

Boon *et al.* [39] have proposed a methodology, as shown in figure 2.10, for distributing clocks in a $1\mu M$ standard cell technology. Because of the affiliation

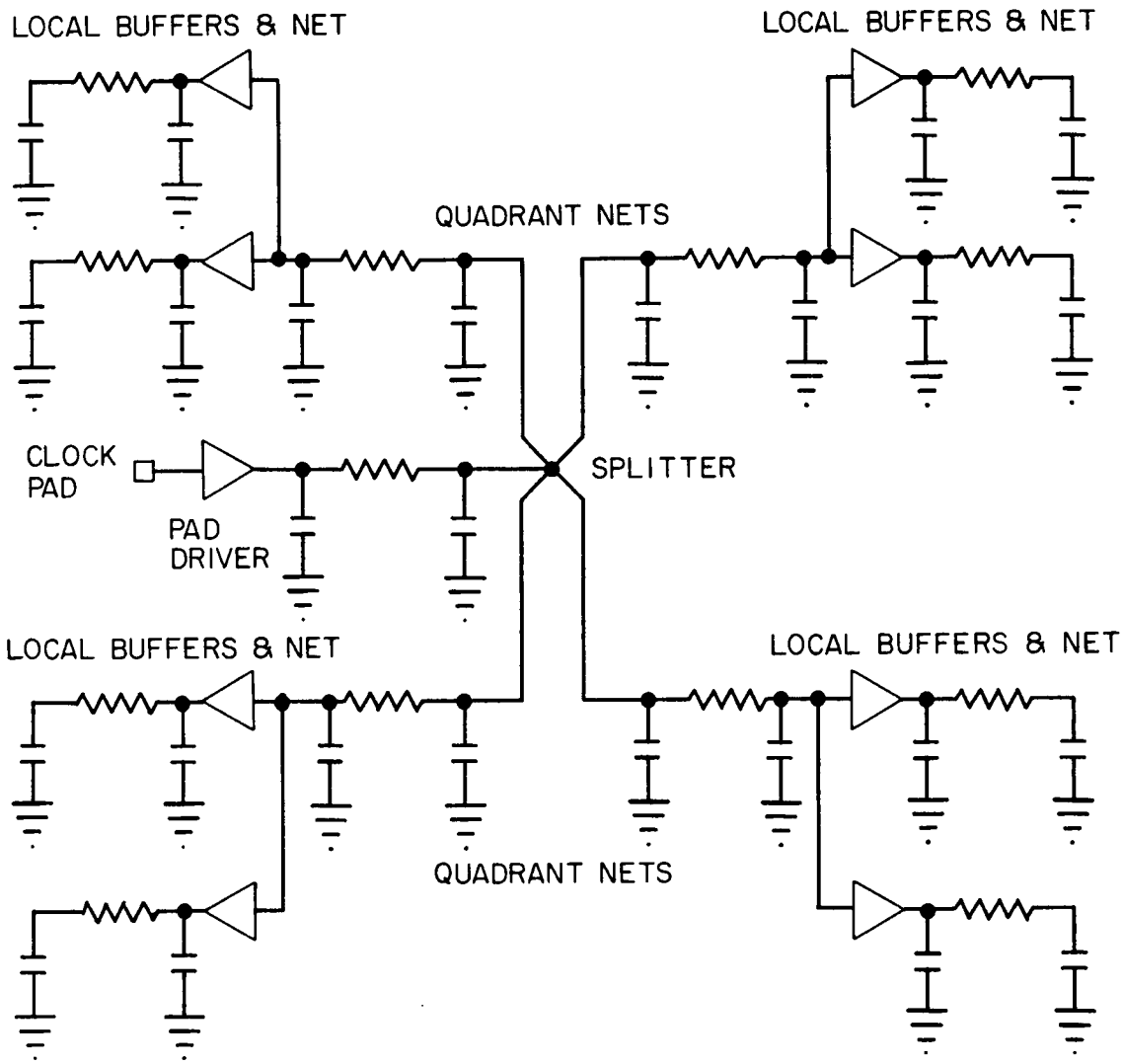


Figure 2.10: Boon's Methodology

of the authors, importance has been given to compatibility with commercial routers. Even though prototypes based on this methodology have not been reported in the work, it is claimed that the methodology is targeted at operation up to frequencies of $100MHz$. The clock generator is assumed to be external and the signal is sent from the pad driver to a central splitter in heavy metal. Even though this line could exhibit LC behavior at frequencies in the range of $100MHz$, the authors have chosen to model it as an RC network. The signal splitter sends the signal to four sets of local buffers via intermediate width interconnects. Typically, each of these interconnects feeds five local buffers which in turn feed the local networks. The interconnect from the pad driver is in the range of $20\mu M$ width and it is reduced to around $3 - 4\mu M$ after the split. The authors also claim that using a high performance distribution scheme as opposed to simplistic routing costs only 2% in area overhead.

The last work that we examine in this section is that of Cox *et al.* [40] who have come up with a unique solution to controlling clock skew due to variations in process parameters, temperature and supply voltage. The uniqueness in their methodology comes from the adoption of a feedback network for correction of the skew. The methodology is for the design of the off-chip clock driver that has to provide the clock signal to various chips. The monitoring of the performance of the chip is accomplished by placing a performance sense element (PSE) on each chip that senses variations in performance and transmits the information to selected drivers and clock circuitry. The characteristics of the driver are

changed to suit the information from the PSEs. This continuous monitoring provides constant error detection and correction in the distribution system.

2.5 Deficiencies in Previous Work

In this section, a critical review of the various clock distribution methodologies presented in the preceding section is provided. First, the author's views on the desired characteristics of an ideal clock distribution system are presented. It should be noted that these are ideal characteristics and are extremely difficult to achieve in a working practical system since process dependent parameters cannot be usually modeled very accurately. However, they serve the objective of having a reference to which all systems can be compared.

An ideal clock distribution system should incorporate the following features:

1. Clock skew should be totally eliminated.
2. Performance should be maximized by reducing the delay in the critical path.
3. The number of buffers utilized should be minimal.
4. Buffers and interconnects should be modeled accurately while calculating or estimating delay.
5. The methodology should be hierarchical.
6. All commonly used techonologies should be accomodated.

7. The methodology should be compatible with commercially available software.

8. Routing should be an integral part of the methodology.

9. Complexity of computation should not be prohibitive.

In Anceau's work [5], the biggest drawback is the slowing down of the communication clock by the divider circuit. The method is not particularly suitable at small feature sizes and high frequencies since the delays in interconnects are not given any importance. Buffer sizing and placement is also not discussed in the work. Lastly, routing of the signal is not an integral part of the distribution methodology.

Friedman and Powell's work [35,36] has many of the desired features in that it is hierarchical, sizes the buffers for minimal skew and considers the impedances of the interconnects. However, the methodology has two apparent drawbacks. Firstly, all the buffers are centrally located and no attempt is made to select the optimal number of buffers. Secondly, routing is not an integral part of the methodology. As such, the distribution system is not fine-tuned for maximum performance.

A thorough treatment of modeling impedances has been accomplished in Bakoglu's work [37]. The methodology is symmetric and hierarchical. Further, it is suitable for designing at very high frequencies and small feature sizes in both ULSI and WSI. Again, using the reduced clock skew as the main design

criterion, routing has not been integrated in the methodology for maximum speed. Lastly, buffer placement has not been incorporated.

Mijuskovic's work [38] is suitable only for small chips and relatively low frequencies. This is due to the fact that the impedances in the interconnects are thought to be negligible. The main advantage in his work is the selection of an optimal number of inverters for buffering in a standard cell design environment. As in the other works, routing has not been integrated into the design methodology for optimal performance.

In the work of Boon *et al.* [39], it is not apparent why the splitting into quad nets was performed. Even though it is true that keeping the same distance to the local buffers from a central location would introduce equal delays in quadrant nets, the optimization of speed is not necessarily accomplished. Also, the choice of connecting five local buffers in each quadrant net does not seem to be based on any optimality criteria. The decision to use the RC time constant of the interconnect as an estimate of the delay is correct only when all the loads are the same. One major advantage in this work is the integration of the methodology in commercial software.

Since the use of PSE in the design of clock distribution as reported by Cox *et al.* [40] is more relevant to the design of the off-chip drivers, no particular criticism is offered on this work in this section.

2.6 Scope of Research

Based on related work and their shortcomings portrayed in this chapter, it is apparent that not only has the design of clock distribution methodologies been an important research activity, but also one that needs further study and improvements. This dissertation extracts the advantageous components from related work and builds upon them to derive a methodology that is closer to the ideal distribution system than any reported work.

Specifically, this work compares to the ideal system as follows:

1. Clock skew is not totally eliminated, but reduced using clustering techniques in the assignment of cells to buffers.
2. Performance in speed is maximized in each cluster of cells.
3. The number of buffers used is minimal.
4. Accurate modeling of interconnects is accomplished even though buffers are modeled in a relatively simplistic method.
5. The methodology is hierarchical.
6. In this work, only CMOS technology is considered. But changes to accomodate other technologies should be minimal.
7. No commercial software has been integrated.
8. Routing is an integral part of the methodology.

9. Complexity of computation is reasonable.

As can be inferred, the methodology presented in this dissertation is not ideal, but is superior to existing reported methodologies.

CHAPTER 3

Problem Formulation

3.1 General

The distribution of the clock signal in a large system could span many technologies, architectures and interconnections. The formal mathematical representation of the distribution system has to be performed so that the constraints in the formulation accurately portray the constraints in the real physical system. In this section, a general description of some of the more predominant design environments and their structures, variations in interconnects and the resulting limitations on the modeling is provided.

3.1.1 Design Environment

Perhaps the simplest VLSI system is a semi-custom chip that utilizes standard cells as building blocks. Generally, these cells have the same heights and varying widths. The designer has to basically utilize the available cells in the system for the circuit design. Such a design environment is somewhat similar to the design of a discrete circuit using available MSI parts. Because of the standard height, standard cells are usually placed in regular rows and columns. Such a

design philosophy is generally adopted in the design of relatively small (order of $0.1cm \sim 0.3cm$) chips. The clock signal in such a system is usually routed along the rows and columns without much attention to the delay in the interconnects since the maximum length of the interconnect may not be large enough to cause significant delays. Further, the pad buffer might be sufficient to buffer the input clock signal and drive all the cells in the circuit.

In a larger full-custom design, the distribution becomes somewhat more critical. Here, cell sizes need not be standardized and the placement of functional blocks could be relatively irregular. Since the chip dimension is also larger (order of $1cm$ per side is not unrealistic), the interconnect delay needs to be considered at the initial design stage. In such a system, the fan-out of the pad driver is generally not large enough to drive all the functional blocks directly. It will then be essential to have local buffers that reduce the load on the pad buffer, but also introduce built-in delays.

If the entire wafer is used in the design of the system (wafer scale integration), clock distribution would be a serious problem in synchronous design. The timing problem has caused otherwise feasible projects to fail, such as the Trilogy design [41]. In WSI, both the sizing of the interconnect, as well as buffer design become critical. Since it is impractical to run a $1\mu m$ interconnect for a distance of several inches, due to the very large delays in such long and narrow interconnects, the width and height of the interconnect needs to be varied across the wafer.

The problem associated with WSI, especially yield and fault location problems, have made multi-chip module systems more attractive. Several technologies have been used in multi-chip modules, including thick-film and thin-film hybrid technologies. One particular type of multi-chip module utilizes a silicon wafer for all the routing [42]. Chips are bonded to the wafer and all the interconnections are run on the wafer. Of course, in a such a system, the impedance of the wire from the pad of each chip to the wafer is also an important consideration.

3.1.2 Hierarchical Model

Irrespective of the technology and the specific architecture, all systems can be represented by a hierarchical model similar to the one derived by Hatamian [14]. The model derived by Hatamian is shown in figure 3-1(a). Here, the board to board interconnection node is at the highest level and each connector is represented by a node at a lower level. The on-chip functional module is at the lowest level. Based on this model, a similar model has been derived for a large VLSI chip as shown in figure 3-1(b). In spite of the similarity in structures, there are several important differences in the two models. The nodes in this model are not connection nodes, but buffers or functional modules. For an external clock, the pad driver would drive all the local buffers. In turn, each local buffer would drive certain functional modules. This representation is hierarchical and the optimization performed at the lowest level in this model could be iteratively

performed at the previous levels.

3.1.3 Variations in Interconnects

The frequency of operation, the length and cross-sectional dimensions of the interconnect are all interdependent. The difference in bandwidth between cross-sectional dimensions of $1\mu m \times 1\mu m$ and $6\mu m \times 12\mu m$ is about three orders of magnitude, with rise time decreasing from $100nS$ to $100pS$ [29]. Therefore, some of the WSI designers have adopted wide and thick lines for distributing the global signals, in spite of the extra processing steps involved. To reduce the RC time constant in a thin line, it is essential to buffer the line at more than one level as the buffer would essentially reduce the RC load to a pure capacitive load. The thick lines, on the other hand, behave like LC lines and the reflections along the line become critical. Even within a thin line, it is essential that at the maximum frequency of operation, $R \gg \omega L$ before the line can be treated as an RC network. The region of transition where the interconnects need to be switched from LC type thick lines to RC type thin lines is unclear [29] and is a future goal of the author.

While formulating the clock distribution problem, it is necessary that the scope of the formulation with respect to the type and size of the system be very clearly defined along with the maximum frequency of operation so that the constraints of the formulation accurately reflect the true constraints in the physical system. For example, if the scope of the problem formulation includes WSI,

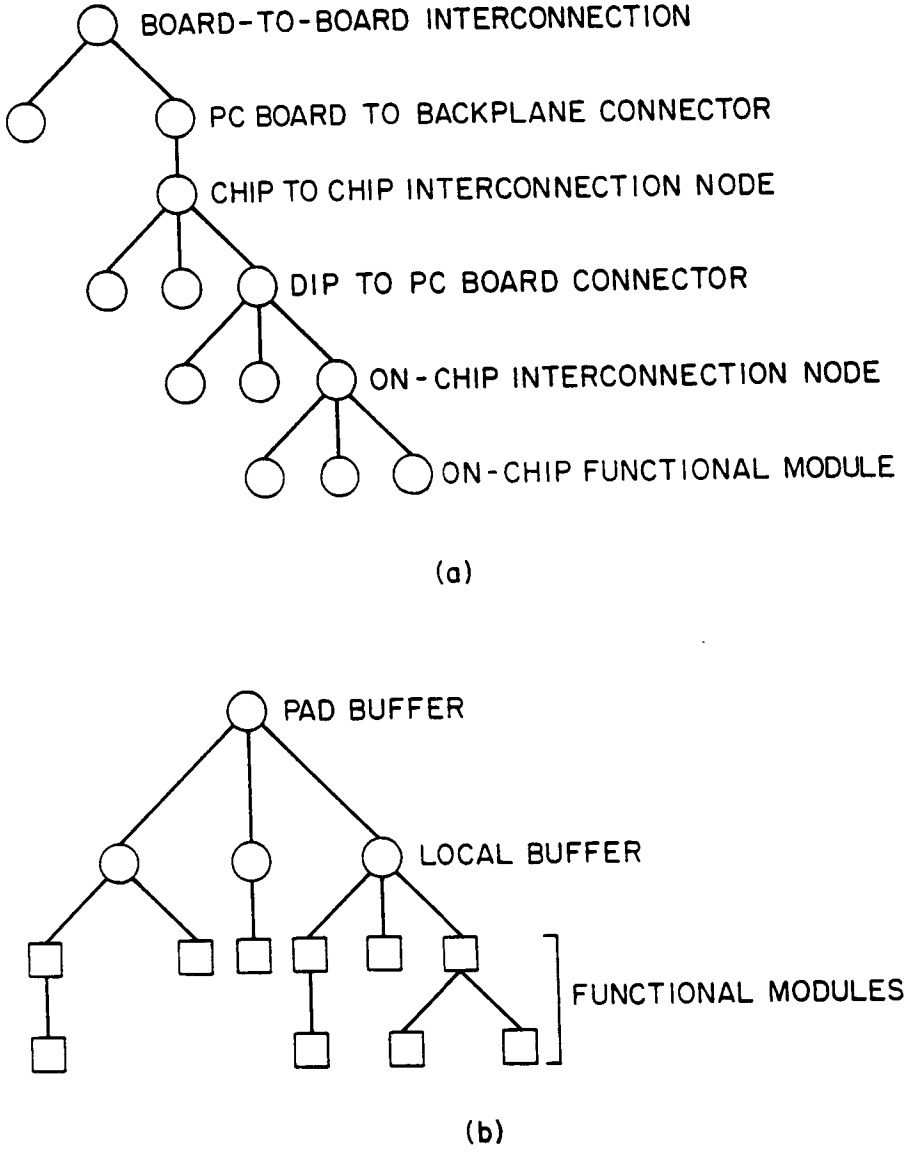


Figure 3.1: (a) Hatamian's Interconnect Modeling (b) Clock Distribution in a Large Chip

then the interconnect modeling should accomodate both RC and LC networks.

3.2 Representation

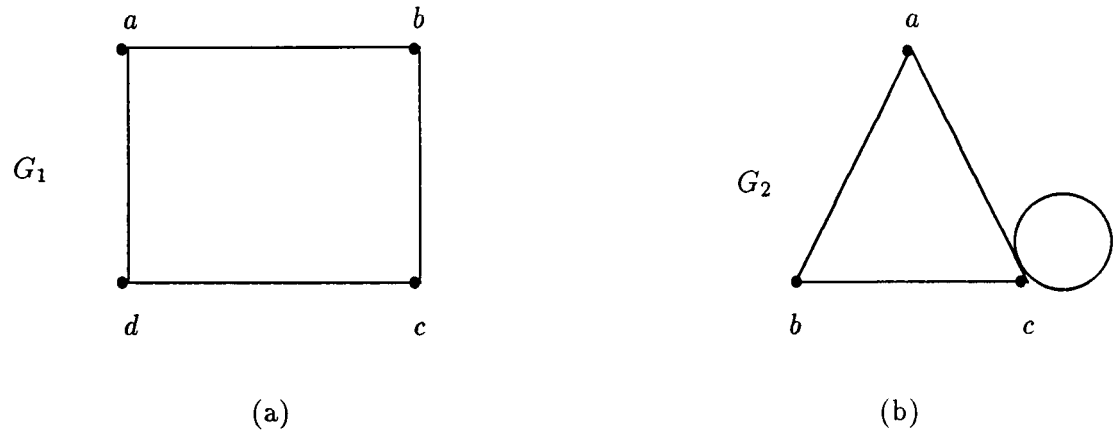
3.2.1 General

Representation of a problem in known mathematical form is generally a prerequisite to solving the problem. This step not only aids the understanding of the problem but also allows the use of standard mathematical techniques in arriving at a solution to the problem. In this section, a model for the clock distribution problem is presented using graph theoretic techniques.

3.2.2 Elements of Graph Theory

In this subsection, some basic definitions of elementary graph theory are presented. Examples and further details can be found in Mott [43].

Definition 3.1 A graph G is a pair of sets (V, E), where V is a set of vertices and E is a set of edges that connect the vertices in V. If the elements of E are ordered pairs of vertices, then the graph G is said to be a *directed graph* or a *digraph*. In a digraph, an edge (u, v) is said to *join* u to v. An edge that is between a vertex and itself is said to be a *loop* and a graph with no loops is called a *simple graph*. It is generally assumed that the number of vertices, and the number of edges, in a graph is finite. If this assumption is true, then the graph is said to be a *finite graph*. In a finite graph, $|V(G)|$ denotes the number



(a) (b)

Figure 3.2: (a) Simple graph. (b) Nonsimple graph.

of vertices in the graph G, and is called the *order* of G. Similarly, $|E(G)|$ is the number of edges in G, and is called the *size* of G. Graphs are generally represented by diagrams in which each vertex is represented by a point in a plane, and each edge by a curve joining the two points.

Figure 3-2 illustrates a simple graph and a nonsimple graph. In graph G_1, $V(G_1) = a, b, c, d$ and $E(G_1) = ab, bc, cd, da$. Similarly, in graph G_2, $V(G_2) = a, b, c$ and $E(G_2) = ab, bc, ca, cc$. The orders of G_1 and G_2 are 4 and 3, respectively, whereas the sizes of both the graphs are 4.

Definition 3.2 In a digraph, an edge (u, v) is said to be *incident from* u, and to be *incident to* v. The number of edges incident to a vertex is called the *in-degree* of the vertex and the number of edges incident from a vertex is called its *out-degree*. The *degree* of a vertex is determined by counting each loop incident on it twice and all other edges once. A vertex of degree zero is called

an *isolated vertex.* If an edge is incident from u to v or incident on u and v, then u and v are said to be neighbors. The minimum of all the degrees of the vertices of a graph G is denoted by $\delta(G)$, and the maximum of the degrees of the vertices by $\Delta(G)$. If $\delta(G) = \Delta(G) = k$, then G is said to be *regular of degree k.*

Definition 3.3 Two graphs G_1 and G_2 are said to be *isomorphic* if there is a function $f : V(G_1) \mapsto V(G_2)$ such that

1. f is one-to-one
2. f is onto, and
3. for each pair of vertices u and v of G_1, $\{u, v\} \in E(G_1)$ iff $\{f(u), f(v)\} \in E(G_2)$.

Definition 3.4 A graph G_2 is said to be a *subgraph* of G_1 iff $V(G_2)$ is a subset of $V(G_1)$ and $E(G_2)$ is a subset of $E(G_1)$.

Definition 3.5 In a graph G, a sequence P of zero or more edges of the form $\{v_0, v_1\}, \{v_1, v_2\}, ..., \{v_{n-1}, v_n\}$ is called a *path* from v_0 to v_n. v_0 is the *initial vertex* and v_n is the *terminal vertex.* If the initial vertex is the same as the terminal vertex, then P is a *closed path.* Otherwise, P is an *open path.* P is *simple* if all edges and vertices on the path are distinct except possibly the endpoints. A path with no repeated edges and whose endpoints are equal is called a *circuit.* A *cycle* is a circuit with no other repeated vertices except its endpoints.

Definition 3.6 Two vertices u and v of a graph G are said to be *connected* iff there is a nondirected path between u and v in G, and the graph G is said to be connected iff each pair of its vertices is connected.

3.2.3 Trees and Their Properties

There are many special classes of graphs whose properties make them important models of representations. One of these classes is the *tree* whose basic definitions and properties are presented here. Proofs of the theorems can be found in Mott [43].

Definition 3.7 A *tree* is a simple graph such that there is a unique simple non-directed path between each pair of vertices of G. A *rooted tree* is a tree in which there is one designated vertex, called a *root*. A rooted tree is a *directed tree* if there is a root from which there is a *directed* path to each vertex. In a directed tree, there can be only a single root. The *level* of a vertex v in a rooted tree is the length of the path to v from the root. A tree T with only one vertex is called a *trivial tree*; otherwise, T is a *nontrivial tree*.

An example of a directed tree is shown in figure 3-3. In this case, the directed tree $T = (V, E)$ where $V = \{a, b, c, d, e, f, g, h\}$ and $E = \{(a, b), (a, c), (a, d), (b, e), (d, f), (e, g), (e, h)\}$. The root of T is a, and the level of f is 2 whereas the level of g is 3.

Definition 3.8 If a graph G is connected and e is an edge such that $G - e$ is not connected, then e is said to be a *bridge* or a *cut edge*. If v is a vertex in

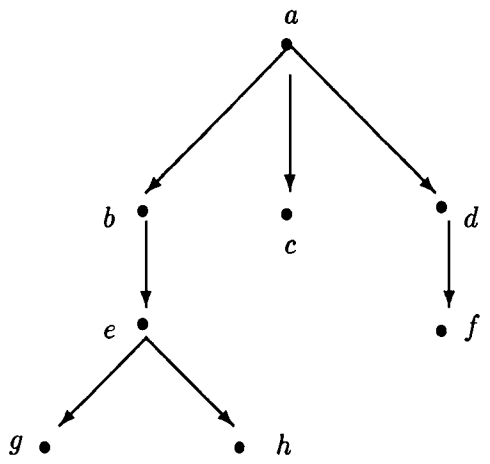


Figure 3.3: A directed tree.

G such that $G - v$ is not connected, then v is a *cut vertex*.

The following are a few interesting and important properties of trees [43] in general:

Theorem 3.1 A simple nondirected graph G is a tree iff G is connected and contains no cycles.

Theorem 3.2 In every nontrivial tree there is at least one vertex of degree 1.

Theorem 3.3 A tree with n vertices has exactly $n - 1$ edges.

Theorem 3.4 If two non-adjacent vertices of a tree are connected by adding an edge, then the resulting graph will contain a cycle.

3.2.4 Tree Representation

Comparing the model in figure 3-1(b) with figure 3-3, it is apparent that we can represent the clock distribution as a graph in general, and as a directed tree in

particular. The nodes in the tree would be either buffers or functional modules.

Let the clock distribution be represented by

$$T = (V, E), \tag{3.1}$$

where V is the set of vertices and E is the set of corresponding edges. Suppose there are n number of functional modules, m number of local buffers and 1 pad driver. Then, the tree T will have $(m+n+1)$ elements in the set V and $(m+n)$ elements in the set E. Further, the pad driver will be the root of the tree and all the elements of V whose level is 1 are the representations of local buffers. All the elements in V whose levels are 2 or greater represent the functional modules. Since T contains no cycles, every edge in the tree is a cut edge and every vertex is a cut vertex. Also, since the tree is acyclic, the clock signal will be routed to each cell in only one path.

In distributing the clock, the inputs are the functional cells and the buffers. Therefore, the problem is to find the elements of the set E, given all the elements of the set V, and some optimality criteria. Hence, the problem is essentially one of assignment and routing. To satisfy the optimality criteria, it may be necessary to not assign any functional modules to certain local buffers.

3.3 Constraints

3.3.1 System Level Constraints

Constraints at the system level have to be clearly stated for the design methodology derived in this work to be valid. Some of these constraints are dictated by the design environment, and some others by the necessity of having to limit the complexity of the problem. The constraints imposed at the system level in this work are as follows:

1. *The entire system is on a single substrate.* This limitation is imposed to ensure that the type of interconnect does not vary within the system. For example, chip-to-chip clock distribution on a printed-wiring-board or a thick-film hybrid circuit would not come within the purview of this methodology. Even though WSI could theoretically be included, the methodology is ideally suited for a large VLSI chip. This constraint can be removed if precise information about the interconnects between modules is available.

2. *The system is totally synchronous.* Therefore, self-timed modules within the system, as well as globally-asynchronous-locally-synchronous systems are eliminated. Further, it is assumed that the architecture is such that large delays in the data path do not make a certain amount of clock skew *essential.* If there are variations in the arrival of signals at different points

within the system, time windows would have to be provided to the clock signal rendering the problem extremely complex.

3. *The clock signal is externally generated.* This implies that the clock signal is brought into the system through a pad. The buffer near the pad then drives local buffers, which in turn supply the clock to all the functional blocks.

4. *There is only one clock signal to be distributed.* This constraint has two implications. One is that the clocking scheme is a single-phase methodology. In most CMOS VLSI circuits, at least a two-phase clocking scheme is generally adopted. However, the two phases have to be distributed in parallel. Therefore, the distribution methodology would still be the same. The second implication is that the communication clock between the modules is the same as the process clock, and not a slower derived signal.

5. *The clock signal is one of the global signals to be routed after placement of the functional blocks [38,39].* The placement of the functional blocks is based on architectural constraints. Commercial routers that interconnect these modules do not route global signals such as power, ground and the clock signal. Therefore, when the clock signal is distributed, the functional blocks cannot be re-routed.

6. *Functional blocks are either standard cells or custom macro-cells whose input capacitances are known.* If the module is a standard cell, the input

capacitance is provided in the data sheet. If the module is a custom macro-cell, the dimensions of the input device and the process parameters are known, and the input capacitance can be calculated.

7. *The physical distances between any two modules or a module and a buffer can be accurately represented by the Manhattan distance.* This assumes that all the interconnections run either horizontally or vertically, which is a common practice in digital VLSI design. Also, it is assumed that a feasible minimum distance routing can be performed between any two points in the system.

8. *The locations of all the functional modules and possible buffer sites are known.* This also implies that the number of modules and the number of possible buffer sites are finite and known quantities.

3.3.2 Buffer Constraints

The selection, location and sizing of buffers are crucial to any clock distribution system. The constraints imposed on these factors in this work are as follows:

1. *Buffers are used in a hierarchical tree-based structure.* If the interconnect lengths are small, or if the interconnect resistance is small, the use of a single centralized buffer would suffice to drive all the functional blocks [44]. However, due to the scaling down of feature sizes and the resulting higher interconnect resistance, a single central buffer will not be sufficient. The

RC load of the interconnect and the functional modules would have to be occasionally buffered to convert it to a purely capacitive load [35].

2. *Buffers can be placed only at predetermined sites.* Except for the pad buffer, all other buffers would have to be accomodated after all the functional blocks are placed and routed. The free space available not only determines the location of the buffers but also the upper limit on the number of buffers that can be used in the clock distribution.

3. *All the local buffers are of the same size.* In reality, the number of stages and the sizing of each buffer can be fine tuned in a given system to suit the loads. This would result in better performance. However, to limit the complexity of the problem, this extra step is not considered in this work.

4. *The output impedances of all the buffers are the same and are known.* Even though each local buffer may be made up of several stages with gradually increasing sizes, the impedance of the final stage is the only one that affects delay calculations. This impedance is mainly resistive and can be calculated knowing the W/L ratio, C_{gox} and V_T of the final stage.

5. *The local buffers have the capability to drive the maximum load that can be connected within the system.* This can be accomplished while sizing the local buffers. This constraint is imposed mainly as a result of constraint 3 above, and the fine tuning of the buffers is not considered.

3.3.3 Interconnect Constraints

The dimensions of the interconnects as well as the maximum frequency of operation have direct bearings on the modeling of the interconnect in delay estimation. These and other constraints imposed on the interconnect are as follows:

1. *The width and the height of the interconnect are in the* $1 - 2\mu M$ *range and the maximum length of the interconnect is in the* $100,000\mu M$ *range.* This basically excludes large interconnects used in wafer scale integration. Since this work is aimed at large VLSI chips, the dimensions assumed here are fairly typical.

2. *The maximum frequency of operation is assumed to be in the region of* $50MHz$. Since CMOS is the dominant technology at present, and most CMOS systems operate in the $16-40MHz$ range [20], the constraint is reasonable. However, ECL systems that can operate in the $100MHz$ region and GaAs systems that can operate as high as $500MHz$ are excluded.

3. *Interconnects exhibit RC instead of LC or RLC behaviour.* This is a direct result of the first two constraints. Let us suppose that a given interconnect has a width of $2\mu M$, a height of $1\mu M$ and a length of $1cM$. This interconnect, if made of a *metal1* layer, would have a resistance of 425Ω, capacitance of $1.5pF$ and an inductance of $1.67nH$ [18]. At a frequency of $50MHz$, $X_C = 2120\Omega$ and $X_L = 0.525\Omega$. It is readily apparent from these numbers that we are amply justified in using a RC model for the

interconnect.

4. *There are no vias in the interconnect.* This constraint implies that the impedance of an interconnect can be derived from the dimensions. The constraint is imposed because of two reasons. Firstly, it is impossible to find the number of *vias*, if there are any, at the initial design stage before performing the routing. Secondly, it is common to reserve the *metal2* layer for the distribution of the global signals, thereby minimizing, if not totally eliminating, the impedances in the *vias*.

5. *Cross talk between adjacent wires is negligible.* Cross talk is the coupling between adjacent lines that can result in noise. The amount of noise generated will be directly proportional to the length and inversely proportional to the distance between the two interconnects. In this work, it is assumed that the distance between any two adjacent interconnects is large enough to make the cross talk noise negligible.

3.4 Optimality Criteria

3.4.1 General

Optimality is always measured in terms of an objective function, whose validity authenticates the correctness of the solution methodology. With that perspective, it is correct to state that this section is the most important in this chapter. Further, optimality connotates a fairly wide range of interpretations. As such,

in this section, the two main issues that limit performance, clock skew and system speed, are examined briefly. Further, the objective function that is used to measure optimality is derived.

3.4.2 Clock Skew

Clock skew occurs when the clock signal arrives at different times at different registers. These differences depend on the delays in the buffers and interconnects. Further, the input capacitances of the functional modules also affect the signal delays. The assignment of the functional modules to each buffer directly affects the maximum delay in that particular sub-circuit. The differences in these maximum sub-circuit delays constitute the clock skew. Since the system is totally synchronous, these differences should be eliminated for proper operation of the circuit. Therefore, minimizing the clock skew is a major criterion in the objective function.

3.4.3 System Speed

The other major criterion for optimality is system speed. Minimizing clock skew does not necessarily result in maximum system speed. System speed depends upon the selection of the functional modules assigned to each local buffer, as well as the actual routing of the clock signal from each local buffer to all the assigned functional modules. Therefore, minimizing the critical path of each local buffer has to be incorporated in the objective function, where the critical

path is the path of the largest delay.

3.4.4 Objective Function

It should be recalled from section 3.2.4 and equation 3.1 that the clock distribution system is represented as a directed tree $T = (V, E)$, where V is the set of vertices and E is the set of edges. The clock distribution problem is to find E, given V. At level 0, T contains m local buffers. Therefore, let T be divided into one trivial tree containing the pad driver, and m nontrivial trees, $T_i, i = 1, ..., m$ by removing all the edges connecting the root of the tree T and the vertices at level 1. In each tree T_i, the root is a local buffer and all the vertices with level ≥ 1 are the functional modules assigned to that particular tree. The n functional modules will then have to be assigned to m trees T_i such that,

$$Min(d_i), i = 1, ..., m \tag{3.2}$$

where d_i is the delay in the critical path of tree T_i

and

$$Min \sum_{j,k=1}^{m} \|(d_j - d_k)\|, j \neq k \; and \; 1 \leq m \leq b, \tag{3.3}$$

where b is the maximum number of potential buffer sites. This implies that not all the potential local buffer sites need to be utilized. On the contrary, the optimal number of local buffers, m, needs to be determined. The second objective function above is the sum of all the individual skews, taken two nodes at a time. It should be noted that the two minimizations stated above are

interdependent, and therefore, assignment and routing have to be simultaneously performed in each iterative step of the selection of m.

CHAPTER 4

Algorithm

4.1 General

The solution methodology presented in this chapter encompasses the assignment of the functional modules to local buffers, routing of the clock signal from each local buffer to the assigned functional modules, estimation of the delay in the critical path of each local buffer, and iterative refinement of the distribution to achieve the objective function presented in the previous chapter. The algorithm does not specifically include the routing between the pad buffer and the local buffers. Since local buffers are selected to minimize the objective function, and the position of the pad driver is fixed, any standard routing algorithm can be utilized to connect the local buffers to the pad buffer.

The algorithm is centered around assigning the functional modules to local buffers and the subsequent routing. The solution to the assignment problem is based on classical pattern analysis techniques. This is explained in section 2 of this chapter. In section 3, the shortest path algorithm is shown to be ideal for routing the clock signal. The estimation of the delay in the critical path is a major part of evaluating the objective function, which is explained in section 4.

The actual algorithm is presented in section 5.

4.2 Assignment of Functional Modules

4.2.1 The Assignment Problem

The problem of generating the clock distribution tree $T = (V, E)$ has been divided, as discussed in section 3.4.4, to one of basically generating the nontrivial trees $T_i = (V_i, E_i)$, $i = 1, ..., m$, where m is the optimal number of local buffers. Even though the assignment problem consists of assigning each of the n functional modules to one of the m local buffers, several complexities arise in the process. To start with, the number m is the optimal number of buffers that have to be selected from the b number of possible buffer sites. Moreover, functional modules cannot be assigned to the nearest possible buffer site since the delay in the interconnect is proportional to the square of the length of the interconnect, and the maximum delay in each tree T_i needs to be minimized. It is necessary to use a methodology which is hierarchical. Also, the assignment cannot be performed local to each tree T_i, but globally to all the trees simultaneously. This can be performed by starting with all the functional modules and iteratively combining them into larger groupings. In such a methodology, the iteration can be discontinued when the optimal number of groupings have been found.

4.2.2 Relationship to Cluster Analysis

Classical pattern analysis techniques have grown out of machine perception study, and are normally based on features extracted from various sensing modalities [45]. In spite of its origins, the classification methodologies are fairly general and can be applied in many areas. Pattern classification can be either statistical or structural. In this work, structural techniques are utilized to solve the assignment problem. Statistical methods are based on the assumption that probability density functions of all the clusters are known before classification. Since such a priori knowledge is not available in the assignment problem, statistical methods cannot be utilized.

The structural analysis techniques are very diverse, both in terms of methodologies as well as applications. The choice of a particular algorithm is based on the application and the similarity measure used in classification. In completely unsupervised classification, the samples that need to be formed into clusters have no specific labels attached to them. The clustering process is then reduced to finding natural groupings amongst the samples. The grouping is based on similarity measures, normally called distance measures. However, the distance need not be Euclidian distance, but any other homogeneity criterion [45].

4.2.3 Agglomerative Clustering

A brief description of a specific clustering technique known as *agglomerative clustering* is provided in this section. A detailed description can be found in the

book on pattern analysis by Duda & Hart [45]. A simple example is provided to elucidate the algorithm.

Consider the problem of partitioning n samples into m clusters. In hierarchical clustering, when two samples are in the same cluster at any level in clustering, they remain in the same cluster at all subsequent levels. A major advantage of hierarchical clustering is its conceptual and computational simplicity. In hierarchical clustering, algorithms could be either *divisive* or *agglomerative*. In a divisive clustering algorithm, all the samples are formed into one cluster initially. This cluster is then divided into smaller clusters based on a *dissimilarity measure*. In agglomerative clustering, the samples are initially divided into n clusters, each cluster containing only one sample. These clusters are iteratively merged to form bigger clusters. Therefore, at the k^{th} step, there are $n - k + 1$ clusters. At the n^{th} level, all the samples are merged into one cluster. The merging of the clusters is generally represented in the form of a tree known as a *dendrogram*.

The major steps in agglomerative clustering are as follows [45]:

1. Let $c = n$ and $\mathcal{X}_i = \{x_i\}, i = 1, ..., n$
2. If $c \leq m$, STOP.
3. Find the nearest pair of distinct clusters $\mathcal{X}_i$ and $\mathcal{X}_j$.
4. Merge $\mathcal{X}_i$ and $\mathcal{X}_j$, delete $\mathcal{X}_j$, and decrement c by one.
5. Go to step 2.

Here, $\mathcal{X}_i$ is the i^{th} cluster and x_i is the i^{th} sample. The algorithm will stop when $c = m$. However, if we let $m = 1$, the entire dendrogram will be generated. It should also be noted that in step 3, the nearest pair of distinct clusters are not necessarily nearest to each other physically. Any appropriate distance measure can be used. Since the major concern in the clock distribution is the longest delay in each cluster, consider the use of the minimum of the maximum distances between any two elements belonging to the two given clusters $\mathcal{X}_i$ and $\mathcal{X}_j$. This distance measure is given by

$$d_{max}(\mathcal{X}_i, \mathcal{X}_j) = max \parallel x - x' \parallel, x \in \mathcal{X}_i, x' \in \mathcal{X}_j \tag{4.1}$$

It should be noted that this is only a different form of the first objective function that was depicted in equation (3.3).

A set of eight samples are shown as an example in figure 4-1(a). The complete dendrogram, using the d_{max} distance measure, is shown in figure 4-1(b). The merging of clusters is not based on the nearest clusters, but on the minimum of the maximum distances between clusters. To illustrate this point, consider level 7 where x_8 is merged with $\{x_1x_2x_4x_5\}$. This merging is performed not because x_8 is closer to x_5 than x_7, but because x_2 is closer than x_3. One point to note is that even though clusters obtained at any level are compact, the clusters are unlikely to have the same number of elements in them.

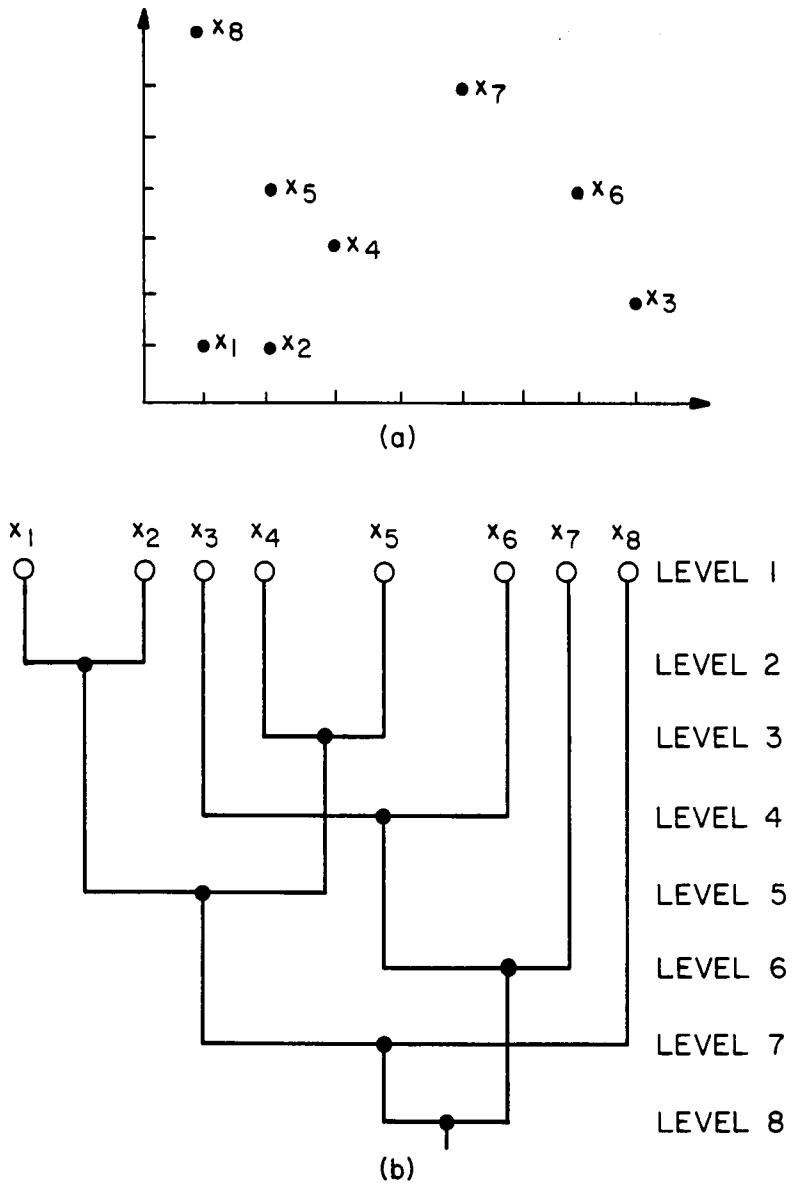


Figure 4.1: Example of Agglomerative Clustering (a) Samples (b) Dendrogram

4.3 Routing of the Clock Signal

4.3.1 General Routing Issues

The material in the previous section dealt with the selection of the vertices V_i in the trees $T_i = (V_i, E_i), i = 1, ..., m$. This section deals with the derivation of the edges E_i. In other words, once a set of functional modules are assigned to a particular local buffer, the clock signal has to be routed from the buffer to all the modules. This routing has to be performed so as to minimize the time taken by the clock signal to arrive at all the modules. This time depends on the length of the interconnect, per unit interconnect resistance, per unit interconnect capacitance, and the input capacitance of each functional module. We can, therefore, attach a cost c_{ij} to each pair of nodes, which would be the time taken for the signal to traverse from node i to node j. It is essential to minimize the total cost in the critical path in each tree T_i.

The simplest methodology would be to generate the *minimum spanning tree (MST)*, which is a graph that connects all the nodes with minimum total length. Computationally, generating the MST from the dendrograph is very simple. However, the MST does not always provide the smallest cost in the critical path. Another standard procedure is the *shortest path algorithm* generally attributed to E. W. Dijkstra [46]. This algorithm is explained in this section along with an example. Details of the algorithm can be found in all standard books on network flow programming [46,47].

4.3.2 Dijkstra's Shortest Path Algorithm

The shortest path algorithm assumes that the input to the procedure is a graph $G = (V, E, C)$, where V is the set of vertices, E is the set edges, and C is the set of costs associated with the edges in E. Further, it is assumed that none of the costs in C are negative. The algorithm then provides the shortest path (in terms of cost) to every single node in V from a designated root node. In the clock distribution system, the root node is the local buffer.

The algorithm constructs the path in a hierarchical method. Once an edge is included in the given path, it is never removed. The included edge will be the best available at that level in the procedure. Therefore, the algorithm is called a *greedy algorithm.* To maintain an efficient data structure, the algorithm actually generates a label of the form (P_i, d_i) for each node, where P_i is the predecessor node in the chain and d_i is the distance from the root node to the node i. Generating the shortest path from any node to the root node would simply be a matter of traversing through predecessor nodes successively.

Slightly varying versions of the basic shortest path algorithm are available. The algorithm, as explained by Murty [47] is as follows:
Let X, Y and Z denote the sets of permanently labelled, temporarily labelled and unlabelled nodes, respectively.

1. Label the origin 1 with $(0, o)$ and make it permanently labelled. For each j such that $(1, j) \in E$, label j with $(1, c_{ij})$ and make it temporarily labelled. Go to step 2.

2. If $Y \neq 0$, find an $i \in Y$ with the property that the distance index d_i of i is the least among nodes in Y at this stage. Break ties arbitrarily. Make the current label on the selected node i, its permanent label. That is, move i from the set Y to the set X. Let the permanent label on i be (P_i, π_i).

   For each $j \in Y$, let d_j be the distance in the present label of j. If $(i,j) \notin E$, or if $(i,j) \in E$ and $d_j \leq \pi_i + c_{ij}$, leave the temporary label on j unchanged. If $(i,j) \in E$ and $d_j > \pi + c_{ij}$, change the label on j to $(i, \pi_i + c_{ij})$.

   For each $j \in Z$ such that $(i,j) \in E$, temporarily label j with $(i, \pi_i + c_{ij})$, and move j from Z to Y.

   Go to step 3.

3. If $X = V$, terminate. If $X \neq V$, but $Y = 0$, there exists no path from 1 to any node in the set Z at this stage. Terminate.

   If $X \neq V$ and $Y \neq 0$, go back to step 2.

4.3.3 Example

The shortest path algorithm is best illustrated by means of an example. Consider the directed graph shown in figure 4-2(a). The circled numbers are the node numbers, and the numbers next to the edges are the costs associated with every edge.

Figures 4-2(b), 4-2(c), 4-3(a) and 4-3(b) show the sequence of operations. In the first stage in figure 4-2(a), node 1 is the root and is permanently labelled,

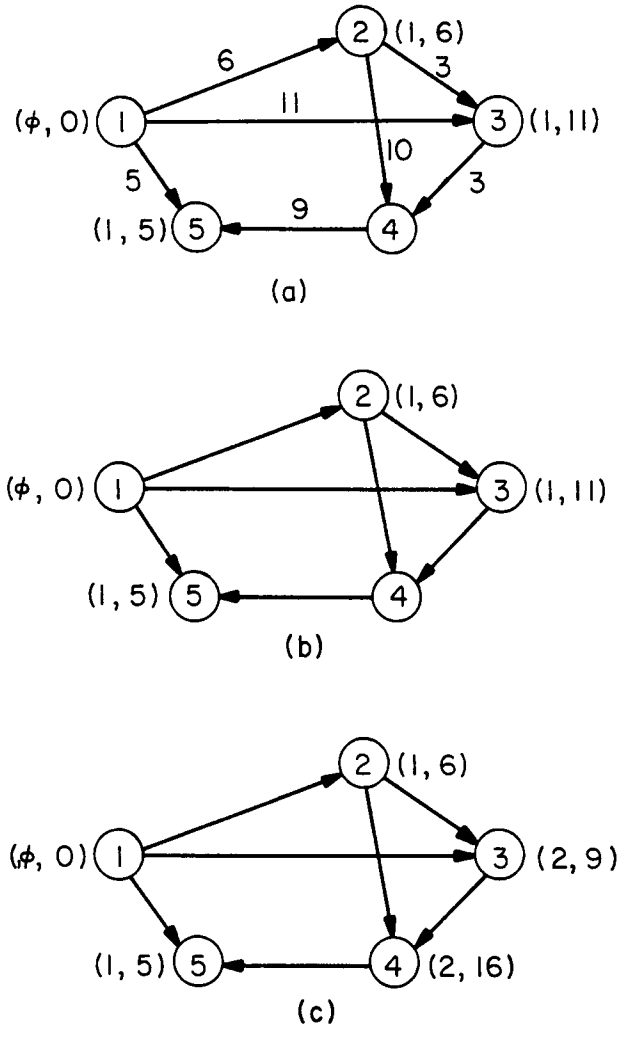


Figure 4.2: Example of Shortest Path Algorithm

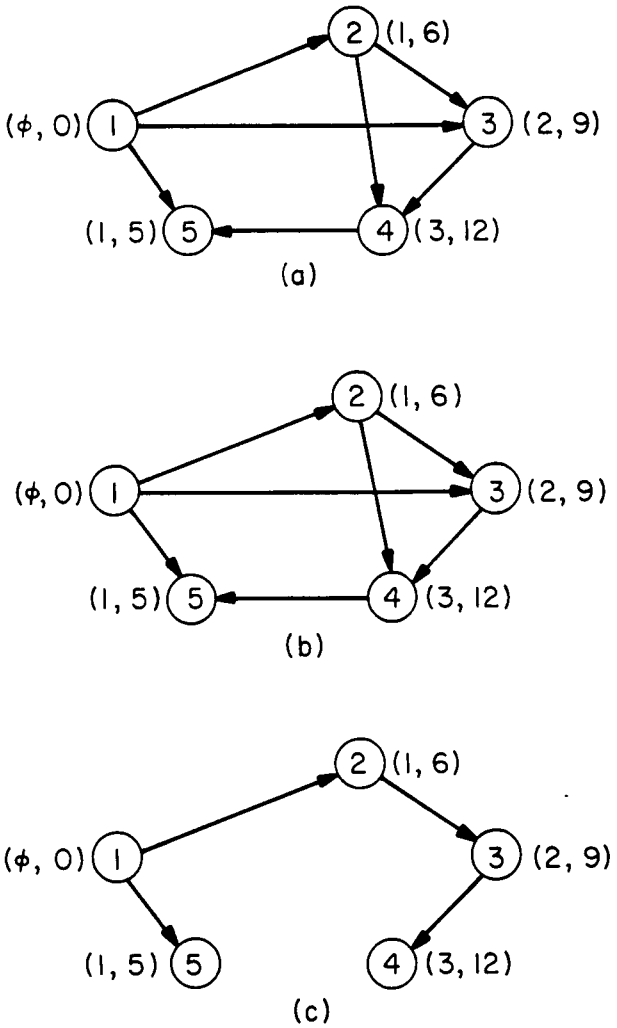


Figure 4.3: Result of Shortest Path Algorithm

where as nodes 2, 5 and 3 have temporary labels. In figure 4-2(b), node 5 is permanently labaled whereas nodes 2 and 3 continue to have temporary labels. In the next step, as shown in figure 4-2(c), node 2 is permanently labelled. The temporary label on node 3 has changed since the cost is less to go through node 2 rather than directly. Also, node 4 has acquired a temporary label. Figures 4-3(a) and 4-3(b) show two more iterations of the algorithm until all the nodes have permanent labels. Finally, figure 4-3(c) shows the shortest path to all the other nodes from node 1 with all the extraneous edges eliminated.

4.4 Delay Estimation

4.4.1 RC Mesh

The modeling of the interconnect as an RC mesh has been espoused and justified in the previous chapters. Since the resistance and the capacitance are evaluated as lumped elements, the input capacitances to the functional modules add to the interconnect capacitance. The delay in the critical path of a tree T_i would be the delay from the root of the tree to the node with the largest cost as obtained by the shortest path algorithm. In this section, the estimation of the time delay in the critical path is explained, along with an example. The model and estimation methodology used in this work is based on the work of Horowitz [48] and improved by Wyatt [23].

The model used is an RC tree where all the elements are resistors and ca-

pacitors driven by a single source. This model is appropriate since each tree T_i has only one buffer. Further, in the RC tree, none of the resistors are grounded and one side of every capacitor is grounded. The circuit model is a tree, and not a general mesh, since there are no loops in any tree generated by the shortest path algorithm. The estimation of the delay in such a tree is explained in the next section.

4.4.2 Delay Estimation

Delays in RC trees can be accurately calculated by resorting to numerical solutions. Such an approach becomes computationally prohibitive even for moderately sized VLSI circuits [23]. Further, detailed information such as precise rise and fall times, provided by a numerical solution, are not essential in the algorithm developed in this dissertation. An estimation that is computationally reasonable, even for large circuits, is presented here.

The delay estimation is based on the *Elmore time constant* [23,49]. The Elmore time constant at the i^{th} node in the RC tree is given by

$$T_{D_i} \equiv \sum_{k=1}^{n} R_{ik} C_k \tag{4.2}$$

which is the sum of the elements in the i^{th} row of the $[RC]$ product matrix.

If an impulse were to be applied to the RC tree, its response would be similar to the waveform depicted in figure 4-4(a), with an area of 1. The response to a unit step function would be similar to the waveform shown in figure 4-4(b). The value of T_{D_i} is equal to the area above the response to the step input [23].

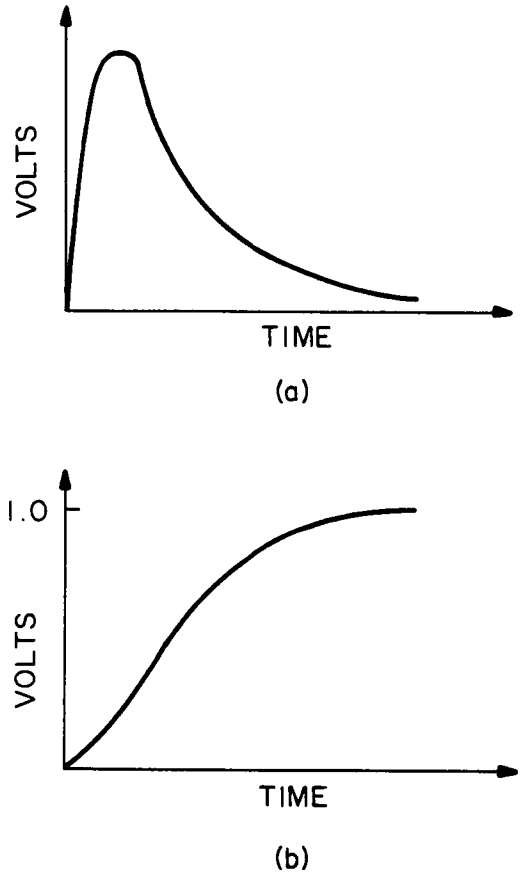


Figure 4.4: Response of the RC Tree to (a) Impulse (b) Step Input

The impulse response is similar to a probability density function, and therefore using T_{D_i} is an approximation to the average value of the impulse response.

4.4.3 Example

Consider the example circuit shown in figure 4-5(a). The square boxes represent functional modules and the distances between each pair of nodes in the tree are indicated. Further, let nodes 1, 3 and 4 have input capacitance of $1pF$, whereas node 2 has a value of $2pF$ and node 5 has a value of $0.8pF$. Let the output resistance of the buffer be 100Ω. The equivalent RC tree is shown in figure 4-5(b). The node numbers are circled, and are not the same as the numbers of the functional modules.

Suppose we want to estimate the delay in the critical path, which happens to be from the buffer to the functional module 3. This is equivalent to finding the delay from node 0 to node 4 in the RC tree.

$$T_{02} = 100 \times (0.2 + 1 + 0.2 + 2 + 0.6 + 1 + 0.3 + 1 + 0.4 + 0.8)pF \quad (4.3)$$

$$+20 \times (\frac{0.2}{2} + 1 + 0.2 + 2 + 0.6 + 1)pF \quad (4.4)$$

$$= 0.75nS + 0.098nS = 0.848nS \quad (4.5)$$

$$T_{24} = 60 \times (\frac{0.6}{2} + 1)pF = 0.078nS \quad (4.6)$$

$$Total\_Delay = 0.848nS + 0.078nS = 0.926nS \quad (4.7)$$

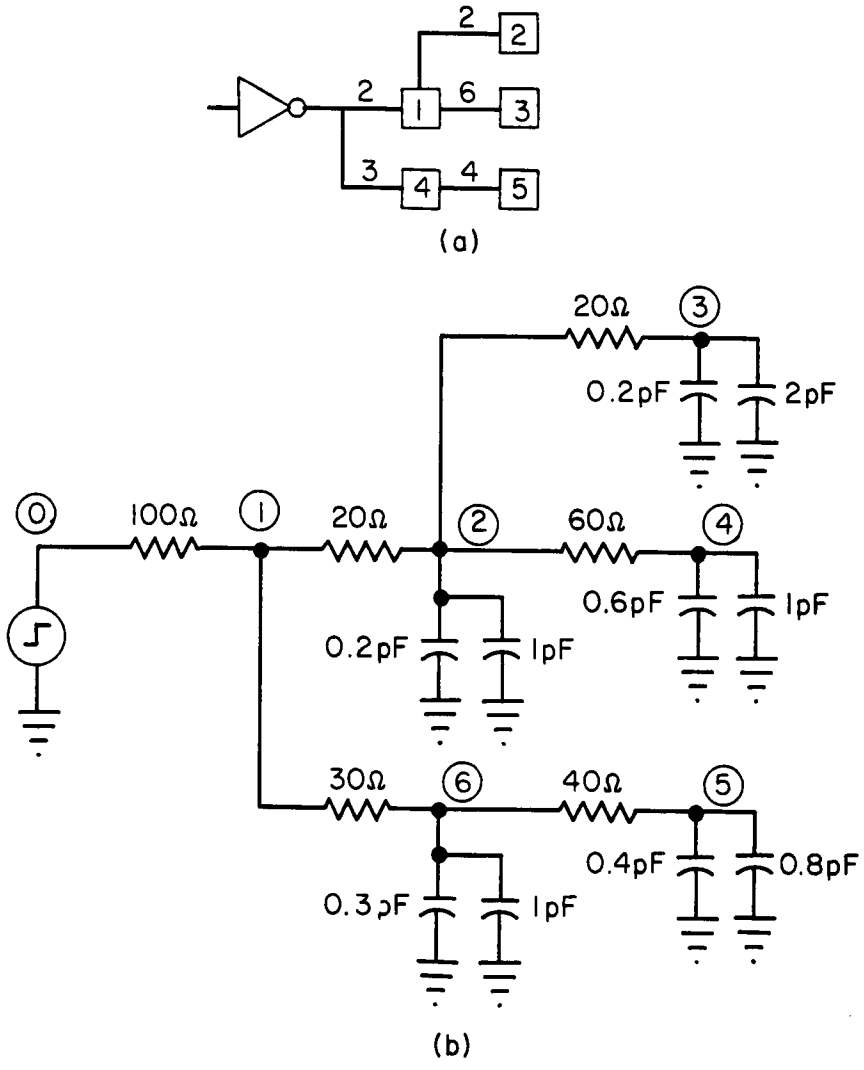


Figure 4.5: Example of Delay Estimation (a) Circuit (b) RC Tree

4.5 Algorithm

The various steps that comprise the algorithm developed to solve the clock distribution problem is presented in this section. The inputs to the system are the number of functional modules, X and Y coordinates of all the functional modules, the maximum number of buffer sites, X and Y coordinates of all potential buffer sites, per unit resistance and capacitance of the interconnect, input capacitances of all the functional modules and the output resistance of the local buffer.

The variables used in the algorithm are as follows:

i - index for the local buffer

$j\&k$ - indices for functional modules

n - number of functional modules

m - optimal number of local buffers

b - maximum number of buffer sites

R_i - output resistance of buffer i

r_{jk} - interconnect resistance between nodes j and k

c_{jk} - interconnect capacitance between nodes j and k

c_j - input capacitance of node j

γ_{jk} - cost function of the edge (j, k)

T_i - tree i

V_i - vertices of tree T_i

E_i - edges of tree T_i

T_{D_i} - Elmore time constant at node i

J - Error function

The following steps make up the algorithm:

1. Compute r_{jk} and c_{jk} for $1 \leq j, k \leq n, j \neq k$

2. Compute γ_{jk} for $1 \leq j, k \leq n, j \neq k$ using the equation

$$\gamma_{jk} = r_{jk}(c_{jk} + c_k) \tag{4.8}$$

3. Let $m = b$ and $J = \infty$

4. Using $max\{\gamma_{jk}\}$ as the distance metric, perform agglomerative cluster analysis, with the number of clusters equal to m. Assign each cluster to the nearest local buffer. This step generates V_i, the vertices of the tree T_i. Also, this step will satisfy the objective function in equation (3.3).

5. Use the shortest path algorithm to generate the edges E_i in each T_i.

6. Calculate the Elmore time constant T_{D_i} in the critical path of each T_i.

7. Compute the error function

$$J = max(T_{D_j}),\ 1 \leq j \leq m \tag{4.9}$$

8. If the value of J has decreased from the previous value, set $m = m - 1$ and go back to step 4.

9. If the value of J has increased from the previous value, or remained the same, let $m = m + 1$ and generate trees T_i, $i = 1, m$.

CHAPTER 5

Simulation Results

5.1 General

5.1.1 Introduction

The clock distribution methodology developed in this dissertation has been tested on three example problems. An explanation of the problems, the results of the application of the methodology to these problems, and some conclusions that can be drawn from the results are presented in this chapter. The problems were chosen such that different design environments were represented.

The values of interconnect parameters were obtained from real fabrication runs by MOSIS using the Hewlett-Packard foundry [50]. The data on the functional blocks were obtained from the standard cell library of the Microelectronic Design Division of the Mississippi State University [51]. The input capacitances of the functional blocks have been kept constant in the first two problems and this constant value is taken directly from the data sheet of cell 1830 in the standard cell library [51]. Since the methodology was designed to accept varying load capacitances also, the functional blocks in the third problem had different input capacitances and were not derived from data sheets.

The results are generally presented in the form of netlists connecting each local buffer, directly or indirectly, to each functional block. When the number of functional blocks is not too large, the resulting netlist is also depicted graphically.

5.1.2 Simulation

Ideally, delays in distribution need to be measured as opposed to simulated and/or estimated. This would provide an exact evaluation of the advantages of the derived distribution methodology. However, the results in this work were simulated due to two main reasons. Firstly, facilities capable of fabricating large chips were not available to the author. Secondly, and more importantly, the timing estimation used in this work has been used in computer-aided design of integrated circuits before, and is known to have a resonably low upper bound in error [23]. It should also be noted that the upper bound on the error can be further reduced using better timing analysis without changing the basic methodology. Even though an error may have been introduced due to some of the constraints used in this work, it should be small compared to the error in timing analysis, which itself is in the order of $3-5\%$ [23].

5.1.3 Implementation

The methodology was implemented using the FORTRAN language on a VAX 11/785 computer running the VMS operating system. The methodology was partitioned into its three main components: Clustering, routing by means of the

shortest path algorithm, and delay estimation. The assignment of a local buffer to each cluster and the termination of the algorithm was performed manually.

In the clustering routine, the X and Y coordinates, and the input capacitance of each functional block constituted the input data. The routine partitioned the functional blocks into m clusters, where m is the specified number of local buffers. Since the clustering was performed hierarchically so as to minimize the maximum delay between any two cells in each cluster, the interconnect resistance and capacitance had to be specified also.

The shortest path routine performed the routing after a local buffer was assigned to each cluster. It should be noted that for this routine to work, no negative costs can exist in the problem. Since all the resistances and capacitances are always positive, the problem of negative costs will never arise. The shortest path routine outputs the predecessor node of each node as well as the cost from node 0, which is always assumed to be the local buffer.

The output of the shortest path routine was input directly to the delay estimation routine that outputs the maximum delay in the critical path of each cluster in picoseconds. The execution of each routine was performed interactively to obtain optimal distribution.

5.2 Description of the Problems

The three example problems that were used in this dissertation, labelled Problem A, Problem B and Problem C, are shown in figures 5-1, 5-2 and 5-3, respectively. The three problems vary in complexity, with Problem A being the simplest.

Problem A consists of 25 functional cells, each represented by a square in figure 5-1, and 4 possible buffer sites, each represented by a circle. This could be typical of a 40 pin MOSIS *tiny chip* design environment. The cells and the buffers are evenly spaced with the horizontal spacing being 200λ and the vertical spacing being 150λ. In all the problems in this chapter, the feature size λ is assumed to be 2μ, whose per unit capacitance is $0.037fF$ and per unit resistance is 0.033Ω [50]. The local buffers are assumed to have output impedances of 100Ω and all functional cells are assumed to have an input capacitance of $166fF$ [51].

Problem B is more typical of a larger standard cell design containing a large number of standard functional cells, as well as a multiplier cell and a RAM block. Typically, the clock input to each row would be fed by a separate buffer. In this problem, there are 10 rows of 24 cells each. All cells are assumed to have the same height of 76λ, which is typical of standard cell libraries, and the spacing between any two rows is 100λ. Three different types of cells are used, whose widths are 32λ, 56λ, and 160λ, respectively. The three types of cells are distributed evenly in each row in blocks of three cells.

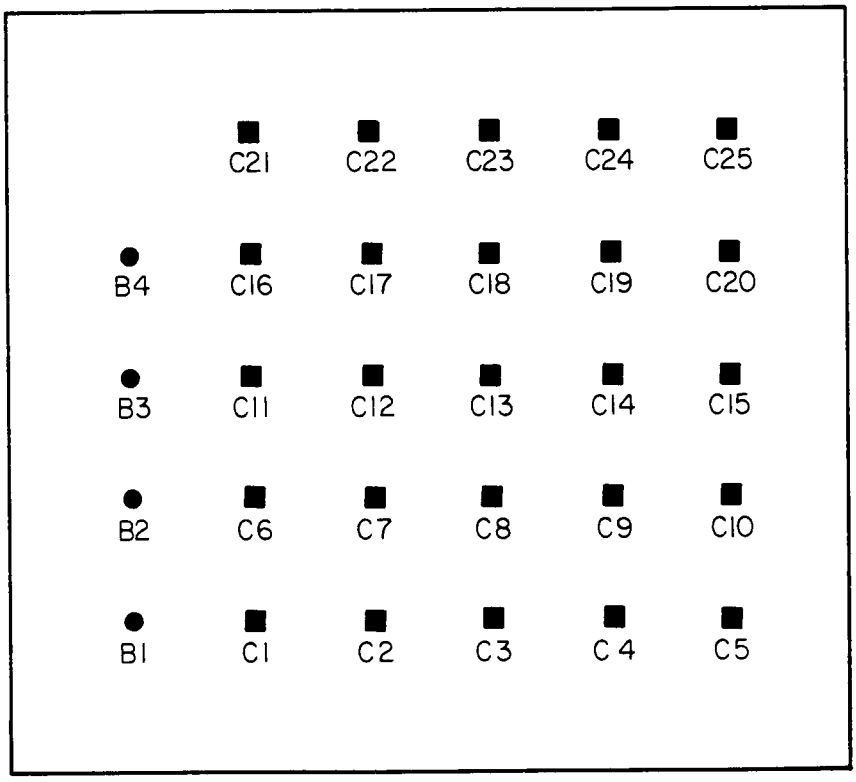


Figure 5.1: Problem A

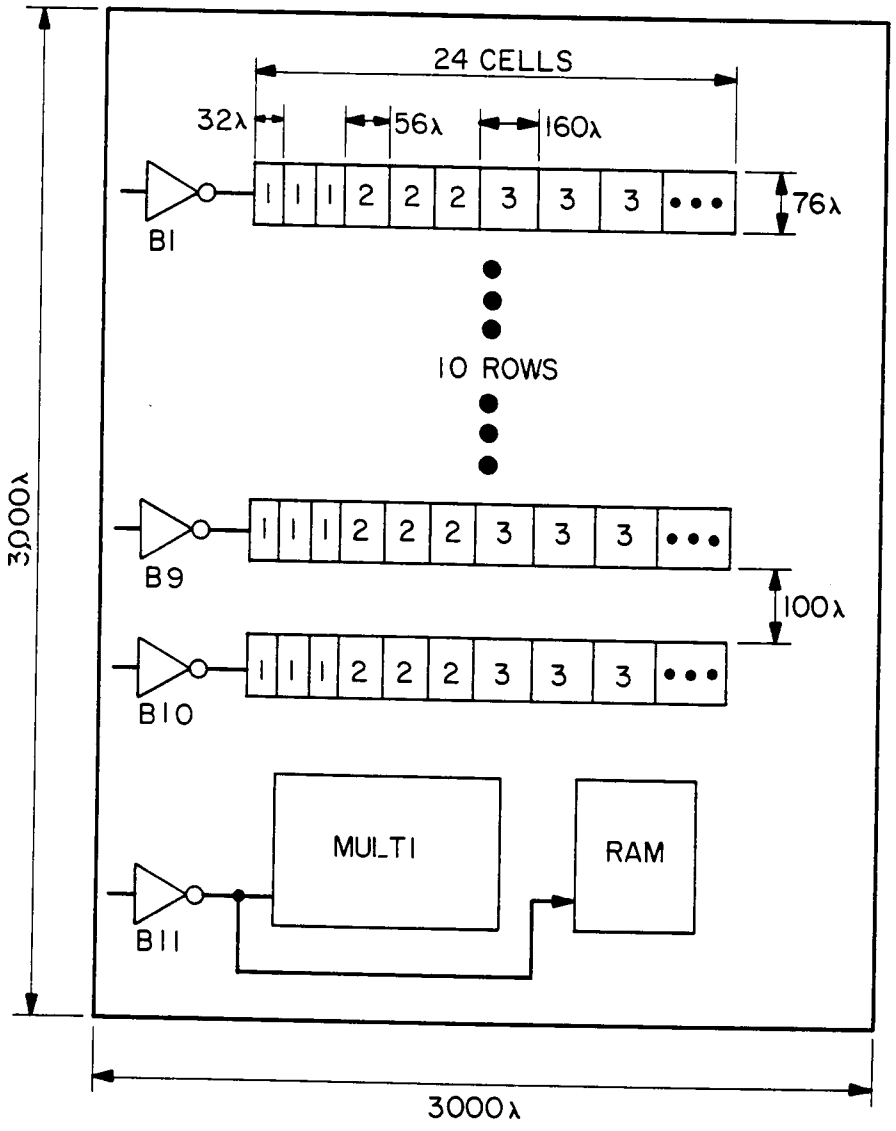


Figure 5.2: Problem B

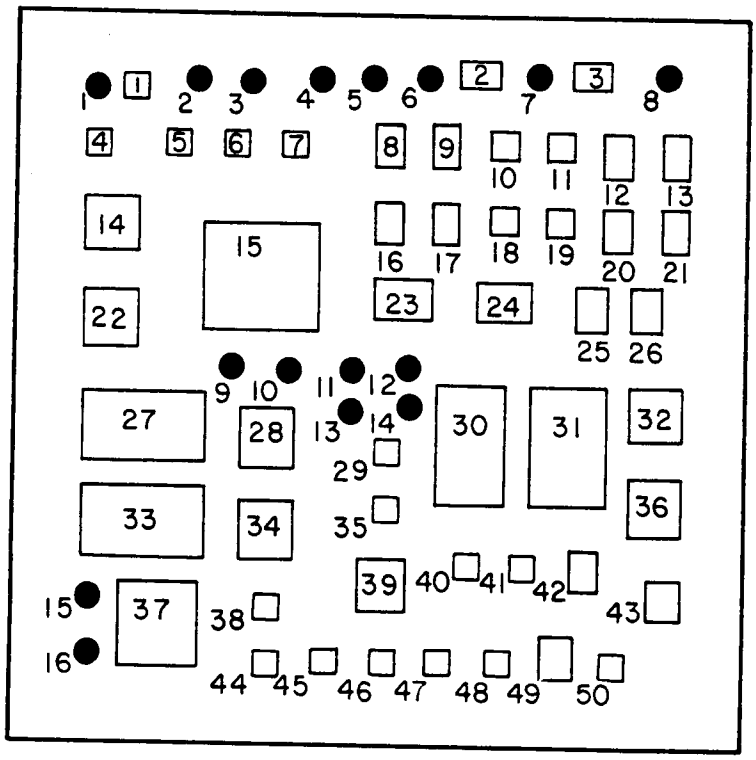


Figure 5.3: Problem C

Problem C is typical of a multi-chip module and is assumed to have dimensions of $100,000\lambda \times 100,000\lambda$. Both the potential buffer sites and and functional cells are randomly distributed. There are 50 cells and 16 potential buffer sites. The input capacitance of the cells vary between $50pF$ and $150pF$. All buffers are assumed to have the same output impedance of 100Ω.

5.3 Simulation Results

In Problem A, a manual assignment was attempted utilizing all four buffers. The results of this assignment are shown in table 5-1 and figure 5-4. The cells were evenly distributed among the four buffers and the maximum delay was $179.13pS$ in cluster 1. Using the derived methodology, the results of using four buffers are shown in table 5-2 and figure 5-5. Here, the maximum delay was $217.74pS$. However, when m was reduced to three, the delay reduced to $181.42pS$, as shown in table 5-3 and figure 5-6. When m was further reduced to two, the results are shown in table 5-4 and figure 5-7. Since the delay has now increased to $290.3pS$, the optimal distribution would be when $m = 3$. It should be noted that the number of buffers is less than the manual assignment while the delay is about the same. The final netlists for routing are shown in tables 5-5, 5-6 and 5-7.

In problem B, an assignment was made manually that connected all the cells in a row to a buffer and therefore, the delay was the same in all the rows. This

Table 5.1: Problem A: Manual Assignment

Cluster Number	Buffer	Functional Modules	Critical Path Delay In Pico Seconds
1	1	1, 2, 3, 4, 5, 9, 10	179.13
2	2	6, 7, 8, 11, 12, 13	114.53
3	3	14, 15, 19, 20, 24, 25	142.9
4	4	16, 17, 18, 21, 22, 23	114.53

Table 5.2: Problem A: $m = 4$

Cluster Number	Buffer	Functional Modules	Critical Path Delay In Pico Seconds
1	1	3, 4, 5, 8, 9, 10	142.15
2	2	1, 2, 6, 7,	75.25
3	3	13, 14, 15, 18, 19, 20, 23, 24, 25	217.74
4	4	11, 12, 16, 17, 21, 22	109.6

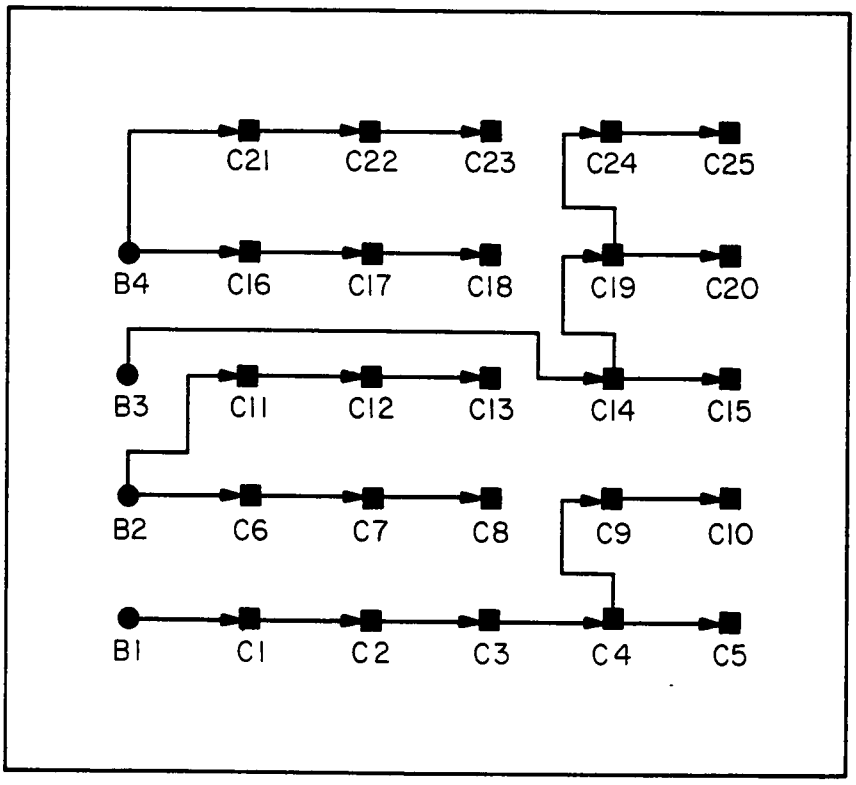


Figure 5.4: Problem A: Manual Assignment

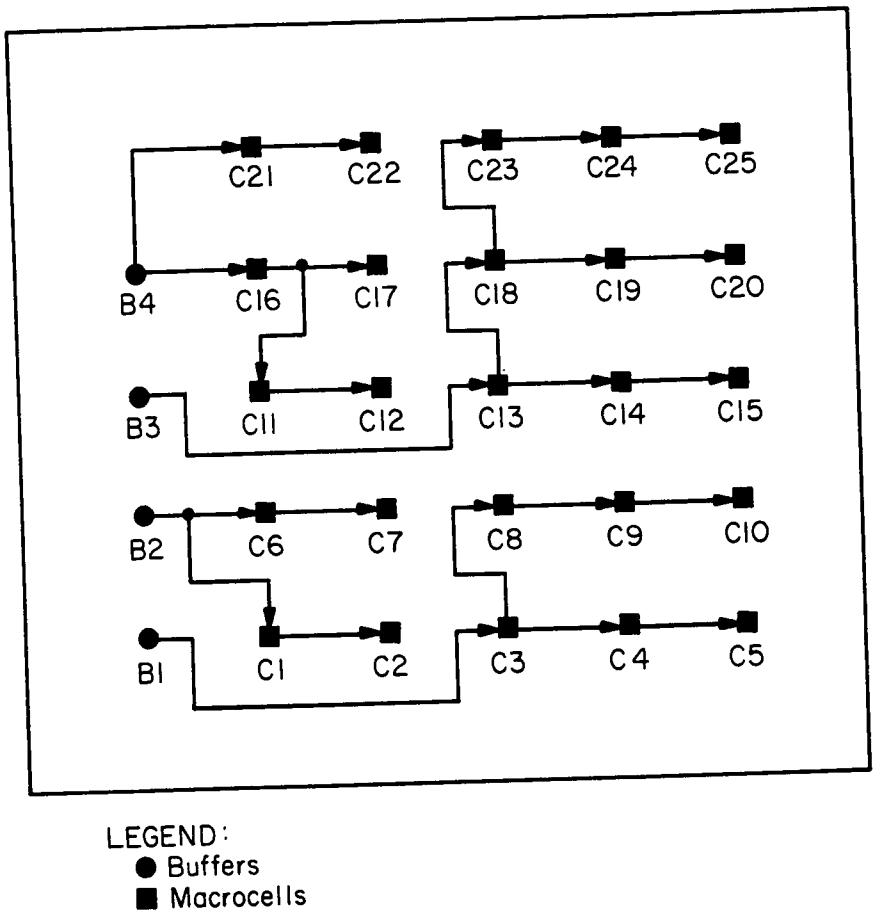


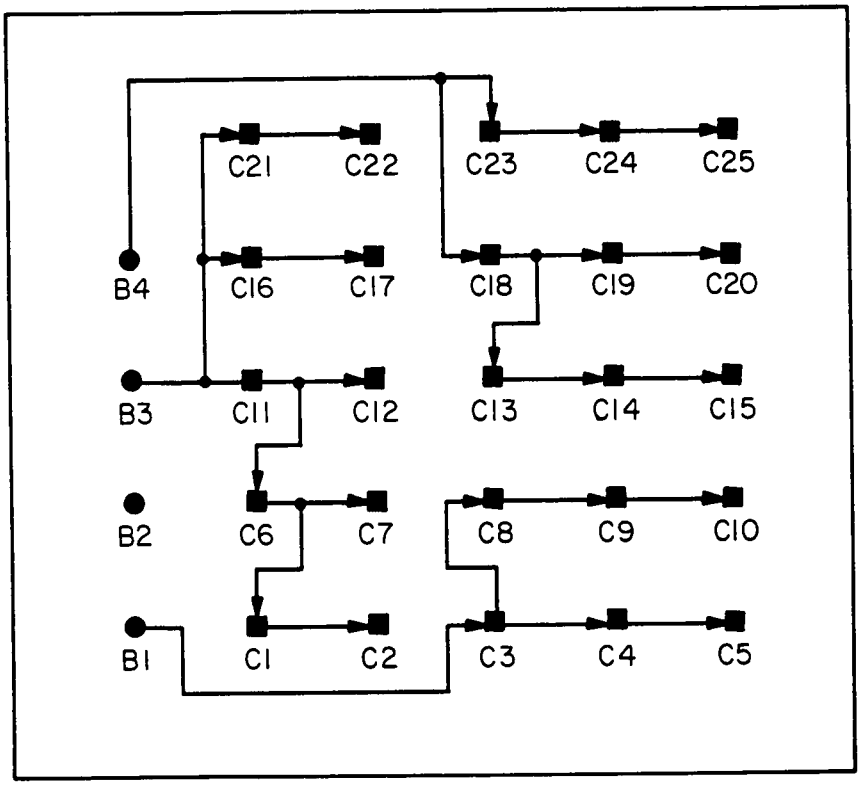
Figure 5.5: Problem A: $m = 4$

Table 5.3: Problem A: $m = 3$

Cluster Number	Buffer	Functional Modules	Critical Path Delay In Pico Seconds
1	1	3, 4, 5, 8, 9, 10	142.15
2	3	1, 2, 6, 7, 11, 12, 16, 17, 21, 22	181.42
3	4	13, 14, 15, 18, 19, 20, 23, 24, 25	176.62

Table 5.4: Problem A: $m = 2$

Cluster Number	Buffer	Functional Modules	Critical Path Delay In Pico Seconds
1	2	3, 4, 5, 8, 9, 10, 13, 14, 15, 18, 19, 20, 23, 24, 25	290.3
2	3	1, 2, 6, 7, 11, 12, 16, 17, 21, 22	181.42



LEGEND:
● Buffers
■ Macrocells

Figure 5.6: Problem A: $m = 3$

Table 5.5: Routing: Cluster 1 (Problem A)

Node	Previous Node
3	B1
4	3
5	4
8	3
9	8
10	9

delay was found to be $558pS$. The result of clustering the cells to nine clusters is shown in table 5-8. However, in this case, the maximum delay increased to $654.84pS$. Therefore, in Problem B, the manual assignment would be the optimal distribution.

In Problem C, due to the complexity of the problem, a reasonable manual assignment is not possible, especially since the input capacitances of the functional cells vary. The results of using the methodology with $m = 16$ is shown in table 5-9. The worst delay was in cluster 16, and was equal to $3816.63pS$. This is mainly because there were no potential buffers sites close to cells 45-50. When m was reduced to 15, the critical delay increased to $4610.71pS$ in cluster 10. Therefore, the optimal distribution is to use all 16 buffers. The netlists for routing for this problem are shown in tables 5-10 and 5-11.

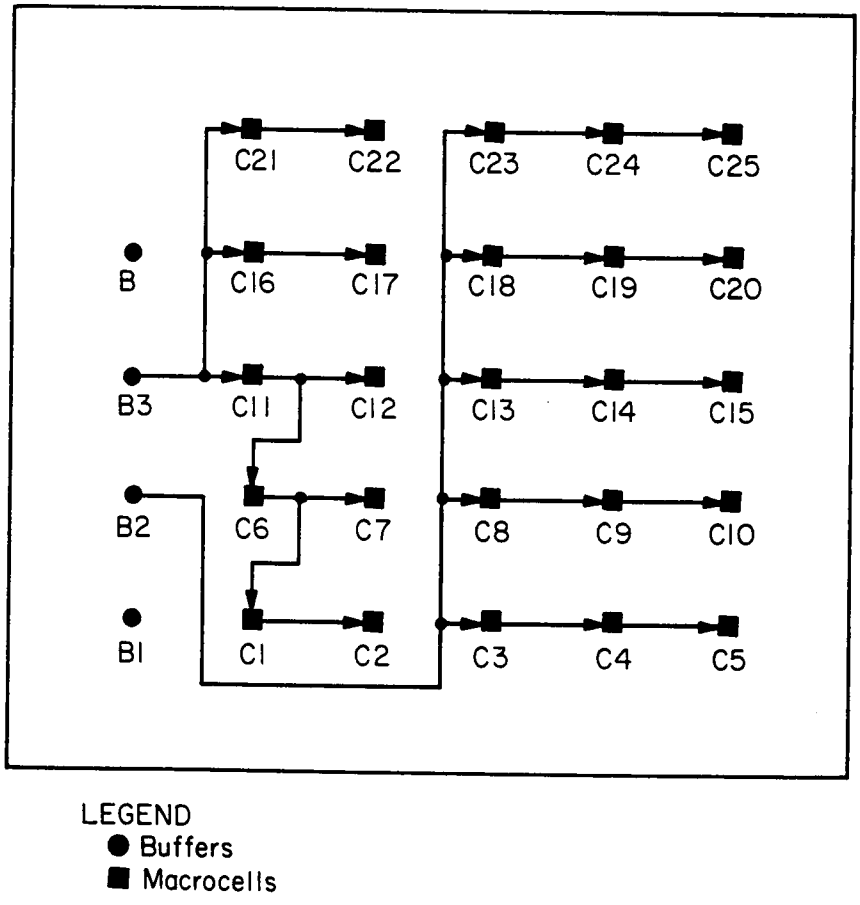


Figure 5.7: Problem A: $m = 2$

Table 5.6: Routing: Cluster 2 (Problem A)

Node	Previous Node
1	6
2	1
6	11
7	6
11	B3
12	11
16	B3
17	16
21	B3
22	21

Table 5.7: Routing: Cluster 3 (Problem A)

Node	Previous Node
13	18
14	13
15	14
18	B4
19	18
20	19
23	B4
24	23
25	24

Table 5.8: Problem B: $m = 9$

Cluster Number	Functional Modules
1	1,2,3,4,5,6,7,8,9,25,26,27,28,29 30,31,32,33,49,50,51,52,53,54,55 56,57,73,74,75,76,77,78,79,80,81
2	97,98,99,100,101,102,103,104,105,121,122,123 124,125,126,127,128,129,145,146,147,148,149 150,151,152,153,169,170,171,172,173,174,175,176,177
3	193,194,195,196,197,198,199,200,201 217,218,219,220,221,222,223,224,225
4	10,11,12,13,14,15,16,34,35,36,37,38,39,40 58,59,60,61,62,63,64,82,83,84,85,86,87,88
5	106,107,108,109,110,111,112,130,131,132 133,134,135,136,154,155,156,157,158,159 160,178,179,180,181,182,183,184
6	202,203,204,205,206,207,208 226,227,228,229,230,231,232
7	17,18,19,20,21,22,23,24,41,42,43,44,45,46,47,48,65,66 67,68,69,70,71,72,89,90,91,92,93,94,95,96
8	113,114,115,116,117,118,119,120,137,138 139,140,141,142,143,144,161,162,163,164,165 166,167,168,185,186,187,188,189,190,191,192
9	209,210,211,212,213,214,215,216 233,234,235,236,237,238,239,240

Table 5.9: Problem C: $m = 16$

Cluster Number	Buffer	Functional Modules	Delay in Critical Path In Pico Seconds
1	2	1, 4, 5, 6,	454.6
2	7	2, 9, 10, 3, 12, 11	659.13
3	5	7, 8	306.18
4	8	13, 21	375.79
5	1	14, 22	943.15
6	4	15, 16, 23	1063.75
7	6	17, 18, 24	1297.42
8	12	19, 20, 25, 26	1862.47
9	3	27, 28, 33, 34	2835.57
10	9	30, 31	2577.58
11	10	32, 36, 43	3987.76
12	13	37, 38, 44	2372.57
13	14	29, 35, 39	619.11
14	11	40, 41, 42	2433.18
15	15	45, 46, 47, 48	3033.93
16	16	49, 50	3816.63

Table 5.10: Problem 2: $M = 15$

Cluster Number	Buffer	Functional Modules	Delay in Critical Path In Pico Seconds
1	2	1, 4, 5, 6,	454.6
2	7	2, 3, 9, 10, 11, 12	659.13
3	5	7, 8	306.18
4	8	13, 21	375.79
5	1	14, 22	943.15
6	4	15, 16, 23	1063.75
7	6	17, 18, 24	1297.42
8	12	19, 20, 25, 26	1862.47
9	3	27, 28, 33, 34	2835.57
10	11	30, 31, 40, 41, 42	4610.71
11	10	32, 36, 43	3987.76
12	13	37, 38, 44	2372.57
13	14	29, 35, 39	619.11
14	11	40, 41, 42	2433.18
14	15	45, 46, 47, 48	3033.93
15	16	49, 50	3816.63

Table 5.11: Routing: Problem C

Cluster Name	Node	Previous Node
1	1	B2
	4	1
	5	B2
	6	B2
2	2	B7
	3	B7
	9	2
	10	B7
	11	B7
	12	11
3	7	B5
	8	B5
4	13	B8
	21	13
5	14	B1
	22	14
6	16	B4
	23	16
	15	B4

5.4 Summary of Results

There are several inferences that can be drawn from the results that were presented in this chapter. To start with, many problems look deceptively simple with respect to clock distribution. But, since the delay is not directly proportional to the distance and capacitive loads of the cells can vary, distribution of the clock signal by physical inspection is usually not feasible. Therefore, the utility of the derived methodology is apparent.

Yet, the methodology does not always provide better solutions compared to manual distribution, as was apparent in Problem B. This is due to the fact that, the hierarchical cluster analysis, due to the nature of the algorithm, does not distribute the cells evenly among the buffers. Other clustering techniques, such as the *nearest neighbor*, *k-nearest neighbor*, and *k-means*, need to be examined to improve this aspect of the methodology [45]. In particular, it is essential to come up with a clustering technique which will always provide *the optimal* clustering, in terms of skew.

The modeling of the buffer needs more attention. Firstly, the output resistance used is only the static resistance and would not be valid during transition. Secondly, the output resistance would not be constant in a bipolar transistor or a GaAs transistor. Therefore, the modeling used in this work would be suitable mostly in CMOS technology. The delay in the buffer itself need to be minimized by determining the optimal size of the transistors. When this is done ,

Table 5.12: Continued Routing: Problem C

Cluster Name	Node	Previous Node
7	17	B6
	18	17
	24	18
8	19	B12
	20	19
	25	B12
	26	25
9	27	B3
	28	B3
	33	28
	34	28
10	30	B9
	31	30
11	32	B10
	36	32
	43	36

Table 5.13: Additional Routing: Problem C

Cluster Name	Node	Previous Node
12	37	B13
	38	B13
	44	38
13	29	B14
	35	29
	39	35
14	40	B11
	41	40
	42	41
15	45	B15
	46	45
	47	46
	48	47
16	49	B16
	50	49

the overall delay in the system can be calculated. The results in this chapter do not provide the routing from the root node of the tree, the pad buffer, to the local buffers. The sizing of the interconnects connecting the pad buffer to the local buffers not only affect the delay, but also the design of the pad buffer itself. However, the derived methodology would work even in the case of symmetrically placed local buffers by assigning the functional blocks to the local buffers and providing the routing.

The computational complexity of the methodology is $\Theta(n^2)$. The shortest path algorithm is $\Theta(n^2)$, but can be reduced to $\Theta(n \log n)$ by using a *heap* data structure. The most expensive routine is the cluster analysis since it operates on all the functional cells. In Problem B, to generate nine clusters, the VAX 11/785 took about 3.36×10^7 cycles that corresponds to approximately 15 minutes of run time with low load on the machine. The shortest path routine and the delay estimation routine executed in just a few seconds.

CHAPTER 6

Conclusions

As stated in chapter 1, the specific technical objectives of this work have been:

- Evaluate the constraints that properly reflect the problems associated with clock distribution.
- Derive an objective function for optimization utilizing the derived constraints.
- Design an algorithm for optimal distribution of the clock signal.
- Test the utility of the algorithm on realistic problems.

In this context, this dissertation has provided a methodology for clock distribution that is application specific. This is a major improvement over methodologies that do not consider the placement of functional cells while selecting the placement of buffers and performing routing. Also, in the derived methodology, an optimal number of buffers is used, thereby reducing area and power dissipation.

In retrospect, some shortcomings are obvious. Firstly, sizing of the interconnect and buffers is not performed. Secondly, only netlists are provided for routing which need to be interfaced to a routing tool. Thirdly, by ignoring the

inductive component of the interconnect impedance, very wide ($> 4\mu$) interconnects are excluded. Finally, one is not sure that a more optimal grouping exists subject to a different interpretive subset stated initially. These issues will be addressed in future research on this topic.

Another related future research goal is the determination of the threshold point where one needs to switch over from RC type interconnects to LC type wide interconnects. Since the design of the buffer would be different in the two cases, the determination of this threshold would be vital to the successful implementation of multi-chip modules and ultra-large integrated circuits.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] J. C. Mudge, "An Illustration of Micropipelines Using Two-Dimensional Fourier Transform Architectures," in *Proceedings of VLSI 89*, pp. 359–368, 1989.

[2] A. J. Martin, S. M. Burns, T. K. Lee, D. Borkovic, and P. J. Hazewindus, "The Design of an Asynchronous Microprocessor," in *Proceedings of the Decennial Caltech Conference on VLSI*, pp. 351–373, 1989.

[3] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989.

[4] L. A. Glasser and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuits*. Reading, MA: Addison-Wesley, 1985.

[5] F. Anceau, "A Synchronous Approach for Clocking VLSI Systems," *IEEE Journal of Solid-State Circuits*, vol. SC-17, no. 1, pp. 51–56, 1982.

[6] D. Jeong, G. Borriello, D. A. Hodges, and R. H. Katz, "Design of PLL-Based Clock Generation Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 2, pp. 255–261, 1987.

[7] M. G. Johnson and E. L. Hudson, "A Variable Delay Line PLL for CPU-Coprocessor Synchronization," *IEEE Journal of Solid-State Circuits*, vol. SC-23, no. 5, pp. 1218–1223, 1988.

[8] S. Y. Sun, "An Analog PLL-Based Clock and Data Recovery Circuit with High Input Jitter Tolerance," *IEEE Journal of Solid-State Circuits*, vol. SC-24, no. 2, pp. 325–330, 1989.

[9] C. Mead and L. Conway, eds., *Introduction to VLSI Sytems*, pp. 218–254. Reading, MA: Addison-Wesley, 1980.

[10] F. U. Rosenberger, C. E. Molnar, T. J. Chaney, and T. P. Fang, "Q-Modules: Internally Clocked Delay-Insensitive Modules," *IEEE Transactions on Computers*, vol. 37, no. 9, pp. 1005–1018, 1988.

[11] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design - A Systems Perspective*. Reading, MA: Addison-Wesley, 1985.

[12] A. Mukherjee, *Introduction to nMOS & CMOS VLSI Design*. Englewood Cliffs, N.J.: Prentice-Hall, 1986.

[13] M. Shoji, "Elimination of Process-Dependent Clock Skew in CMOS VLSI," *IEEE Journal of Solid-State Circuits*, vol. SC-21, no. 5, pp. 875–880, 1986.

[14] M. Hatamian, L. A. Hornak, T. E. Little, S. K. Tewksbury, and P. Franzon, "Fundamental Interconnection Issues," *AT& T Technical Journal*, vol. 66, pp. 13–30, July/August 1987.

[15] M. S. Nakhla, "Analysis of Pulse Propagation on High-Speed VLSI Chips," in *Proceedings of IEEE 1989 Custom Integrated Circuits Conference*, pp. 13.5.1–13.5.4, 1989.

[16] O. Akcasu, H. Hingarh, S. Martin, and R. A. Kertis, "Modeling of Delay and Crosstalk in Interconnects," in *ISSCC 88 Digest of Technical Papers*, pp. 80–81, 1988.

[17] H. R. Kaupp, "Waveform Degradation in VLSI Interconnections," *IEEE Journal of Solid-State Circuits*, vol. SC-24, no. 4, pp. 1150–1153, 1989.

[18] J. Dykstra and R. B. Brown, "A Comparison of Gold and Superconductors Used as Air-Bridge and Microstrip Interconnects for High-Speed VLSI," *IEEE Journal of Solid-State Circuits*, vol. SC-24, no. 3, pp. 842–844, 1989.

[19] D. J. Pedder, "Interconnection and Packaging of Solid-State Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-24, no. 3, pp. 698–703, 1989.

[20] H. B. Bakoglu, ed., *Circuits, Interconnections, and Packaging for VLSI.* Reading, MA: Addison-Wesley, 1990.

[21] H. B. Bakoglu and J. D. Meindl, "Optimal Interconnection Circuits for VLSI," *IEEE Transactions on Electron Devices*, vol. ED-32, no. 5, pp. 903–909, 1985.

[22] R. J. Antinone and G. W. Brown, "The Modeling of Resistive Interconnects for Integrated Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-18, no. 2, pp. 200–203, 1983.

[23] J. L. Wyatt, ed., *Signal Propagation Delay in RC Models for Interconnect*, pp. 254–291. Amsterdam: North-Holland, 1987.

[24] D. Chengson, C. Frazao, H. W. Wang, and B. Billett, "Signal Delay in Distributed RC Tree Networks," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 2835–2837, 1988.

[25] A. Deng and Y. Shiau, "Generic Linear RC Delay Modeling for Digital CMOS Circuits," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 4, pp. 367–376, 1990.

[26] C. B. Reynolds, "Analysis and Guidelines for High-Speed VLSI System Interconnections," in *Proceedings of IEEE 1988 Custom Integrated Circuits Conference*, pp. 23.5.1–23.5.4, 1988.

[27] L. T. Pillage and R. A. Rohrer, "Delay Evaluation with Lumped Linear RLC Interconnect Circuit Models," in *Proceedings of the Decennial Caltech Conference on VLSI*, pp. 143–158, 1989.

[28] R. Plitschka, "Transmission Line Effects in Testing High-Speed Devices with a High- Performance Test System," *Hewlett-Packard Journal*, vol. 40, pp. 58–67, December 1989.

[29] J. McDonald. Rensselaer Polytechnic Institute, private communication (november 1989).

[30] H. J. M. Vandrick, "Short-Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 4, pp. 468–473, 1984.

[31] H. C. Lin and L. W. Linholm, "An Optimized Output Stage for MOS Integrated Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-10, no. 2, pp. 106–109, 1975.

[32] C. M. Lee and H. Soukup, "An Algorithm for CMOS Timing and Area Optimization," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 5, pp. 781–787, 1984.

[33] H. U. Chou and M. A. Franklin, "Optical Distribution of Clock Signals in Wafer Scale Digital Circuits," in *Proceedings of ICCD 87*, pp. 117–122, 1987.

[34] B. C. Clymer and J. W. Goodman, "Timing Uncertainty for Recievers in Optical Clock Distribution for VLSI," *Optical Engineering*, vol. 27, no. 11, pp. 944–954, 1988.

[35] E. G. Friedman and S. Powell, "Design and Analysis of a Hierarchical Clock Distribution System for Synchronous Standard Cell/Macrocell VLSI," *IEEE Journal of Solid-State Circuits*, vol. SC-21, no. 2, pp. 240–246, 1986.

[36] E. Friedman, "A Partitionable Clock Distribution System for Sequential VLSI Circuits," in *Proceedings of the 1986 IEEE International Symposium on Circuits and Systems*, pp. 743–746, 1986.

[37] H. B. Bakoglu, J. T. Walker, and J. D. Meindl, "A Symmetric Clock-Distribution Tree and Optimized High-Speed Interconnections for Reduced Clock Skew in ULSI and WSI Circuits," in *Proceedings of ICCD 86*, pp. 118–122, 1986.

[38] D. Mijuskovic, "Clock Distribution in Application Specific Integrated Circuits," *Microelectronics Journal*, vol. 18, no. 4, pp. 15–27, 1987.

[39] S. Boon, S. Butler, R. Byrne, B. Setering, M. Casalanda, and A. Scherf, "High Performance Clock Distribution for CMOS ASICs," in *Proceedings of the IEEE 1989 Custom Integrated Circuits Conference*, pp. 15.4.1–15.4.5, 1989.

[40] D. T. Cox, D. L. Guertin, C. L. Johnson, B. G. Rudolph, R. R. Williams, R. A. Piro, and D. W. Stout, "VLSI Performance Compensation for Off-Chip Drivers and Clock Generation," in *Proceedings of the IEEE 1989 Custom Integrated Circuits Conference*, pp. 14.3.1–14.3.4, 1989.

[41] J. F. McDonald, E. H. Rogers, K. Rose, and A. J. Steckl, "The Trials of Wafer-Scale Integration," *IEEE Spectrum*, pp. 32–39, october 1984.

[42] R. R. Johnson, "Multichip modules: next-generation packages," *IEEE Spectrum*, vol. 27, pp. 34–48, March 1990.

[43] J. L. Mott, A. Kandel, and T. P. Baker, *Discrete Mathematics for Computer Scientists and Mathematicians*. Englwood Cliffs, NJ: Prentice-Hall, 1986.

[44] R. Woudsma and J. M. Noteboom, "The Modular Design of Clock Generator Circuits in a CMOS Building-Block System," *IEEE Journal of Solid-State Circuits*, vol. SC-20, no. 3, pp. 770–774, 1985.

[45] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York, NY: John-Wiley, 1973.

[46] P. A. Jensen and J. W. Barnes, *Network Flow Programming*. New York, NY: John-Wiley, 1980.

[47] K. G. Murty, ed., *Network Programming*. Englwood Cliffs, NJ: Prentice-Hall, in press.

[48] M. A. Horowitz, "Timing Models for MOS Circuits," Tech. Rep. SEL83-003, Stanford University, Department of Electrical Engineering, Stanford, CA 94305, 1983.

[49] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wide-Band Amplifiers," *Journal of Applied Physics*, vol. 19, no. 1, pp. 55–63, 1948.

[50] MOS Implementation Service. MOSIS parameter test results, run=M83I, vendor=Hewlett-Packard, feature size=3.0micron (september 1988).

[51] Mississipi State University. Standard Cell Library data sheet (november 1986).

APPENDIX

Program Code

```
C*************************************************************
C*                      CLUSTER.FOR                          **
C*                                                           **
C* This routine performs the clustering of nodes based       **
C* on agglomerative clustering scheme. The number of         **
C* desired clusters can be input as a variable. The          **
C* distance measure is the maximum distance based on         **
C* delay cost in an interconnect.                            **
C*                                                           **
C* Programmer: P.R. Mukund                                   **
C* Last update: May 6, 1990                                  **
C*                                                           **
C*************************************************************
C
C Declarations
C
integer numnode,maxbuf,numbclus
integer cluster(250,250),numemb(250),valclus(250)
integer numbnode,numbcnode,temp(250),ibuf(8)
real*8 node(250,3),cost(250,250)
```

```
real*8 rint,cint,dist(250,250),res(250,250)
        real*8 maxcost,mincost,cap(250,250)
C
C Initialization
C
type *,'Number of nodes ?'
accept *,numnode
type *,'Number of potential buffer sites ?'
accept *,maxbuf
type *,'enter the interconnect resistance/sq (in Ohms)'
accept *,rint
type *,'enter the interconnect capacitance/sq (in fF)'
accept *,cint
open(unit=20,file='node.dat',status='old')
read (20,*)((node(i,j),j=1,3),i=1,numnode)
call time(ibuf)
do i=1,8
type *,ibuf(i)
end do
numbclus=numnode
do 101 i=1,numnode
   numemb(i) = 1
```

```
   valclus(i) = 1
   cluster(i,1) = i
101 continue
C
C Map from distance space to delay space
C
do 301 i=1,numnode
   do 201 j=1,numnode
      dist(i,j) = (dabs(node(i,1)-node(j,1))+
     1                    dabs(node(i,2)-node(j,2)))
      res(i,j) = dist(i,j)*rint
      cap(i,j) = dist(i,j)*cint
      cost(i,j) = res(i,j)*(node(j,3)+(0.5*cap(i,j)))
201    continue
301 continue

C
C Start of the loop. Merge clusters until it is the same
C as the number of potential buffer sites.
C
do while (numbclus.GT.maxbuf)
   mincost = 100000000.0
   do 701 i=1,numnode
```

```
      do 601 j=1,numnode
         if ((valclus(i).EQ.1.AND.valclus(j).EQ.1)
     1            .AND.(i.NE.j)) then
            maxcost = 0.0
    do 501 k=1,numemb(i)
       do 401 l=1,numemb(j)
  if (cost(cluster(i,k),cluster
     1                    (j,l)).GT.maxcost) then
     maxcost=cost(cluster(i,k),cluster(j,l))
  endif
401        continue
501     continue
    if (maxcost.LT.mincost) then
       mincost=maxcost
                       mini=i
                       minj=j
                    end if
                 end if
601       continue
701    continue
   i=mini
   j=minj
```

```
   valclus(j)=0
   do 801 k=numemb(i)+1,numemb(i)+numemb(j)
      cluster(i,k) = cluster(j,k-numemb(i))
801    continue
   numemb(i) = numemb(i) + numemb(j)
   numbclus = numbclus - 1
end do
C
C Output the results of clustering
C
open(unit=22,file='cluster.out',status='new')
write (22,1)numbclus
jc=1
do 1000 ic=1,numbclus
900    if (valclus(jc).EQ.1) then
       write (22,2)ic
       write (22,3)(cluster(jc,l),l=1,numemb(jc))
       jc = jc + 1
   else
       jc = jc + 1
       go to 900
   end if
```

```
1000 continue
call time(ibuf)
do i=1,8
type *,ibuf(i)
end do
1 format ('The number of clusters is ',i2)
2 format ('The members of the cluster ',i2,' are:')
3 format (8i4)
stop
end

C*********************************************************
C*                                                      **
C*                      SPATH                           **
C*                                                      **
C* This routine finds the shortest path from the        **
C* origin, designated as node 1, to every other node    **
C* in the graph. The cost criteria can be other than    **
C* the Euclidian distance and is to be provided as      **
C* one of the input to the system. The program runs     **
C* on VAX 11/785. The algorithm is generally credited   **
C* to Dijkstra.                                         **
C*                                                      **
```

```
C* Programmer: P.R. Mukund                                    **
C* Last update: May 17, '90.                                  **
C*                                                            **
C************************************************************
C
C declaration
C
integer arc(4950,2),cost(100,100)
integer permlabel(100,3),templabel(100,3)
integer unlabel(100),numbnode
real*8 node(100,3)
real*8 rint,cint,dist(100,100)
        real*8 cap(100,100),res(100,100),lambda
C
C Initialization
C
type *,'Number of nodes ?'
accept *,numbnode
type *,'enter the interconnect resistance/sq (in Ohms)'
accept *,rint
type *,'enter the interconnect capacitance/sq (in fF)'
accept *,cint
```

```
open(unit=20,file='spath.dat',status='old')
read (20,*)((node(i,j),j=1,3),i=1,numbnode)
C
C Map from distance space to delay space
C
do 301 i=1,numbnode
   do 201 j=1,numbnode
      dist(i,j) = (dabs(node(i,1)-node(j,1))+
     1                    dabs(node(i,2)-node(j,2)))
      res(i,j) = dist(i,j)*rint
      cap(i,j) = dist(i,j)*cint
      cost(i,j) = nint(res(i,j)*(node(j,3)+(0.5*cap(i,j))))
201    continue
301 continue
C
C Remove all arcs whose length is greater than lambda
C
numbarc = 0
do 21 i=1,numbnode
   do 22 j=1,numbnode
      if (i.lt.j) then
         numbarc = numbarc + 1
```

```
 arc(numbarc,1) = i
 arc(numbarc,2) = j
      end if
22    continue
21 continue

C
do 100 i=1,numbnode
   unlabel(i) = i
100 continue
do 150 i=1,numbnode
   templabel(i,1) = 0
150 continue
C
C Make the first node at origin as the source and make
C it a permanent node.
C
unlabel(1) = 0
permlabel(1,1) = 1
permlabel(1,2) = 0
permlabel(1,3) = 0
numbul = numbnode - 1
numbtl = 0
```

```
numbpl = 1
C
C For each j in unlabel, such that (1,j) is in arc, label j
C with (1,cost(1,j)) and move it to templabel.
C
do 300 i=2,numbul+1
   k = unlabel(i)
   do 200 j=1,numbarc
      if (arc(j,1).EQ.1.AND.arc(j,2).EQ.k) then
         numbtl = numbtl + 1
         l = numbtl
 templabel(l,1) = k
 templabel(l,2) = 1
 templabel(l,3) = cost(1,k)
 unlabel(i) = 0
      end if
200    continue
300 continue
C
C while unlabel is not empty, find an i in templabel such
C that the distance index of i is the least among the nodes
C in templabel at this stage. Make i the selected node and
```

```
C move i from templabel permlabel.
C For each j in templabel, let d(j) be the distance in the
C present label of j. If (i,j) is in arc and d(j) is greater
C than d(i)+cost(i,j) change the label on j to
(i,d(i)+cost(i,j)).
C For each j in unlabel such that (i,j) is in arc, move j to
C templabel and label with (i,d(i)+cost(i,j).
C
do while (numbtl.NE.0)
   mindist = 10000000
   do 400 i=1,numbnode
      if (templabel(i,1).NE.0) then
 if (templabel(i,3).LT.mindist) then
    mindist = templabel(i,3)
    mini = i
         end if
      end if
400    continue
   numbtl = numbtl - 1
   numbpl = numbpl + 1
   permlabel(numbpl,1) = templabel(mini,1)
   permlabel(numbpl,2) = templabel(mini,2)
```

```
   permlabel(numbpl,3) = mindist
   templabel(mini,1) = 0
   i = permlabel(numbpl,1)
   do 600 k=1,numbnode
      j = templabel(k,1)
      if (j.NE.0) then
         do 500 l=1,numbarc
    if (arc(l,1).EQ.i.AND.arc(l,2).EQ.j) then
       if (templabel(k,3).GT.(mindist+cost
     1                                          (i,j))) then
  templabel(k,2) = i
  templabel(k,3) = (mindist + cost(i,j))
       end if
    end if
500          continue
      end if
600    continue
   do 800 k=1,numbnode
      j = unlabel(k)
      if (j.NE.0) then
         do 700 l=1,numbarc
    if (arc(l,1).EQ.i.AND.arc(l,2).EQ.j) then
```

```
        m = 1
        do while (templabel(m,1).NE.0)
  m = m+1
        end do
        templabel(m,1) = j
        templabel(m,2) = i
        templabel(m,3) = mindist + cost(i,j)
        unlabel(k) = 0
        numbul = numbul - 1
        numbtl = numbtl + 1
             end if
700           continue
      end if
800    continue
end do
open (unit=21,file='spath.out',status='new')
write (21,1)((permlabel(i,j),j=1,3),i=1,numbnode)
1 format (3i8)
stop
end

c*************************************************************
c*                                                          **
```

```
c*                              DELAY                                  **
c*                                                                     **
c* This routine determines the 0-50 % delay in a R-C mesh.             **
c*                                                                     **
c* Programmer: P.R. Mukund                                             **
c* Last revision: May 25, 1990                                         **
c*                                                                     **
c*********************************************************************
c
c intialization
c
integer path(100),spath(100,3),tree(100,100)
integer numbnode,numbcnode,numbtree(100),temp(100)
integer maxdist,thisnode,lastnode,pointer,tempnode
integer nextnode,selectnode,stack(500),tnumbtree(100)
real intres,intcap,bufres,node(100,3),res(100,100)
        real delay(100),cap(100,100),totcap
c
c input data
c
type *,'enter the number of nodes'
accept *,numbnode
```

```
type *,'enter interconnect resistance / square in  Ohms'
accept *,intres
type *,'enter interconnect capacitance / square in fF'
accept *,intcap
type *,'enter buffer output resistance in  Ohms'
accept *,bufres
open(20,file='delay.dat',status='old')
read (20,*)((spath(i,j),j=1,3),i=1,numbnode)
read (20,*)((node(i,j),j=1,3),i=1,numbnode)
c
c initialize arrays
c
do 200 i=1,numbnode
   numbtree(i) = 0
   do 100 j=1,numbnode
      tree(i,j) = 0
100    continue
200 continue
c
c generate the tree
c
do 300 i=1,numbnode
```

```
   k = spath(i,1)
   l = spath(i,2)
   numbtree(l) = numbtree(l) + 1
   m = numbtree(l)
   tree(l,m) = k
300 continue
c
c generate critical path
c
maxdist = 0
do 400 i=1,numbnode
   if (spath(i,3).GT.maxdist) then
      maxdist = spath(i,3)
      maxi = i
   end if
400 continue
numbcnode = 0
pointer = 1
thisnode = spath(maxi,1)
do while (thisnode.NE.1)
   do 500 i=1,numbnode
      if (spath(i,1).EQ.thisnode) then
```

```
         lastnode = spath(i,2)
      end if
500    continue
   temp(pointer) = thisnode
   pointer = pointer + 1
   thisnode = lastnode
   numbcnode = numbcnode + 1
end do
path(1) = 1
do 600 i=2,numbcnode+1
   path(i) = temp(pointer-1)
   pointer = pointer - 1
600 continue
c
c calculate distances between nodes and
c resistances and capacitances in the
c interconnects
c
do 800 i=1,numbnode
   do 700 j=1,numbnode
      dist = abs(node(i,1)-node(j,1))
     1               + abs(node(i,2)-node(j,2))
```

```
      res(i,j) = dist * intres
      cap(i,j) = dist * intcap
700    continue
800 continue
c
c calculate delays between each pair of nodes
c in the critical path
c
do 1500 i=1,numbcnode+1
   if (i.EQ.1) then
      totcap = 0.0
      do 900 j=1,numbnode
 totcap = totcap + node(j,3)
900       continue
      do 1100 k=2,numbnode
         totcap = totcap + cap(spath(k,1),spath(k,2))
1100       continue
      delay(i) = (bufres*totcap)
   else
      thisnode = path(i-1)
      nextnode = path(i)
      do 1200 j=1,numbnode
```

```
         tnumbtree(j) = numbtree(j)
1200       continue
      pointer = 0
      selectnode = nextnode
      tempnode = thisnode
      totcap = 0.0
      k=1
1250       if (tnumbtree(selectnode).NE.0) then
         tempnode = selectnode
 if (k.eq.1) then
            pointer = pointer + 1
            stack(pointer) = tempnode
 end if
                  do 1300 j=1,tnumbtree(selectnode)
    pointer = pointer+1
    stack(pointer) = tree(selectnode,j)
type *,i,stack(i)
1300          continue
         selectnode = stack(pointer)
 k=k+1
         go to 1250
      else
```

```
          totcap = totcap + node(selectnode,3)
     1                    + cap(selectnode,tempnode)
          if (tempnode.NE.thisnode) then
     tnumbtree(tempnode) = tnumbtree(tempnode) - 1
     pointer = pointer - 1
     selectnode = stack(pointer)
             if (tnumbtree(tempnode).EQ.0) then
        do 1400 n=1,numbnode
   if (spath(n,1).EQ.tempnode) then
      tempnode = spath(n,2)
   end if
1400         continue
     end if
     k = 1
     go to 1250
          end if
       end if
       totcap = totcap - (0.5*cap(selectnode,tempnode))
       delay(i) = res(selectnode,tempnode) * totcap
    end if
1500 continue
c
```

```
c compute total delay
c
totaldelay = 0
do 1600 i=1,numbcnode+1
   totaldelay = totaldelay + delay(i)
1600 continue
totaldelay = totaldelay/1000.
type *,'totaldelay=',totaldelay,'pico seconds'
stop
end
```

VITA

P. R. Mukund was born in Bangalore, India on February 4^{th}, 1952.

He obtained an A.E. degree in Electronic Engineering Technology from the Nashville State Technical Institute in 1976. He graduated from the University of Tennessee, Knoxville with a B.S.E.E. in 1980, an M.S. in 1981, and a Ph.D. in 1990, all in Electrical Engineering.

During 1976-1977, he was an electronic technician at Teledyne Lewisburg. During 1978-1980, he was an electronic technician at the University of Tennessee. During 1980-1981, he was a GTA and a GRA at the University of Tennessee. He joined International Instruments Ltd., Bangalore, India as a Research Engineer in 1981, and was promoted to Senior Engineer in 1982, and to Dy. Manager in 1985. Since 1986, he has been an Instructor, a GRA and a GTA, at different intervals at the University of Tennessee.

Mukund is a member of Tau Beta Pi, Eta Kappa Nu, IEEE and ACM.