

Automated calculation of device sizes for digital IC designs

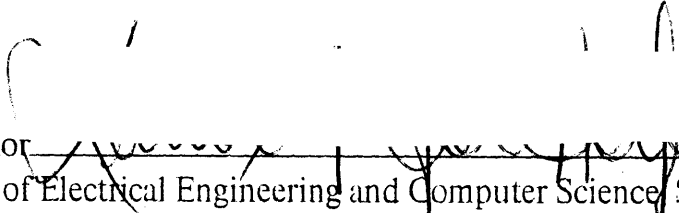
by

Lennox P. John Hoyte

Submitted in partial fulfillment of the requirements  
For the degree of Master of Science  
at the  
Massachusetts Institute of Technology  
September 1982

© Lennox P. John Hoyte

The author hereby grants to M.I.T. permission to reproduce and to  
distribute copies of this thesis document in whole or in part

Signature of Author   
Department of Electrical Engineering and Computer Science, Sept. 3, 1982.

Certified by    
Thesis Supervisor

Accepted by   
Chairman, Departmental Committee on Graduate Students

Archives

MASSACHUSETTS INSTITUTE  
OF TECHNOLOGY

OCT 20 1982

LIBRARIES

# Automated calculation of device sizes for digital IC designs

by

**Lennox P. John Hoyte**

Submitted to the Department of Electrical Engineering on September 3, 1982 in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering and Computer Science.

## **Abstract:**

A speed and power optimizer for a class of digital MOS integrated circuit designs is discussed. Simple models are used to create an optimization algorithm which is low in computational complexity but reasonably accurate. The discussion proceeds from a theoretical treatment of the optimization algorithm to an actual implementation of a software system for circuit optimization. Schemes for mapping the MOS device characteristics into the optimizer's operating parameters are also addressed.

Thesis Supervisor: Lance A. Glasser  
Title: Assistant Professor of Electrical Engineering

## Acknowledgements

The author would like to thank all those who helped with this work. In particular, discussions with Leo Bain proved to be rewarding and encouraging. Mark Matson's well placed critical comments and useful consultations served to enlighten the author, and his help with the early drafts of this paper was highly appreciated. John Wroclawski's thorough understanding of the MIT SPEECH computer system helped to make software development on that system much less tedious than it could have been. The members of the VLSI circuits group at MIT have been of great help in the development of the theory. Professor Paul Penfield's insights into device modeling has been invaluable. Steve McCormick contributed a name for the program.

Finally, deepest thanks and appreciation must go to the thesis advisor, Professor Lance Glasser. This thesis could not have started without him. Indeed without his expert guidance, moral support, and editorial contributions, this work might not have come to a successful completion. Also, his impact on the author's personal and academic development continues to be significant. It has been a pleasure working with him.

## Table of contents

Abstract . . . . .	2
Acknowledgements . . . . .	3
1.0 Introduction . . . . .	6
2.0 Models and Theory . . . . .	11
2.1 Models . . . . .	11
2.2 Theory . . . . .	16
2.2.1 Worst case considerations . . . . .	16
2.2.2 Delay . . . . .	17
2.2.3 Relating power to delay . . . . .	18
2.2.4 Minimization of delay with respect to power . . . . .	20
3.0 Solving the optimization equations . . . . .	28
3.1 The special case . . . . .	28
3.2 The general case . . . . .	31
3.3 Convergence of the solution . . . . .	32
4.0 Parameter Extraction . . . . .	37
5.0 Implementation as a software system . . . . .	42
5.1 Decoupler . . . . .	44
6.0 Results . . . . .	51
6.1 Comparison to theoretical results . . . . .	51
6.2 Comparison to circuit simulation . . . . .	53
6.3 Measured computational complexity . . . . .	53
6.3.1 Runtimes . . . . .	53
6.3.2 Convergence dependence . . . . .	54
7.0 Conclusion . . . . .	55
8.0 References . . . . .	56
Appendix A: Sample program run . . . . .	58

## List of figures

1.1	General input circuit form . . . . .	10
2.1	Zero order $RC$ model . . . . .	24
2.2	$\tau$ as a measure of delay . . . . .	25
2.3	Typical output voltage waveforms . . . . .	26
2.4	Channel resistance versus input and output waveshapes . . . . .	26
2.5	Speed vs Power . . . . .	27
2.6	Delay vs Power . . . . .	27
4.1	Test circuit I . . . . .	41
4.2	Test circuit II . . . . .	41
5.1	Form of process data file . . . . .	46
5.2	Form of circuit data file . . . . .	47
5.3	User block diagram of optimizer program . . . . .	48
5.4	Optimization algorithm . . . . .	49
5.5	List of optimizer parameters . . . . .	50

## 1.0 Introduction

Computer aids are crucial to the design of today's integrated circuits. Computer aided design (CAD) tools are used to check for design rule errors in IC layouts[1], to verify logical circuit behaviour[2], to analyze electrical circuit behaviour[3], and also as sophisticated "bookkeepers" in the actual piecing together of designs themselves[4]-[6]. Lately though, CAD tools have been becoming more and more a part of the IC synthesis effort[7] - [9]. Most of the synthesis tools to date have concentrated on achieving functionality and on the automatic layout problem e.g, placement, routing, compaction and other topological issues, with little emphasis on performance considerations. A notable exception to this is the PLA generator developed by Glasser[10]. This tool optimized the PLA's for speed, power consumption and area.

The importance of circuit performance, cannot be overstated: circuit speed, a prime determinant of computational power, is generally proportional to power consumption, itself a scarce resource. As the circuit complexity and density of a chip increases, the amount of power that may be feasibly dissipated per unit of chip area stays fairly constant. On average, this requires that the power consumption of individual elements on the chip be decreased. A more judicious distribution of power among elements is required in order to sustain circuit speeds (and computational power) in the face of decreasing average element power.

Also, as more synthesis tools become widely available as circuit design aids, we will surely get to the place where people with minor (or no) backgrounds in electrical circuits would be able to design their own integrated circuits. To make the design process amenable to such designers, some attempt must be made to remove the tedium associated with having to calculate those parameters which determine the

performance characteristics of their integrated circuits. By simplifying this dimension of the design procedure, such designers are freed to concentrate on the issues of interest to them. A computer scientist for example, might be able to realise his wildest architectural dreams without having to worry much about the specifics of signal delay in one of the adder circuits of his system.

This thesis discusses the theory and implementation of a CAD tool which attempts to solve a part of the performance problem. Specifically, the tool will optimize the class of combinational circuits generalized in fig(1.1), with respect to a given set of speed and power constraints. In particular, the inputs to the optimizer would be: a circuit specification (with critical path and impedance boundary conditions), a delay requirement, and some information about the process to be used in fabricating the circuit. The system would then compute a set of widths and lengths of devices in the circuit so that it will meet its speed requirement while consuming the minimum possible power.

This tool is useful either to a circuit designer to optimize circuits of his own creation or, as part of a larger CAD tool which might automatically convert high level system specifications into layouts of circuits implementing the high level function. If it were made to operate at a low level of computational complexity, then this tool could be incorporated into the IC design process without significantly lengthening the design cycle. Of course, the computational complexity of the algorithm is proportional to the complexity of the device models used and to the accuracy of the results obtained. A model which trades off 10% to 15% accuracy in exchange for speeds suitable for an interactive system will be an acceptable one. Such an error is not bad considering that the widely used full circuit simulator (which is also very

CPU intensive), has errors in the 5 to 10% range [11].

The urge to optimize electronic circuits is not a new one. On the contrary, it has been the object of scholarly research for some time. The abundance of circuit optimization research in both analog and digital contexts in the late sixties and early seventies lead to the publication of works which attempted to catalog the then known methods of circuit optimization [12] ,[13]. In general though, most of the work was conducted on a theoretical level [14], [15]. There were some exceptions which produced working computer programs for optimizing specific kinds of circuits [16] -[18].

In 1974, the first significant attempt to produce a usable CAD tool for digital circuit optimization was reported [19]. A computer program called OPTISEM was developed to optimize bipolar and MOS circuits for delay or noise margins, with respect to power consumption, circuit area, or any other user definable criteria. OPTISEM had the capability to monitor and correct optimization parameters which violated user defined constraints during the course of the optimization. This ability enabled the optimizer to avoid producing circuits with unsatisfactory noise margins or tendencies to saturate (in the case of bipolar ECL designs). OPTISEM also understood important interdependencies such as the area-capacitance relationship. The major drawback of this tool was its slowness of operation. Because of the complexity of the optimization algorithms, OPTISEM required on the order of one hour of CPU time (on a Siemens 4004/150) to optimize a 100 element circuit. Also, this system was unable to handle more than 350 circuit elements. The combination of high computational complexity and limited capacity made this optimizer unsuitable for the applications envisioned earlier in this section.



Since OPTISEM, work on practical CAD tools for digital circuit optimization has been scarce. Lin and Lindholm [20], reported a technique for optimizing the output stage of a design. Their method involves the introduction of successively larger buffers between the output driver and the load, sizing them so as to decrease total output delay. They define a figure of merit  $F$  which is based on an area-delay product and is a function of the number of intermediate buffers  $m$  between the load and driver. For a given load and driver, the optimal number of buffers is determined by minimizing  $F$  with respect to  $m$ .

Later, Mohsen and Mead [21], developed a method for optimizing the driver and receiver circuits on different ends of a long capacitive line, for minimum delay. Kang [22], worked with CMOS circuits and reported a technique for minimizing the product of chip area and delay, with respect to the widths of the devices in the circuit<sup>1</sup>. None of these efforts mention working software for performing these optimizations.

---

<sup>1</sup>Kang pointed out that since static power dissipation in CMOS designs was very low, area was the limiting scarce resource for CMOS.

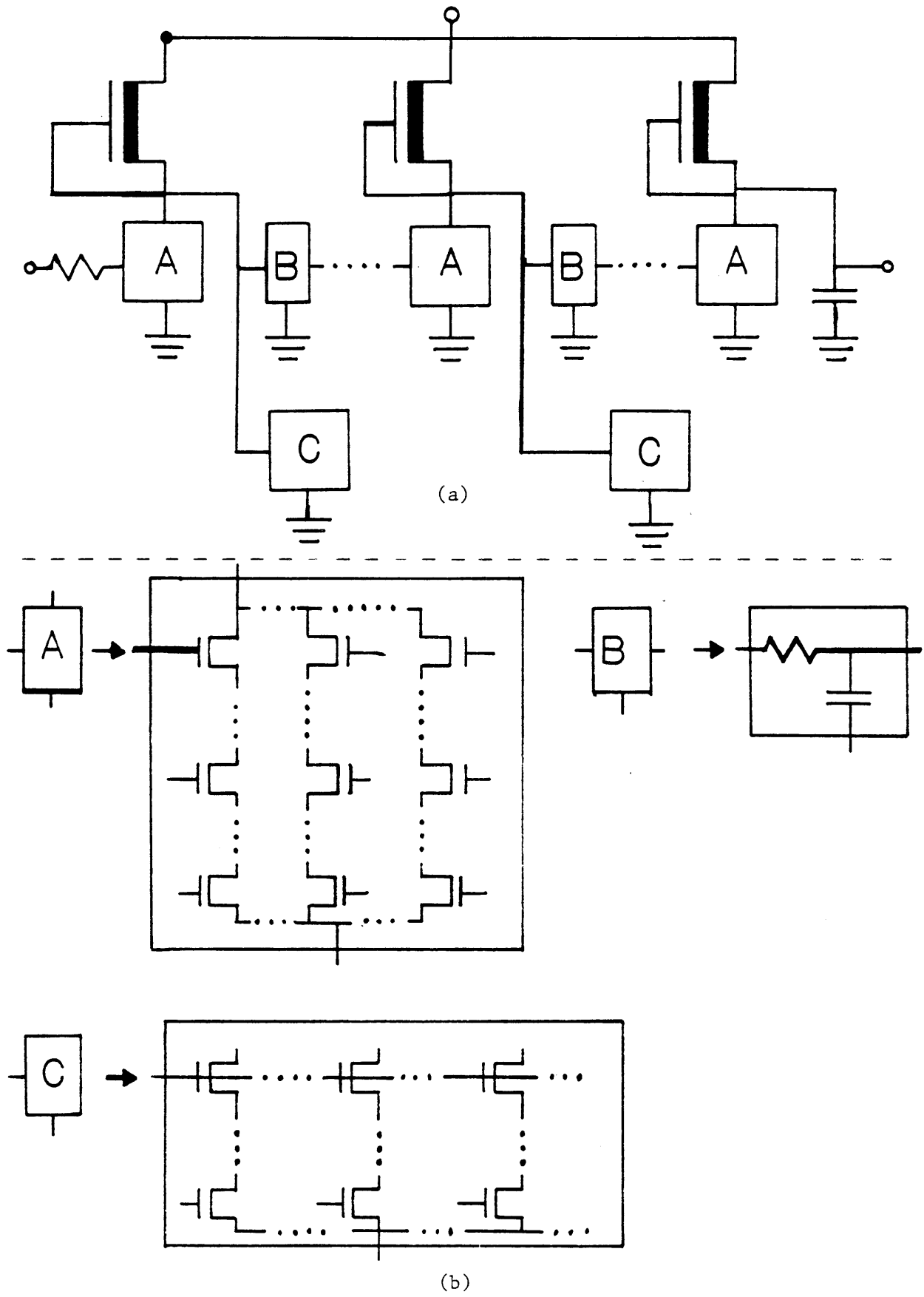


Fig 1.1 General input circuit form: (a)- circuit, (b)- legend.

## 2.0 Models and Theory

In this section, a simplified MOS device model is derived starting from the fundamental device equations. We use the model to present our definition of circuit delay and then proceed to develop an optimization theory based on the relationship between total circuit delay and power consumption in a given circuit element group (stage). On the way, issues of range of validity of the model and worse case considerations are dealt with. Some expectations about the complexity of the optimization algorithm are ventured.

### 2.1 Model

The optimizer is based on the zero order resistance concept which models the MOS transistor with a capacitor  $C_{gate}$  at the input and a switched linear resistor  $R_{ch}$  between the drain and source regions, where the switch is controlled by the voltage at the device's input. This model is illustrated in fig(2.1). We begin our discussion by obtaining the relationship between  $C_{gate}$  and  $R_{ch}$  for any given device. Following this, the model's range of validity is examined. An expression for delay, based on the model, is then presented.

We have,

$$R_{ch} \equiv \int \frac{1}{I_{ds}} dV_{ds}$$

and

$$I_{ds} = Wf(V_{ds})$$

where  $W$  is the width of the device and  $I_{ds}$  and  $V_{ds}$  are the channel current and drain to source voltage respectively of the device. The function  $f(V_{ds})$  is as follows:

$$f(V_{ds}) = \begin{cases} 0, & \text{if } V_{gs} < V_{th}; \\ \left(\frac{\mu\epsilon}{LT_{ox}}\right)(V_{gs} - V_{th})V_{ds}, & \text{if } (V_{gs} - V_{th}) > V_{ds}; \\ \left(\frac{\mu\epsilon}{LT_{ox}}\right)(V_{gs} - V_{th})^2, & \text{if } 0 < (V_{gs} - V_{th}) < V_{ds}; \end{cases}$$

where  $V_{gs}$  is the device input (gate) voltage,  $V_{th}$  is its threshold voltage and  $L$  is its channel length. The parameters  $T_{ox}$  and  $\epsilon$  are the thickness and dielectric constant respectively of the gate oxide used in the fabrication of the device and  $\mu$  is the mobility of the carriers in the channel of the device. For each fabrication process,  $T_{ox}$ ,  $\epsilon$  and  $\mu$  are taken to be constant.

$C_{gate}$  is given by

$$C_{gate} = Wg$$

where

$$g = \frac{\mu\epsilon L}{T_{ox}}$$

Taking the product of  $C_{gate}$  and  $R_{ch}$  we obtain

$$R_{ch}C_{gate} = C_{gate} \int \frac{1}{I_{ds}} dV_{ds} = g \int \frac{1}{f(V_{ds})} dV_{ds}.$$

If  $V_{gs}$  is a step function, then for a given  $L$  and fabrication process, the function

$$g \int \frac{1}{f(V_{ds})} dV_{ds}$$

becomes a constant (with respect to time,  $V_{gs}$  and  $V_{ds}$ ) which is independent of  $W$ . We define this constant as  $\tau$ . Specifically,

$$R_{ch}C_{gate} = \tau \tag{2.1}$$

$\tau$  may be viewed as a measure of delay i.e, the time taken to discharge (or charge) the gate of a given device, through the channel of a similar sized device as illustrated in fig(2.2) [23].

To examine the range of validity of our model, we must study those parameters which affect  $\tau$ . Since this model is valid only for the case where  $V_{gs}$  is a step function, it can be expected to change when  $V_{gs}$  is less than ideal. Furthermore, because perfect step functions rarely occur, there is a need to extend the model to cover a more practical set of input waveshapes. Fortunately, the dependence of  $\tau$  on the shape of the input waveform is readily determined for any given fabrication process. This means that the relationship between  $C_{gate}$  and  $R_{ch}$  may be found over any range of input waveshapes. In fact, since  $R_{ch}$  is the only term in (2.1) which is dependent on  $V_{gs}$ , the input shape dependence of  $\tau$  may be specified by the input shape dependence of  $R_{ch}$ <sup>1</sup>. Before presenting a method for determining this dependence, a metric for signal waveshape must be given. Also, a useful definition of delay will be needed.

We define the slope of a waveform in terms of the time taken by the signal to travel between 10 and 90% of its total path. Specifically the slope ( $S$ ) is

$$S \equiv \left( \frac{V(90\%) - V(10\%)}{t(90\%) - t(10\%)} \right). \quad (2.2)$$

Using this definition, the input shape dependence of  $\tau$  may be written concisely as  $\tau(S_{in})$ . Similarly  $R_{ch}(S_{in})$  may be written for the input shape dependence of  $R_{ch}$ . Our definition of delay will be facilitated by the graph of a typical set of voltage waveforms for a circuit with  $N$  inverters. Such a graph is presented in fig(2.3). The delay through a gate is defined as the time interval beginning when the gate's input signal crosses a specified reference voltage and ending when the gate's output signal crosses the reference. In fig(2.3), gate delays are marked as  $t_0, t_1, \dots, t_N$ . For this work, we have picked the reference as  $V_{inv}$  or the point at which, under D.C conditions, the gate's output voltage equals that of its input<sup>2</sup>. The implications of this definition are

<sup>1</sup>There are some nonlinearities in  $C_{gate}$ , but these are minor compared to those in  $R_{ch}$ . We neglect these in our computations.

<sup>2</sup>Our choice of voltage reference is substantiated in [24].

- 1) for a set of cascaded gates, total delay is the sum of the individual stage delays and
- 2) at DC,  $V_{out}$  crosses  $V_{inv}$  at the same time as  $V_{in}$  does and so for a D.C input, stage delay is zero.

$R_{ch}(S_{in})$  is determined by using a circuit simulator (SPICE [3]) to apply different input slopes to an inverter with a fixed load capacitor. For each  $S_{in}$  the gate delay is measured and equated to  $\tau$ .  $R_{ch}(S_{in})$  is computed by

$$R_{ch} = \frac{\tau}{C_{load}}$$

This experiment is repeated for rising and falling inputs to determine  $R_{ch}(S_{in})$  for the pulldown and pullup devices respectively. Furthermore, to find  $R_{ch}(S_{out})$  (that is, the dependence of  $R_{ch}$  on the output waveshape), the process is repeated for varying values of output signal slopes ( $S_{out}$  is varied by changing  $C_{load}$ ).

$R_{ch}$  was found to be primarily dependent upon the ratio ( $\frac{S_{in}}{S_{out}}$ ). This result is not surprising considering that all of the time dependent terms in the output of the gate's dynamic equations come from the input and the  $C \frac{d}{dt}$  terms. If we assume that  $C_{load}$  dominates over the internal gate capacitances (as is generally the case), then scaling the time and  $C_{load}$  parameters by the same amount (as is done by scaling  $S_{in}$  and  $S_{out}$ ) does not change the dynamic equations. This implies that  $R_{ch}$  would remain unchanged.

Figure (2.4) presents the measured  $R_{ch}(\frac{S_{in}}{S_{out}})$ .  $R_{ch}$  goes to zero as  $S_{in}$  approaches zero (i.e DC input) because delay was defined to be zero at DC. The effective pullup resistance is greater than that of the pulldown because of the 4:1 pullup/pulldown ratio used in the test circuit. In general, it is expected that, for a downgoing input transient (active pullup) the stage is being driven by a pulldown device and is driving

a stage of roughly the same size. In such a case one would expect the input signal to be faster than the signal at the output i.e,  $S_{in} > S_{out}$ , because pulldowns are generally smaller than pullups. By a similar argument, the upgoing input signal can be expected to be slower than the output ( $S_{in} < S_{out}$ ). These expected regions of operation are shaded in fig(2.4) and show that the pullup  $R_{ch}$  is fairly stable in its normal region of operation while the pulldown  $R_{ch}$  may vary significantly over its expected region of operation. This instability in the pulldown is normally overwhelmed by the stable pullup device, which is generally 4 to 20 times as large as the pulldown. Since the signal flow direction alternates as the signal propagates through the circuit, any resistance contributions will be made by a more or less equal number of pullups as pulldowns. Because of the large pullup/pulldown ratios, the total resistance will be dominated by the larger pullups thus minimizing the total error.

Using our model, the expression for delay through a single stage may be written

$$t_i = (R_i + Z_i)(C_{i+1} + X_i) \quad (2.3)$$

Here  $R_i$  is the channel resistance of the active device(s) in the chain i.e, pullup or pulldown, and  $C_i$  is the gate capacitance at the input to stage  $i$ .  $Z_i$  and  $X_i$  are the parasitic resistance and capacitance respectively in stage  $i$ . For a set of  $N$  cascaded gates, the individual stage delays may be summed to arrive at the total circuit delay:

$$T_{total} = \sum_i t_i = \sum_i (R_i + Z_i)(C_{i+1} + X_i) \quad (2.4)$$

for

$$0 \leq i \leq N$$

## 2.2 Theory

We begin our discussion of the optimizer theory by showing the importance of worst case analysis. The discussion continues by applying the zero order resistance MOS model to obtain an expression for circuit delay as a function of one parameter. The relationship between total worst case delay and the worst case power consumption of the individual stages is then developed, following which we apply conventional minimization methods to obtain a set of solution equations.

### 2.2.1 Worst Case Considerations

Even though the process of integrated circuit fabrication is a refined and meticulous one, many device parameters are subject to variations which come about due to deviations in process control. After fabrication, the devices are subject to operating conditions which may also vary. The extremes of these variations change the properties of the resulting circuits in predictable ways.

In particular, one group of extremes might cause the circuit's power consumption to be higher than nominal, while the other extreme might cause the circuit to run more slowly than expected. The undesirable extremes e.g, high power and low speed are called worst case conditions and represent the "worst" that the circuit could do and still be considered acceptable.

It is important that the optimizer produce worst case results because the circuit designer must be assured that his optimized circuit will not do worse than stated. For example, adverse deviations in circuit speed can create races which might render a circuit useless, and increases in stated power might be enough to burn out a power bus or overheat a chip. For this reason, worst case speed is optimized with respect



to worst case power. This is done by using worst case speed parameters when performing speed calculations, and worst case power parameters to perform power calculations. In this way, the optimizer is guaranteed to produce worst case results.

### 2.2.2 Delay

Using (2.1), the equation for delay through a chain of gates (2.4) may be rewritten

$$T = \left(\frac{\tau n_0}{C_0} + Z_1\right)(C_1 + X_0) + \dots + \left(\frac{\tau n_i}{C_i} + Z_i\right)(C_{i+1} + X_i) + \dots + \left(\frac{\tau n_N}{C_N} + Z_N\right)(C_{load} + X_N)$$

or,

$$T = \sum_{i=0}^N \left(\frac{\tau n_i}{C_i} + Z_i\right)(C_{i+1} + X_i) \quad (2.5)$$

where

$$C_{N+1} \equiv C_{load}$$

As before

$$t_i = \left(\frac{\tau n_i}{C_i} + Z_i\right)(C_{i+1} + X_i) \quad (2.6)$$

For notational convenience, the input (source) resistance is defined as

$$R_{in} \equiv \frac{\tau}{C_0}$$

The parameter  $n_i$  in (2.5), (2.6) is used to account for the cases where  $R_i$  might be different from  $\frac{\tau}{C_i}$ . In a NAND gate for example, the total pulldown resistance is the sum of all pulldown resistances in the structure whereas the total NOR gate pulldown resistance is the parallel combination of all pulldowns in the stage. The total pulldown resistance of a stage then, may be written

$$R(\text{pulldown})_i = \frac{\tau a_i}{C_i}$$

where  $\alpha_i$  is that fraction of  $\frac{\tau}{C_i}$  which is the total pulldown resistance of that stage. For a 2-input NAND gate  $\alpha_i = 2$ . The case of  $\alpha_i = 1/2$  specifies a 2-input NOR gate (that is, 2 “turned on” pulldown devices in parallel<sup>1</sup>). An inverter would have an  $\alpha_i$  of 1. In addition, the direction of the signal at the input to the stage may be such that the pullup device is active. In that case

$$R(\text{pullup})_i = \beta_i R(\text{pulldown})_i = \frac{\tau \alpha_i \beta_i}{C_i}$$

where  $\beta_i$  is the total pullup/pulldown resistance ratio of that stage. Thus for an upgoing input transient (pulldown active),

$$n_i = \alpha_i$$

and for a downgoing input (pullup active),

$$n_i = \alpha_i \beta_i$$

### 2.2.3 The delay – power relationship

A typical relationship between total worst case delay and worst case power used in the  $i^{\text{th}}$  stage of a chain of gates is given by fig(2.5). This function is readily understood if the relationship between worst case stage power ( $P_i$ ) and stage input capacitance ( $C_i$ ) is made plain. Worst case power is defined as

$$P_i = \frac{V_{dd}^2}{R_{power}} \quad (2.7)$$

where  $V_{dd}$  is the worst case supply voltage and  $R_{power}$  is the total “on” resistance of stage  $i$  measured under worst case power conditions. We have

$$R_{power} = R_{power}(\text{pullup}) + R_{power}(\text{pulldown})$$

<sup>1</sup> In fact, worst case delay for a NOR stage occurs only if 1 of the NOR-pulldowns are activated (i.e., for  $\alpha_i = 1$ ). That way, pulldown resistance is its largest and thus  $t_i$  takes on its largest possible value.

where  $R_{power}(pullup, pulldown)$  can be written in terms of the worst case speed resistance ( $R_{speed}(pullup, pulldown)$ ) for a given device. Specifically

$$\gamma_{pullup} = \frac{R_{power}(pullup)}{R_{speed}(pullup)} \quad (2.7a)$$

and

$$\gamma_{pulldown} = \frac{R_{power}(pulldown)}{R_{speed}(pulldown)} \quad (2.7b)$$

If we take  $R_{speed}$  as the total "on" worst case speed resistance, i.e.,

$$R_{speed} = R_{speed}(pulldown) + R_{speed}(pullup) = \frac{\tau\alpha_i}{C_i} + \frac{\tau\alpha_i\beta_i}{C_i}$$

then

$$R_{power} = \frac{\tau\alpha_i\gamma_{pulldown}}{C_i} + \frac{\tau\alpha_i\beta_i\gamma_{pullup}}{C_i} \quad (2.7c)$$

Using (2.7a) and (2.7c),

$$P_i = \frac{V_{dd}C_i}{\tau\alpha_i(\gamma_{pulldown} + \beta_i\gamma_{pullup})} \quad (2.7d)$$

For a given stage, an increase in power (drive) increases the capacitance at the input to the stage (2.7d). Thus, while increased power speeds up the stage output (for a fixed load), the input to the stage is slowed down (for a fixed driver) because of its increased input capacitance. By a similar argument, a decrease in power (in the same stage) would slow down the stage output, and speed up its input by reducing its input capacitance.

For a change in power then, there are two opposing components (input and output) which determine the effect on total circuit speed. Depending on the relative magnitudes of these components, one or the other might dominate, or they might combine to nullify each other's effects. A given stage may therefore operate in one

of three regions. In the first case, the change in total circuit speed is dominated by the change in speed at the stage output. Thus for an increase in power, total circuit speed increases whereas total circuit speed decreases for a power decrease. This region is represented by the dashed part of fig(2.5). In the second case, the change in total speed is dominated by the change in speed at the stage input. That is, an increase/decrease in stage power decreases/increases total circuit speed. This region is represented by the solid part of the curve in fig(2.5). Finally, in the third case, the change in total speed is zero because the contributions of each side are equal and opposite and cancel each other out. This is marked by the place where the solid and dashed lines meet in fig(2.5). A translation of fig(2.5) into the language of the optimizer is given in fig(2.6) in which delay ( $T$ ) is substituted for circuit speed.

#### 2.2.4 Minimization of delay with respect to power

Further study of fig(2.6) shows the slope of this curve to be a measure of the sensitivity of the total delay to changes in the power of a given stage. Let this slope be:

$$K = \frac{dT}{dP_i} \quad (2.8)$$

Figure(2.6) shows that the case of  $K = 0$  produces the minimum delay possible and is the point where total delay ( $T$ ) is least sensitive to changes in power. Negative values of  $K$  give the minimum power required to meet a given delay. The case of  $K > 0$  produces a given delay at a power consumption that is other than minimum. In this region, power and delay may be simultaneously increased or decreased. Since optimum, in our context, suggests that any change in power will increase delay, the region  $K > 0$  cannot an optimum part of the space.

The optimization is accomplished by solving (2.8) for  $C_i$  in terms of  $K$ ,  $C_{i+1}$  and  $C_{i-1}$ .

$$K = \frac{dT}{dP_i} = \frac{dT}{dC_i} \frac{dC_i}{dP_i} = \left[ \left( \frac{\tau n_{i-1}}{C_{i-1}} + Z_i \right) - \frac{\tau n_i (C_{i+1} + X_i)}{C_i^2} \right] A_i \quad (2.9)$$

which leads to the solution equation

$$C_i = \left\{ \frac{\tau n_i (C_{i+1} + X_i)}{\left( \frac{\tau n_{i-1}}{C_{i-1}} + Z_i \right) - \frac{K}{A_i}} \right\}^{\frac{1}{2}} \quad (2.10)$$

For an  $N$  stage circuit, there are  $N$  curves of the form of fig(2.6) and thus  $N$  equations of the form of (2.10), in  $N$  unknowns.

The factor  $K$  is made equal in all the stages because because this assures the most optimum distribution of power among stages. We demonstrate this by using fig(2.6) to show that, for any pair of stages with unequal  $K$ 's, it is possible to maintain constant total delay while decreasing total circuit power. This is shown to be equivalent to lessening the difference between the two  $K$ 's. We show that when the  $K$ 's become equal, no further decreases in total power (at constant delay) are possible. We then point out that the process may be applied to pairs of stages with unequal  $K$ 's with the result of making  $K$  equal throught the system. Consider a pair of stages  $S_m$  and  $S_n$  such that

$$K_{S_m(t_m)} > K_{S_n(t_n)}$$

i.e,  $S_m$  is more sensitive to changes in delay than  $S_n$ . Let  $\Delta K$  be defined as

$$\Delta K = K_{S_m(t_m)} - K_{S_n(t_n)}$$

where,

$$T(S_m + S_n) = t_m + t_n$$

and

$$T_{total} = \sum_i t_i = T(rest) + T(S_m + S_n)$$

According to fig(2.6), if a given amount of delay ( $T_0$ ) is removed from  $S_n$  and added to  $S_m$  the effect on total circuit power is,

$$\Delta P_{total} = \Delta P[S_n(t_n - T_0)] + \Delta P[S_m(t_m + T_0)] < 0$$

where,

$$T(S_m + S_n) = t_m + t_n$$

and thus,

$$T_{total} = T(S_m + S_n) + T(rest)$$

is left unchanged. The total circuit power now is

$$P_{total} = P_{previous} + \Delta P_{total} < P_{previous}$$

That is, the total power consumption is decreased without changing the total delay.

Also,

$$S_n(t_n - T_0) \Rightarrow K_{S_n(t_n)} + a_1 = K'_{S_n(t_n - T_0)}$$

and,

$$S_m(t_m + T_0) \Rightarrow K_{S_m(t_m)} - a_2 = K'_{S_m(t_m + T_0)}$$

where  $a_1, a_2 > 0$ . Thus

$$\Delta K' = K'_{S_m(t_m + T_0)} - K'_{S_n(t_n - T_0)} = K_{S_m(t_m)} - K_{S_n(t_n)} - (a_1 + a_2) < \Delta K$$

This "trading" of power between stages is responsible for decreasing the total power at constant delay, bringing the  $K$ 's closer together. The trading process may be

repeated until

$$\Delta P\{S_n(t_n - T_0)\} = \Delta P\{S_m(t_m + T_0)\} \quad (2.11)$$

i.e.,

$$\Delta P_{total} = 0$$

Equation (2.11) is satisfied when  $K_{S_m} = K_{S_n}$ . At this point, it ceases to be profitable to trade delay between the two stages.

This trading scheme may be applied to every pair of stages with unequal  $K$ 's, reducing the total power at constant total delay and ultimately, making  $K$  the same, through the system. To use a constant value for  $K$  in equations (2.10) is to anticipate this development.

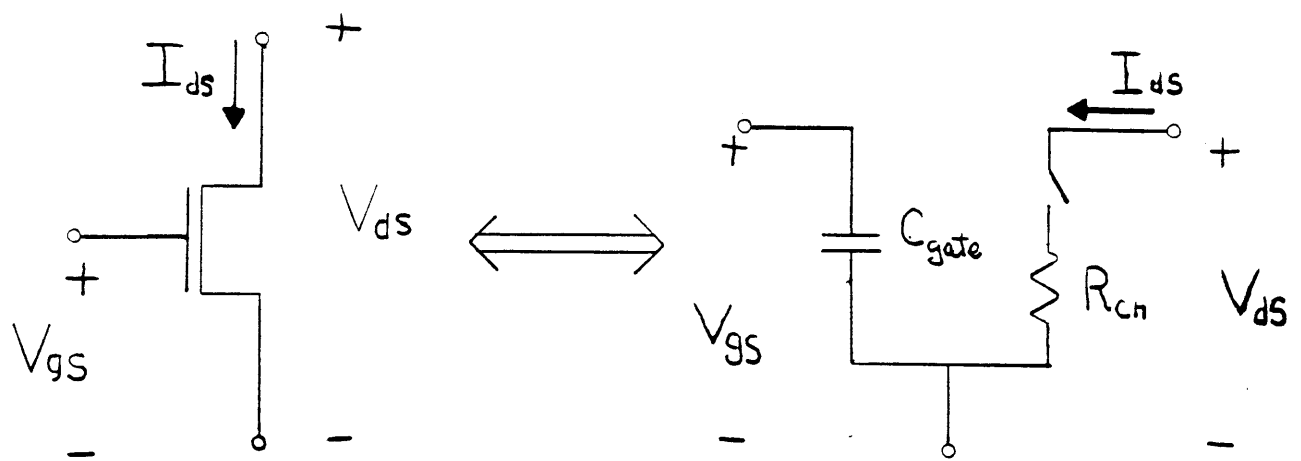
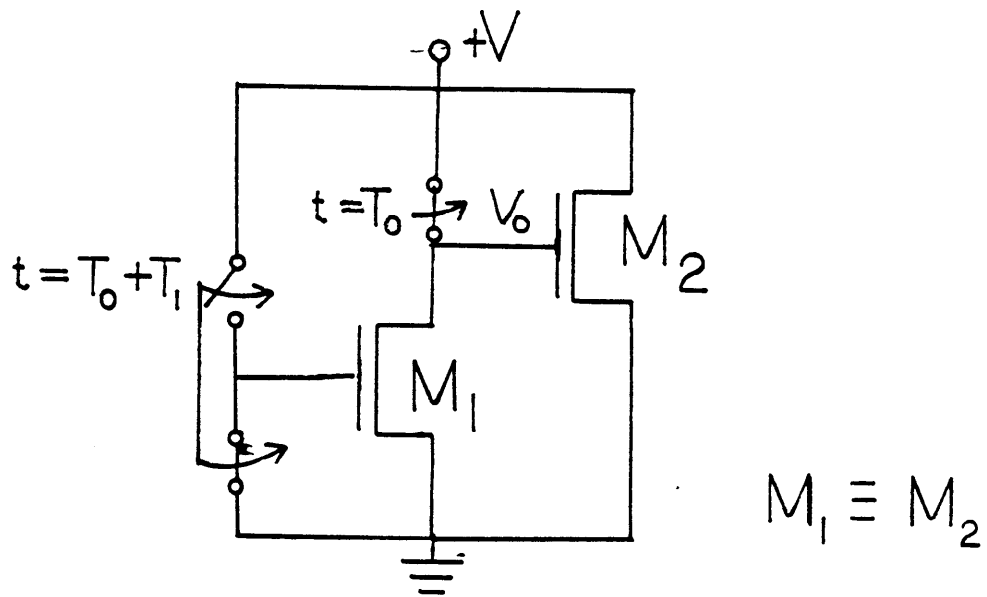


Fig 2.1 The Zero order RC model of MOS device





(a) TEST CIRCUIT

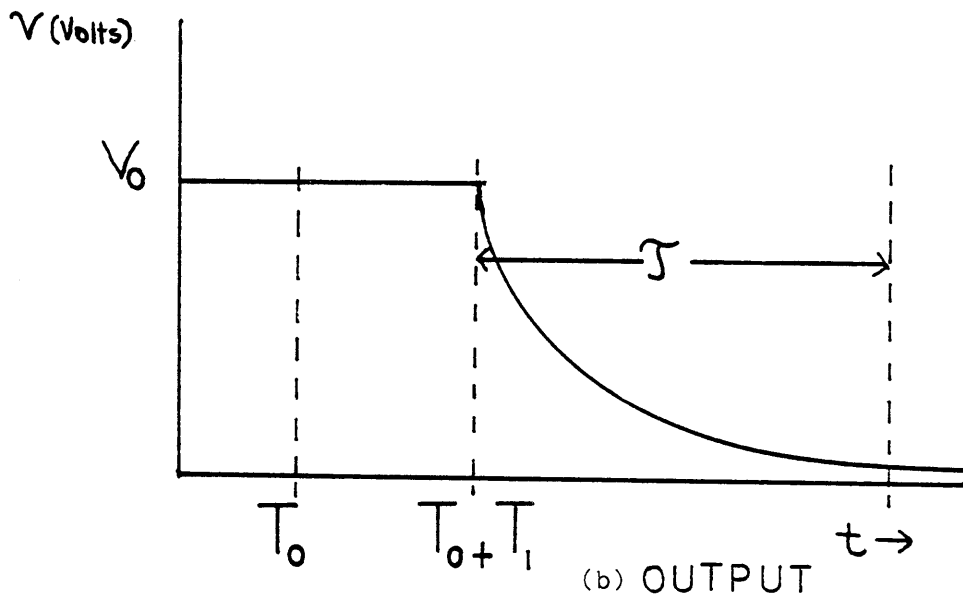


Fig 2.2  $\mathcal{T}$  as a measure of delay

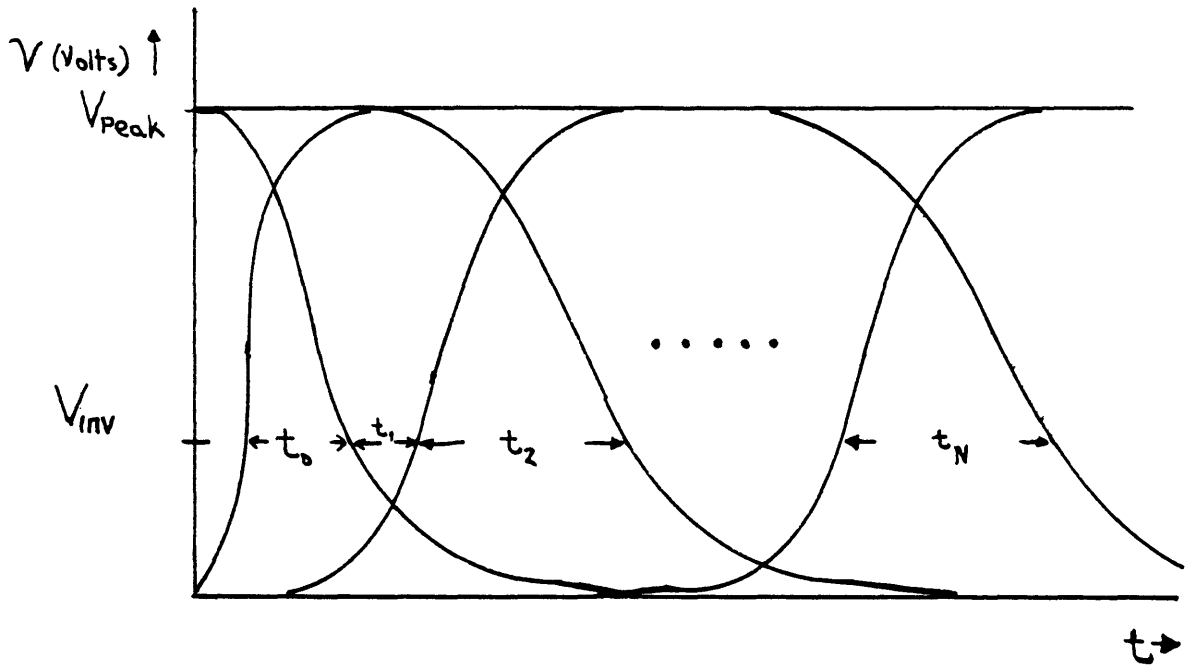


Fig 2.3 Some typical voltage waveforms of a MOS circuit

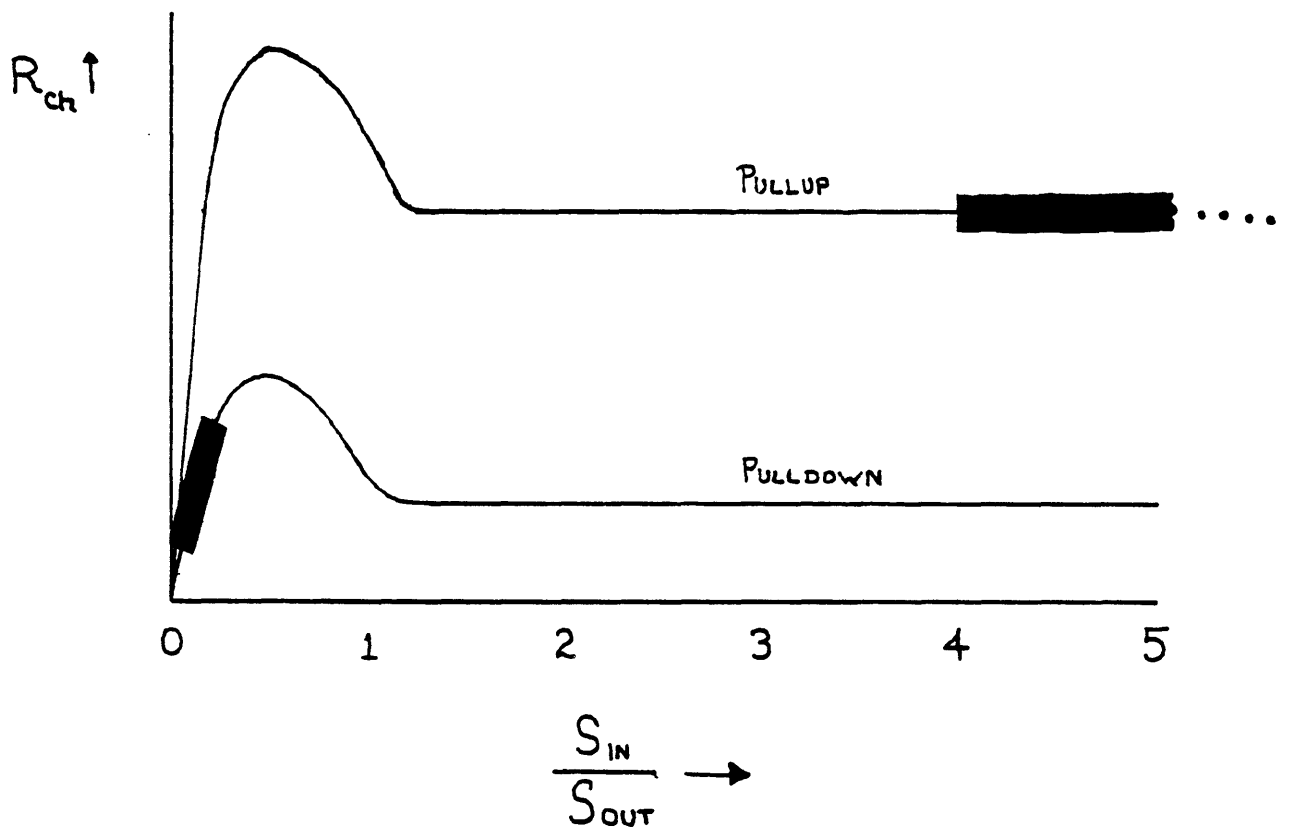


Fig 2.4 Channel resistance vs input and output waveshape

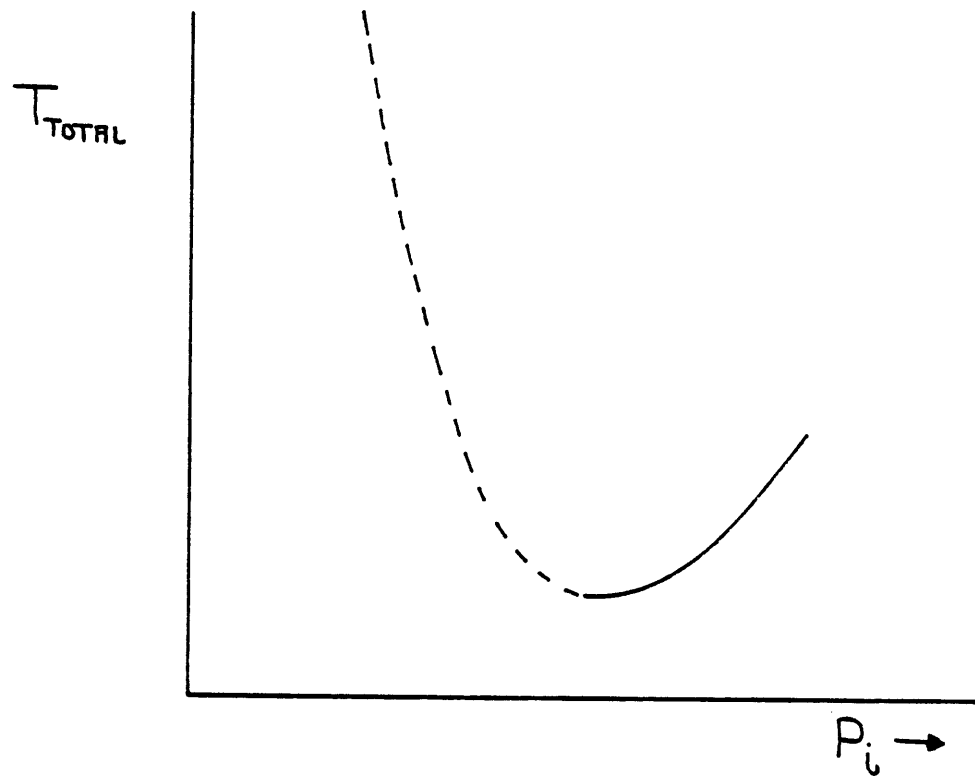


Fig 2.6 Delay vs stage power

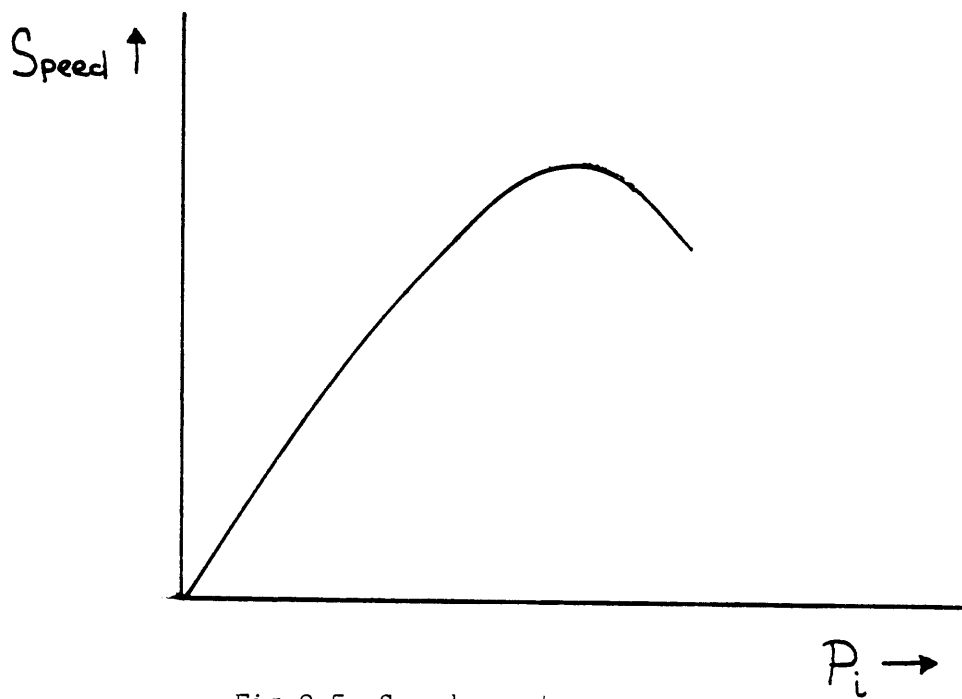


Fig 2.5 Speed vs stage power

### 3.0 Solving The System Of Equations

#### 3.1 The special case

Closed form solutions of (2.10) are difficult. For a special case, analytic solutions are readily obtained. In the case where  $K$  and all parasitics are set to zero, (2.10) reduces to

$$C_i = \left( \frac{n_i}{n_{i-1}} C_{i-1} C_{i+1} \right)^{\frac{1}{2}} \quad (3.1)$$

or,

$$n_{i-1} \left( \frac{C_i}{C_{i-1}} \right) = n_i \left( \frac{C_{i+1}}{C_i} \right) \quad (3.2)$$

Comparison to (2.8) shows that the solution to this special case requires that the stage delays be equal

$$t_{i-1} = t_i \equiv t_0 \quad (3.3)$$

for all  $i$ . Using this relation, the closed form solution for  $C_i$  may be found

$$\begin{aligned} t_0 &= R_0 C_1 \\ t_1 &= R_1 C_2 = \frac{n_1 \tau C_2}{C_1} \\ C_1 &= \frac{n_1 \tau}{t_1} C_2 \\ t_0 &= R_0 C_2 \frac{n_1 \tau}{t_1} \end{aligned} \quad (3.4)$$

using (3.3), (3.4) becomes

$$\begin{aligned} t_0^2 &= R_0 n_1 \tau C_2 \\ C_2 &= \frac{t_0^2}{R_0 n_1 \tau} \end{aligned}$$

in general

$$t_0^i = R_0 \prod_{j=1}^{i-1} n_j \tau^{i-1} C_i$$

and

$$C_i = \frac{t_0^i}{R_0 \prod_{j=1}^{i-1} n_j \tau^{i-1}} = C_0 \left( \frac{t_0}{\prod_{j=1}^{i-1} n_j \tau} \right)^i \quad (3.5)$$

$t_0$  is obtained by applying the boundary condition

$$t_0^{N+1} = R_0 \prod_{j=1}^N n_j \tau^N C_{N+1}$$

or

$$t_0 = \left( \prod_{j=1}^N n_j \tau^N R_0 C_{N+1} \right)^{\frac{1}{N+1}} = \tau \left( \frac{\prod_{j=1}^N n_j C_{N+1}}{C_0} \right)^{\frac{1}{N+1}} \quad (3.6)$$

Boundary evaluation of this result shows that the delay for a degenerate chain ( $N = 0$ ) is given (as expected) by  $R_0 C_{load}$ .

Using (3.6), it can also be shown that, for a special case, stage delay approaches a limiting value as the chain gets longer and longer that is, as  $N \rightarrow \infty$ . For the special case of a chain with  $N$  identical stage types, that is to say:

$$\alpha_i = \alpha_{i+1}$$

and

$$\beta_i = \beta_{i+1},$$

individual stage delays may be written as

$$t_i = \tau \left( \frac{\alpha^{\frac{N}{2}} (\alpha\beta)^{\frac{N}{2}} C_{N+1}}{C_0} \right)^{\frac{1}{N+1}}$$

which leads to

$$t_i = \tau \left( \frac{\alpha^{\frac{N+1}{2}} (\alpha\beta)^{\frac{N+1}{2}} C_{N+1}}{\alpha^{\frac{1}{2}} (\alpha\beta)^{\frac{1}{2}} C_0} \right)^{\frac{1}{N+1}}$$

or

$$t_i = \tau \alpha \beta^{\frac{1}{2}} \left( \frac{C_{N+1}}{\alpha \beta^{\frac{1}{2}} C_0} \right)^{\frac{1}{N+1}} \quad (3.6a)$$

Clearly (3.6a) approaches  $\tau\alpha\beta^{\frac{1}{2}}$  as  $N \rightarrow \infty$ . We assumed here that there are an even number of stages, that is  $\frac{N}{2}$  stages with active pullups and  $\frac{N}{2}$  with active pulldowns. We find that for an odd number of stages, there are two possibilities which depend upon the direction of the applied input signal. They are; 1)  $\frac{N+1}{2}$  stages with active pullups and  $\frac{N-1}{2}$  with active pulldowns, and 2)  $\frac{N-1}{2}$  stages with active pullups and  $\frac{N+1}{2}$  with active pulldowns. In the first case, stage delay is

$$t_i = \tau \left( \frac{\alpha^{\frac{N-1}{2}} (\alpha\beta)^{\frac{N+1}{2}} C_{N+1}}{C_0} \right)^{\frac{1}{N+1}} = \tau\alpha\beta^{\frac{1}{2}} \left( \frac{C_{N+1}}{\alpha C_0} \right)^{\frac{1}{N+1}} \quad (3.6b)$$

and in the second case

$$t_i = \tau \left( \frac{\alpha^{\frac{N+1}{2}} (\alpha\beta)^{\frac{N-1}{2}} C_{N+1}}{C_0} \right)^{\frac{1}{N+1}} = \tau\alpha\beta^{\frac{1}{2}} \left( \frac{C_{N+1}}{\alpha\beta C_0} \right)^{\frac{1}{N+1}} \quad (3.6c)$$

As with (3.6a), (3.6b) and (3.6c) approach  $\tau\alpha\beta^{\frac{1}{2}}$  as  $N \rightarrow \infty$ .

Additionally, equation (3.6) provides a way to determine the optimum length of a chain with fixed boundary conditions. The delay through  $N$  stages may be written as

$$T_{total} = (N + 1)t_0 = \tau(N + 1) \left( P \frac{C_{N+1}}{C_0} \right)^{\frac{1}{N+1}}$$

where

$$P = \prod_{j=1}^N n_j.$$

Setting

$$\frac{dT_{total}}{dN} = 0$$

we obtain,

$$N = \ln \left( P \frac{C_{N+1}}{C_0} \right) - 1$$

For an optimum number of buffers, (3.5) becomes

$$C_m = C_0 \left\{ \frac{\left( P \frac{C_{N+1}}{C_0} \right)^{\ln \left( P \frac{C_{N+1}}{C_0} \right)}}{\prod_{j=1}^{m-1} n_j} \right\}^m$$

or

$$C_m = C_0 \left( \frac{e}{\prod_{j=1}^{m-1} n_j} \right)^m$$

Also, the optimal individual stage delays become

$$t_{opt} = \tau e$$

and thus the lowest possible circuit delay is

$$T_{best} = (N + 1)t_{opt} = \left[ \ln \left( P \frac{C_{N+1}}{C_0} \right) \right] \tau e$$

### 3.2 Solving the general case

A numerical technique is used to solve (2.10) in the general case. Initial guesses are made for the  $C_i$ 's and iterations are made towards the final solution. This numerical technique is low in computational complexity since there are only 6 floating point operations required per stage per iteration for circuit optimization (2.10). Using today's "megaflop" computers (i.e, machines capable of more than  $10^8$  floating point operations per second), this means that 10 iterations of a circuit with  $10^4$  stages could take place in under 1 second of CPU time. Indeed the processor time required for managing the data and other "overhead" operations may well turn out to be the factor which determines the amount of CPU time required for this optimizer. Each iteration can be computed in linear time because each  $C_i$  depends

solely upon the characteristics of its immediate neighbours. Thus, the number of computations required for each  $C_i$  is independent of the number of stages in the chain. For example, doubling the length of the chain only doubles the number of  $C_i$ 's to be computed.

### 3.3 Convergence

As with most iterative solution schemes, the question of convergence of the solution must be raised. Consider the general form of solution equations (2.10)

$$C_i = \left\{ \frac{a_i \{C_{i+1} + X_i\}}{\frac{a_i-1}{C_{i-1}} + \frac{\mathfrak{K}}{A_i}} \right\}^{\frac{1}{2}} \quad (3.7)$$

where,  $\mathfrak{K}, X_i, A_i, C_k > 0$  and  $a_k > 1$ . Here,  $\mathfrak{K} = -K$ .

For an  $N$  stage system, there are a set of  $N$  equations with

$$C_1 = \left\{ \frac{a_1 \{C_2 + X_1\}}{\frac{a_1-1}{C_0} + \frac{\mathfrak{K}}{A_1}} \right\}^{\frac{1}{2}} \quad (3.8)$$

where  $C_0 = \text{Constant}$  and,

$$C_N = \left\{ \frac{a_N \{C_{load} + X_N\}}{\frac{a_N-1}{C_{N-1}} + \frac{\mathfrak{K}}{A_N}} \right\}^{\frac{1}{2}} \quad (3.9)$$

with  $C_{load} > C_0$ .

Consider the case of a 2 stage system given by equations(3.8), (3.9) with  $N = 2$ . Substituting for  $C_1$  and  $C_2$  the results

$$C_1^{j+1} = \left\{ \frac{a_1 \left[ \frac{a_2 \{C_{load} + X_2\}}{\frac{a_2-1}{C_1^j} + \frac{\mathfrak{K}}{A_2}} \right]^{\frac{1}{2}} + X_1}{\frac{a_1-1}{C_0} + \frac{\mathfrak{K}}{A_1}} \right\}^{\frac{1}{2}}$$



and

$$C_2^{j+1} = \left\{ \frac{a_2 \{C_{load} + X_2\}}{a_1 / \left[ \frac{a_0 C_2^j + X_2}{C_0^j + A_1} \right]^{\frac{1}{2}} + \frac{K}{A_2}} \right\}^{\frac{1}{2}}$$

are obtained. (Note that  $C_i^m$  is the value of  $C_i$  after the  $m^{th}$  iteration.) The solutions are considered to be convergent if they move in the same direction under iteration.

That is, if

$$C_i^j \geq C_i^{j-1}$$

then

$$C_i^{j+1} \geq C_i^j$$

or if

$$C_i^j < C_i^{j-1}$$

then

$$C_i^{j+1} < C_i^j$$

Convergence in () may be verified by studying the term

$$V_i^j = a_1 \left\{ \frac{a_2 (C_{LOAD} + X_2)}{\frac{a_1}{C_i^j} + \frac{K}{A_2}} \right\}^{\frac{1}{2}}$$

whose derivative with respect to  $C_i^j$  is always positive. Let  $C_i(0) = C_i^0$  be the initial estimate of  $C_i^{FINAL}$ . If  $C_1^1 < C_1^0$  then  $V_1^1 < V_1^0$  which implies that  $C_1^2 < C_1^1$ . Recall that

$$C_1^1 = \left\{ \frac{V_1^0 + X_1}{\frac{a_0}{C_0^0} + \frac{K}{A_1}} \right\}^{\frac{1}{2}}$$

$$C_1^2 = \left\{ \frac{V_1^1 + X_1}{\frac{a_0}{C_0^0} + \frac{K}{A_1}} \right\}^{\frac{1}{2}}$$

Since  $V_1^1 \leq V_1^0$ ,  $C_1^2 \leq C_1^1$ . Similarly, if  $C_1^1 > C_1^0$ , then  $V_1^1 > V_1^0$  and  $C_1^2 > C_1^1$ . A similar argument is used to show convergence of () i.e, if  $C_2^0 > C_2^1$ , then  $C_2^2 > C_2^1$  and conversely if  $C_2^0 < C_2^1$ , then  $C_2^2 < C_2^1$ .

An expansion to the general case is now undertaken. The general form of the system equation was given as

$$C_i^j = \left\{ \frac{a_i(C_{i+1}^j + X_i)}{\frac{a_{i-1}}{C_{i-1}^j} + \frac{b_i}{A_i}} \right\}^{\frac{1}{2}}$$

Let there be an optimum  $C_i$  for stage  $i$  called  $C_{opt}$ . The error in the  $i^{th}$  stage after the  $j^{th}$  iteration may be defined as

$$\mathfrak{E}_i^j = (C_{opt} - C_i^j)^2$$

The equations converge if this error is made smaller after each iteration, that is if

$$\mathfrak{E}_i^j \geq \mathfrak{E}_i^{j+1}$$

for all  $j$ . Now,

$$\mathfrak{E}_i^j = C_{opt}^2 + C_i^j(C_i^j - 2C_{opt})$$

$$\mathfrak{E}_i^{j+1} = C_{opt}^2 + C_i^{j+1}(C_i^{j+1} - 2C_{opt})$$

For  $\mathfrak{E}_i^j \geq \mathfrak{E}_i^{j+1}$ ,  $C_i^{j+1} \leq C_i^j$ . That is, the  $C_i$ 's may not grow under iteration.

To satisfy this condition, the set of initial guesses for the  $C_i$ 's must be larger than the optimum values. After finding such a set of initial values, it will be shown that they become smaller after the first iteration. It will then be argued that if they shrink after the  $j^{th}$  iteration, then they will not grow after the  $j + 1^{st}$  iteration.

A set of initial values  $C_{g(i)}$  are sought so that  $C_i < C_{g(i)}$ . At the end of the chain,

$$C_N = \left\{ \frac{a_N(C_{LOAD} + X_N)}{\frac{a_{N-1}}{C_{N-1}} + \frac{b_N}{A_N}} \right\}^{\frac{1}{2}}$$

the condition

$$C_N^1 \leq C_{g(N)}$$

is desired.

This means that

$$C_{g(N)}^2 > \left\{ \frac{a_N C_{g(N)} (C_{LOAD} + X_N)}{a_{N-1} + \frac{C_{g(N)}^2}{a_N}} \right\} \quad (3.10)$$

For the case  $K = 0$ ,

$$\frac{a_N}{a_{N-1}} (C_{LOAD} + X_N) < C_{g(N)}$$

or,

$$C_{g(N)} = a_N \frac{a_N}{a_{N-1}} C_{LOAD} + X_N$$

where  $0 < \alpha < 1$ . This inequality is made stronger for  $K > 0$  (see (3.10)).

For the  $N - 1^{st}$  stage,

$$C_{N-1} = \frac{a_{N-1}}{a_{N-2}} (a_N C_{g(N)} + X_{N-1} C_{g(N-1)}) < C_{g(N-1)}^2$$

or,

$$C_{g(N-1)} > \frac{a_{N-1}}{a_{N-2}} (a_N C_{g(N)} + X_{(N-1)}) \quad (3.11)$$

In general,

$$C_{g(n)} > \frac{a_{n-1}}{a_{n-2}} (a_{n+1} C_{g(n+1)} + X_n) \quad (3.12)$$

where

$$C_{g(N)} = \frac{a_N}{a_{N-1}} (a_{load} C_{g(load)} + X_N)$$

Here, the  $\alpha$ 's provide a way to uphold the inequalities (3.11) and (3.12). These initial estimates will produce lower values after the first iteration.

The general equations show that if a member's immediate neighbours  $i + 1^{st}$ ,  $i - 1^{st}$  get smaller, then that member will also become smaller. This shrinking has the effect of decreasing the  $i - 1^{st}$  member, repeating down to the  $0^{th}$  stage. Since all optimum values are expected to lie between  $C_0$  and  $C_{load}$  where

$$C_{load} > C_0$$

and all initial guesses are larger than  $C_0$ , the  $0^{th}$  stage will be "slowed down" when it gets close to its optimum. This argument is justified by pointing out that the expression for  $C_1$  is of the form

$$C_1 = (\mathcal{C}C_2C_0)^{\frac{1}{2}}$$

(see (3.7) for  $\mathcal{K} = 0$ ). As  $\mathcal{C}C_2$  approaches  $C_0$ ,  $C_1$  becomes more like  $C_0$ . This slowing down propagates through to  $C_2$  and  $C_3$  on to  $C_{N-1}$ . When the optimum  $C_i$ 's are reached, no further changes occur.

#### 4.0 Parameter Extraction

In order to optimize a circuit according to our theory, two sets of parameters are required. These are;

- (1) The relationship between  $C_{gate}$  and  $R_{ch}$  ( $\tau$ , equation (2.1)).
- (2) The ratios between the worst case speed and worst case power estimates of  $R_{ch}$  ( $\gamma_{pullup}$ ,  $\gamma_{pulldown}$ , eqs(2.7a,b)).

Also, in order to specify the results of the optimizer as real circuits, the mapping between physical device and the gate capacitance parameter ( $C_i$ ) used by the optimizer theory, must be found. The  $\tau$  and  $\gamma$  parameters are process dependent as is the relationship between device geometry and  $C_i$ . Any attempt to determine these must therefore take the fabrication process into account. In this section, we specify a method for extracting these process parameters. The method is an experimental one owing to the sheer complexity of the device models.

Briefly, the method consists of simulating a test circuit under worst case speed conditions and equating the measured circuit delay to  $\tau$  in order to determine the effective worst case speed values of  $C_{gate}$  and  $R_{ch}$  ( $C_{eff}$  and  $R_{eff}$  respectively). By comparison of  $C_{eff}$  and  $R_{eff}$  to the dimensions of the test devices, these parameters may be determined in terms of device dimensions: that is, gate capacitance per square ( $C_{/sq}$ ) and channel resistance per square ( $R_{/sq}$ ) may be found. Then, for a device of given dimensions, the relationship between  $C_{gate}$  and  $R_{ch}$  may be calculated. The  $\gamma$ 's is determined by measuring the power of an "on" inverter under worst case power conditions to determine its pullup and pulldown resistances. The  $\gamma$ 's are then computed according to (2.7a) and (2.7b).

We begin by using SPICE[3] to determine  $R_{eff}(\frac{S_{in}}{S_{out}})$  over a useful range of  $S_{in}$  and  $S_{out}$ . Knowledge of  $R_{eff}(\frac{S_{in}}{S_{out}})$  will then be used to compute  $C_{eff}$ . The experimental circuit shown in fig(4.1), consists of an inverter whose input signal slope is directly controlled and whose output slope is varied by changing  $C_{out}$ . Slopes are measured according to (2.2). For a given output slope, signals of varying slopes are applied to the input.  $R_{eff}$  is defined as

$$R_{eff} = \frac{\tau}{C_{out}}$$

where  $\tau$  (delay) is measured as given in (2.1). The process is repeated with different values of output slope to determine  $R_{eff}$  over the  $\frac{S_{in}}{S_{out}}$  space. This experiment is performed for both upgoing and downgoing input transients to determine  $R_{eff}(\frac{S_{in}}{S_{out}})$  for the pulldown and pullup devices respectively.

Knowing  $R_{eff}(\frac{S_{in}}{S_{out}})$ ,  $C_{eff}$  may be determined by replacing  $C_{out}$  with an identical inverter (fig(4.2)) and again applying another set of signal slopes to the input. For the load gate,  $C_{eff}$  is given by:

$$C_{eff} = \frac{\tau}{R_{eff}}$$

where  $R_{eff}$  is the previously computed effective channel resistance of the of the driver inverter.

$R_{eff}$  and  $C_{eff}$  may then be related to the widths and lengths of the devices used, in order to derive the mapping between the effective parameters and the drawn device dimensions.

$$R_{/sq} = R_{eff} \frac{W_{channel}}{L_{channel}}$$

and

$$C_{/sq} = \frac{C_{eff}}{W_{channel}L_{channel}}$$

where  $W$  and  $L$  are the device dimensions of the experimental circuit.  $R_{/sq}$  and  $C_{/sq}$  are the channel resistance per square and channel capacitance per square for the tested fabrication process. The units of  $W$  and  $L$  are also the area units of  $C_{/sq}$ , e.g., if  $W$  and  $L$  are in microns then  $C_{/sq}$  would be in Farads per square microns.

The relationship between  $C_{gate}$  and  $R_{ch}$ , of any device made by this process, may then be found

$$W_{channel} = \frac{C_{gate}}{C_{/sq}L_{channel}}$$

then,

$$R_{channel} = \frac{L_{channel}}{W_{channel}R_{/sq}} = \frac{L_{channel}^2 C_{/sq} R_{/sq}}{C_{gate}} = \frac{\tau'}{C_{gate}}$$

Because

$$\tau \propto L_{pulldown}$$

the extracted parameters are valid only for those pulldown lengths simulated.

We compute the  $\gamma$ 's by first measuring the total power consumption of our test inverter in the "on" state, under worst case power conditions.  $R_{power}$  (eq(2.7)) is then computed by

$$R_{power} = \frac{V_{dd}^2}{power}$$

The pullup and pulldown resistances are then determined using the drawn pullup/pulldown ratio ( $\beta_T$ ) of the test circuit

$$R_{power(pulldown)} = \frac{R_{power}}{\beta_T + 1}$$

and

$$R_{power(pullup)} = \frac{R_{power}\beta_T}{\beta_T + 1}$$

The  $\gamma$ 's are then readily computed

$$\gamma_{\text{pulldown}} = \frac{R_{\text{power}}(\text{pulldown})}{R_{\text{eff}}(\text{pulldown})}$$

and

$$\gamma_{\text{pullup}} = \frac{R_{\text{power}}(\text{pullup})}{R_{\text{eff}}(\text{pullup})}$$



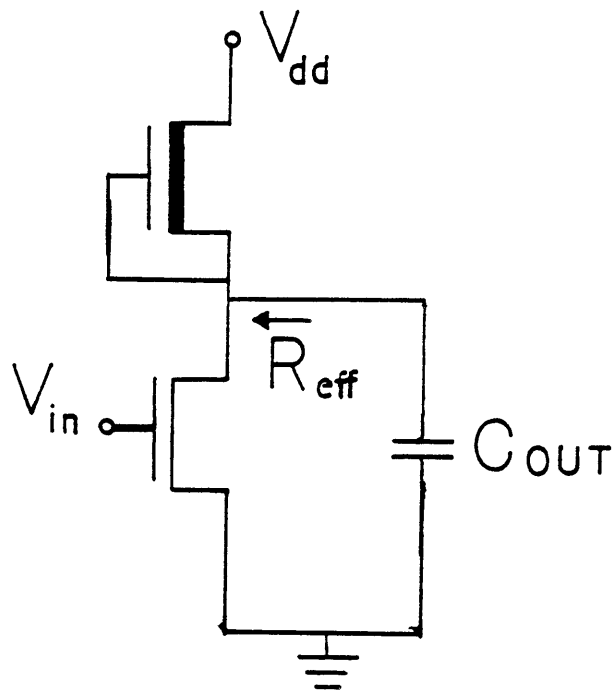


Fig 4.1 Test Circuit for  $R_{eff}$

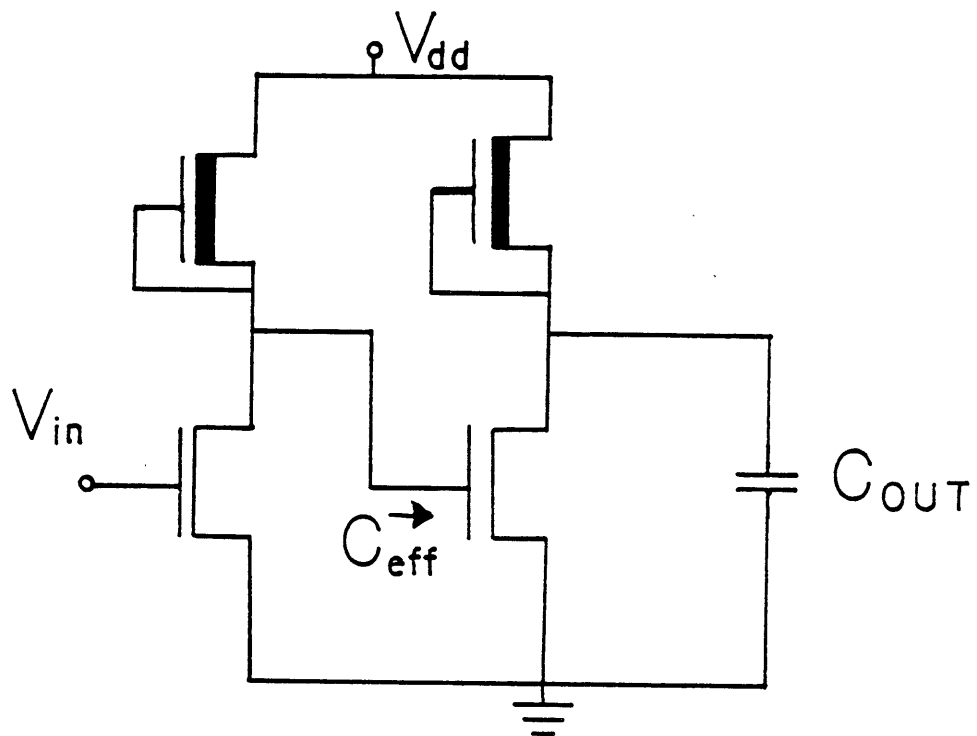


Fig 4.2 Test Circuit for  $C_{eff}$

## 5.0 Implementation as a Software System

A computer program called WIZARD and written in the C language was developed to implement the optimization process. Inputs to WIZARD are

- (i) A circuit datafile specifying the form of the circuit, see fig(5.1).
- (ii) A process datafile which gives the parameters of the process to be used when optimizing the circuit, fig(5.2).

WIZARD is a three level system with each level handling a specific group of chores. The highest level ("READY>") is responsible for interacting with the user to obtain process and circuit information from datafiles. The program's second level, called ("SETUP>"), permits the user to edit the process parameters and also to bind the circuit specification to the given fabrication process. The deepest level ("OPTIONS>>") is also interactive and allows the modification of user parameters (including boundary conditions, number of iterations, and required delay). Circuit optimization may also be initiated from this level.

The user may optimize either for a specific input transient or ask WIZARD to optimize for the fastest input transient. Upon request for an optimization, WIZARD performs it, informs the user of the results and queries for the name of a file in which to place the output. This output consists of Speed and Power information, along with the optimized dimensions of all the transistors in the circuit. At this time, WIZARD returns to the "OPTIONS>>" level. A user picture of the system is given in fig(5.3).

The "OPTIONS>>" level controls and implements the optimization algorithm presented in fig(5.4) for different senses of input signal. In fig(5.4) the variable  $T_0$

refers to the circuit delay requested by the system user. The blocks "SADDLE  $T_0$ " and "CHANGE  $K$ " specify a linear interpolation scheme which is used to change  $K$  so as to "zero in" on the required delay. Specifically, "SADDLE  $T_0$ " finds an initial pair of  $K$ 's,  $K_1$  and  $K_2$  so that

$$\text{delay}(K_1) < T_0 < \text{delay}(K_2)$$

where the function  $\text{delay}(K_n)$  is the total optimized circuit delay when  $K = K_n$ . The amount by which  $K$  is changed when searching for  $K_1$  and  $K_2$  is specified by a parameter in the input process file. The module "CHANGE  $K$ " in fig(5.4) applies the interpolation algorithm. The delay tolerance factor determines the region of values around the requested delay in which computed delay will be considered acceptable to the user.

The "OPTIMIZE" block in fig(5.4) applies the optimization equation (2.10) to successive stages in the circuit, starting at the output end and finishing at the input end. Each such traversal of the circuit is considered an iteration. Thus, the first iteration uses the boundary conditions and/or the initial guesses of adjacent stages to compute new stage variables. Subsequent iterations make use of the previously computed stage variables and/or the boundary conditions. The optimization is considered complete when either; the user specified number of iterations is performed, or; the user specified convergence tolerance is achieved. The convergence tolerance factor specifies the maximum difference between delays, computed after successive iterations, which is acceptable to the user as a converged result.

Total power consumption is found by assuming that half of all stages have a conductive path between the supply ( $V_{dd}$ ) and ground. This is an acceptable assumption since the direction of signal flow alternates as the signal propagates down

the chain e.g. if the input to stage  $i$  is charging, then the input to stage  $i + 1$  would be discharging and so on. By such a token, approximately half of the stages would have a conductive path to ground owing to the charge at their inputs which would turn on their pulldown transistors. Specifically, total power is computed by summing individual stage powers (2.7d) and dividing by 2. Delay is calculated according to (2.6).

A list of all user controlled parameters used by WIZARD is presented, along with their effects and the level at which they may be accessed, in fig(5.5).

### 5.1 Decoupler

While the optimization is taking place, WIZARD makes sure that all parameters stay within acceptable limits. In particular, each new value of  $C_i$  is checked to see that it stays above  $C_{MIN}$  as calculated using  $W_{MIN}$ ,  $L_{MIN}$  and  $C/sq$ . If  $C_i$  falls below  $C_{MIN}$ , as it might in low power situations, a special handler takes over and resolves that situation. This handler, called a decoupler, looks at the present stage and figures out, based on the direction of the input transient, which device is active. If the pullup is found to be active, WIZARD computes a new pullup resistance based on the existing pullup/pulldown resistance ( $\beta$ ) ratio and the new (illegal) value of  $C_i$

$$R_{pullup} = \frac{\tau\beta}{C_i}$$

where  $R_{pullup}$  is limited by  $R_{MAX}$ .  $C_i$  is then forced to  $C_{MIN}$  and a new pullup/pulldown ratio is computed so that

$$R_{pullup} = \frac{\tau\beta'}{C_{MIN}}$$

or

$$\beta' = \frac{R_{pullup}C_{MIN}}{\tau}$$

In the case of the active pulldown, WIZARD forces  $C_i$  to  $C_{MIN}$  and computes a new  $\beta$  ratio

$$\beta' = \frac{R_{MAX}C_{MIN}}{\tau}$$

Process parameters:

Channel Resistance per square : Pullup and Pulldown.

Gamma: Pullup and Pulldown.

Gate Capacitance per square: Pullup and Pulldown.

Pass device Capacitance per square.

Limitations on device geometries:

Minimum channel length.

Minimum channel width.

Maximum channel length. (Used to compute maximum pullup length).

User parameters also included in process file:

Initial guess at channel width.

Maximum number of iterations to perform.

Convergence tolerance factor.

Delay search tolerance factor.

Delay search modifier.

Fig 5.1 General form of input process file

Circuit parameters:

Source Resistance

Load Capacitance

Number of Stages

Stage parameters:

Structure of pulldown network ( $\alpha_i$  ).

Total Area of pass devices in this stage (i);  
(Used to compute  $Z_i$ ,  $X_i$  and the pullup/pulldown  
ratio of stage i+1 i.e., stage i+1 would have an  
8:1 ratio if stage i had a pass device, otherwise  
its ratio would be 4:1.).

Total fanout area; This is the area of all device  
gate inputs, not along the critical path, driven  
from stage i. (Also used to compute  $X_i$  ).

Total area of polysilicon lines driven from stage i.  
(Used also to compute  $X_i$  ).

Fig 5.2 General form of circuit data file.

```

READY>
    Get process data from user specified file.
    Get circuit data " " " "
    Specify a circuit to WIZARD.
    Go to next level (SETUP>>).
        Default process and/or circuit data are
        used if not specified.
    Quit WIZARD.

SETUP>>
    Edit process parameters.

    Bind circuit to process and go to next
    level (OPTIONS>>)

    Go directly to "OPTIONS>>" if circuit is
    already bound.

    Save process data in user specified file.

    Save circuit " " " " "

    Return to previous level (READY>>).
        Circuit and process data are erased.

    Include/neglect parasitics in calculations.

OPTIONS>>
    Edit user parameters and circuit boundary
    conditions.

    Initiate an optimization for a user specified
    input signal direction.

    Optimize for all input signal directions and
    release the results for the fastest case.
        After optimization, WIZARD prints the
        results and stores them in a user specified
        file (see Appendix A).

    Return to previous level (SETUP>>).

```

Fig 5.3 User diagram of the optimizer program



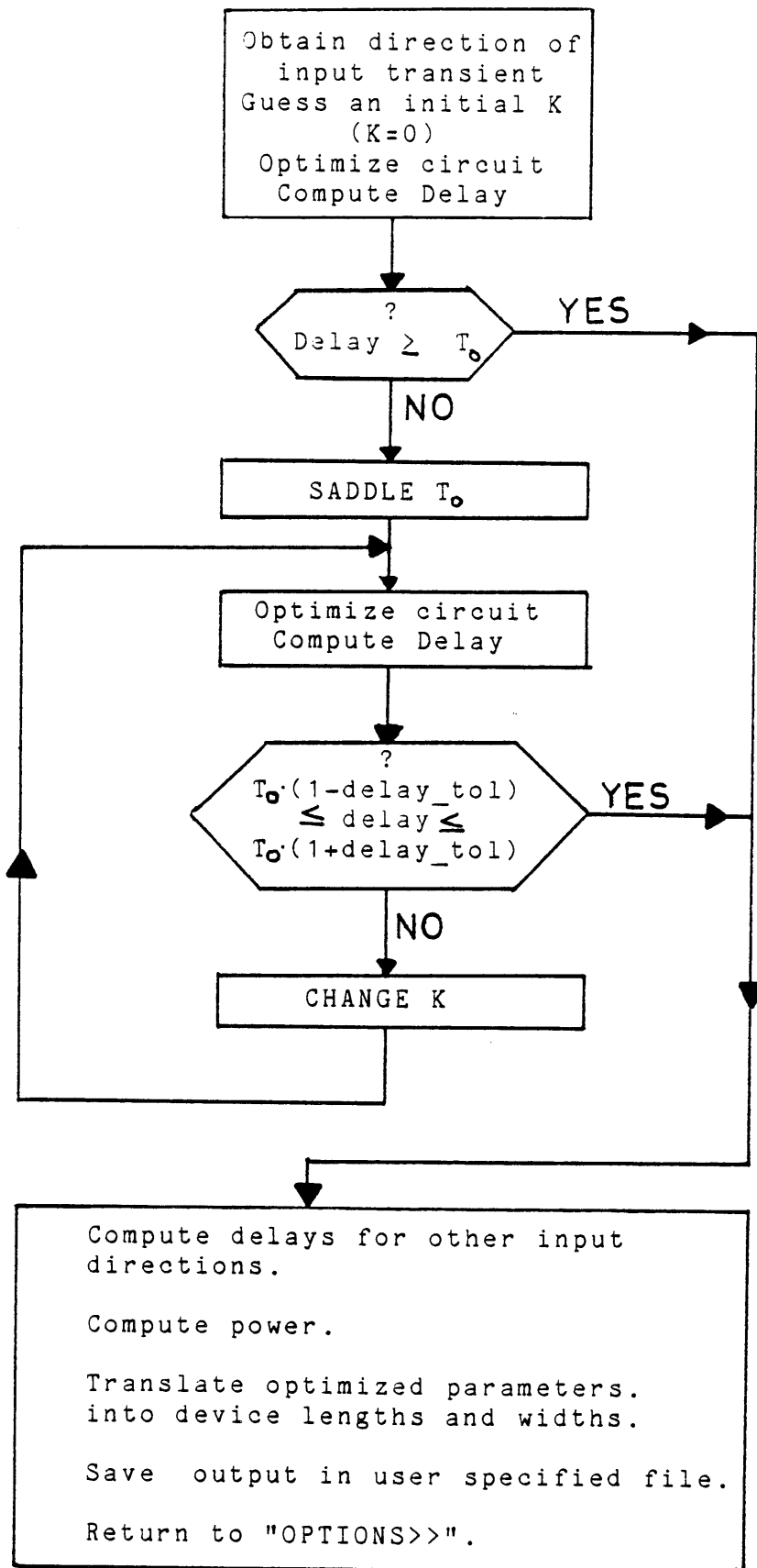


Fig 5.4 The Optimization algorithm.

SETUP:

Max gate length	default	20	microns.
Min " "	"	2	"
Min " width	"	2	"
Initial guess of gate width	"	4	"
Edge capacitance per unit length	"	.2	fF
Poly capacitance per square.	"	.03	ff
Pass device capacitance per square	"	1	"
Channel resistance " "	"		
Pullup	"	50	Kilohms
Pulldown	"	40	"
Gamma			
Pullup	"	1	-
Pulldown	"	1	-
Supply voltage (worst case power)	"	5.5	Volts

OPTIONS:

Source resistance (up and down inputs)	"	10	Kilohms
Load capacitance	"	.1	pF
Requested ckt. delay	"	0.0	Sec
Max number of iterations	"	50	-
K search factor	"	1e3	Sec/Watt
Delay tolerance	"	20	percent
Convergence tolerance	"	.1	"

Fig 5.5 Optimizer parameters: defaults and levels of access.

## 6.0 Results

The performance of WIZARD may be gauged from three points of view; how well it implements the theory, how well its results fare under circuit simulation, and its measured computational complexity. Experiments were carried out with WIZARD to determine its performance in these respects. The results of these experiments are presented in this section.

### 6.1 Theoretical purity

To evaluate WIZARD in the first light, we must again refer to the optimization theory. Recall that:

$$T = \sum_{i=0}^N \left( \frac{n_i}{C_i} + Z_i \right) (C_{i+1} + X_i) \quad (2.5)$$

and

$$K = \frac{dT}{dP_i} = A_i \left[ \left( \frac{\tau n_{i-1}}{C_{i-1}} + Z_i \right) - \frac{\tau n_i (C_{i+1} + X_i)}{C_i^2} \right] \quad (2.9)$$

which leads to

$$\frac{KC_i}{A_i} = \frac{\tau n_{i-1} C_i}{C_{i-1}} + Z_i C_i - \frac{\tau n_i C_{i-1}}{C_i} - \frac{\tau n_i X_i}{C_i} \quad (6.1)$$

and

$$t_i = t_{i-1} - \frac{\tau X_i n_i}{C_i} + Z_i C_i - \frac{KC_i}{A_i} \quad (6.2)$$

Using equation (6.2), successive stage delays may be related for different circuit conditions. As presented in an earlier section, the case of  $K = X_i = Z_i = 0$  gives the result

$$t_i = t_{i-1}$$

i.e equal stage delays. For  $K = Z_i = 0$  and  $X_i \geq 0$ , we find that  $t_i \geq t_{i-1}$  and for  $K \leq 0$

and  $X_i = Z_i = 0$

$$t_i \leq t_{i-1}$$

Application of WIZARD to an experimental circuit showed all of these results to hold when the decoupler was not activated.

Inclusion of the decoupler produced, for  $K = 0$ , the same result as without it under normal circumstances. This is to be expected since the decoupler is not designed to come into play except in low power applications i.e,  $K \ll 0$ . When its used was forced (by making  $C_{MIN}$  abnormally large), WIZARD would fix the sizes of all stages which tried to become smaller than  $C_{MIN}$ . Since the number of stages that are fixed depends on the value of  $C_{MIN}$ , the general relationship between the stage delays in this case is not very useful.

When the decoupler is used on a chain of  $N$  identical stages (i.e, where  $\alpha_i = \alpha_{i+1}$  and  $\beta_i = \beta_{i+1}$ ), one would expect that, for the case  $K \ll 0$  and  $X_i = Z_i = 0$ , all the stages would eventually be decoupled. That is

$$C_i = C_{i+1}$$

for  $0 < i < N$ . In this case, there would only be two unique values of stage delay in the system: those stages with active pullups would all have one value of delay and those stages with active pulldowns would have another delay value. The delays at the circuit boundaries would be different because of the user defined boundary conditions. We found this result to hold when WIZARD was applied to such a circuit.

## 6.2 Circuit simulation of results

A 5 stage circuit, optimized by WIZARD for minimum delay, was analyzed using the SPICE[3] circuit simulator. The purposes of this analysis were: 1) to compare the results computed by WIZARD to the results computed by the simulator, and 2) to determine if the circuit was indeed an optimal one.

The optimizer's estimate of total delay was found to be within 12% of the SPICE delay, whereas its power estimate was within 53% of the SPICE power estimate.

To test whether the circuit was an optimal one, we randomly changed the widths of various stages in the circuit. If the circuit was optimal (as claimed), then any change in the width of any stage would cause an increase in total delay<sup>1</sup>. SPICE showed an increase in delay for all of the width changes made (increases and decreases), suggesting that WIZARD did produce an optimum circuit according to the theory.

## 6.3 Complexity

We judge the complexity of the optimization algorithm on both its speed of operation (runtime) and the dependence of its convergence on the size of the circuit optimized.

### 6.3.1 Runtime

When WIZARD (running on a DEC 20/60 computer) was used on a 1000 stage circuit, the optimize time was found to be 6 CPU seconds for 4 iterations which implies an average iteration time of 1.5 CPU seconds. This result shows that the

<sup>1</sup> Since  $W \propto Power$ , changing stage width has the effect of changing stage power. Because delay was optimized with respect to power, both increases and decreases in power should cause increases in delay.

algorithm is fast enough to be used as part of an interactive system.

### 6.3.2 Convergence dependence

It was argued that our optimization algorithm would run in linear time. Under experiment however, it was found that the number of iterations required for convergence varies slightly with the number of stages in the chain. In particular, convergence was defined as the point at which successive iterations produced total delays which were arbitrarily close.

We found that, for a circuit of fixed boundary conditions (input resistance, output capacitance) and fixed initial guesses, the number of iterations required for convergence in a 5 stage circuit was slightly larger than that required for a 50 stage circuit. This difference was due to the fact that the optimum  $C_i$ 's for the 50 stage circuit were much closer in values to the initial guesses to the optimum  $C_i$ 's for the 5 stage chain. The number of iterations required for convergence is dependent on the difference between the optimum  $C_i$ 's and the initial guesses. The dependence on chain length is present only because the optimum  $C_i$ 's depend upon the chain length.

## 7.0 Conclusion

We have developed and implemented a circuit optimizer for combinational MOS circuits, which optimizes a given signal path to meet a speed specification at the minimum possible power consumption. This optimizer runs in linear time and took 6 CPU seconds (on a DEC 20/60) to optimize a 1000 stage circuit (i.e., at least 2000 transistors). Compared to circuit simulation, the optimizer's results (for a 5 stage circuit) were accurate to within 12% for circuit delay and 53% for power consumption. From these results, it is obvious that this circuit optimizer is fast enough to be used interactively and yet accurate enough to be used in serious design applications. Indeed we believe that, if coupled with a reasonable critical path analyzer, this optimizer would be ideal for use in automatic circuit synthesis tools.

Further work on circuit optimization is being conducted by Mark Matson of the VLSI circuits group at MIT. In particular, Matson's work includes the development of more accurate circuit models, methods of optimizing critical paths, and techniques for hierarchical circuit optimization to improve execution time. He will also address other optimization criteria, such as circuit area and dynamic power consumption. It is hoped that his work will produce a system which will optimize the performance of large designs with respect to a designer's objectives.

## References

- [1] Baker C. and Terman C., private communication .
- [2] Bryant R., *MOSSIM: A logic level simulator for MOS LSI* , PhD thesis, MIT, (1980).
- [3] Nagel L. W., ERL memo ERL-M520, University of California, Berkeley , May 1975.
- [4] Armstrong R.C., HPEDIT User's Manual MIT, 1982 .
- [5] Batali J. and Hartmaier A., "The Design Procedure Language manual," *MIT Artificial Intelligence Laboratory Memo 598* , 1980.
- [6] Penfield P., AIDS User's Manual MIT 1980 .
- [7] MIT Lincoln Laboratory, "Semiannual Technical Report on the Restructurable VLSI Program," *Technical report* , MIT Lincoln Laboratory, Jan 1981.
- [8] Hsueh M.Y., "Symbolic Layout Compaction," *NATO Advanced Study Institute on Computer aided Designs for VLSI Circuits* , June 1980.
- [9] Cho Y, Korenjak A. and Stockton D., "FLOSS: An approach to Automated Layout for High-Volume Designs," *Proceedings 14th IEEE Design Automation Conference* , IEEE, pp. 14-18, June 1977.
- [10] Glasser L.A. and Penfield P., "An Interactive PLA generator as an Archetype for a New Design Methodology," *Proceedings IEEE International Conference on Circuits and Computers* , IEEE, pp. 608-611, Oct 1980.
- [11] Vladimirescu A and Liu S., "The Simulation of MOS Integrated Circuits Using SPICE2," *Memo UCB/ERL M80/7* , University of California, Berkeley, pp. 41, Oct 1980.
- [12] Temes G.C and Calahan D.A., "Computer-aided network optimization the state of the art," *Proc. IEEE* , vol. 55, pp. 1832-1863, Nov 1967.
- [13] Director S., "Survey of circuit oriented optimization techniques," *IEEE trans. on Circuit Theory* , vol. CT-18, pp. 3-10, Jan 1971.
- [14] Hatchel G.D. and Rohrer R.A., "Techniques for optimal design and synthesis of switching circuits," *Proc. IEEE* , vol. 55, pp. 1864-1877, Nov 1967.
- [15] Fletcher R. and Powell M.J.D., "A rapidly convergent descent method for minimization," *Computer Journal* , vol. 6, pp. 163-168., June 1963.
- [16] Mokari-Bolhassan M.E. and Trick T.N., "Computer aided Design of Distributed Lumped



Active Networks." *IEEE trans. on Circuit Theory* , vol. CT-18, no. 1, pp. 187-190, Jan 1971.

- [17] Russo P.M and Rohrer R.A., "Computer Optimization of the transient response of an ECL gate," *IEEE trans on Circuit Theory* , vol. CT-18, no. 1, pp. 197-199, Jan 1971.
- [18] Kakihana S. and Hwayanwang P., "Simple CAD Technique to Develop High Frequency Transistors," *IEEE J. Solid State Circuits* , vol. SC-6, no. 4, Aug 1971.
- [19] Essl D.V.J., Mitterer R.W., Rehn B.F. and Domitrowich J.R., "Automated Design Optimization of Integrated Switching Circuits," *IEEE J. Solid State Circuits* , vol. SC-9, no. 1, Feb 1974.
- [20] Lin H.C and Lindholm L.W., "An Optimized Output Stage for MOS Integrated Circuits," *IEEE J. Solid State Circuits* , vol. SC-10, no. 2, April 1975.
- [21] Mohsen A.M. and Mead C. A., "Delay-time Optimization for Driving and Sensing of Signals on High Capacitance Paths of VLSI Systems," *IEEE J. Solid State Circuits* , vol. SC-14, no. 2, April 1979.
- [22] Kang S.M., "A Design of CMOS Polycells for LSI Circuits," *IEEE trans. Circuits and Systems* , vol. CAS-28, no. 8, Aug 1981.
- [23] Glasser L.A., Private communication .
- [24] Koppel A., Shah S. and Puri P., A High Performance Delay Calculation Software System for MOSFET Digital Logic Chips. Private communication. .

APPENDIX A: Sample run of the optimizer program.

Items in curly braces {} are user inputs.

TOPS-20 Command processor 4A(653)  
@wizard

The WIZARD is on your side....

READY>{<cr>}

options are:

'p': get process data

'd': get circuit

's': show process parameters

'g': prepare to optimize

'e': exit from WIZARD

READY>{g}

Using default circuit

there are 5 stages

Rin[UP] = <1.00e + 4> Rin[DWN] = < 1.00e + 4> Cout = <1.00e-13>

using default PROCESS values

SETUP>{<cr>}

options are:

c: edit process parameters

destroys low-level circuit specification

t: prepare to optimize

create low-level circuit specification

g: return to 'OPTIONS>>'

if low-level circuit specification exists

s: show process parameters

p: include parasitics in optimization

n: ignore parasitics " "

k: save process data on mass storage

l: save circuit data " " "

q: quit options

e: exit from WIZARD

SETUP>{t}

ready to optimize for... 0.00e-1 delay

OPTIONS>>{<cr>}

options are:

c: edit user parameters

s: show stage parameters

u: optimize for 'up' input transient

d: " " 'down' " "

a: " " 'average' "

m: " " 'BEST' "  
q: quit options  
e: exit from WIZARD

ready to optimize for... 0.00e-1 delay  
OPTIONS>>{m}

Best optimized delay is for: UP input transient  
Requested delay is: 0.000e-1  
optimized delay is: 2.241e-9  
Other delays are:  
DWN input: 5.076e-9  
AV input: 3.658e-9  
Power: 4.650e-3  
Requested # of iterations: 50  
Actual # of iterations: 11  
Requested convergence tolerance factor: 1.00e-3  
Final convergence tolerance factor: 7.40e-4  
Search tolerance factor: .20  
Optimization time: 3.670e-1 Sec  
Average time per iteration: 3.336e-2 Sec  
Parasitics ignored

Output file: {tty:}  
Output from WIZARD using:  
  datafile: default  
  processfile: default  
optimized for UP input transient  
Requested # of iterations: 50  
Actual # of iterations: 11  
Requested convergence tolerance factor: 1.00e-3  
Final convergence tolerance factor: 7.40e-4  
Search tolerance factor: .20  
Optimization time: 3.67e-1 Sec  
Average time per iteration: 3.34e-2 Sec  
Boundaries are: Rin = 1.00e +4, Cout = 1.00e-13  
Parasitics ignored

pwr = 4.65e-3:UP\_dly = 2.24e-9:DWN\_dly = 5.08e-9:AV\_dly = 3.66e-9

listing by stage, width and length

stage 0: W = 1.63e +1: Lpd = 2.00e +0: Lpup = 6.40e +0  
stage 1: W = 3.32e +1: Lpd = 2.00e +0: Lpup = 6.40e +0  
stage 2: W = 1.71e +1: Lpd = 2.00e +0: Lpup = 6.40e +0  
stage 3: W = 3.64e +1: Lpd = 2.00e +0: Lpup = 6.40e +0  
stage 4: W = 2.00e +1: Lpd = 2.00e +0: Lpup = 6.40e +0

ready to optimize for... 0.00e-1 delay  
OPTIONS>>{e}

@