Modeling and Optimization of CMOS Logic Circuits with Application to Asynchronous Design

by

Maitham Shams

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical Engineering

Waterloo, Ontario, Canada, 1999

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

CMOS remains the mainstream IC technology for the foreseeable future. This thesis addresses modeling and optimization of conventional and differential CMOS logic styles, provides insightful analysis, and derives formulas for optimal transistor sizing of mixed logic-style CMOS circuits. Furthermore, as an application platform, the thesis deals with the less developed area of asynchronous circuits, rather than the commonly used synchronous circuits.

The scope of the modeling and optimization technique presented in this work covers the device, switch, logic, and module levels of abstraction. At the device level, we propose a simple model for evaluating the saturation current of submicron MOS devices. This model reproduces the short-channel characteristics of modern MOS transistors accurately. At the switch level, we recognize and model four types of delays: PMOS rising delay, NMOS falling delay, NMOS rising delay, and PMOS falling delay. The delay models at this level, capture the effect of input signal slope and characterize the behaviour of MOS transistors connected in series. At the logic level, we apply the switch-level delay models to formulate delay macromodels for different CMOS logic styles including conventional, DCVSL, and CPL. We also derive closed-form optimal transistor sizing formulas for several popular CMOS logic styles. At the module level, using the optimal transistor sizing formulas, we demonstrate that it is feasible to optimize the delay of a circuit involving mixed CMOS logic styles.

Part of this work is devoted to comparing different CMOS implementations of logic gates. We develop a fair method for this purpose and study the performance and energy consumption of various conventional and differential CMOS implementations of the C-element and XOR gate, which are the most widely used primitives in asynchronous control circuits. For each primitive, we express our recommendation regarding the most appropriate implementation. We also introduce a differential logic style that has a static memory and, hence, is suitable for implementing primitives such as the C-element.

Finally, a theory of delay optimization evolves from our work that states *the delay in a circuit consisting of conventional CMOS logic gates is minimal if for each stage along the critical path of the circuit, the delay due to that stage as a load equals the delay through that stage as a drive.*

Acknowledgments

Whoever doesn't thank others, hasn't indeed thanked God.

Apostle of God, Mohammad (S)

This thesis would not have been possible without the generous support, patient guidance, and constructive criticism of my supervisors Dr. Mohamed Elmasry and Dr. Jo Ebergen. I am deeply grateful to them, especially for their trust on my work. The multi-dimensional personality of Dr. Elmasry has taught me that it is possible to combine academic excellence with social activities and observation of religion duties. Dr. Ebergen's high standards of clarity, conciseness, and rigour will certainly benefit all my future endeavours.

I would like to thank the other members of my Ph.D. thesis examination committee, Dr. Graham Jullien, Dr. Manoj Sachdev, Dr. Cathy Gebotys, and Dr. Farhad Mavaddat for reading this thesis and for their comments. I would like to extend my thanks to Dr. John Brzozowski for his fruitful remarks in our Maveric Group meetings. I am thankful to the graduate secretary of our department, Wendy Boles, for her friendly and invaluable help. I would also like to thank the computer system administrator of VLSI Research Group, Phil Regier, and the group's secretary, Gehan Sabry, for their prompt assistance.

I greatly appreciate the companionship of many wonderful friends and colleagues who have made my stay in Waterloo a rewarding experience. Although I cannot list all their names, I will definitely remember their favours. Special thanks to Nasser Masoumi and Majid Soleimanipour, my officemates during the last couple of years of my Ph.D. program.

Most importantly, I would like to express my sincere gratitude to my caring parents for their countless blessings, to my brothers and sister for their unconditional love, to my wife Sara for her encouragement and prayers, and to our son Mohammad Amin for all the happiness he has brought to our lives.

Finally, I must admit that I am indebted to many other people in one way or another. To all of them, I would like to say "Thank You."

Maitham Shams

Waterloo, Canada

19 May 1999

In the Name of God,
the Compassionate, the Merciful

To Imam Baqer (A)

Who split the seed of knowledge Widely!

My Ph.D. defense date coincides with the inspiring occasion of the 1363rd birthday of Imam Baqer (A). He is the grandson of Imam Hussain (A), the grandson of Prophet Mohammad (S).

Contents

List of Figures

List of Tables

Nomenclature

Γ	Width ratio of PMOS transistor to NMOS transistor
Λ	NMOS to PMOS driveability ratio
α	Velocity saturation index
β	Tapering factor in buffer chain
ϑ	Captures mobility degradation effect
κ	Technology related parameter
μ	Carrier mobility
ν	MOSFET resistance times unit width
ξ	Captures velocity saturation effect
ρ	NMOS to PMOS driveability ratio
τ	Transition time of input signal
C	Capacitance
C_{ox}	Gate oxide capacitance per unit area
D	Delay
E	Energy
F	Energy-delay product
I	Current
I_D	MOSFET drain current
K_D	Unit falling delay due to transistor gate
K'_D	Unit falling delay due to transistor diffusion
K_E	Unit energy dissipation due to transistor gate
K'_D	Unit energy dissipation due to transistor diffusion
L	MOSFET effective gate length

R Resistance

S Signal slope factor

W MOSFET effective gate width

V_{DD} Power supply voltage

V_T Threshold voltage

X Delay degradation factor for load capacitance due to serially connected MOSFETs

Y Delay degradation factor for internal capacitances due to serially connected MOSFETs

d Diffusion capacitance

g Gate capacitance

m Number of transistor gates related by symmetry

n Number of transistors connected in series

q Number of transistor diffusions related by symmetry

r Width ratio of PMOS transistor to NMOS transistor

Subscripts

C Correspondence to cell or logic gate

D Correspondence to driving logic gate or Delay

L Correspondence to output load

d Correspondence to transistor diffusion

g Correspondence to transistor gate

n Correspondence to NMOS transistor

p Correspondence to PMOS transistor

Accents

$\acute{}$ Correspondence to rising delay

$\grave{}$ Correspondence to falling delay

$\hat{}$ Correspondence total or average delay

Chapter 1

Introduction

Since its inception in the early 1960s, Complementary Metal Oxide Silicon (CMOS) technology has sustained a tremendous evolution in complexity and performance. Today's CMOS microprocessors contain millions of transistors and operate at speeds approaching one GHz. Without the low power consumption of CMOS circuits, it would have been impossible to integrate so many transistors on a die running at such speeds. This low power consumption combined with large noise margins, another intrinsic feature of CMOS circuits, has made portable computers and personal digital assistants a reality. In other words, CMOS is the enabling technology for the modern information age. Moreover, the trend towards higher integration densities and the increasing demands for high-speed and low-power integrated electronics secure the position of CMOS as the mainstream VLSI technology for the years to come.

This dissertation deals with an essential concept in the field of CMOS technology; that is, *modeling and optimization of digital CMOS circuits*. This concept is almost as old as CMOS technology itself and, yet, remains a major focus of research. The thesis, however, attempts to present a more *comprehensive* and *intuitive* perspective of this concept. The thesis covers the modeling and optimization of a variety of CMOS logic styles, provides insightful analysis, and derives closed-form formulas for optimal transistor sizing of digital CMOS circuits. Furthermore, as an application platform, the thesis deals with the less developed area of *asynchronous circuits*, rather than the commonly used synchronous circuits. Although the primary concern of optimization in this work is the *performance* of a design, the *energy* dissipation is carefully observed to avoid wasting resources. In particular, when comparing two designs performing a similar function, both the performance and energy are considered.

This introductory chapter continues with a section on the perspective and motivation of this work. This is followed by some background material organized in three sections. The first one defines basic terminologies like the delay and energy. The second one explains the operation of MOS transistors. And the third background section briefly discusses different CMOS logic styles. Due to the nature of this work and the chosen area of application, it relates to previous work from various areas. A major part of this chapter is devoted to explaining the scope of the thesis in relation to related work from the literature. This chapter concludes with an overview of the chapters of the thesis.

1.1 Perspective and Motivation

Although the literature is abundant with accurate models for evaluating and optimizing the performance of digital CMOS gates, especially the inverter, they are often too complicated for intuitive analysis and quick optimization. To clarify our point, consider Figure 1.1. The figure shows the simplest CMOS logic gate, an inverter, and its environment. This is a basic and the least complicated scenario in a digital circuit. The environment usually consists of other CMOS gates, but the details are irrelevant now. Even for this case, without resorting to circuit simulators and CAD tools, the following problem seems quite challenging.

- Design the gate such that the rising delay is minimal.

Similar problems may be posed for the falling and average delays. Unfortunately, the literature does not have accurate, short, and explicit answers to these problems. Instead, the literature usually offers a solution using mathematical programming in dealing with such problems.

Why is such a simple case important, one may ask, especially when there are CAD tools that handle much more complicated situations? Well, there are a number of good reasons.

1. The basic cases are related to our fundamental knowledge of handling digital circuits. The approach of mathematical programming actively pursued in the literature,

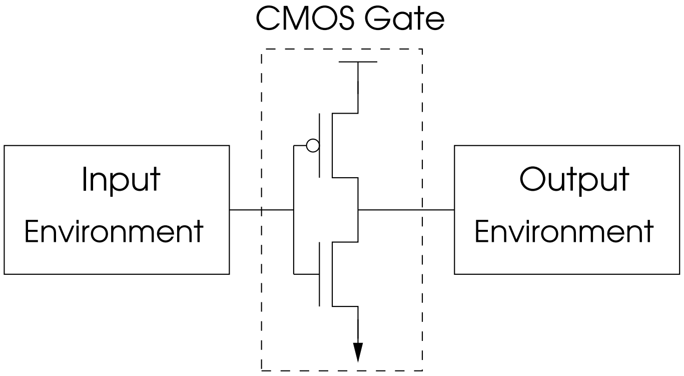


Figure 1.1: A CMOS gate (inverter) and its environment.

although capable of treating circuits with a large number of gates and sometimes inevitable, fails to provide an intuitive understanding of optimal circuit design.

2. If we derive a relation between the optimal sizing of a gate and its input and output environment, we would be able to use the results to optimize a critical path of an arbitrary number of gates by dividing it into smaller sections of three gates each. As far as the literature is concerned, this seems an innovative and quick technique.

3. There is usually more than one CMOS topology to implement a logic function. We may enhance our optimization technique by choosing the right gate topology in addition to the right transistor sizing. To perform a *fair* comparison between different implementations of a function, however, it is necessary to first optimize each implementation for the given environment and then evaluate its performance. So far, this concept of fairness has been ignored in the literature to a large extent.

These points summarize the perspective from which this thesis approaches the problem of modeling and optimization of digital CMOS circuits.

Another concern is that the literature on modeling and optimization of digital CMOS circuits is confined to the conventional CMOS design style. This confinement is in spite of the fact that in many cases unconventional styles may be advantageous in terms of speed or energy dissipation. Models for evaluation and optimization of unconventional CMOS styles enables mixing conventional and unconventional gates for optimizing the overall performance of a circuit. The reader may think of unconventional CMOS styles as implementations not resembling the structure of the inverter in Figure 1.1, which is a conventional CMOS implementation.

For the applications of this work, we have turned our attention towards the area of asynchronous design, because this area has become increasingly promising. In synchronous circuits, it is vitally important to simultaneously supply all components of a system with a clock edge. In the view of demands for even higher speeds and chip densities, this is becoming an increasingly challenging task. Since the late 1980s, the VLSI community has been exerting an intensifying effort on designing asynchronous circuits that, unlike synchronous circuits, operate without a global clock. Asynchronous circuits may also be beneficial in terms of performance, power dissipation, noise immunity, electromagnetic compatibility, and ease of interfacing. A separate chapter details the motivations for asynchronous circuit design.

In short, this work deviates from the current literature in the following aspects.

- *Approach*: Provides an intuitive understanding and explicit models leading to closed-form formulas.
- *Comprehensiveness*: Covers both the conventional and unconventional CMOS styles, such as DCVSL and CPL.
- *Application*: Investigates optimal implementations of asynchronous primitives and circuits.

1.2 Basic Terms and Definitions

This section introduces and defines the basic mathematical expressions for the terms performance, delay, energy, and power, which are frequently used throughout this thesis.

The performance is defined as the inverse of the propagation delay, also called the average delay $\bar{D}$, which is expressed in terms of the rising delay $\acute{D}$ and the falling delay $\grave{D}$.

$$\bar{D} = \frac{\acute{D} + \grave{D}}{2} \tag{1.1}$$

The falling and rising delays are defined as the time interval between the middle of the input voltage swing ($V_{DD}/2$) and the middle of the output voltage swing ($V_{DD}/2$), when the output signal is falling and rising, respectively. In CMOS circuits, the power supply voltage is usually denoted by V_{DD}. We often find the notion of total delay $\hat{D}$ more convenient than the average delay. The total delay signifies the delay over one *switching cycle* of the output. A switching cycle consists of a rising and a falling transition.

$$\hat{D} = \acute{D} + \grave{D} \tag{1.2}$$

The delay is always associated with charging or discharging capacitances. The general delay equation states that the delay for charging an initially empty capacitance C to a potential V with an average current I equals

$$D = \frac{C\ V}{I} \tag{1.3}$$

This equation also governs the discharging delay of C from an initial potential V to ground. In transistor circuits, (1.3) can be directly applied by substituting V with $V_{DD}/2$, if the input signal of a gate is a step waveform. The delay in response to such an input is called the *step delay*. In practice, however, a circuit signal exhibits an exponential behaviour and is best approximated by a ramp waveform. The delay in response to a ramp input is called the *ramp delay*. Deriving a general model for the ramp delay has proven complicated. Under certain restrictive, but realistic, assumptions the ramp delay of a gate can be expressed as a linear combination of two step delays, the step delay of the gate itself and the step delay of the preceding gate.

The power dissipation is the energy per unit of time. Therefore, the average power dissipation P over a period T is given by

$$P = \frac{E}{T} = \frac{V\ Q}{T} = V\ I \tag{1.4}$$

where E is the total amount of energy dissipated, Q is the total charge consumed, V is the average voltage, and I is the average current. Accordingly, if the energy consumed by a certain task is reduced in proportion to the time taken to perform the task, the power remains constant, while obviously fewer number of electric charges are spent. The power is only a measure of expenditure per second, but doesn't give any indication of what is accomplished for the cost. This can be better understood in terms of the energy per task. Thus, we prefer to use the term energy, which implies the energy dissipation per switching cycle.

Like the delay, the energy is also associated with capacitances. The energy for charging an initially empty capacitance C to a potential V through a power supply voltage V_{DD} is expressed by

$$E = Q\ V_{DD} = C\ V\ V_{DD} \tag{1.5}$$

Since no energy is spent for discharging C, (1.5) represents the energy per switching cycle of C as well. It is important to make a distinction between the energy stored in C and the energy consumed for charging C given by (1.5). The energy stored in C when charged to V can be formulated by

$$E_C = \int_0^V Q\ dV_C = \int_0^V C\ V_C\ dV_C = \frac{1}{2}\ C\ V^2$$

where V_C represents the potential across C. Therefore, E is always larger than E_C. Since the energy is conserved, the amount of energy $E - E_C$ must have been dissipated in the form of heat by the resistance connecting the power supply to C. The amount of electrical energy converted to heat is, interestingly, independent of the value of this resistance. In a similar fashion, it can be deduced that during the discharging process, the total energy stored in C is also converted to heat by the resistance connecting C to ground. Another point worth mentioning is that there are two types of energy loss in transistor circuits: dynamic and static. The dynamic energy is only dissipated when the nodal capacitances are switching, whereas the static energy is continuously consumed as long as the circuit is supplied with power. Nevertheless, the relative magnitude of the static energy is so small that it is always ignored during switching. Hence, (1.5) only applies to the dynamic energy in circuits.

1.3 MOSFET Operation

In a MOSFET the channel current is modulated through the gate voltage. The conventional MOSFET model proposed by Shockley in 1952 [87], when device channels exceeded 10 μm, is summarized in Table 1.1. This model is still used in textbooks for hand-analysis of MOS circuits [72, 102]. It is, however, very inaccurate in predicting the behaviour of today's submicron short-channel MOS devices. We use the model here to introduce a number of parameters.

Table 1.1: Shockley Model for an MOS transistor.

Operation Mode	Current	Voltages
Cutoff	$I_D = 0$	$\|V_{GS}\| \leq \|V_T\|$
Linear	$I_D = (\frac{W}{L})\ \mu\ C_{ox}\ [(V_{GS} - V_T)\ V_{DS} - \frac{1}{2}\ V_{DS}^2]$	$0 < \|V_{DS}\| < \|V_{GS} - V_T\|$
Saturation	$I_D = \frac{1}{2}\ (\frac{W}{L})\ \mu\ C_{ox}\ (V_{GS} - V_T)^2$	$0 < \|V_{GS} - V_T\| < \|V_{DS}\|$

Parameters L and W are the effective device length and width respectively, μ is the mobility of the carriers in the channel, and C_{ox} is the gate oxide capacitance per unit area given by

$$C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}} \tag{1.6}$$

where ε is the permittivity of the gate oxide and t_{ox} is the gate thickness. Often, the process-dependent factors are combined into $k = \mu C_{ox}$. Then, the transistor gain factor β can be expressed in terms of technology and geometry factors as

$$\beta = k\ \frac{W}{L} \tag{1.7}$$

The expressions given in Table 1.3 are equally valid for N- and P-type devices when V_T is replaced with V_{TN} and V_{TP}, respectively; except that the negative sign of the P-type threshold voltage must be accounted for. According to Shockley's model, the operation of an NMOS transistor is as follows. If a voltage greater than some threshold, V_{TN}, is applied to the gate, the substrate surface beneath the gate is inverted and an N-type channel is formed. No current passes through the channel if the source and drain are both grounded (cutoff mode). However, as the drain-to-source voltage V_{DS}, is raised, an almost linearly proportional current I_{ds} is established (linear mode). When V_{DS} is increased such that V_{GD}

falls below V_{TN}, the channel no longer reaches the drain and is pinched-off. In this case, channel electrons are injected into the drain and the current is controlled by the gate voltage alone, independent of the drain voltage (saturation mode). The voltage across the pinched-off channel also remains fixed at $V_{GS} - V_{TN}$ regardless of the drain voltage. Shockley's Model is particularly unreliable in reproducing the saturation mode behaviour of short-channel devices. For an intuitive understanding of the so called short-channel effects, the reader may refer to [76, 95].

1.4 CMOS Logic Styles

A PMOS transistor is a good transmitter of a logic 1 and a weak transmitter of a logic 0. An NMOS transistor, on the other hand, is a good transmitter of a logic 0 and a weak transmitter of a logic 1. Therefore, PMOS transistors are normally used in the pull-up sections of CMOS logic gates and NMOS transistors are normally used in the pull-down sections of CMOS logic gates. This original style of implementing a CMOS logic gate is termed *conventional*. Some other popular CMOS logic styles, however, use transistors of the same type to play both pull-up and pull-down roles. These CMOS logic styles are generally referred to as *unconventional*. The conventional logic style is still the most widely practiced, because it is more familiar, easier to automate and, most importantly, offers a good balance of performance and energy dissipation. Nevertheless, using an unconventional logic style is sometimes beneficial in reducing the device count for implementing a function which, in turn, may improve the delay, energy consumption, and area. There is a variety of unconventional CMOS logic styles. Occasionally, a logic function may even have more than one conventional CMOS implementation, like that of the C-element studied in later chapters of this thesis. Within the conventional category, we refer to the CMOS implementation obtained from a Boolean function through the standard procedure outlined in text books like [102] as the *standard* CMOS implementation of a logic gate. Figure 1.2 illustrates the schematics of a conventional and two unconventional CMOS implementations of the XOR gate. The primary inputs of the gate are denoted by a and b and their complements are denoted by a' and b', respectively. Similarly, c and c' denote the output of the gate and its complement, respectively.

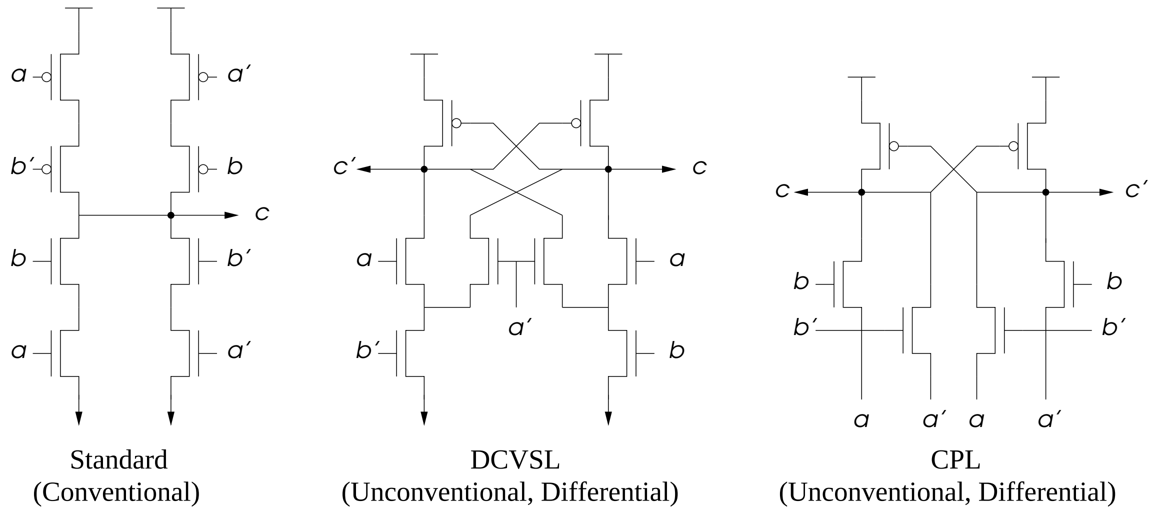


Figure 1.2: CMOS implementations of XOR in different logic styles.

A dominant group of unconventional CMOS logic gates belong to the category of *differential* logic styles. A differential logic gate usually has a symmetrical structure and requires both the input signals and their complements. In return, it produces the output and its complement. Although differential logic circuits typically double the wiring requirements, they counterbalance this deficiency by lowering the device count. The most well-known differential CMOS logic families are Differential Cascade Voltage Switch Logic (DCVSL) [42] and Complementary Pass-Transistor Logic (CPL) [105]. The unconventional implementations of the XOR gate depicted in Figure 1.2 belong to these two logic styles.

A DCVSL gate consists of a network of NMOS transistors and a couple of cross-coupled PMOS transistors forming a dynamic latch. When one of the outputs is pulled down by the NMOS network, the other output is pulled up through the corresponding PMOS transistor. Hence, unlike the case of a conventional gate, the NMOS network of a DCVSL gate is partially involved in the pull-up process. DCVSL was first introduced in [42]. Later, [16] described a design procedure for DCVSL circuits, and [17] presented a comparison between conventional and DCVSL full-adder circuits. DCVSL is, especially, very efficient in designing full-adders [73]. DCVSL gates are used in asynchronous circuits as completion signal detectors [61].

The switching activity of a CPL gate is entirely controlled by a network of NMOS transistors. A CPL gate usually uses a minimum size PMOS latch to avoid excessive static energy dissipation. CPL was first introduced in [105], where it was incorporated to design a fast multiplier circuit. Design issues regarding CPL circuits are discussed in [69]. Double Pass-transistor Logic (DPL) is another differential CMOS logic style, which has a similar structure to CPL. Compared to CPL, DPL doubles the number of transistors to achieve higher performance [94]. We study CPL, DPL, and similar gate topologies under the category of pass-transistor logic (PTL). By PTL we refer to the general category of CMOS circuits in which signals, not necessarily V_{DD} or ground, are passed from their inputs to their outputs through a chain of MOS transistors.

1.5 Scope of Thesis Based on Abstraction Levels and Related Work

Digital circuits are studied at different abstraction levels. Commonly used abstraction levels in digital circuits are, in order of increasing abstraction, the device, switch (or circuit), logic (or gate), module (or functional block), and system levels. Although there is a generally shared understanding of these abstraction levels, there is no consensus on a precise definition for each level. As indicated, more than one term may refer to the same abstraction level. We have noted our preference by enclosing the terms we find more ambiguous within parentheses. For example, we prefer the term switch level over circuit level, because circuit is a very general term and may have different meanings depending on the context. We also like to avoid using the term gate level, because gate refers to one of the three terminals of an MOS transistor as well as a logic circuit primitive.

The scope of the modeling and optimization technique presented in this work covers a few abstraction levels extending from the device level to the module level. Nevertheless, the study of the details is not equally distributed among the levels, and most of the effort is focused at the logic level. This section outlines our method related to the device, switch, logic, and module levels of abstraction. Our contribution at each abstraction level is also stated. We have followed a bottom-up approach in our modeling, such that the results obtained at one level are incorporated into the next higher abstraction level. Figure 1.3 illustrates the concepts studied at each abstraction level. The reader is asked to refer to the

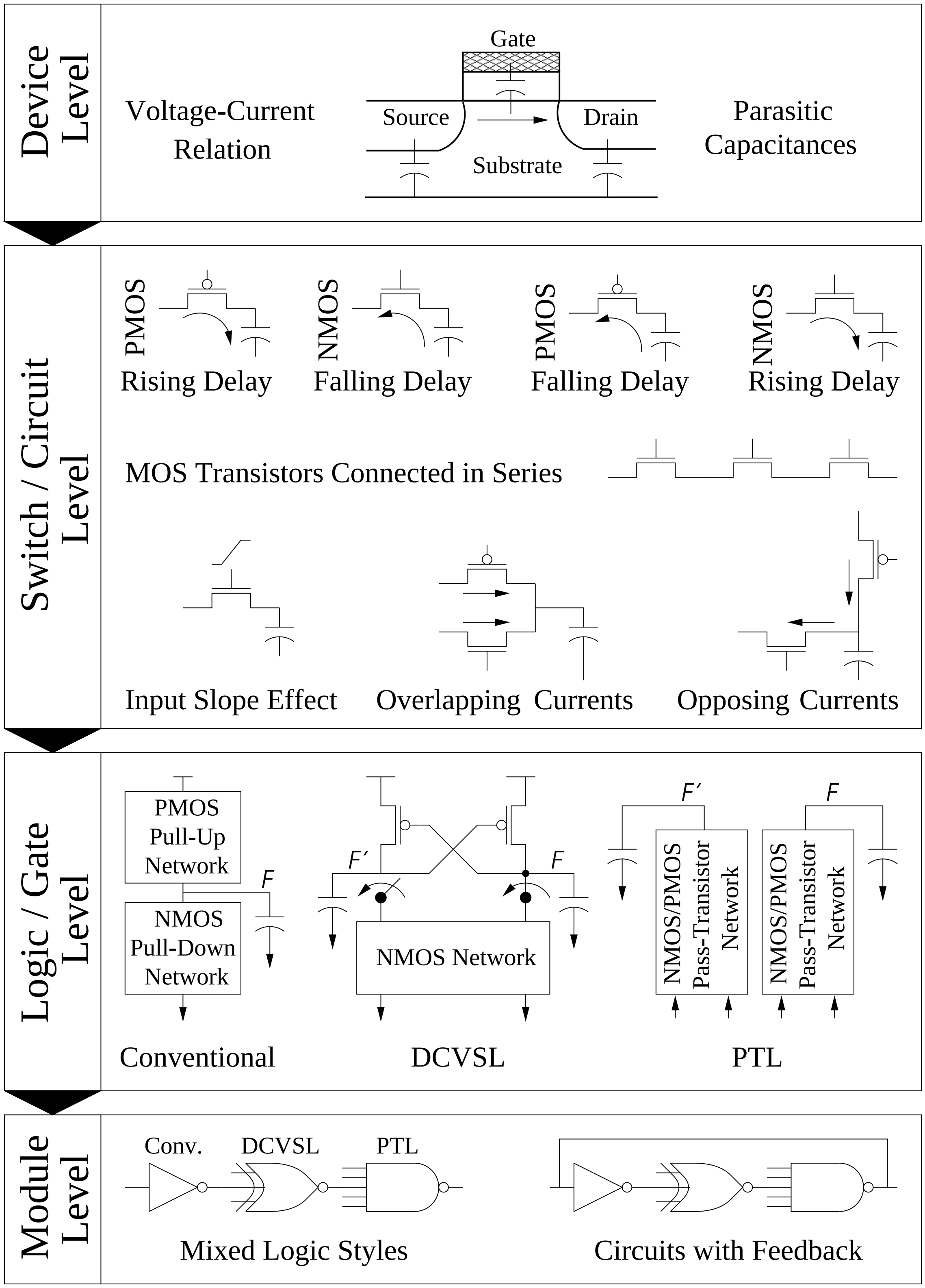


Figure 1.3: Scope of our modeling technique presented in this thesis.

corresponding part of this figure when scanning the following subsections.

1.5.1 Device Level

The basic delay equation (1.3) suggests that accurate estimation of the current and capacitances is essential in predicting the delay. Both of these delay components are related to the device level of abstraction.

In a delay calculation, usually the saturation current of an MOS transistor is of concern. With the advent of IC technology, resulting in small device geometries, the conventional MOSFET model of Shockley [87] is no longer valid. Shockley's model doesn't include short-channel effects, such as *velocity saturation* and *mobility degradation*; hence, new MOSFET models have been proposed. A modified version of Shockley's MOSFET model, which accounts for *channel length modulation*, is used in SPICE as the elementary Level 1 MOSFET model. Higher level SPICE MOSFET models, including Level 13 used in our simulations, are based on the semi-empirical model known as BSIM [86]. This model is too complicated for hand calculation and intuitive analysis. A simpler model is proposed in [95] that gives an intuitive understanding of a number of parameters. Sakurai and Newton have developed the so called α-power law for short-channel MOSFETs [76–78]. The α-power model is popular due to its simplicity and similarity to Shockley's model. Motivated by the considerable mismatch between HSPICE simulations and the α-power law, we introduce another model for the saturation current of submicron MOS transistors. The new model shows a high precision in reflecting short-channel effects. Our MOSFET saturation current model uses three empirical parameters, which are extracted with HSPICE simulations.

MOSFET capacitances consist of several types and can be categorized into the total effective gate capacitance and the total effective capacitances at the source and the drain. All of these capacitances are non-linear functions of the bias voltage. Therefore, the values for a rising transition are different from a falling transition. Although there are simple expressions for calculating these capacitances, they are not very accurate [72, 102]. The more accurate expressions, however, are complicated and suitable for CAD tools. We present a method for extracting these capacitances for rising and falling transitions with the aid of HSPICE simulations. This assures that the average effect of junction overlap capacitances is also taken into account.

1.5.2 Switch Level

At the switch level we study the switching delay of a number of transistor circuit scenarios that are frequently encountered in CMOS logic styles. Most importantly, in order to cover different CMOS logic styles we define and accommodate four types of switching delays: 1) Rising delay through a PMOS transistor, 2) Falling delay through an NMOS transistor, 3) Falling delay through a PMOS transistor, and 4) Rising delay through an NMOS transistor. Expressions for these delays are derived by applying the current model, developed at the device level, to each case. The distinction between these delays and the consequent modeling of each has not been addressed in the literature before.

Another concern at this level is the delay through MOS transistors connected in series. The usual practice is to use the well known theory of Elmore [35] to approximate the delay in such cases. First, we present an RC model of serially connected MOS transistors. Then, we apply Elmore's theory in conjunction with some simulation results to come up with a new expression for the delay of these structures. The delay expression has two components; one related to the capacitances of the internal nodes and one related to the output capacitance. A particularly important conclusion of this part of the investigation is that the delay optimization of similarly sized, serially connected MOS transistors is independent of the internal capacitances.

The delay of a gate is affected by the finite slope of its input signal. To simplify the delay modeling, the input signal is usually approximated by a ramp waveform. Hedenstierna and Jeppson [40] were the first to suggest that if the input slope exceeds one-third of the output slope, then the delay of a gate can be obtained by adding a fraction of the input rise or fall time to the step delay of the gate. This idea has been well received by circuit designers and CAD tool developers. The delay expression derived in [40] is based on the characteristics of long channel MOS devices which are now outdated. Sakurai and Newton [76] generalized that delay expression to comply with the behaviour of today's short-channel MOS devices. We demonstrate the extension of this concept to the four delay types introduced earlier.

The combination of the step delay, input slope factor, and the effect of serial connections suffices for the delay modeling of conventional CMOS gates. In order to cover non-conventional CMOS gates as well, we introduce additional delay models for the cases where the currents of two branches in a circuit overlap at a node or oppose each other. These cases are schematically illustrated in Figure 1.3. A prominent example of the overlapping currents situation is a CMOS transmission gate. The case of opposing currents, on the other hand, is most evident in DCVSL gates.

1.5.3 Logic Level

The majority of the modeling, optimization, and application work presented in this thesis concerns the logic level of abstraction. We present two methods of modeling and optimization at this level: One which employs simplifying approximations and another which is considerably more accurate and rigorous. The former is restricted to conventional CMOS gates, and its application to modeling and comparing conventional CMOS implementations of an important asynchronous circuits primitive, the C-element, is detailed. The latter is more comprehensive, because it is systematically developed based on the previous levels of abstraction. This formulation is applied to optimization and comparison of conventional and unconventional CMOS implementations of the XOR gate, which is a widely used primitive in asynchronous and synchronous circuits.

Many related publications at this level have treated the characterization of the conventional CMOS inverter in detail; for some recent ones the reader may refer to [7,23,31]. Often, it is not clear how these delay models can be generalized to cover more complex conventional CMOS gates. Separate papers, like [66], have dealt with the issue of generating an inverter equivalent for conventional CMOS gates. In addition, most of the published delay models express the delay in terms of a load capacitance from which the effect of changing the sizes of the transistors is not immediately clear to a designer. Circuit optimization work that is specifically related to the logic level of abstraction is mainly concerned with optimal buffer design for driving large loads [15, 53, 98]. We demonstrate that this concept is a special case of our general formulation. In contrast to conventional CMOS logic, the literature seems to lack any substantial work dealing with the delay modeling of unconventional CMOS logic styles. Therefore, one part of our contribution at the logic level can be summarized as follows: developing delay models for CMOS logic styles, including unconventional ones, that are explicit in terms of the width of the transistors and that lead to closed-form formulas for the delay optimization of CMOS gates implemented in any style. The formulas relate the optimal transistor sizing of a gate to the size of its succeeding, loading gate and the size of its preceding, driving gate.

Another part of our work at the logic level concerns developing methodologies for performing a fair comparison between different implementations of a logic gate. These methodologies are exemplified by applying them to the implementations of the C-element and the XOR gate. The message is that, since the performance of a gate is characterized by its operating environment, one should optimize the transistor sizing of each implementation for the given environment before evaluating and comparing their performance. This has not been strictly observed in the literature, even in recent publications such as [109]. Hence, one may question the credibility of the results and the suggestions reported in papers comparing different logic styles [18, 52, 59, 94, 105]. The difficulty of performing such a fair comparison is that it requires exhaustive simulations, since the available optimization tools usually do not handle unconventional CMOS logic styles. Our formulation greatly facilitates this task.

When examining the CMOS implementations of the C-element, we realized that all reported designs belonged to the conventional family. We introduce a differential CMOS logic style (DIL) which is similar to DCVSL, but uses an inverter latch instead of a PMOS latch. Thus, DIL has an inherent static memory. This property of DIL makes it suitable for implementing primitives like the C-element. Part of this work discusses the cons and pros of the DIL C-element and a number of modified versions of it.

The delay and energy for DCVSL and DIL gates are affected by the race between the pull-up PMOS transistor and the pull-down NMOS network during output switching. Neglecting the race problem results in considerable underestimation of the delay and energy. A method of calculating the delay and energy of DCVSL gates that captures the race problem had not been reported before. A similar type of race problem also exists in conventional CMOS logic styles using an inverter latch at the output. This is also addressed and treated in the thesis.

1.5.4 Module Level

Delay modeling and optimal transistor sizing eventually targets the module and system levels of abstract. Circuit optimization at this level has been actively studied since the late 1970's [75] for over two decades [68]. A number of techniques from the 1980's express the delay with posynomials and solve them using geometric programming or heuristic approaches [36], [90]. However, the delay models in these earlier works, including [75], [19], and [41], do not accommodate the effect of the finite waveform slopes. Gate sizing is formulated as a nonlinear programming problem in [19] and [41]. In [45], although the delay models include the input slope effect, the accuracy of the optimization is somewhat compromised by assuming a fixed size ratio of PMOS transistors to NMOS transistors within a gate. A convex programming formulation solved by an interior point method is used for the problem of gate sizing in [79]. In [25], a gate sizing algorithm is developed based on a table lookup nonlinear delay model. More recently, an approach for minimizing total power dissipation under delay constraint is presented in [68]. In general, the literature on this subject is confined to conventional CMOS logic styles and evolves around the method of mathematical programming, which seems inevitable for constrained delay optimization.

This thesis follows a different approach in delay estimation and optimal transistor sizing at the module level. The technique evolves naturally from our delay models and optimization formulas developed at the logic level. At the logic level, we have derived a set of delay models and optimal transistor sizing formulas for the popular CMOS logic styles. This enables us to deal with modules consisting of mixed logic styles. For delay estimation along a path, we simply add the delay equations for the gates on that path. For delay optimization along a critical path, we solve a set of nonlinear equations by iteration. This method is more convenient, faster, and more intuitive than mathematical programming. Furthermore, it doesn't show any convergence problem. We investigate the optimization of modules with and without feedback loops. Loop structures play a significant role in asynchronous pipelines.

We should mention that the only work which, to some extent, shares our outlook is the so called theory of logical effort by Sutherland and Sproull [91, 93]. Their method, however, is limited to standard CMOS logic gates, uses a uniform PMOS to NMOS size ratio for all logic gates, assumes rising and falling delays are equal and, in general, does not support branching within a path. Moreover, they use a delay model which does not include signal slope effect and a short-channel view of series-connected transistors.

A theory of delay optimization evolves from our work that states "the delay in a circuit consisting of conventional CMOS logic gates is minimal if for each stage along the critical path of the circuit, the delay due to that stage as a load equals the delay through that stage as a drive". This theory also generally holds for logic gates which experience no cases of overlapping and opposing currents. We show that the theory is valid even when the input slope factor and the effect of serial connection of transistors are taken into account. Moreover, branches and spurious capacitances along the path do not void the theory.

1.6 Thesis Overview

The order at which the chapters of the thesis appear follows the natural course of the development of this work, rather than the abstraction levels presented in Section 1.5. Figure 1.4 illustrates the organization of the chapters in the thesis and indicates the relation between them.

Chapter 2 is entirely devoted to an overview of asynchronous circuit design and covers motivations, models, methodologies, and design techniques. Chapter 3 presents a formulation for quick evaluation and optimization of conventional CMOS circuits and its applications to an inverter chain, tapered buffer design, and generation of complementary signals. Chapter 4 applies the model derived in Chapter 3 to single-rail implementations of the C-element and identifies the most qualified implementation for a high-performance, energy-efficient design environment. Chapter 5 introduces the DIL logic style and compares the DIL C-element and its modified versions with the conventional implementations of the C-element. The chapter demonstrates cases where an asynchronous pipeline using differential C-element gates may fail due to the divergence of complementary signals. A more comprehensive, rigorous, and accurate delay modeling technique starts from Chapter 6, which introduces a model for the saturation current of MOS transistors. This chapter also deals with the delay modeling issues related to the device and switch levels of abstraction. Chapter 7 incorporates the arguments of Chapter 6 to derive delay models and optimal transistor sizing formulas

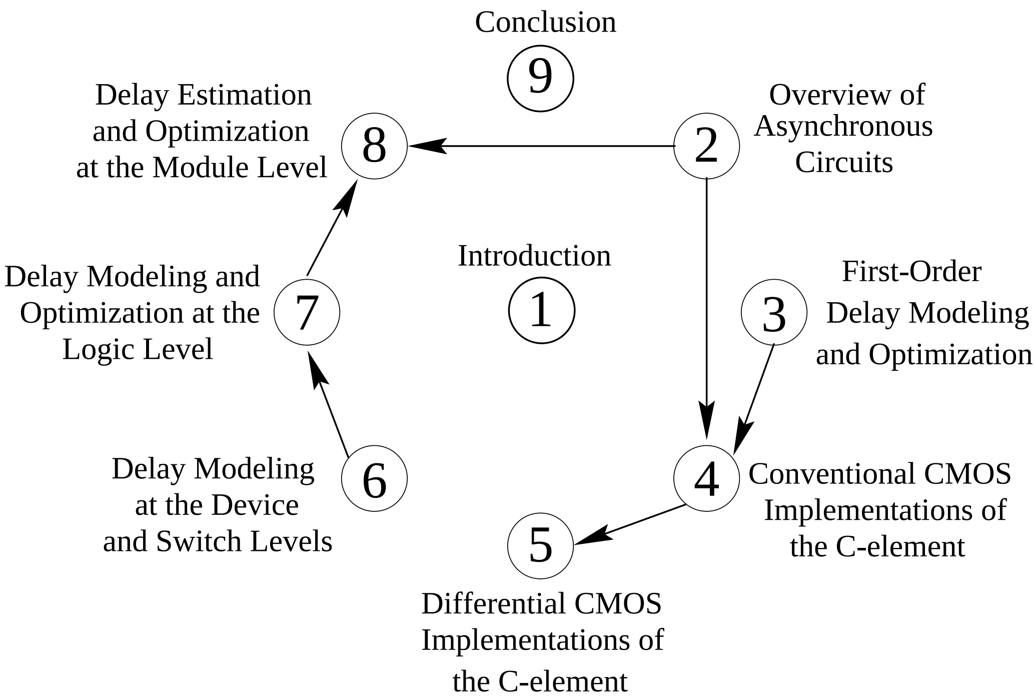


Figure 1.4: Organization of the chapters and their relations in the thesis.

for various conventional and differential CMOS logic styles. The same chapter includes a comparison between the optimized CMOS implementations of the XOR gate. Chapter 8 presents an optimization technique for modules with and without feedback loops. Chapter 9 concludes the thesis with a summary of the contributions and a list of directions for future research.

Chapter 2

Asynchronous Circuits

Digital VLSI circuits are usually classified into synchronous and asynchronous circuits. Synchronous circuits are generally controlled by global synchronization signals provided by a clock. Asynchronous circuits, on the other hand, do not use such global synchronization signals. Between these extremes there are various hybrids. Digital circuits in today's commercial products are almost exclusively synchronous. Despite this big difference in popularity, there are a number of reasons why asynchronous circuits are of interest.

In this chapter, we present a brief overview of asynchronous circuits [1]. First we address some of the motivations for designing asynchronous circuits. Then, we discuss different classes of asynchronous circuits and briefly explain some asynchronous design methodologies. Finally, we present a typical asynchronous design in detail.

2.1 Motivations for Asynchronous Circuits

Throughout the years researchers have had various reasons for studying and building asynchronous circuits. Some of the often mentioned advantages of asynchronous circuits are speed, low energy dissipation, modular design, immunity to metastable behavior, freedom from clock skew, and low generation of and low susceptibility to electromagnetic interfer-

[1] This chapter is based on our invited article on the topic of asynchronous circuits for the Encyclopedia of Electrical and Electronics Engineering published by John Wiley [84].

ence. We elaborate here on some of these potentials and indicate when they have been demonstrated through comparative case studies.

2.1.1 Speed

Speed has always been a motivation for designing asynchronous circuits. The main reasoning behind this advantage is that synchronous circuits exhibit worst-case behavior, whereas asynchronous circuits exhibit average-case behavior. The speed of a synchronous circuit is governed by its clock frequency. The clock period should be large enough to accommodate the worst-case propagation delay in the critical path of the circuit, the maximum clock skew, and a safety factor due to fluctuations in the chip fabrication process, operating temperature, and supply voltage. Thus, synchronous circuits exhibit worst-case performance. This worst-case behavior is dictated by the global clock and, in spite of the fact that the worst-case propagation in many circuits, particularly arithmetic units, is improbable and may be much longer than the average-case propagation.

Many asynchronous circuits are controlled by local communications and are based on the principle of initiating a computation, waiting for its completion, and then initiating the next one. When a computation is completed early, the next computation can start early. For this reason, the speed of asynchronous circuits equipped with completion-detection mechanisms depend on the computation time of the data being processed, not the worst-case timing. Accordingly, such asynchronous circuits exhibit average-case performance. An example of an asynchronous circuit where the average-case potential is nicely exploited is reported in [103], an asynchronous divider that is twice as fast as its synchronous counterpart. Nevertheless, to date, there are few concrete examples demonstrating that the average-case performance of asynchronous circuits is higher than that of synchronous circuits performing similar functions. The reason is that the average-case performance advantage is often counterbalanced by the overhead in control circuitry and completion-detection mechanisms.

Besides demonstrating the average-case potential, there are case studies in which the speed of an asynchronous design is compared to the speed of a corresponding synchronous version. Molnar et al. report a case study of an asynchronous FIFO that is every bit as fast as any synchronous FIFO using the same data latches [64]. Furthermore, the asynchronous

FIFO has the additional benefits that it operates under local control and is easily expandable. At the end of this chapter we give an example of a FIFO with a slightly different control circuit.

2.1.2 Immunity to Metastable Behavior

Any circuit with a number of stable states also has metastable states. When such a circuit gets into a metastable state, it can remain there for an indefinite period of time before resolving into a stable state [12, 54]. Metastable behavior occurs, for example, in circuit primitives that realize mutual exclusion between processes, called *arbiters*, and components that synchronize independent signals of a system, called *synchronizers*. Although the probability that metastable behavior lasts longer than period t decreases exponentially with t, it is possible that metastable behavior in a synchronous circuit lasts longer than one clock period. Consequently, when metastable behavior occurs in a synchronous circuit, erroneous data may be sampled at the the computation time of the clock pulses. An asynchronous circuit deals gracefully with metastable behavior by simply delaying the computation until the metastable behavior has disappeared and the element has resolved into a stable state.

2.1.3 Modularity

Modularity in design is an advantage exploited by many asynchronous design styles. The basic idea is that an asynchronous system is composed of functional modules communicating along well-defined interfaces. Composing asynchronous systems is simply a matter of connecting the proper modules with matching interfacial specifications. The interfacial specifications describe only the sequences of events that can take place and do not specify any restrictions on the timing of these events. This characteristic reduces the design time and complexity of an asynchronous circuit, because the designer does not have to worry about the delays incurred in individual modules or the delays inserted by connection wires. Designers of synchronous circuits, on the other hand, often pay considerable attention to satisfying the detailed interfacial timing specifications.

Besides ease of composability, modular design also has the potential for better technology migration, ease of incremental improvement, and reuse of modules [91]. Here the idea is that an asynchronous system adapts itself more easily to the advances in technology. The obsolete parts of an asynchronous system can be replaced with new parts to improve system performance. Synchronous systems cannot take advantage of new parts as easily, because they must be operated with the old clock frequency or other modules must be redesigned to operate at the new clock frequency.

One of the earliest projects that exploited modularity in designing asynchronous circuits is the Macromodules project [21]. Another nice example where modular design has been demonstrated is the TANGRAM compiler developed at Philips Research Laboratories [6].

2.1.4 Low Power

Due to rapid growth in the use of portable equipment and the trend in high-performance processors towards unmanageable power dissipation, energy efficiency has become crucial in VLSI design. Asynchronous circuits are attractive for energy-efficient designs, mainly because of the elimination of the clock. In systems with a global clock, all of the latches and registers operate and consume dynamic energy during each clock pulse, in spite of the fact that many of those latches and registers might not have new data to store. There is no such waste of energy in asynchronous circuits, because computations are initiated only when they need to be done.

Two notable examples that demonstrated the potential of asynchronous circuits when in energy-efficient design are the work done at Philips Research Laboratories and at Manchester University. The Philips group designed a fully asynchronous digital compact-cassette (DCC) error detector which consumed 80% less energy than a similar synchronous version [4]. The AMULET group at Manchester University successfully implemented an asynchronous version of the ARM microprocessor, one of the most energy-efficient synchronous microprocessors. The asynchronous version achieved a power dissipation comparable to the fourth generation of ARM, around 150 mW [37], in a similar technology.

Recently, power management techniques are being used in synchronous systems to turn the clock on and off conditionally. However, these techniques are only worthwhile implementing at the level of functional units or higher. Besides, the components that monitor the environment for switching the clock continue dissipating energy.

It is also worth mentioning that unlike synchronous circuits, most asynchronous circuits do not waste energy on *hazards*, which are spurious changes in a signal. Asynchronous circuits are essentially designed to be hazard-free. Hazards can be responsible for up to 40% of energy loss in synchronous circuits [11].

2.1.5 Freedom from Clock Skew

Because asynchronous circuits generally do not have clocks, they do not have many of the problems associated with clocks. One such problem is *clock skew*, the technical term for the maximum difference in clock arrival time at different parts of a circuit. In synchronous circuits, it is crucial that all modules operating with a common clock receive this signal simultaneously, that is, within a tolerable period of time. Minimizing clock skew is a difficult problem for large circuits. Various techniques have been proposed to control clock skew, but generally they are expensive in terms of silicon area and energy dissipation. For instance, the clock distribution network of the DEC Alpha, a 200 MHz microprocessor at a 3.3 V supply, occupies 10% of the chip area and has a 40% share in the total chip power consumption [30]. Although asynchronous circuits do not have the clock skew problem, they have their own set of problems in minimizing the overhead needed for synchronization among the parts.

2.2 Models and Methodologies

There are many models and methodologies for analyzing and designing asynchronous circuits. Asynchronous circuits can be categorized by the following criteria: signaling protocol and data encoding, underlying delay model, mode of operation, and formalism for specifying and designing circuits. This section presents an informal explanation of these criteria.

2.2.1 Signaling Protocols and Data Encodings

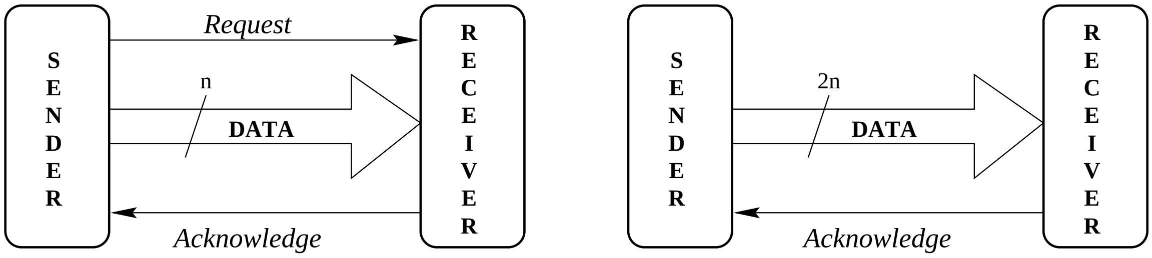
Modules in an asynchronous circuit communicate data with some signaling protocol consisting of request and acknowledgment signals. There are two common signaling protocols for communicating data between a sender and a receiver: the four-phase and the two-phase protocol. In addition to the signaling protocol, there are different ways to encode data. The most common encodings are single-rail and dual-rail encoding. We explain the two signaling protocols first and then discuss the data encodings.

If the sender and receiver communicate through a *two-phase signaling* protocol, then each communication cycle has two distinct phases. The first phase consists of a request initiated by the sender. The second phase consists of an acknowledgment by the receiver. The request and acknowledgment signals are often implemented by voltage transitions on separate wires. No distinction is made between the directions of voltage transitions. Both rising and falling transitions denote a signaling event.

The *four-phase signaling protocol* consists of four phases: a request followed by an acknowledgment, followed by a second request, and finally a second acknowledgment. If the request and acknowledgment are implemented by voltage transitions, then at the end of every four phases, the signaling wires return to the same voltage levels as at the start of the four phases. Because the initial voltage is usually zero, this type of signaling is also called *return-to-zero signaling*. Other names for two-phase and four-phase signaling are *two-cycle* and *four-cycle signaling*, respectively, or *transition* and *level signaling*, respectively.

Both signaling protocols can be used with single and dual-rail data encodings. In *single-rail data encoding* each bit is encoded with one wire, whereas in *dual-rail encoding*, each bit is encoded with two wires.

In single-rail encoding, the value of the bit is represented by the voltage on the data wire. When communicating n data bits with a single-rail encoding, during periods where the data wires are guaranteed to remain stable, we say that the data are *valid*. During periods where the data wires are possibly changing, we say the data are *invalid*. A two-phase or four-phase signaling protocol is used to tell the receiver when data are valid or invalid . The sender informs the receiver about the validity of the data through the request signal, and the receiver, in turn, informs the sender of the receipt of the data through the acknowledgment signal. Therefore, to communicate n bits of data, a total number of $(n+2)$ wires are necessary between the sender and the receiver. The connection pattern for single-rail encoding and two or four-phase signaling is depicted in Figure 2.1(a).



(a) Bundled Data Convention (b) Dual-Rail Data Encoding

Figure 2.1: Two different data communication schemes

Figure 2.2(a) shows the sequence of events in a two-phase signaling protocol. The events include the times when the data become valid and invalid. The transparent bars indicate the period when data are valid, during the other periods, data are invalid. Notice that a request signal occurs only after data become valid. This is an important timing restriction associated with these communication protocols, namely, the request signal that indicates that data are valid should always arrive at the receiver *after* all data wires have attained their proper value. The restriction is referred to as the *bundling constraint.* For this reason the communication protocol is often called the *bundled data protocol.* Figure 2.2(b) shows a sequence of events in a four-phase protocol and single-rail data encoding. Other sequences are also applicable for the four-phase protocol.

The dual-rail encoding scheme uses two wires for every data bit. There are several dual-rail encoding schemes. All combine the data encoding and signaling protocol. There is no explicit request signal, and the dual-rail encoding schemes all require $(2n+1)$ wires as illustrated in Figure 2.1(b). In the case of four-phase signaling, there are several encodings that can be used to transmit a data bit. The most common encoding has the following meaning for the four states in which each pair of wires can be in: 00 = reset, 10 = valid 0, 01 = valid 1, and 11 is an unused state. Every pair of wires has to go through the reset state before becoming valid again. In the first phase of the four-phase signaling protocol, every pair of wires leaves the reset state for a valid 0 or 1 state. The receiver detects the arrival of a

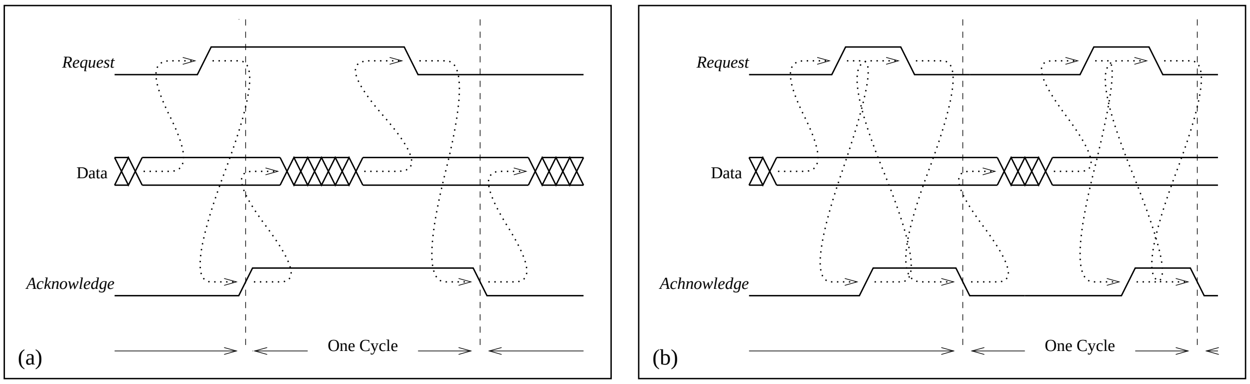


Figure 2.2: Data transfer in two-phase signaling (a), and four-phase signaling (b)

new set of valid data when all pairs of wires have left the reset state. This detection replaces an explicit request signal. The second phase consists of an acknowledgment to inform the sender that data has been consumed. The third phase consists of the reset of all pairs of wires to the reset state, and the fourth phase is the reset of the acknowledgment.

In a two-phase signaling protocol, a different dual-rail encoding is used. An example of an encoding is as follows. Each pair of wires has one wire associated with a 0 and one wire associated with a 1. A transition on the wire associated with 0 represents the communication of a 0, whereas a transition on the other wire represents a communication of a 1. Thus, a transition on one wire of each pair signals the arrival of a new bit value. A transition on both wires is not allowed. In the first phase of the two-phase signaling protocol, every pair of wires communicates a 0 or a 1. The second phase is an acknowledgment sent by the receiver.

Of all data encodings and signaling protocols, the most popular are the single-rail encoding and four-phase signaling protocol. The main advantages of these protocols are the small number of connection wires and the simplicity of the encoding, which allows using conventional techniques for implementing data operations. The disadvantages of these protocols are the bundling constraints that must be satisfied and the extra energy and time wasted in the additional two phases compared with two-phase signaling. Dual-rail data encodings have been used to communicate data in asynchronous circuits free of any timing constraints. Dual-rail encodings, however, are expensive in practice, because of the many interconnection wires, the extra circuitry to detect completion of a transfer, and the difficulty in data processing.

2.2.2 Delay Models

An important characteristic distinguishing different asynchronous circuit styles is the delay model on which they are based. For each circuit primitive, gate or wire, a *delay model* stipulates the sort of delay it imposes and the range of the delays. Delay models are needed to analyze all possible behavior of a circuit for various correctness conditions, like the absence of hazards.

A circuit is composed of gates and interconnection wires, all of which impose delays on the signals propagating through them. The delay models are categorized into two classes: pure delay models and inertial delay models. In a *pure delay* model, the delay associated with a circuit component produces only a time shift in the voltage transitions. In reality, a circuit component may shift the signals and also filter out pulses of small width. A delay model which captures this fact is called an *inertial delay* model. Both classes of delay models can have several ranges for the delay shifts. We distinguish the *zero-delay*, *fixed-delay*, *bounded-delay*, and *unbounded-delay* models. In the zero-delay model, the values of the delays are zero. In the fixed-delay model, the values of the delays are constant, whereas in the bounded-delay model the values of the delays vary within a bounded range. The unbounded-delay model does not impose any restriction on the value of the delays except that they cannot be infinite. Sometimes two different delay models are assumed for the wires and the gates in an asynchronous circuit. For example, the operation of a class of asynchronous circuits is based on the zero-delay model for wires and the unbounded-delay model for gates. Formal definitions of the various delay models are given in [10].

A concept closely related to the delay model of a circuit is its *mode of operation*. The mode of operation characterizes the interaction between a circuit and its environment. Classical asynchronous circuits operate in the *fundamental mode* [58,97], which assumes that the environment changes only one input signal and waits until the circuit reaches a stable state. Then the environment is allowed to apply the next change to one of the input signals. Many modern asynchronous circuits operate in the *input-output* mode. In contrast to the fundamental mode, the input-output mode allows for input changes immediately after receiving an appropriate response to a previous input change, even if the entire circuit has not yet stabilized. The fundamental mode was introduced in the sixties to simplify the analysis and design of gate circuits with Boolean algebra. The input-output mode evolved in the eighties from event-based formalisms to describe modular design methods that abstracted from the internal operation of a circuit.

2.2.3 Formalisms

Just as in any other design discipline, designers of asynchronous circuits use various formalisms to master the complexities in the design and analysis of their artifacts. The formalisms used in asynchronous circuit design can be categorized into two classes: formalisms based on Boolean algebra and formalisms based on sequences of events. Most design methodologies in asynchronous circuits use some mixture of both formalisms.

The design of many asynchronous circuits is based on Boolean algebra or its derivative switching theory. Such circuits often use the fundamental mode of operation, the bounded-delay model, and have, as primitive elements, gates that correspond to the basic logic functions, like AND, OR, and inversion. These formalisms are convenient for implementing logic functions, analyzing circuits for the presence of hazards, and synthesizing fundamental-mode circuits [10, 97].

Event-based formalisms deal with sequences of events rather than binary logic variables. Circuits designed with an event-based formalism operate in the input-output mode, under an unbounded-delay model, and have, as primitive elements, the JOIN, the TOGGLE, and the MERGE, for example. Event-based formalisms are particularly convenient for designing asynchronous circuits when a high degree of concurrency is involved. Several tools have been generated for the automatic verification of asynchronous circuits with event-based formalisms [29, 32]. Examples of event-based formalisms are Trace Theory [3, 34, 99], DI Algebra [50], Petri nets, and Signal Transition Graphs [18, 60].

2.3 Design Techniques

This section introduces the most popular types of asynchronous circuits and briefly describes some of their design techniques.

2.3.1 Types of Asynchronous Circuits

There are special types of asynchronous circuits for which formal and informal specifications have been given. Here are brief informal descriptions of some of them in a historical context.

There are two types of logic circuits: *combinational* and *sequential*. The output of a combinational circuit depends only on the current inputs, whereas the output of a sequential circuit depends also on the previous sequences of the inputs. With this definition of a sequential circuit, almost all asynchronous circuit styles fall into this category. However, the term *asynchronous sequential* circuits or machines generally refers to those asynchronous circuits based on *finite state machines* similar to those in synchronous sequential circuits [48, 97].

Muller was the first to give a rigorous formalization of a special type of circuits for which he coined the name *speed-independent* circuits. An account of this formalization is given in [63, 65]. Informally, a speed-independent circuit is a network of gates that satisfies its specification irrespective of any gate delays.

From a design discipline that was developed as part of the Macromodules project [21] at Washington University in St. Louis, the concept of another type of asynchronous circuits evolved, which was given the name *delay-insensitive* circuit, that is, a network of modules that satisfies its specification irrespective of any element *and* wire delays. It was realized that proper formalization of this concept was needed to specify and design such circuits in a well-defined manner. Such a formalization was given by Udding [96].

Another name frequently used in designing asynchronous circuits is *self-timed systems*. This name was introduced by Seitz [80]. A self-timed system is described recursively as either a self-timed element or a legal connection of self-timed systems. The idea is that self-timed elements can be implemented with their own timing discipline, and some may even have synchronous implementations. In composing self-timed systems from self-timed elements, however, no reference to the timing of events is made; only the sequence of events is relevant. In other words, the elements "keep time to themselves."

Some have found the unbounded gate-and-wire delay assumption, on which the concept of a delay-insensitive circuit is based, to be too restrictive in practice. For example, the unbounded gate-and-wire delay assumption implies that a signal sent to multiple recipients by a fork can incur a different unbounded delay for each of the recipients. They proposed to relax this delay assumption slightly by using *isochronic forks* [56]. An isochronic fork is a fork whose difference in the delays of its branches is negligible compared with the delays in the element to which it is connected. A delay-insensitive circuit that uses isochronic forks is called a *quasi-delay-insensitive* circuit [3, 56]. Although the use of isochronic forks gives more design freedom in exchange for less delay insensitivity, care has to be taken with its implementation [2].

2.3.2 Asynchronous Sequential Machines

The design of asynchronous sequential finite state machines was initiated with the pioneering work of Huffman [48]. He proposed a structure similar to that of synchronous sequential circuits consisting of a combinational logic circuit, inputs, outputs, and state variables [97]. *Huffman circuits*, however, store the state variables in feedback loops containing *delay elements*, instead of in latches or flip-flops, as synchronous sequential circuits do. The design procedure begins with creating a *flow table* and reducing it through some *state minimization* technique. After a *state assignment*, the procedure obtains the Boolean expressions and implements them in combinational logic with the aid of a *logic minimization* program. To guarantee a hazard-free operation, Huffman circuits adopt the restrictive single-input-change fundamental mode, that is, the environment changes only one input and waits until the circuit becomes stable before changing another input. This requirement can substantially degrade the circuit performance. Hollaar realized this fact and introduced a new structure in which the fundamental mode assumption is relaxed [44]. In his implementation, the state variables are stored in NAND latches, so that inputs are allowed to change earlier than the fundamental mode would allow. Although Hollaar's method improves the performance, it suffers from the danger of producing hazards. Besides, neither technique seem to be adequate for designing concurrent systems. Models and algorithms for the analysis of asynchronous sequential circuits have been developed by Brzozowski and Seger [10].

The quest for more concurrency, higher performance, and hazard-free operation, resulted in the formulation of a new generation of asynchronous sequential circuits known as *burst-*

mode machines [22,26]. A burst-mode circuit does not react until the environment performs a number of input changes called an *input burst*. The environment, in turn, is not allowed to introduce the next input burst until the circuit produces a number of outputs called an *output burst*. A state graph is used to specify the transitions caused by the input and output bursts. Two synthesis methods have been proposed and automated for implementing burst-mode circuits. The first method employs a locally generated clock to avoid some hazards [67]. The second method uses three-dimensional flow tables and is based on Huffman circuits [108]. One limitation of burst mode circuits is that they restrict concurrency within a burst.

2.3.3 Speed-Independent Circuits and STG synthesis

Speed-independent circuits are usually designed by a form of Petri nets [71]. A popular version of Petri nets, *signal transition graphs* (STG), was introduced by Chu. He also developed a synthesis technique for transforming STGs into speed-independent circuits [18]. Chu's work was extended by Meng, who produced an STG-based tool for synthesizing speed-independent circuits from high-level specifications [61]. In this technique, a circuit is composed of computational blocks and interconnection blocks. Computational blocks range from a simple shifter module to more complicated ones, such as ALUs, RAMs, and ROMs. Interconnection blocks synchronize the operation of computational blocks by producing appropriate control signals. Computational blocks generate *completion signals* after their output data become valid. The interconnection blocks use the completion signals to generate four-phase handshake protocols.

2.3.4 Delay-Insensitive Circuits and Compilation

Several researchers have proposed techniques for designing delay-insensitive circuits. Ebergen [33] has developed a synthesis method based on the formalism of *Trace Theory*. The method consists of specifying a component by a program and then transforming this program into a delay-insensitive network of basic elements. The program notation allows specifying parallel behavior. Ebergen's method has been applied to the design of small components like stacks, various counters, and arbiters [34].

Martin proposes a method [56] that starts with a specification of an asynchronous circuit in a high-level programming language similar to Hoare's *Communicating Sequential Processes* (CSP) [43]. An asynchronous circuit is specified as a group of processes communicating over channels. After various transformations, the program is mapped into a network of gates. This method led to the design of an asynchronous microprocessor [57] in 1989. Martin's method yields quasi-delay-insensitive circuits.

Van Berkel [3] has designed a compiler based on a high-level language called *Tangram*. A Tangram program also specifies a set of processes communicating over channels. A Tangram program is first translated into a *handshake circuit*. Then these handshake circuits are mapped into various target architectures, depending on the data-encoding techniques or standard-cell libraries used. The translation is syntax-directed, which means that every operation occurring in a Tangram program corresponds to a primitive in the translated handshake circuit. This property is exploited by various tools that quickly estimate the area, performance, and energy dissipation of the final design by analyzing the Tangram program. Van Berkel's method also yields quasi-delay-insensitive circuits.

Other translation methods from a CSP-like language to a (quasi-) delay-insensitive circuit can be found in [9, 101].

2.4 A Typical Asynchronous Design

In this section we present a typical asynchronous design, a *micropipeline* [92]. The circuit uses single-rail encoding with the two-phase signaling protocol to communicate data between stages of the pipeline. The control circuit for the pipeline is a delay-insensitive circuit. First we present the primitives for the control circuit, then we present the latches that store the data, and finally we present the complete design.

2.4.1 The Control Primitives

Figure 2.3 shows a few simple primitives used in event-based design styles. The schematic symbol for each primitive is depicted opposite its name.

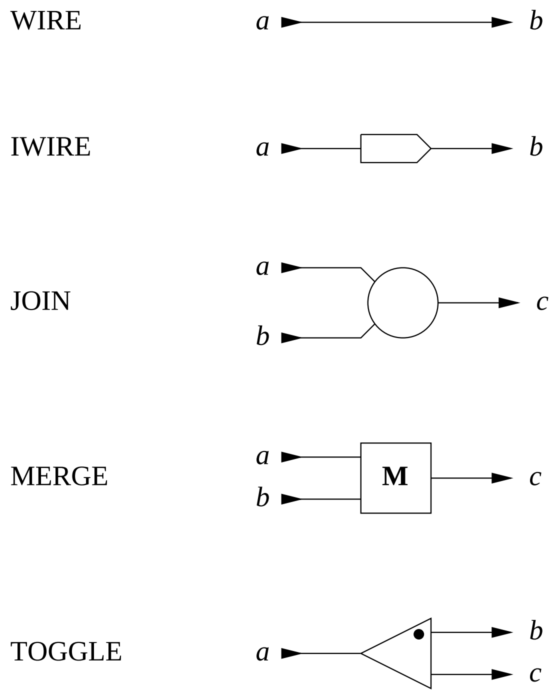


Figure 2.3: Some delay-insensitive primitives

The simplest primitive is the WIRE, a two-terminal element that produces an output event on its output terminal b after every input event on its input terminal a. Input and output events in a WIRE must alternate. An input event a must be followed by an output event b before another event a occurs. A WIRE is physically realizable with a wire, and events are implemented by voltage transitions. An initialized WIRE, or IWIRE, is very similar to a WIRE, except that it starts by producing an output event b instead of accepting an input event a; after this, its behavior exactly resembles that of a WIRE.

The primitive for synchronization is the JOIN, also called the RENDEZVOUS [21]. A JOIN has two inputs a and b and one output c. The JOIN performs the AND operation of two events a and b. It produces an output event c only after both of its inputs, *a and b*, received an event. The inputs can change again after an output is produced. A JOIN can be implemented by a *Muller C-element*, explained in the next section.

The MERGE component performs the OR operation of two events. If a MERGE component receives an event on either of its inputs, *a or b*, it produces an output event c. After an input event, there must be an output event; successive input events are not allowed. A MERGE can be implemented by a XOR gate.

The TOGGLE has a single input a and two outputs b and c. After an event on input a, an event occurs on output b. The next event on a results in a transition on output c. An input event must be followed by an output event before another input event can occur. Thus, output events alternate or *toggle* after each input event. The dot in the TOGGLE schematic indicates the output which produces the first event.

2.4.2 Storage Primitives

Now we discuss two event-controlled latches due to Sutherland [92], as depicted in Figure 2.4. Their operation is managed through two input control signals: capture and pass, labeled c and p respectively. They also have two output control signals: capture done, cd, and pass done, pd. The input data is labeled D, and the output data is labeled Q. Implementation

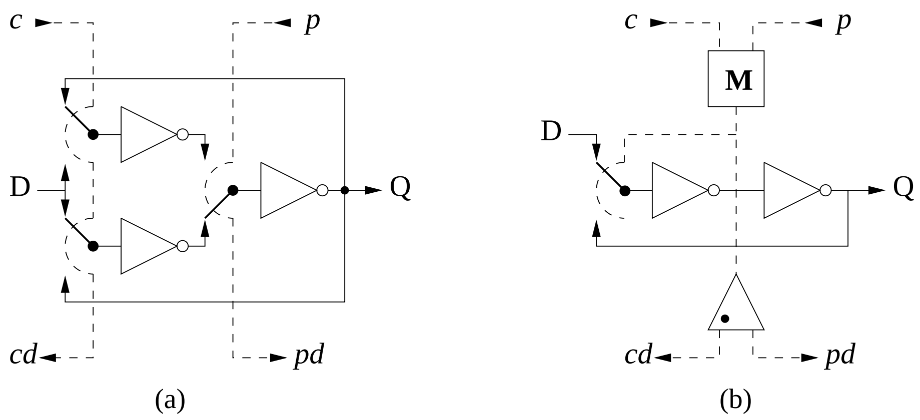


Figure 2.4: Two event-driven latch implementations

(a) is composed of three so-called *double-throw* switches. Implementation (b) includes a MERGE, a TOGGLE, and a level-controlled latch consisting of a double-throw switch and an inverter. A double-throw switch is schematically represented by an inverter and a switching tail. The tail toggles between two positions based on the logic value of a controlling signal. A double-throw switch, in fact, is a two-input multiplexer that produces an inverted version of its selected input. A CMOS implementation of the double-throw switch is shown in Figure 2.5 [92]. The position of the switch corresponds to the state where c is low.

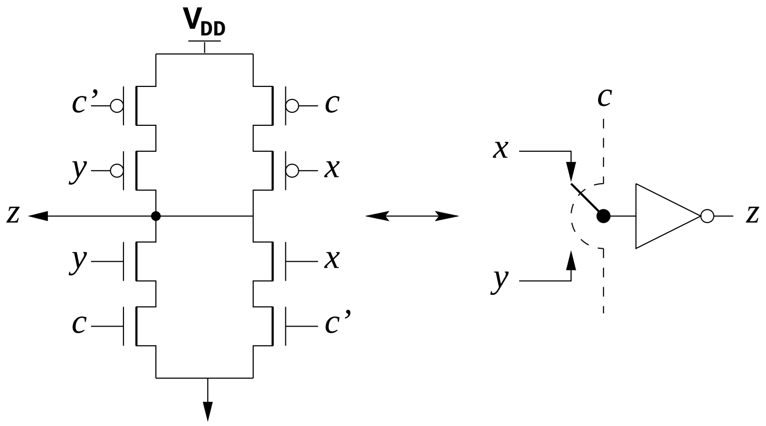


Figure 2.5: A CMOS implementation of a double-throw switch

An event-controlled latch can assume two states: *transparent* and *opaque*. In the transparent state no data is latched, but the output replicates the input, because a path of two inverting stages exists between the input and the output. In the opaque state, this path is disconnected so that the input data can change without affecting the output; the current data at the output, however, is latched. Implementations in Figures 2.4(a) and 2.4(b) are both shown in their initial transparent states. The capture and pass signals in an event-controlled latch always alternate. Upon a transition on c, the latch captures the current input data and becomes opaque. The following transition on cd is an acknowledgment to the data provider that the current data has been captured and that the input data can be changed safely. A subsequent transition on p returns the latch back to its transparent state to pass the next data to its output. The p signal is acknowledged by a transition on pd. Notice that in implementation (a) of Figure 2.4, signals cd and pd are merely delayed and possibly amplified versions of c and p, respectively.

A group of event-controlled latches, similar to implementation (a) of Figure 2.4, can be connected, sharing a capture wire and a pass wire, to form an event-controlled register of arbitrary data width. Implementation (b) of Figure 2.4 can be generalized similarly into a register by inserting additional level-controlled latches between the MERGE and the TOGGLE. A comparison of different micropipeline latches is reported in [28] and later in [107].

2.4.3 Pipelining

Pipelining is a powerful technique for constructing high-performance processors. Micropipelines are elegant asynchronous circuits that have gained much attention in the asynchronous community. Many VLSI circuits based on micropipelines have been successfully fabricated. The AMULET microprocessor [37] is one example.

The simplest form of a micropipeline is a FIFO. A four-stage FIFO is shown in Figure 2.6. It has a control circuit composed solely of interconnected JOINs and a data path of event-controlled registers. The control signals are indicated by dashed lines. The thick arrows show the direction of data flow. Data is implemented with single-rail encoding, and the data path is as wide as the registers can accommodate. Adjacent stages of the FIFO communicate through a two-phase, bundled-data signaling protocol. This means that a request arrive at the next stage only when the data for that stage becomes valid. A bubble at the input of a JOIN is a shorthand for a JOIN with an IWIRE on that input. It implies that, initially, an event has already occurred on the input with the bubble, and the JOIN can produce an output event immediately upon receiving an event on the other input.

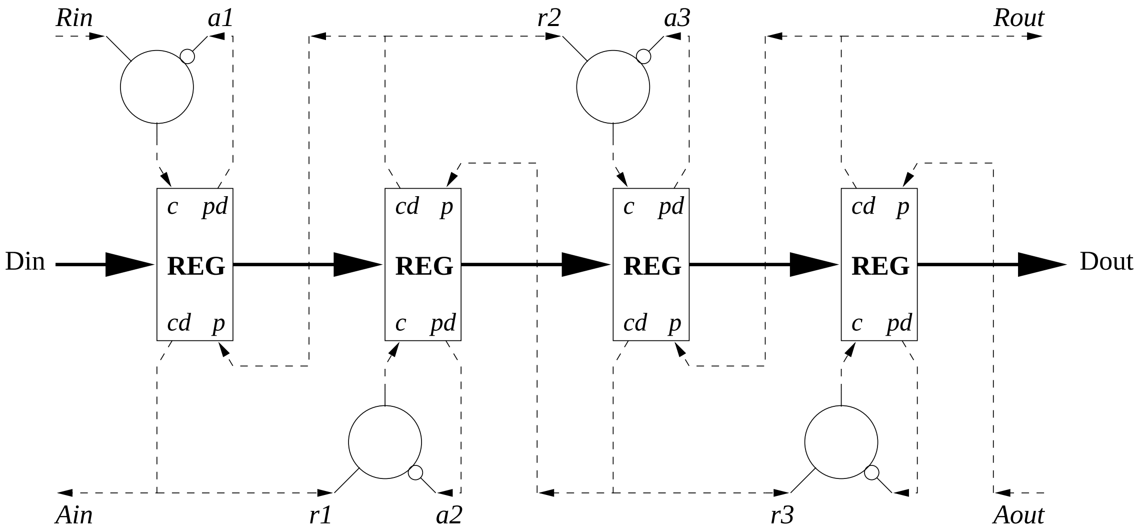


Figure 2.6: A four-stage micropipeline FIFO structure

Initially, all control wires of the FIFO are at low voltage, and the data in the registers are not valid. The FIFO is activated by a rising transition on R_{in}, which indicates that input data is valid. Subsequently, the first-stage JOIN produces a rising output transition. This signal is a request to the first-stage register to capture the data and become opaque. After capturing the data, the register produces a rising transition on its *cd* output terminal. This causes a transition on A_{in} and a transition on $r1$, which is a request to the second stage of the FIFO. Meanwhile, the data has proceeded to the second-stage register and has arrived there before the transition on $r1$ occurs. If the environment does not send any new data, the first stage remains idle, and the data and the request signals propagate further to the right. Notice that each time the data is captured by a stage, an acknowledgment is sent back to the previous stage which causes its latch to become transparent again. When the data has propagated to the last register, it is stored and a request signal R_{out} is forwarded to the consumer of the FIFO. At this point, all control signals are at high voltage except for A_{out}. If the data is not removed out of the FIFO, that is, A_{out} remains low, the next data coming from the producer advance only up to the third-stage register, because the fourth-stage JOIN cannot produce an output. Finally, A_{out} also becomes high when the consumer acknowledges receipt of the data. Further data storage and removal follows the same pattern. The operation of each JOIN can be interpreted as follows. If the previous stage has sent a request for data capture *and* the present stage is empty, then send a signal to capture the data in the present stage.

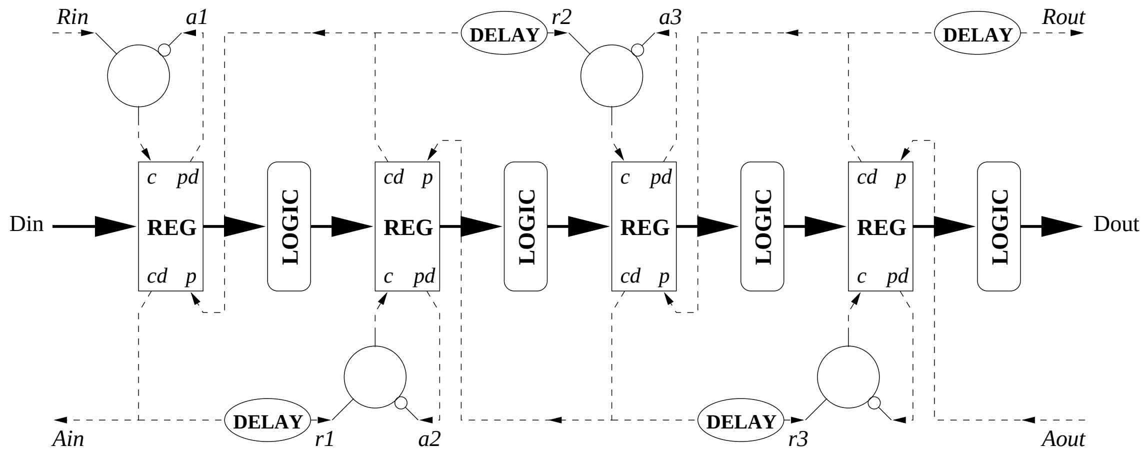


Figure 2.7: A general four-stage micropipeline structure

The FIFO can be modified easily to include data processing. A four-stage micropipeline, in its general form, is illustrated in Figure 2.7. Now the data path consists of alternately positioned event-driven registers and combinational logic circuits. The event-driven registers store the input and output data of the combinational circuits, and the combinational circuits perform the necessary data processing. To satisfy the data bundling constraint, delay elements may occasionally be required to slow down the propagation of the request signals. A delay element must at least match the delay through its corresponding combinational logic circuit, either by some completion detection mechanism or through the insertion of a worst-case delay.

A micropipeline FIFO is flexible in the number of data items it buffers. There is no restriction on the rate at which data enters or exits the micropipeline, except for the delays imposed by the circuit elements. That is why this FIFO and micropipelines generally, are termed *elastic*. In contrast, in an ordinary synchronous pipeline, the rates at which data enter and exit the pipeline are the same, dictated by the external clock signal. A micropipeline is also flexible in the amount of energy it dissipates, which is proportional to the number of data movements. A clocked pipeline, however, continuously dissipates energy as if all stages of the pipeline capture and pass data all the time. Another attractive feature of a micropipeline is that it automatically shuts off when there is no activity. A clocked pipeline, on the other hand, requires a special clock management mechanism to implement this feature. This sensing mechanism, however, constantly consumes energy, because it should never go idle.

2.5 Concluding Remarks

We have touched only on a few topics relevant to the area of asynchronous circuits and omitted many others. Among the topics omitted are the important areas of verification, testing, and performance analysis of asynchronous circuits. We hope, however, that within the scope of these pages we have provided enough information for further readings. For more information on asynchronous circuits, please see [27], [10], or [39]. A comprehensive bibliography of asynchronous circuits can be found in [70]. Up-to-date information on research in asynchronous circuit design can be found at [38].

Chapter 3

A Formulation for Quick Evaluation and Optimization of CMOS Logic Circuits

From a digital circuit designer's point of view, the design parameters are often limited to the widths of the transistors. The supply voltage is usually fixed by the management and the threshold voltage is dictated by the technology. Just recently, altering the supply voltage within the chip and multi-threshold voltage technologies for saving energy are being practiced. But the former produces overhead and the latter is costly. Although literature is abundant with models for the performance of digital CMOS gates (e.g. [1, 31, 40, 49, 76, 90]), they are often too complicated for quick hand-analysis and optimization. Most of these models express the delay in terms of a load capacitance, from which the effect of changing the sizes of the transistors is not immediately clear.

The purpose of this chapter[1] is to introduce a first-order for the delay and energy of CMOS gates explicitly expressed in terms of the widths of the transistors. We demonstrate the convenience of using this model and report some important results obtained through direct applications of the model. We present simple expressions for optimum transistor sizing for minimizing the rising delay, falling delay, total delay, and energy-delay product of

[1]The content of this chapter appears in [85].

standard CMOS logic circuits. To keep the model simple, we ignore some of the secondary effects, such as the contribution of finite signal slopes to delay. We also use a very simple model for MOS capacitances. Later chapters present a more accurate and comprehensive formulation. Considering that in practice the input signal to a gate is the output of the preceding gate and, in turn, the output of a gate is the input to the succeeding gate, the formulation and the parameters are derived for a set of three inverting buffers: a driver, a cell, and a load. In a separate section, we discuss the applications of the model to the classical problem of driving large loads, and generation of complementary signals.

3.1 Single Stage CMOS Inverter

The parasitic capacitances of a circuit characterize its delay and energy consumption. In an MOS transistor, there are two main capacitances: the gate capacitance, and the source and drain diffusion capacitances. Figure 3.1 shows the layout of an MOS transistor.

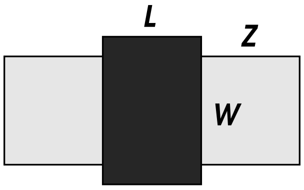


Figure 3.1: Layout of a single MOS transistor.

The gate capacitance can be expressed as

$$C_g = C_{ox}A_g \;=\; C_{ox}LW \;=\; K_{gw}W \tag{3.1}$$

where A_g is the gate area, C_{ox} is the gate oxide capacitance per unit area, L is the effective gate length, and W is the effective gate width of the transistor. The diffusion capacitance at the source or drain can be expressed as

$$C_d = (C_{ja}Z + 2C_{jp})W + 2ZC_{jp} \;=\; K_{dw}\;W + C_z \tag{3.2}$$

where Z is the diffusion length, C_{ja} is the average diffusion junction capacitance per unit area, and C_{jp} is the average diffusion periphery capacitance per unit length. For convenience, we assume that the area and periphery junction capacitances for NMOS and PMOS transistors have the same corresponding values. The above expression shows that a diffusion capacitance has two components: one which linearly grows with W, and another one which is constant independent of W. We have ignored the gate to drain overlap capacitances and the miller effect. If included, however, overlap capacitances grow linearly with W.

Figure 3.2 illustrates the schematic of a CMOS inverter driving a total capacitive load of C which includes all the gate and diffusion capacitances at the output node. The delay of discharging C from V_{DD} to $\frac{V_{DD}}{2}$ through the NMOS transistor of width W_n is given by

$$\grave{D} = \frac{CV_{DD}}{2I_n} \tag{3.3}$$

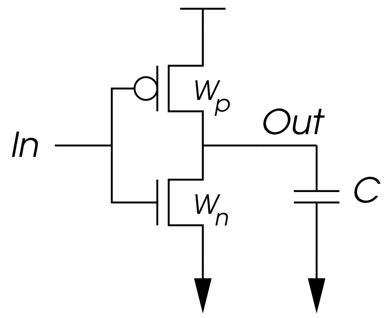


Figure 3.2: Schematic of a CMOS inverter driving a capacitance C.

where I_n is the average value of the current during the discharging period. Assuming a linear relation with W_n, we can write

$$I_n = K_{iw} W_n$$

where the constant K_{iw} is a function of V_{DD}, the threshold voltage V_{TN}, and technology parameters. The capacitance at the output could be split into two components. One component C_g includes all the gate capacitances at the output node, while the other component C_d includes all the diffusion capacitances at the output node. Thus,

$$\begin{aligned} D_n &= \frac{(C_g + C_d)V_{DD}}{2I_n} \\ &= \frac{V_{DD}K_{gw}}{2K_{iw}} \frac{W_g}{W_n} + \frac{V_{DD}K_{dw}}{2K_{iw}} \frac{W_d}{W_n} + \frac{V_{DD}C_z}{2K_{iw}} \frac{N}{W_n} \end{aligned}$$

where W_g is the total width of the gates at the output node, W_d is the total width of the diffusions at the output node including W_n and W_p, and N is the number of diffusions at the output node. The above equation can be simplified as the following by introducing parameters K_D, K'_D, and K''_D.

$$\grave{D} = K_D \frac{W_g}{W_n} + K'_D \frac{W_d}{W_n} + K''_D \frac{N}{W_n} \tag{3.4}$$

A similar equation can be written for the charging delay through the PMOS transistor of width W_p. If ρ represents the ratio of the saturated currents in an NMOS transistor to that of a PMOS transistor of the same width, then

$$\acute{D} = \rho\ \grave{D}, \quad \text{if} \quad W_n = W_p$$

Analytically, ρ is given by

$$\rho = \frac{\mu_n}{\mu_p} \left(\frac{V_{DD} - V_{TN}}{V_{DD} - V_{TP}} \right)^{\alpha}$$

where μ_n and μ_p are the effective electron and hole mobilities, and α equals 2 for long channel devices and around 1.25 for short channel devices [76]. Alternatively, ρ is defined as the width ratio of the PMOS transistor to NMOS transistor at which the rising and falling delays are equal. Using this definition, ρ can be found by simulating a chain of inverters. Figure 3.3 shows that for our technology at $V_{DD} = 3$ V, $\rho = 2.2$. Note that the rise time and fall time (i.e., 10% to 90% of output signal value) are not equal at this value, as illustrated in the figure.

The dynamic energy required to charge and discharge the capacitance C is approximately given by

$$\begin{aligned} E &= V_{DD}^2\ C \ = \ V_{DD}^2 (C_g + C_d) \\ &= V_{DD}^2 K_{gw} W_g + V_{DD}^2 K_{dw} W_d + V_{DD}^2 C_z N \end{aligned}$$

Which can be simplified as

$$E = K_E W_g + K'_E W_d + K''_E N \tag{3.5}$$

Here, we have ignored the effect of short-circuit current.

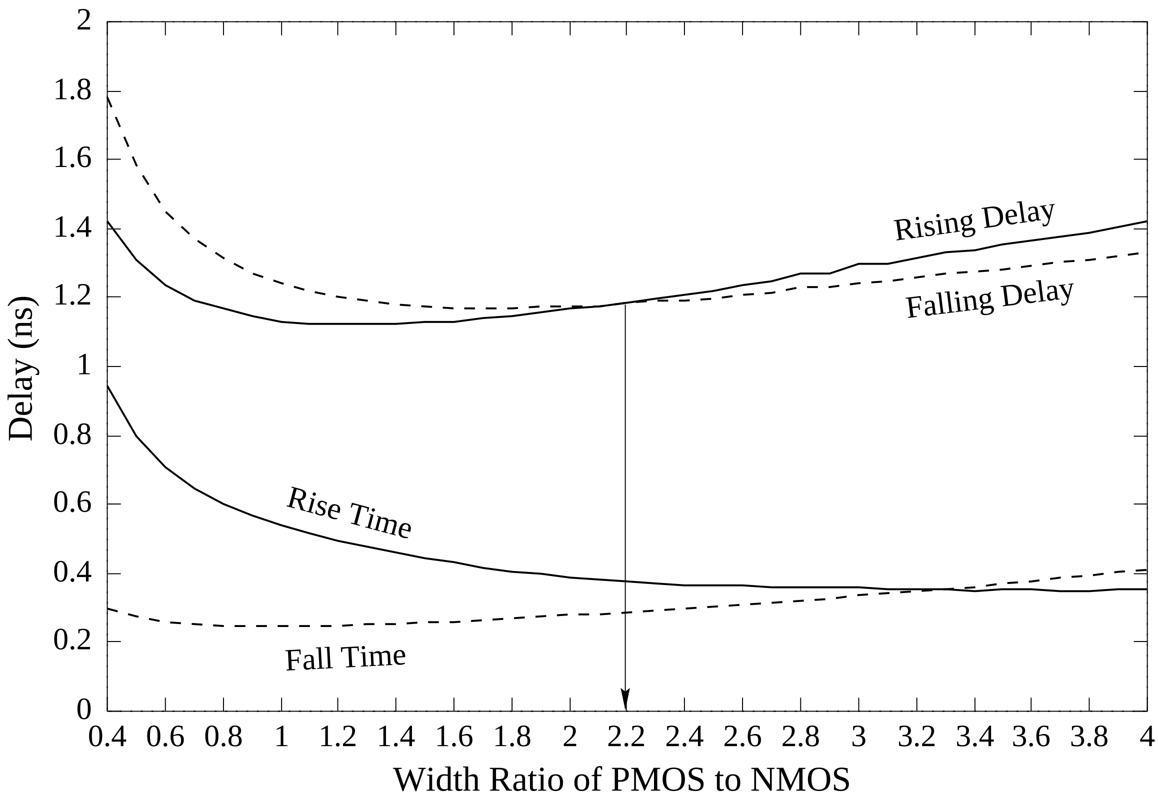


Figure 3.3: Simulation results for a chain of five inverters to extract the value of ρ.

3.2 Cascaded CMOS Gates

The delay of a CMOS gate is characterized by its driving gate and the gates it drives. To simplify design and analysis, a CMOS gate can usually be represented by an equivalent CMOS inverter [66]. In Figure 3.4, a CMOS gate, its driving gate, and its load have all been modeled as inverters. The ratio of PMOS to NMOS transistors in the inverters is represented by r. We are assuming a uniform r for all three gates. This restriction is removed in later chapters.

3.2.1 Total Delay

The total delay, $\hat{D}$, defined here as the sum of a rising transition delay and a falling transition delay between the input of the drive and the output of the cell is given by

$$\hat{D} = \hat{D}_g + \hat{D}_d \tag{3.6}$$

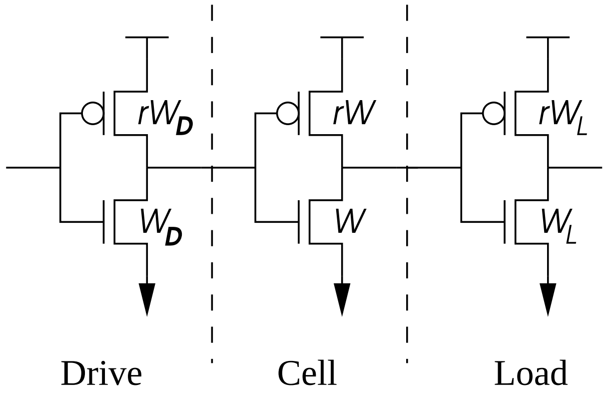


Figure 3.4: A CMOS driver, gate (cell), and load represented by CMOS inverters

Where the delay due to the gate capacitances is denoted by

$$\begin{aligned}\hat{D}_g &= K_D\left[\frac{W(r+1)}{W_D}+\frac{\rho W(r+1)}{rW_D}+\frac{W_L(r+1)}{W}+\frac{\rho W_L(r+1)}{rW}\right]\\ &= K_D\left(\rho+r+\frac{\rho}{r}+1\right)\left(\frac{W}{W_D}+\frac{W_L}{W}\right)\\ &= \delta\left(\frac{W}{W_D}+\frac{W_L}{W}\right)\end{aligned} \tag{3.7}$$

and the delay due to the diffusion capacitances is denoted by

$$\begin{aligned}\hat{D}_d &= K'_D\left[\frac{W_D(r+1)}{W_D}+\frac{\rho W_D(r+1)}{rW_D}+\frac{W(r+1)}{W}+\frac{\rho W(r+1)}{rW}\right]\\ &\quad +K''_D\left[\frac{2}{W_D}+\frac{2\rho}{W_D}+\frac{2}{W}+\frac{2\rho}{W}\right]\\ &= 2K'_D\left(\rho+r+\frac{\rho}{r}+1\right)\ +\ 2K''_D(1+\rho)\left(\frac{1}{W_D}+\frac{1}{W}\right)\\ &= 2\delta'\ +\ 2\delta''\left(\frac{1}{W_D}+\frac{1}{W}\right)\end{aligned} \tag{3.8}$$

Parameters δ, δ', and δ'' are constants defined by the above equations. For further simplification, the second term in D_d can be neglected, since it is relatively small for typical values of W and W_D, and it also decreases as W and W_D increase. Therefore, as a first approximation

$$\hat{D} = \delta\left(\frac{W}{W_D}+\frac{W_L}{W}\right)+2\delta' \tag{3.9}$$

So, the delay consists of two terms: a gate-delay term, which depends on the size of the transistors, and a diffusion-delay term, which depends solely on the number of the diffusions.

The larger the driver and the smaller the load, the smaller is the delay. If $W_D \gg W$ and $W_L \ll W$, then $\hat{D}$ is reduced to its minimal value and approaches $D_d = 2\delta'$. For fixed W_D and W_L, the optimum value of W obtained by solving $\frac{\partial D}{\partial W} = 0$ is expressed by

$$\hat{W} = \sqrt{W_D W_L} \tag{3.10}$$

This is an important result for optimization of CMOS digital circuits. It is interesting to note that the above expression remains valid even if the driver is driving some other gates in addition to the cell. Using the above formula it can be easily shown that the maximum frequency in a chain of inverters is achieved if all inverters are of the same size. Figure 3.5 shows the variations of $\hat{D}$ as a function of W obtained by HSPICE simulations of the circuit in Figure 3.4 with $r = 2$. The driver size W_D is fixed at 5μm and W_L is changed from 5μm to 100μm. The optimum values of W as obtained from Equation 3.10 are indicated on each curve; good agreement with simulation is observed.

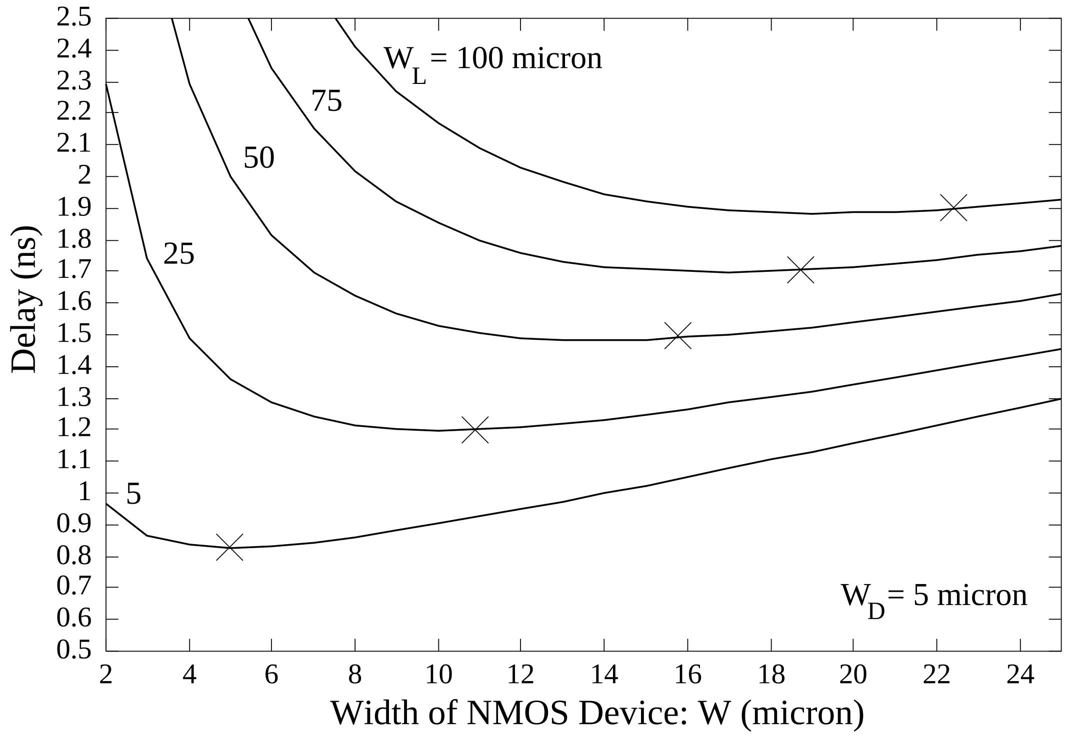


Figure 3.5: Variations of delay as a function of the size of the cell for a fixed driver size and various load sizes. Minimums obtained by the model using equation 3.10 are indicated on each curve.

The optimum ratio of PMOS to NMOS transistors for minimizing the delay can be found from $\frac{\partial D}{\partial r} = 0$, and is governed by

$$\hat{r} = \sqrt{\rho} \tag{3.11}$$

independent of W, W_D, and W_L. In [47] the above has been derived under the assumption that the total width of the PMOS and NMOS transistors $W_p + W_n$ is kept fixed. We emphasize that the above, as derived, is valid in general.

Using these values for W and r in (3.9), the minimum achievable delay for fixed W_D and W_L is

$$D_{\min} = 2\left(\rho + 2\sqrt{\rho} + 1\right)\left[K_D\sqrt{\frac{W_L}{W_D}} + K'_D\right] \tag{3.12}$$

3.2.2 Rising and Falling Delays

In a similar fashion to the total delay, the rising and falling delays can be expressed as

$$\acute{D} = K_D(1+r)\left[\frac{W}{W_D} + \frac{W_L}{W}\frac{\rho}{r}\right] + K'_D\left(\rho + r + \frac{\rho}{r} + 1\right) \tag{3.13}$$

and

$$\grave{D} = K_D(1+r)\left[\frac{W}{W_D}\frac{\rho}{r} + \frac{W_L}{W}\right] + K'_D\left(\rho + r + \frac{\rho}{r} + 1\right) \tag{3.14}$$

respectively. The optimum W for minimizing the rising delay is, then, given by

$$\acute{W} = \sqrt{W_D W_L \frac{\rho}{r}}, \tag{3.15}$$

and that for the falling delay by

$$\grave{W} = \sqrt{W_D W_L \frac{r}{\rho}} \tag{3.16}$$

In the same way, the optimum r for the rising delay is obtained by

$$\acute{r} = \sqrt{\rho\frac{K_D(W_L/W) + K'_D}{K_D(W/W_D) + K'_D}} \tag{3.17}$$

and that of the falling delay is obtained by

$$\grave{r} = \sqrt{\rho \frac{K_D(W/W_D) + K_D'}{K_D(W_L/W) + K_D'}} \tag{3.18}$$

Sometimes a designer may be interested in transistor sizing for equal rising and falling delays. On the one hand, by definition $\acute{D} = \grave{D}$ at $r = \rho$ for any W. On the other hand, if we solve $\acute{D} = \grave{D}$ for W, the result is

$$W = \sqrt{W_D W_L} \tag{3.19}$$

independent of r. This is the same expression as Equation 3.10, which minimizes the average delay. Hence, if the number of cascaded gates in a path is n, then by setting

$$\begin{aligned} W_i &= \sqrt{W_{i-1} W_{i+1}}, \qquad 0 < i < n \\ r &= \sqrt{\rho} \end{aligned}$$

the designer minimizes the total delay while matching the rising and falling delays of the signal along the path.

3.2.3 Energy and Area

The total energy dissipation of the drive, cell, and the load in Figure 3.4 can be approximated by

$$\begin{aligned} E &= (K_E + K_E')(r+1)(W_D + W + W_L) + 2 \times 3\ K_E'' \\ &= (\eta + \eta')(W_D + W + W_L) + 3\eta'' \end{aligned} \tag{3.20}$$

where $\eta = K_E(r+1)$, $\eta' = K_E'(r+1)$, and $\eta'' = 2K_E''$. The last term is relatively small and can be ignored.

Assuming that the total gate and diffusion area of a CMOS cell is a fair indication of its total layout area, the area of a single NMOS transistor of width W can be expressed by

$$A_n = (L + 2Z)W = K_A W \tag{3.21}$$

where K_A is a constant. Hence, the area of the cell in Figure 3.4 (assuming an actual inverter) is

$$A = K_A\ (r+1)\ W = \theta W \tag{3.22}$$

where θ is a constant. For our technology $Z = 2.5\ \mu$m and $K_A = 5.8\ \mu$m.

3.2.4 Energy-Delay Product

A parameter which is sometimes used as the optimization criterion in VLSI design is the energy-delay product denoted here by

$$F = (W_D + W + W_L)\left[\phi\left(\frac{W}{W_D} + \frac{W_L}{W}\right) + \phi'\right] \tag{3.23}$$

with $\phi = \delta(\eta + \eta')$ and $\phi' = 2\delta'(\eta + \eta')$.

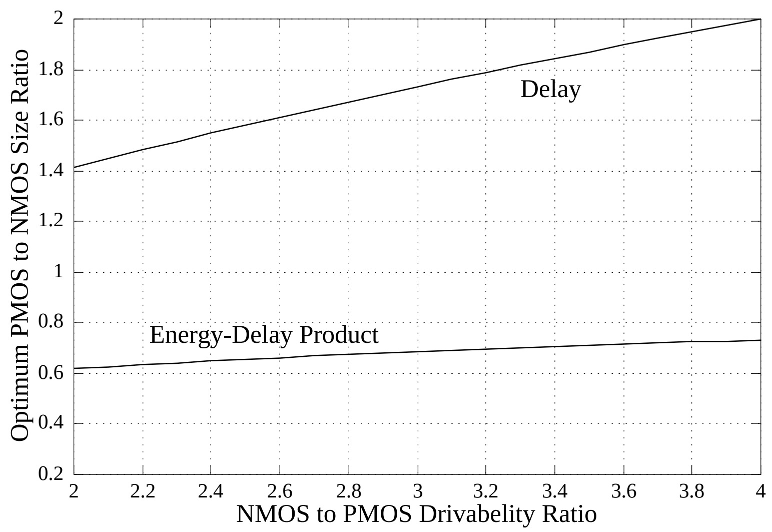


Figure 3.6: Variations of optimal r with ρ for delay and energy-delay product.

The optimum r for minimizing F found from solving $\partial F/\partial r = 0$ is given by

$$r = \frac{1}{4}\left(\sqrt{\rho^2 + 8\rho} - \rho\right) \tag{3.24}$$

which ranges between 0.6 to 0.7 for the typical values of ρ between 2 to 3. The optimum value of W for minimizing F is the root of the following polynomial

$$2W^3 + W^2\left(W_D + W_L + \frac{K'}{K}W_D\right) - (W_L W_D^2 + W_D W_L^2) = 0 \tag{3.25}$$

for which the closed form solution is rather lengthy. The variations of optimal r with ρ for minimizing the delay and energy-delay product are illustrated in Figure 3.6.

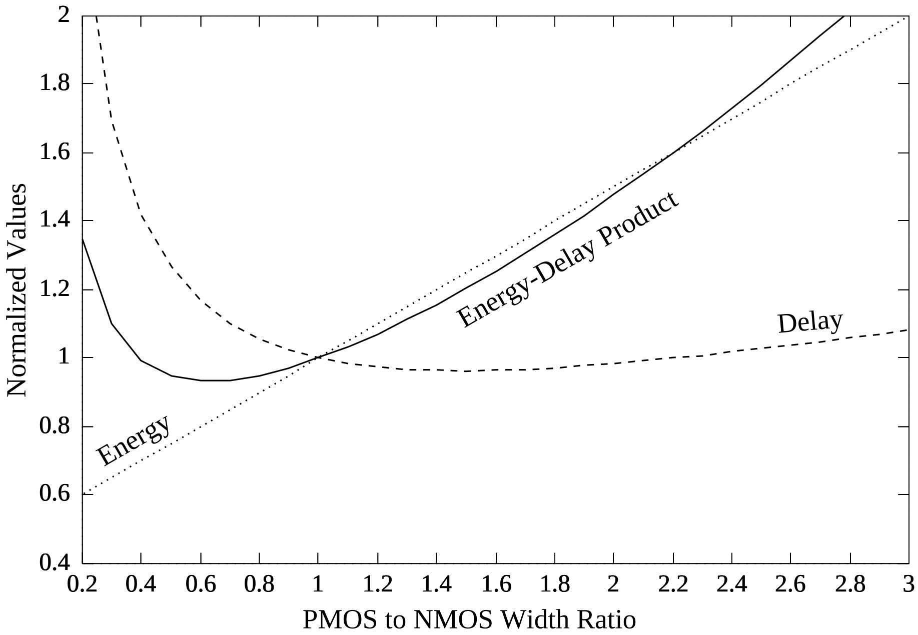


Figure 3.7: Variations of the delay $\hat{D}$, energy E, and energy-delay product F for a driver, cell, and a load based on the formulation.

Figure 3.7 shows the variations of the delay, energy, and delay-energy product as a function of the ratio of the PMOS to NMOS transistor width. The curves are obtained from equations 3.9, 3.20, and 3.23 for $W_D = W_L = W = 1$; each curve is then normalized to its value at $r = 1$. One realizes that while $\hat{D}$ has a broad minimum from $r = 1$ to $r = 2.5$, F has a relatively narrow minimum from $r = 0.5$ to $r = 0.8$. Figure 3.8 show the variations of F as a function of W using simulations of the circuit in Figure 3.4 with $r = 2$. W_D is fixed at 5μm and W_L is changed from 5μm to 100μm. The minimums as obtained from the equations 3.25 are indicated on each graph; good agreement with simulation is observed.

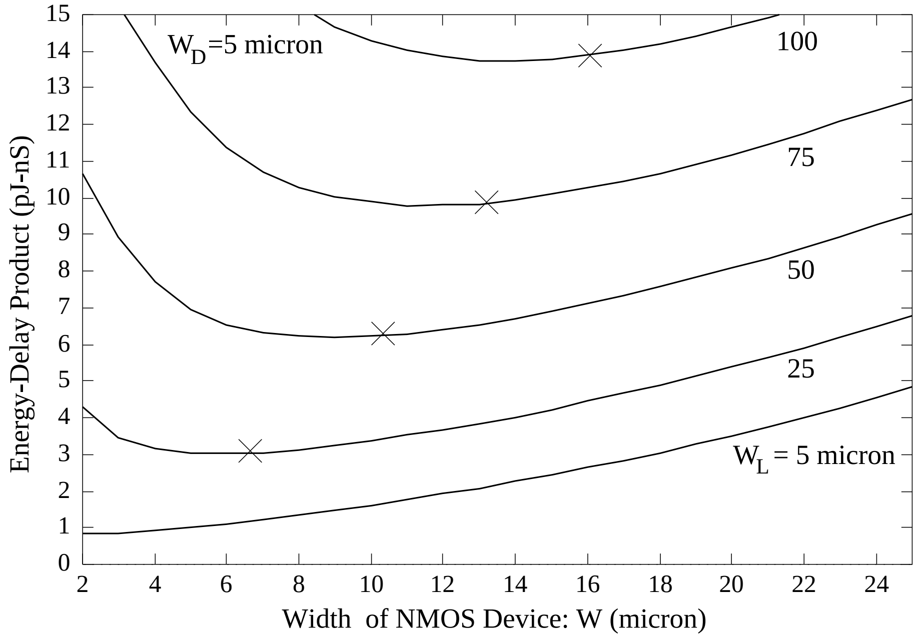


Figure 3.8: Variations of energy-delay product as a function of the size of the cell for a fixed driver size and various load sizes. Minimums obtained by the model using equation 3.25 are indicated on each curve.

3.3 Some Applications and Observations

3.3.1 Inverter Chain

The period of oscillation of a chain of M inverters, where M is an odd integer, can be expressed by

$$P = M(K_D + K'_D)(\rho + 2\sqrt{\rho} + 1) \tag{3.26}$$

independent of the size of the inverters. Figure 3.9 shows the variations of P, E, and F as a function of r for a chain of five inverters. The results are obtained both through the model and through HSPICE simulations; good agreement between the two is observed. The period curve is almost flat at the minimal value of about 2.2 ns for a wide range of r. While the maximum variation of the delay in the flat region is less than 1.4%, the energy variation is 45% in the same region. Hence, by choosing a value of r close to the lower end of the flat region, the energy saving is huge while the delay cost is negligible. Of course, this can only be done where poor rise time is tolerable. Designers usually design for equal rising and falling delays by setting $r = \rho$. In an energy-efficient design environment, this practice should certainly be limited to the exceptional spots. For instance, in this technology at $r = \rho$ although the delay is close to its minimum, the energy is 1.5 times the case when $r = 1$. The energy-delay product curve is also almost flat at the minimal value of 10.5 pJ-ns for the

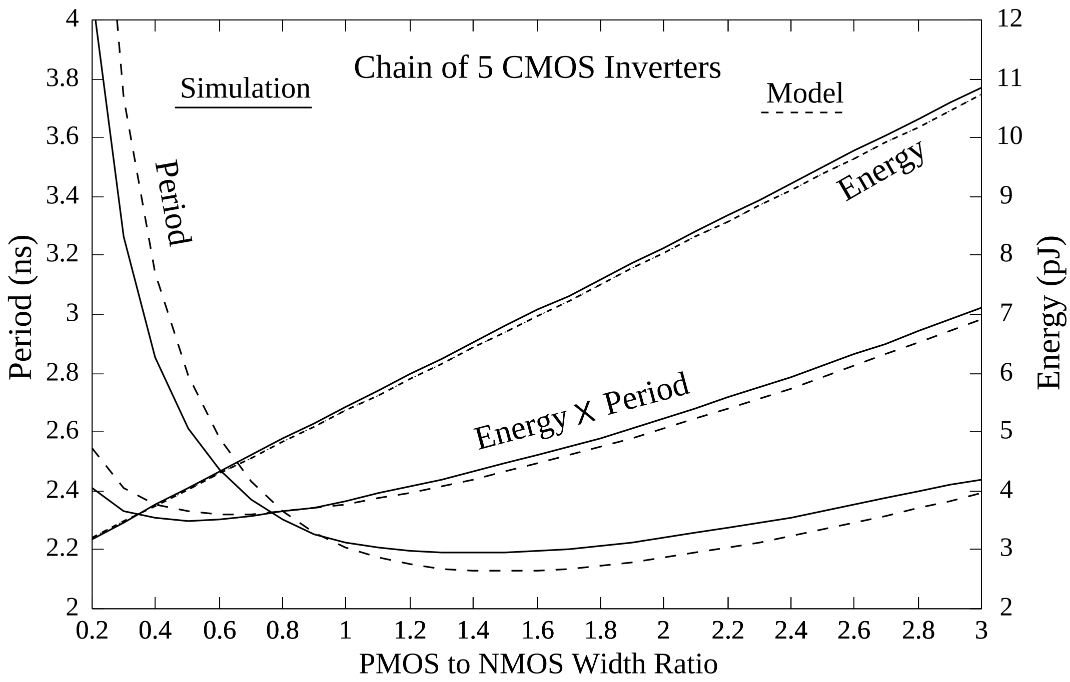


Figure 3.9: Variations of the delay (period P), energy E, and energy-delay product F in an inverter chain as a function of the PMOS to NMOS transistor size ratio r based on simulations and model

range of r between 0.4 and 0.7. The maximum variation of F in this range is less than 4% while variations of P and E for the same range are 17% and 18%, respectively. Thus, by setting $r = 0.7$ instead of 0.4, the designer is gaining the same fractional amount in the delay as losing in the energy, whereas F still remains very close to its minimum. Therefore, we conclude that if the optimization criterion is the delay, it is preferable to choose r close to the lower end of the flat delay curve range because saving in energy is considerable. However, if the optimization criterion is the energy-delay product, it is preferable to choose r close to the upper end of the flat F curve range, because gaining in the delay equals the energy cost. Overall, for energy-efficient designs, $r = 1$ seems a very reasonable choice.

3.3.2 Tapered Buffer Design

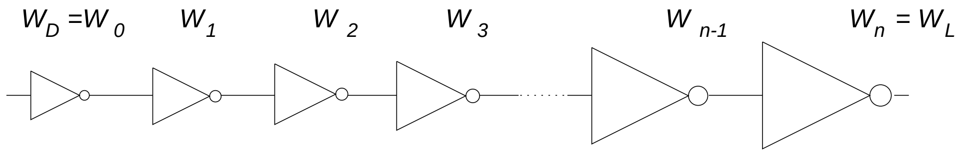


Figure 3.10: Tapered buffer circuit for driving large load.

Figure 3.10 illustrates the classical problem of driving a large load by a group of cascaded buffers. The size of the first stage buffer is known. The load is also known and represented by a buffer. We are interested in finding the optimal number of stages n and the size of each stage to minimize the total delay. Assume that all of the buffers have the same r. Usually it is also assumed that the buffers decrease in size from W_n to W_0 by a constant tapering factor β and, hence, the name tapered buffer chain [53]. Here, however, we don't make such an assumption and start by applying Equation 3.10. To minimize the delay, each stage i, where $0 < i < n$, must satisfy $W_i = \sqrt{W_{i-1}W_{i+1}}$, which can be rewritten as

$$\frac{W_i}{W_{i-1}} = \frac{W_{i+1}}{W_i} = \beta = \left(\frac{W_L}{W_D}\right)^{\frac{1}{n}} \tag{3.27}$$

Thus, the constant tapering factor β appears naturally as a necessary condition rather than an assumption. In [89], this condition is proven using the fact that the arithmetic average of N positive integers is always larger than their geometric average. The total delay can be expressed as

$$D = \delta\beta n + \delta' n \tag{3.28}$$

Solving $\frac{\partial D}{\partial n} = 0$ for optimum n yields

$$n = \frac{\ln \frac{W_L}{W_D}}{\Omega\left(\frac{\delta'}{\delta}e^{-1}\right) + 1} \tag{3.29}$$

where $\Omega(x)$ is the Omega or Lambert W function [24] satisfying

$$\Omega(x)e^{\Omega(x)} = x \tag{3.30}$$

The Omega function is included in some computational tools such as Maple. The optimum β as obtained by substituting 3.29 into 3.27 is given by

$$\beta = e^{\Omega\left(\frac{\delta'}{\delta}e^{-1}\right)+1} \tag{3.31}$$

This result is, in fact, equivalent to those obtained by authors who included the effect of the diffusion capacitances of the buffers, e.g. [53], although none of them has mentioned the relation to the Omega function. Figure 3.11(bottom) shows a plot of β as a function of δ'/δ.

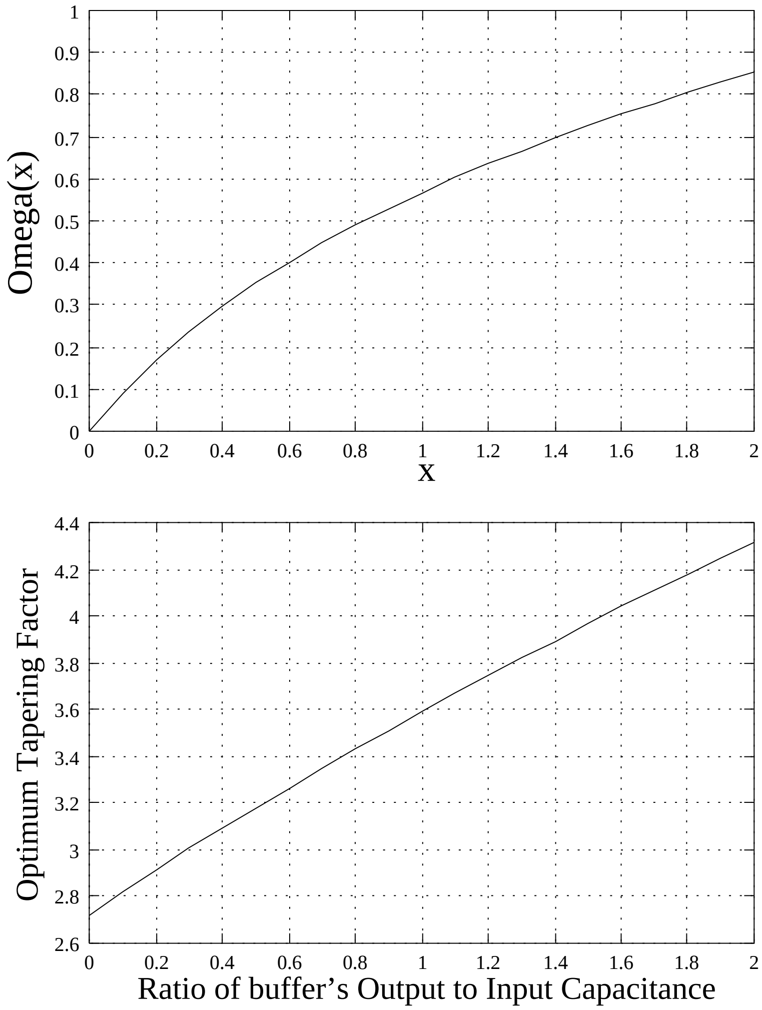


Figure 3.11: The Omega function $\Omega(x)$ (top) and the optimum tapering factor β as a function of the buffer's intrinsic to output capacitance δ'/δ (bottom).

3.3.3 Generation of Complementary Signals

Sometimes a designer may want to generate a pair of complementary control signals, for instance, two phases of a clock signal. In such cases it is important that both signals arrive at the load almost simultaneously. With reference to Figure 3.12, assume that using the signal at the input of the driving buffer of size W_D, we want to produce a pair of corresponding complementary signals at a pair of equal loads represented by two buffers of size W_L each. The values of W_D and W_L are known.

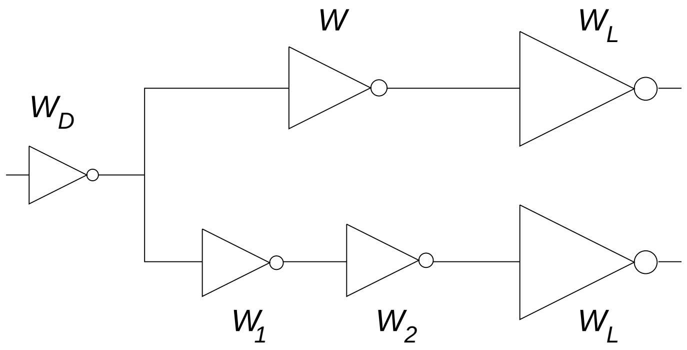


Figure 3.12: Circuit for producing complementary signals.

Let us denote the delay through the upper path D_u and the delay through the lower path D_l, both of which are obtained using Equation 3.9.

$$D_u = \delta\left(\frac{W}{W_D} + \frac{W_1}{W_D} + \frac{W_L}{W}\right) + 2\delta' \tag{3.32}$$

$$D_l = \delta\left(\frac{W}{W_D} + \frac{W_1}{W_D} + \frac{W_2}{W_1} + \frac{W_L}{W_2}\right) + 3\delta' \tag{3.33}$$

From the previous arguments we know that D_l is minimized if

$$\frac{W_1}{W_D} = \frac{W_2}{W_1} = \frac{W_L}{W_2} = \beta = \left(\frac{W_L}{W_D}\right)^{1/3} \tag{3.34}$$

From which we obtain W_1 and W_2. Next, we solve $D_l - D_u = 0$ for W which results in

$$W = \frac{\delta W_L}{2\beta\delta + \delta'} \tag{3.35}$$

For example, if $W_L = 20$ μm and $W_D = 5$ μm, the above formulas give $W = 4.51$ μm, $W_1 = 7.93$ μm, and $W_2 = 12.60$ μm. HSPICE Simulation shows that with these values, D_u

and D_l are apart by less that 10%. Note that this is only a first-order model and the fine tuning is still left to the schematic and layout simulations.

3.4 Extracting Delay and Energy Parameters

A circuit simulator such as HSPICE can be used to extract the constants in the delay and energy expressions. Alternatively, they can be calculated using the standard equations for the capacitances and saturation currents. We have chosen the former method using a chain of five inverters. Although using a chain of three inverters seems equally appropriate and simpler, the parameters obtained in this way may be less accurate due to the possibility of the partial-swing behaviour of the signals. For $V_{DD} = 3$ V with the 0.8 μm BiCMOS technology we have used, the values of the constants are listed in Table 3.1. The values of δ, δ', and δ'' can be calculated using this table and Equations (3.7) and (3.8). Similarly, the values of η, η', and η'' can be calculated using the table and Equation (3.20).

Table 3.1: Delay And Energy Parameters Extracted for a 0.8 μm Bicmos Technology at $V_{dd} = 3$ V and $r = 2$

Delay	**Energy**
$K_D = 0.0256$ ns	$K_E = 0.0176$ pJ/μm
$K_D' = 0.0319$ ns	$K_E' = 0.0207$ pJ/μm
$K_D'' = 0.0194$ ns-μm	$K_E'' = 0.0104$ pJ

The parameters can be obtained through the following three-step simulations:

- Simulate the inverter chain using large transistors, e.g. $W = 100\ \mu$ m, so that the effects of δ'' and η'' can be neglected. Measure the period of oscillation $D1 = 5\delta + 5\delta'$ and the energy $E1 = 5W\eta + 5W\eta'$.

- Insert an inverter load of size $W/5$ at each node of the chain, as shown in Figure 3.13 with the dashed lines, and repeat the simulation. Measure the period of oscillation $D2 = 6\delta + 5\delta'$ and the energy $E2 = 6W\eta + 5W\eta'$. Calculate the values of δ, η, δ', and η'.

- Remove the loads and simulate the chain with small transistors, e.g. $W = 2\mu m$, to capture the effects of δ'' and η''. Measure the period of oscillation $D3 = 5\delta + 5\delta' + (10/W)\delta''$ and the energy $E3 = 5W\eta + 5W\eta' + 5\eta''$. Calculate the values of δ'' and η''.

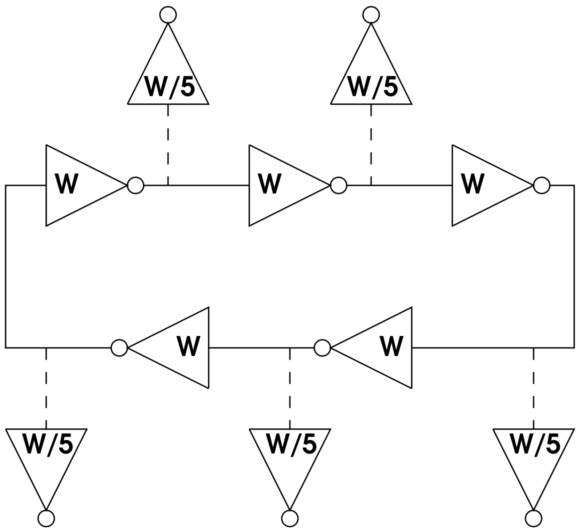


Figure 3.13: The circuit used to extract the delay and energy parameters. through simulation

3.5 Concluding Remarks

In this chapter, we have introduced a formulation for the delay of a CMOS gate based on the widths of the transistors of the gate itself, the preceding gate (driver), and the succeeding gate (load) in a path. We argued that in most cases the widths of the transistors are the only parameters a designer can adjust and, hence, it is important to have an adequate model explicitly expressing the change in performance as a function of the widths of the transistors. This model is intended for fast evaluation of circuit performance, providing appropriate initial guess for starting running a circuit simulator or optimization tool, giving insight into simulation results. The model is kept simpler by ignoring some secondary effects. These effects were added later to improve the accuracy and suitability for CAD tools. Using this model, we have derived expressions for optimal transistor sizing for minimizing the total (average) delay, rising delay, falling delay, and energy-delay product. To demonstrate the convenience of using the model, we applied it to the classical problem of driving large loads and designing a circuit for generation of complementary signals.

Chapter 4

Modeling and Comparing Single-Rail CMOS Implementations of the C-Element

Various applications have demonstrated that asynchronous circuits have great potential for low-power and high-performance design. One of the most primitives frequently used in asynchronous control circuits is the C-element [92]. For instance, the C-element plays an important role in the asynchronous, micropipeline-based version of the ARM microprocessor, AMULET [37]. In this chapter[1] we compare and give optimizations of the most popular CMOS implementations of the C-element using the formulation of Chapter 3. This chapter deals only with non-differential implementations of the C-element, which all happened to be of the single-rail type. Hence, we also refer to them as the single-rail implementations of the C-element. A study on differential (double-rail) implementations of the C-element appears in the next chapter. Sometimes the single-rail implementations seem to be favoured over the differential implementations, because they reduce the wiring requirements and eliminate the potential problems associated with the difference in the propagation delays of the complementary signals.

The CMOS implementations studied here differ in topology for maintaining the state

[1]The content of this chapter has also appeared in [81] and [83].

of the output when the gate is idle. One single-rail, conventional implementation of the C-element has been introduced by Sutherland [92] and is used often in high-performance micropipelines; a second implementation has been introduced by Martin and has been used in the Caltech asynchronous microprocessor [57] and an asynchronous, low-power version of the ARM developed at Manchester University [37]; a third implementation has been introduced by Van Berkel and is being used in the TANGRAM silicon compiler for low-power design at Philips Research Laboratories [5, 6]. The implementations are evaluated with respect to delay, energy consumption, and silicon area using analytical approximations and HSPICE simulations. Simulations are performed for two typical test environments: one to measure the individual performance of the particular implementation and to evaluate the effect of its input capacitance and driving ability, and another one to measure the group performance of a series of cross-coupled C-elements, where the performance of individual elements are mutually dependent.

First, we introduce the C-element and its various single-rail implementations. Then, The first-order model of Chapter 3 is applied to each implementation followed by a comparative analysis. Next, the two test environments and the significance of each are described. For each environment, a comparative study is performed using HSPICE simulations. The simulations support the results deduced from the first-order analysis. Finally, we conclude by indicating the most suitable single-rail CMOS implementation for an energy-efficient, high-speed design and an intuitive justification of the results.

4.1 The C-element

The C-element has been introduced by D. E. Muller [65] and is therefore also called the "Muller C-element." A C-element has two inputs a and b and one output c. Traditionally, its logical behaviour has been described as follows. If both inputs are 0 (1) then the output becomes 0 (1); otherwise the output remains the same. For the proper operation of the C-element, it is also assumed that once both inputs become 0 (1), they will not change until the output changes. A state diagram is given in Figure 4.1 along with a commonly used schematic.

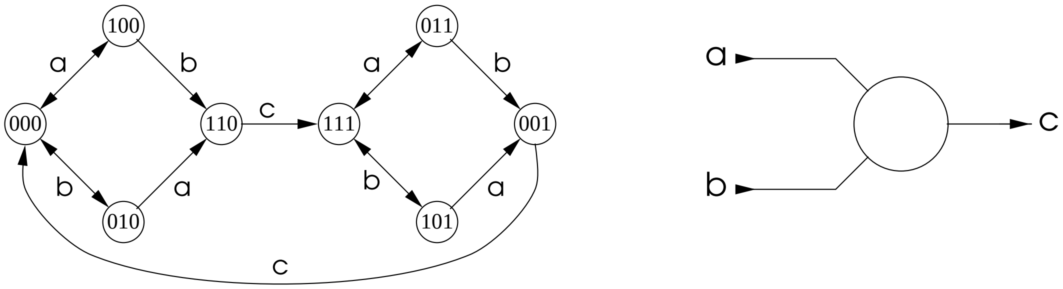


Figure 4.1: State diagram and schematic of the C-element.

The behavior of the output, c, of a C-element can be expressed in terms of the inputs a and b and the previous state of the output, $\hat{c}$, by the following Boolean function

$$c = \hat{c} \cdot (a + b) + a \cdot b$$

The C-element is closely related to the JOIN [34], or RENDEZVOUS [20]. The JOIN has a slightly more restrictive environment behaviour in the sense that an input is not allowed to change twice in succession. So in the state graph, the bidirectional arcs should be replaced by unidirectional arcs. One could also say that the JOIN performs the AND operation of two events, where an event could either be a rising or a falling voltage transition. Thus, a JOIN produces an output transition only after both inputs have received a transition. In asynchronous circuits, a C-element is most often operated as if it was a JOIN. In fact, any C-element implementation can be used to realize a JOIN. In this chapter, we assume that environment behaviour satisfies the restrictions for the JOIN environment.

4.2 Dynamic C-element

The single-rail dynamic C-element of Figure 4.2 constitutes the basic functional part of a static single-rail implementation of the C-element. The static versions, thus, differ only in mechanisms for preserving the state of the output. Since these mechanisms insert additional parasitics, we expect the dynamic C-element to be faster and less energy consuming than the static versions. The design parameters in this circuit are the main body size W, the output inverter size U, and r.

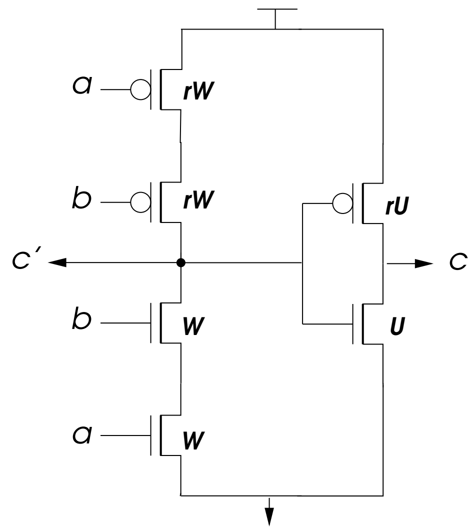


Figure 4.2: Dynamic implementation of the C-element.

In order to derive the delay of the dynamic C-element, assume the general environment of Figure 3.4, such that the C-element is driving an inverter load of W_L (i.e. NMOS width is W_L and PMOS width is rW_L) at the output node c and W_L' at node c'. The delay, hence, is governed by

$$\begin{aligned} D_D &= \delta\left(\frac{W}{W_D} + \frac{U + W_L'}{0.5W} + \frac{W_L}{U}\right) + \delta'\left(\frac{W_D}{W_D} + \frac{W}{0.5W} + \frac{U}{U}\right) \\ &= \delta\left(\frac{W}{W_D} + 2\frac{U + W_L'}{W} + \frac{W_L}{U}\right) + 4\delta' \end{aligned} \tag{4.1}$$

which has been intentionally written out in some detail so that the reader can verify. The energy and area are simply as follows

$$E_D = \eta(2W + U) + \eta'(W + U) \tag{4.2}$$

$$A_D = \theta(2W + U) \tag{4.3}$$

We compare the delay, energy, and area of the dynamic C-element to the values for the static implementations to measure the deviations from an ideal case. Also, we use the dynamic C-element to find the optimal design parameters in a given environment and use the same values for the static C-element implementations.

4.3 Standard Implementation of the C-element

The *standard* pull-up pull-down static realization of the C-element is shown in Figure 4.3. This circuit has been presented in [92] by Sutherland. This implementation is ratioless, i.e. it does not impose any restrictions on the sizes of the transistors. From the operation of the circuit, we conclude that N1, N2, and N6 are the main pull-down transistors which contribute to output switching; they are of size W, W, and U, respectively. Whereas N3, N4, and N5 only provide the necessary feed-back to hold the state of the output when values of the inputs do not match; hence, they are made as small as possible to reduce their loading effect. Similarly, the feed-back transistors P3, P4, and P5 have minimum width w, while P1 and P2, and P6, the normal pull-up transistors, have widths rW, rW, and rU, respectively.

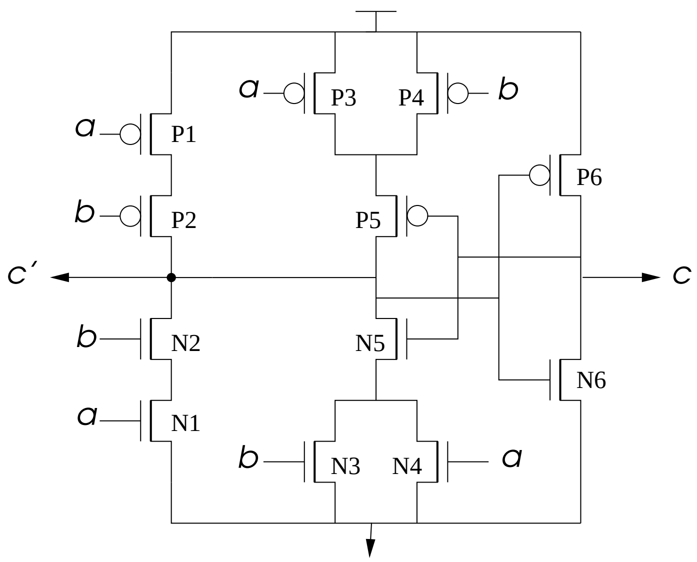


Figure 4.3: Standard implementation of the C-element [92].

The delay of the standard implementation can be expressed by

$$
\begin{aligned}
D_C &= \delta\left(\frac{W}{W_D} + \frac{U + W_L'}{0.5W} + \frac{W_L}{U}\right) + \delta'\left(\frac{W_D}{W_D} + \frac{W}{0.5W} + \frac{U}{U}\right) \\
&\quad + K_D\left(\frac{2\ w}{W_D} + \frac{2\ w\ \rho}{W_D\ r} + \frac{2\ w}{U} + \frac{2\ w\ \rho}{U\ r}\right) \\
&\quad + K_D'\left(\frac{2\ w}{0.5W} + \frac{2\ w\ \rho}{0.5W\ r}\right)
\end{aligned}
$$

$$
\begin{aligned}
&= D_D + 2K_D\left(1+\frac{\rho}{r}\right)\left[\frac{w}{W_D}+\frac{w}{U}\right] \\
&\quad +4K'_D\left(1+\frac{\rho}{r}\right)\left[\frac{w}{W}\right]
\end{aligned} \tag{4.4}
$$

which has two extra terms compared to that of the dynamic implementation. The two terms represent the cost of the output-state-holding mechanism and vanish as W_D, W, and U are increased such that w becomes negligible. The energy and area, as given below, also become approximately equivalent to those of the dynamic implementation for large W and U.

$$
\begin{aligned}
E_C &= \eta(2W+U)+\eta'(W+U)+K_E\ 6\ w+K'_E\ 2\ w \\
&= E_D + 6\ w\ K_E + 2\ w\ K'_E
\end{aligned} \tag{4.5}
$$

$$
A_C = \theta(2W+U) + K_A\ 6\ w = A_D + 6\ w\ K_A \tag{4.6}
$$

4.4 Implementation of the C-element with Weak Feedback

The C-element implementation illustrated in Figure 4.4 has been presented by Martin [55]. This circuit utilizes an inverter latch to maintain the state of the output when the inputs do not have the same logic level. Hence, in this chapter it is referred to as the *Weak Feedback* Implementation. The circuit suffers from a race problem at node c'.

There is an inherent resistance to switching the state of the latch that can be reduced, but cannot be eliminated. For a proper operation of the circuit, certain size ratios must be imposed on the transistors. The feed-back inverter should be a weak one to allow changes in the state of the latch. The race problem reduces as the resistance of the feed-back inverter transistors increases. This can be done by increasing the length of these transistors with respect to their width. This solution, however, increases the load at the output and makes node c' more susceptible to noise. Therefore, minimum size transistors are chosen for this inverter.

The delay of this circuit has extra components due to the race problem at node c'; it can

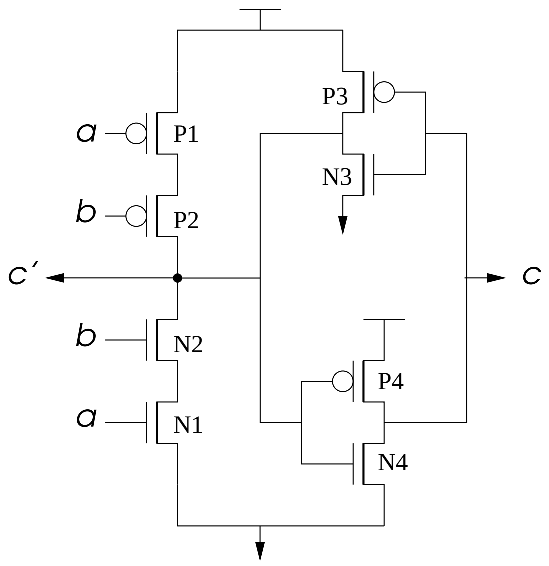


Figure 4.4: Implementation of the C-element with weak feedback inverter [55].

be expressed as

$$
\begin{aligned}
D_F &= \delta\left(\frac{W}{W_D} + \frac{W_L}{U}\right) + 2\delta' \\
&+ K_D\left[\frac{(U+W_L')(1+r)}{0.5W - \frac{w}{2\rho}} + \frac{(U+W_L')(1+r)}{0.5W\frac{r}{\rho} - \frac{w}{2}} + \frac{2w}{U} + \frac{2w\rho}{Ur}\right] \\
&+ K_D'\left[\frac{W(r+1)+2w}{0.5W - \frac{w}{2\rho}} + \frac{W(r+1)+2w}{0.5W\frac{r}{\rho} - \frac{w}{2}}\right]
\end{aligned}
\tag{4.7}
$$

The first two terms represent the gate and diffusion delays for the input node a or b and the output, except the output gate delay due to the feed-back inverter. The third term represents the gate delay at node c' and the gate delay at the output due to the feed-back inverter. The fourth term is the diffusion delay at node c'. When node c' is to switch from high to low, a current proportional to $0.5W$ flows to ground through N1 and N2. This current, however, is opposed by a current proportional to $w/(2\rho)$ which flows from P3 into c'. Note that if c was a stable low, the current through P3 would have been w/ρ, and when c goes to a stable high, the current through P3 will become zero. Hence, here we are taking an average value of the current to capture the effect of c being driven high at the same time as c' is falling. The net current discharging c', thus, is $0.5W - w/(2\rho)$ which appears in the denominators of the appropriate terms in the delay expression. Similarly, the net current charging c' is

$0.5Wr/\rho-(w/2)$ as it appears in the delay expression. Hence, the delay expression demands that $W > w/\rho$ for N1 and N2, and that $W > \rho w/r$ for P1 and P2 for the output low-to-high and high-to-low switchings to take place, respectively. Comparing the delay expression with that of the dynamic implementation, we realize that the third term has replaced $2\delta U/W$ and the fourth term has replaced $2\delta'$. The third and fourth terms are reduced to the mentioned terms, respectively, if $w = 0$.

The energy expression for the weak feedback implementation, stated below, consists of two parts: the useful energy, E^u, spent on charging and discharging the nodes, and the energy wasted in the race at c', E^w.

$$
\begin{aligned}
E_F &= E_F^u + E_F^w && (4.8)\\
E_F^u &= E_D + 2w(K_E + K_E') && (4.9)\\
E_F^w &= [K_E(U + W_L')(1+r) + K_E'(W(r+1)+2w)] \\
&\quad \times \left(\frac{\frac{w}{2\rho}}{0.5W - \frac{w}{2\rho}} + \frac{\frac{w}{2}}{0.5W\frac{r}{\rho} - \frac{w}{2}}\right) && (4.10)
\end{aligned}
$$

The useful energy is the energy of the dynamic implementation plus the energy spent on the feed-back inverter. In general, the wasted energy at a node with a total load capacitance C can be approximated by

$$
\begin{aligned}
E^w &= I^w V_{DD} D^w = I^w V_{DD}\frac{CV_{DD}}{I} \\
&= \frac{I^w}{I}(K_E W_g + K_E W_d) && (4.11)
\end{aligned}
$$

Where, I^w is the wasted current, D^w is the duration (delay) of wasting energy, I is the average net current flow during D^w, and W_g and W_d are the gate and diffusion components of C. Applying this to node c', expression (4.10) is obtained. According to this expression, the wasted energy decreases as W increases, because the duration of wasting energy gets smaller.

The weak feedback implementation has a device count of 8 compared to the 12 of the standard implementation. It also has a smaller area by $4wK_A$.

$$A_F = A_D + 2wK_A \quad (4.12)$$

A modified version of the weak feedback implementation is shown in Figure 4.5. In this circuit, the feed-back inverter has two additional transistors, N5 and P5. These transistors are weak and skinny acting as resistances to limit the current flow and reduce the race problem during switching. We refer to this circuit as the *Resistive* Implementation of the C-element. We have implemented N5 and P5 with a channel length of $L' = 2\mu$ (instead of the minimal 0.8μ) and the minimum channel width $w = 1.4\mu$. A disadvantage of using these skinny transistors is that node c' becomes more sensitive toward noise.

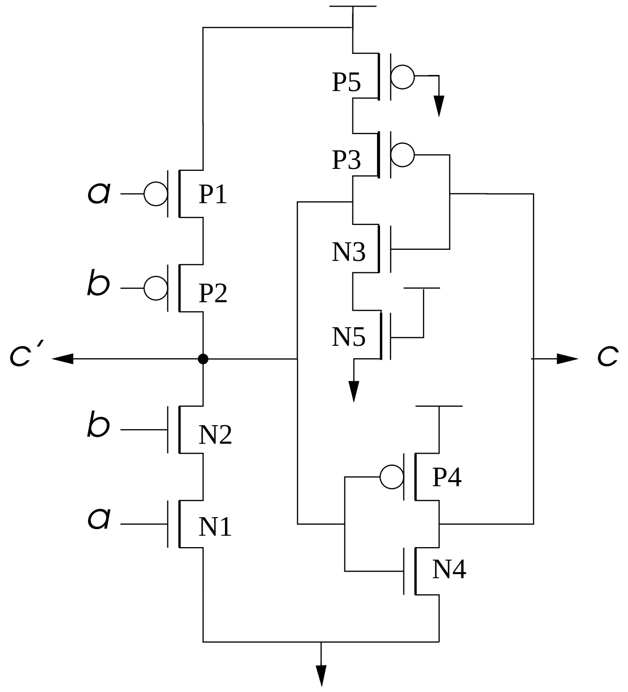


Figure 4.5: Implementation of the C-element with resistive inverter at the feedback.

By inserting N5 and P5, the pull-up and pull-down currents through the feed-back are scaled by a factor of h given by

$$h = \frac{L}{L + L'} \tag{4.13}$$

Therefore, the delay and energy expressions for the resistive implementation can be easily obtained by modifying the corresponding expressions for the weak feedback implementation, i.e., by replacing all the occurrences of $w/2$ by $wh/2$ and those of $w/(2\rho)$ by $wh/(2\rho)$. Since h is less than unity, one can verify that D_R is less than D_F and that E_R is less than E_F.

The area of the resistive implementation, however, increases compared to the weak feed-back implementation by $4w(Z + L')$ as a penalty for the modification made.

$$A_R = A_D + 2K_A w + 2(K_A + L' - L)w \tag{4.14}$$

4.5 Symmetric Implementation of the C-element

The C-element implementation by Van Berkel [2] is shown in Figure 4.6. In this circuit, the output state is maintained through a feed-back conducting path of three transistors in the pull-up tree or the pull-down tree. If the input values are not equal and the output is low, the conducting path consists of either P1, P3, and P5, or P4, P3, and P2. By symmetry, when the output is high, a path of N1, N3, and N5, or N4, N3, and N2 is responsible for latching the output. Two parallel paths of P-devices or N-devices contribute to output switching. During output rising, P1 and P2 in parallel to P4 and P5 are conducting. During output falling, N1 and N2 in parallel to N4 and N5 are conducting. Similar to the standard implementation, this circuit is also ratioless. An advantage of this implementation is that it is symmetric with respect to the inputs and, thus, we call it the *Symmetric* implementation of the C-element. For the circuit to have the same pull-up and pull-down resistances (when switching) as the previous implementations, the normal N-tree and P-tree transistors, except those of the output inverter, must be made half the size (i.e. $W/2$). The feed-back transistors N3 and P3 are, as usual, of minimum size, and N6 and P6 have a normal size to achieve the load driving capability of the previous circuits.

The delay, energy, and area of the symmetric implementation are governed by the following expressions.

$$D_S = D_D + 2K_D \left(1 + \frac{\rho}{r}\right) \frac{w}{U} \tag{4.15}$$

$$E_S = E_D + 2wK_E \tag{4.16}$$

$$A_S = A_D + 2wK_A \tag{4.17}$$

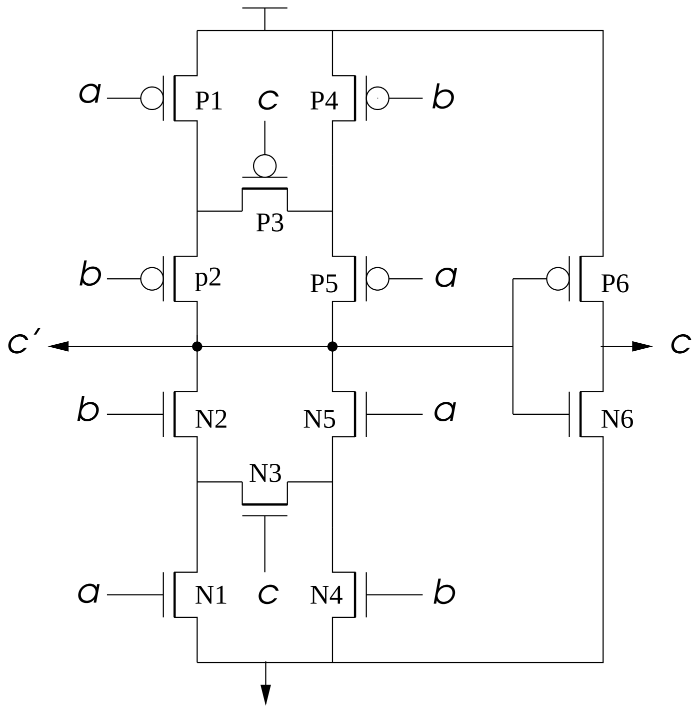


Figure 4.6: Symmetric implementation of the C-element [2].

The delay expression has one additional term compared to the dynamic implementation. Interestingly, this extra term also appears in the delay expressions of the standard, weak feedback, and resistive implementations. Hence, for the same value of W, the symmetric implementation is expected to have the lowest delay compared to the other static implementations. The very same argument applies to the energy when compared to the energy expressions of the other static implementations. The area of the symmetric implementation equals that of the weak feedback implementation and is less than those of the resistive and standard implementations. The symmetric implementation has a device count of 12, equal to that of the standard implementation and higher than the weak feedback and resistive implementations by 4 and 2, respectively.

Our analysis shows that the symmetric implementation is the best candidate for energy-efficient, high-speed designs, as it gives the least delay per unit energy among the static implementations. Our model also indicates that as W and U are increased, eventually, the delay and energy of all of the static implementations coincide with the asymptotic delay and energy of the dynamic implementation.

4.6 Effect of Arriving-Time Order of Inputs

As mentioned before, the dynamic C-element structure is common to all static C-element implementations considered here; hence, the following arguments are made in reference to Figure 4.2. Let's name the internal node between the two NMOS transistors "N", and the internal node between the two PMOS transistors "P". To produce an output, there are two possible orders of input events: a arrives before b and b arrives before a. Assume, initially, the inputs are low, thus c' and c are high and low, respectively. If a rises first, node N is discharged; and when b rises, c' goes low with minimal body effect. If b rises first, however, charge distribution takes place between nodes N and c'. In the dynamic C-element, the charge lost in c' cannot be replaced, and the consequence is that when a rises, c' is discharged relatively slower due to the higher body effect. The scenario is worse in the static C-elements in the case of b arriving after a, because the charge lost in c' is gradually replaced through the keepers and N can charge up to $V_{DD} - V_{TN}$. Hence, the total effective charge in switching could increase by $C_N(V_{DD} - V_{TN})$, where C_N is the total capacitance at N. This of course, increases the delay and energy consumption. A similar argument applies to node P in the process of discharging the output, if b falls before a does. Again, the effective charge in switching could increase by $C_P(V_{DD} - V_{TP})$, where C_P is the total capacitance at P.

The extra delay per cycle, ignoring the body effect, can be approximated by

$$D_O = K'_D \left(\frac{2W}{W} + \frac{2rW}{rW/\rho} \right) v = 2K'_D(1+\rho)v. \tag{4.18}$$

where $v = (V_{DD} - V_t)/V_{DD}$. The extra energy per cycle, can be expressed by

$$E_O = v\eta'(2W) \tag{4.19}$$

The effect of the order of input arrival on the keepers is usually negligible due to the small size of their transistors.

4.7 First Test Environment

Figure 4.7 illustrates the first measurement setup. The C-element is driven by two inverters which are fed by the ideal sources V_a and V_b. At the output, the C-element is driving a

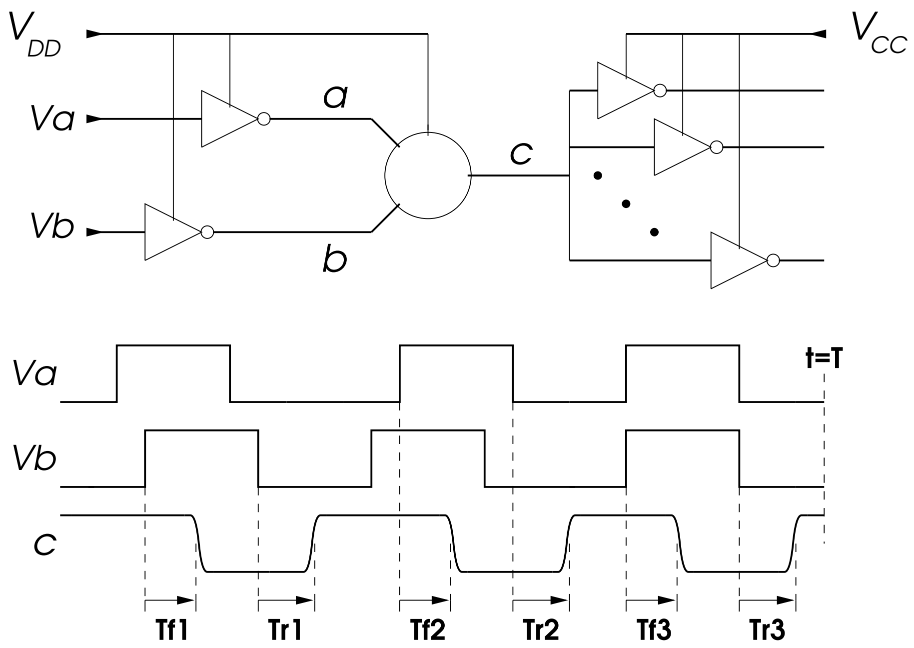


Figure 4.7: First measurement setup: testing for optimal sizing for known fan-out.

number of inverters similar to the input inverters in size. The NMOS and PMOS transistors of the inverters are 4 and 10 microns wide, respectively.

The input inverters and the C-element gate are supplied with the voltage source V_{DD}, whereas the fan-out inverters have a separate power supply, V_{cc}. This is to make sure that, when measuring the energy consumption, the effect of the input capacitance of the C-element and its own internal parasitics are accounted for. The typical testing input waveforms and the expected output waveform are also depicted in Figure 4.7. The delay through the C-element depends on the order and arriving times of the input signals. Exhaustive simulations are needed to find the worst-case delay by altering the arriving interval between the input signals. Simulations have shown that in the case of the C-element, the worst-case delay is obtained when one input arrives shortly (less than 1 ns) after the other. A slightly less accurate, but faster technique is to be only concerned with the arriving order of the input signals. As shown in the figure, using this method, six cases are considered and the respective delays produced are measured: V_a rises before V_b gives T_{f1}, V_a falls before V_b gives T_{r1}, V_b rises before V_a gives T_{f2}, V_b falls then V_a gives T_{r2}, V_a and V_b simultaneously rising gives T_{f3}, and simultaneously falling gives T_{r3}. Accordingly, the worst-case fall time D_f, the worst-case rise time D_r, and the overall worst-case delay D, are obtained by the following.

$$\begin{aligned} D_f &= \max(T_{f1}, T_{f2}, T_{f3}) \\ D_r &= \max(T_{r1}, T_{r2}, T_{r3}) \\ D &= \max(D_f, D_r) \end{aligned}$$

In cases where one input changes before the other, the time difference between them is 5 ns.

The energy consumption of the C-element per output transition also depends on the arriving times and order of the inputs. We measure the average energy consumption per output cycle E, corresponding to the illustrated waveforms, from the following.

$$E = P(V_{DD})\frac{T}{3} \tag{4.20}$$

where $P(V_{DD})$ is the average power extracted from V_{DD}, and T is the total period of the output waveform. The SPICE simulations for this setup were performed at a frequency of 40 MHz.

Assume that we would like to design the C-elements for a standard cell library and test them in the environment of Figure 4.7. We chose $r = 2.5$ very close to the value of ρ (i.e. 3.08 in our technology) to obtain almost equal rise and fall times. We need to decide upon the optimal value of U. For this purpose, we use the dynamic implementation as reference point and take its delay equation as in (4.1).

$$D_D = \delta\left(\frac{W}{W_D} + 2\frac{U + W_L'}{W} + \frac{W_L}{U}\right) + 4\delta' \tag{4.21}$$

This yields the following value for optimum U

$$U_{\text{opt}} = \sqrt{\frac{W W_L}{2}} \tag{4.22}$$

We further assume that our optimal design targets a fan-out of two C-elements. Hence, $W_L = 2W$ and $U_{\text{opt}} = W$. The only remaining design parameter is W, which now can be obtained by assuming some value for W_D and a target delay.

The simulations for this environment are performed for a range of $W = 2 \ldots 30$, $U = W$, and fan-out of three inverters. Each inverter has an NMOS transistor width of 4 μm and a PMOS transistor width of 10 μm. Figure 4.8 shows the energy dissipation versus the propagation delay for the C-element implementations. The size of the C-element gate increases for each curve from the right hand side of the graph toward the top of the graph. Thus, one might get two different energy readings for the same delay and implementation, but they correspond to two different sizes.

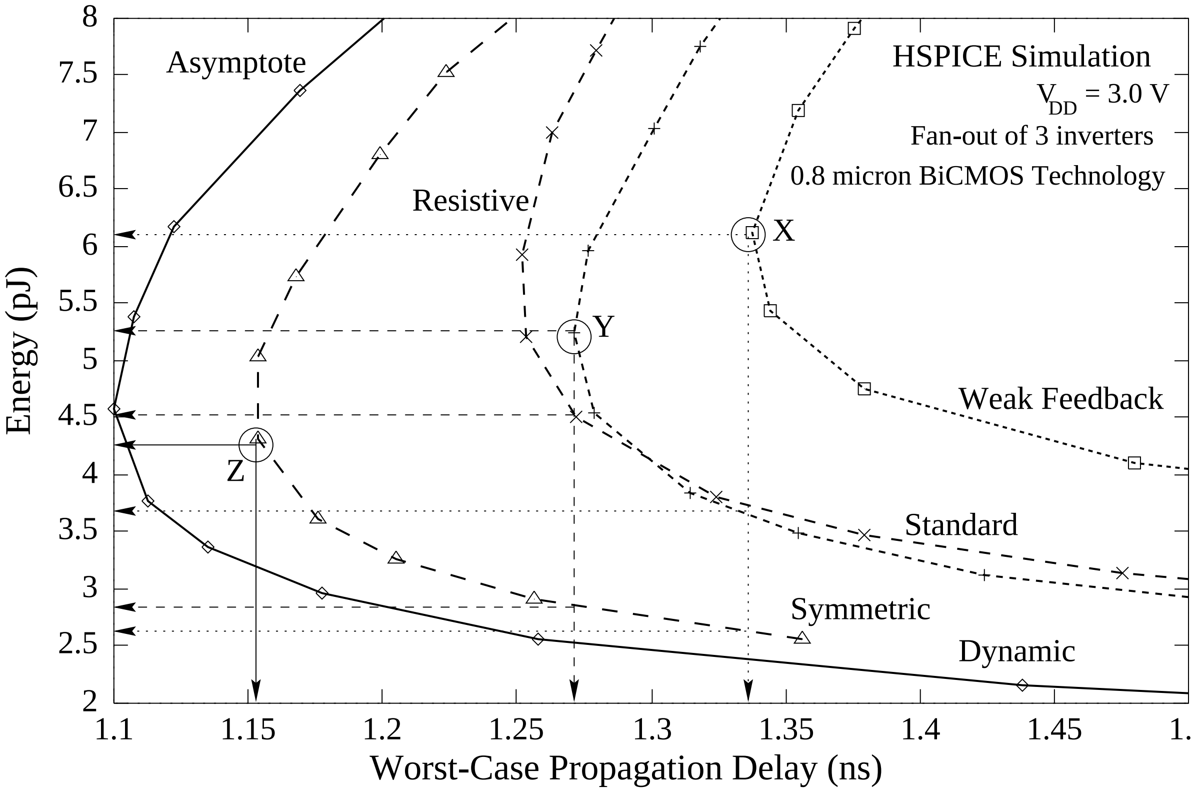


Figure 4.8: SPICE simulation results, energy versus delay, for the C-element gates in the first test setup.

Point "X" in the figure shows the minimum delay achievable using the weak feedback implementation of the C-element. This delay is around 1.34 ns costing 6.2 pJ in energy and 100 μm^2 in area. For the same delay, if we chose the standard implementation (see the cross point between the dotted vertical line and the standard implementation curve), we would spend 3.6 pJ and occupy only 50 μm^2, a saving of 42% in energy and 50% in area. Furthermore, if we chose the symmetric implementation, it would consume only 2.6 pJ and occupy 30 μm^2. A saving of 58% in energy and 70% in area over the weak feedback circuit. Similarly, point "Y", $D = 1.27$ ns and $E = 5.2$ pJ, shows the minimum obtainable delay using the standard implementation with an area of 90 μm^2. The symmetric circuit obtains the same delay for 2.85 pJ and 35 μm^2, saving 45% and 61% in energy and area, respectively. Notice also that delays between 1.15 ns to 1.27 ns can be achieved by the symmetric circuit but not by the other static implementations. The performance of the resistive implementation (with $L' = 2.0$) is very similar to that of the standard implementation.

Energy-delay (E-D) graphs similar to that of Figure 4.8 are convenient to compare alternatives. In such graphs, the outermost curve closest to the origin represents the best choice. The graph demonstrates that the first test is very much in favour of the symmetric implementation of the C-element. Although the minimum delays of the circuits are all within a 10% difference, the symmetric implementation achieves a significant energy savings of about 50% for the same delay. Drawing the E-D graphs for other fan-out values, it is observed that the energy savings of the symmetric implementation for the same delay further increase (with respect to the other implementations) as the fan-out decreases, since energy consumption due to the capacitive load becomes a less significant portion of the total energy consumption. The same is true for energy savings using the standard implementation over the weak feedback implementation.

4.8 Comparing the Model with the Simulation Results

Figure 4.9 shows the Energy-Delay graph of the different C-element implementations based on the model. The calculations include $\frac{1}{3}E_O$, since E_O is only relevant in two out of six input transition cases. For example, for the symmetric implementation at $W = 10$ μm, E_O is over 17% of the total energy per cycle. The delay D calculated by the formula for each implementation is actually the sum of the rising and falling delays. In Figure 4.9 we have plotted $D/2$, which is the average of these delays; thus we have a way of comparing the results obtained from the formulas with the results obtained from the simulations. Notice that the calculated delay D does not distinguish rising and falling delays, nor does it include D_O, the extra delay caused by a difference in arrival time of the inputs. The generated graph is in good agreement with the results obtained by simulations shown in Figure 4.8. Firstly, we observe that the general shape of the curves matches that of the simulations. Secondly, the relative order and position of the curves correspond to those of the simulations; even the

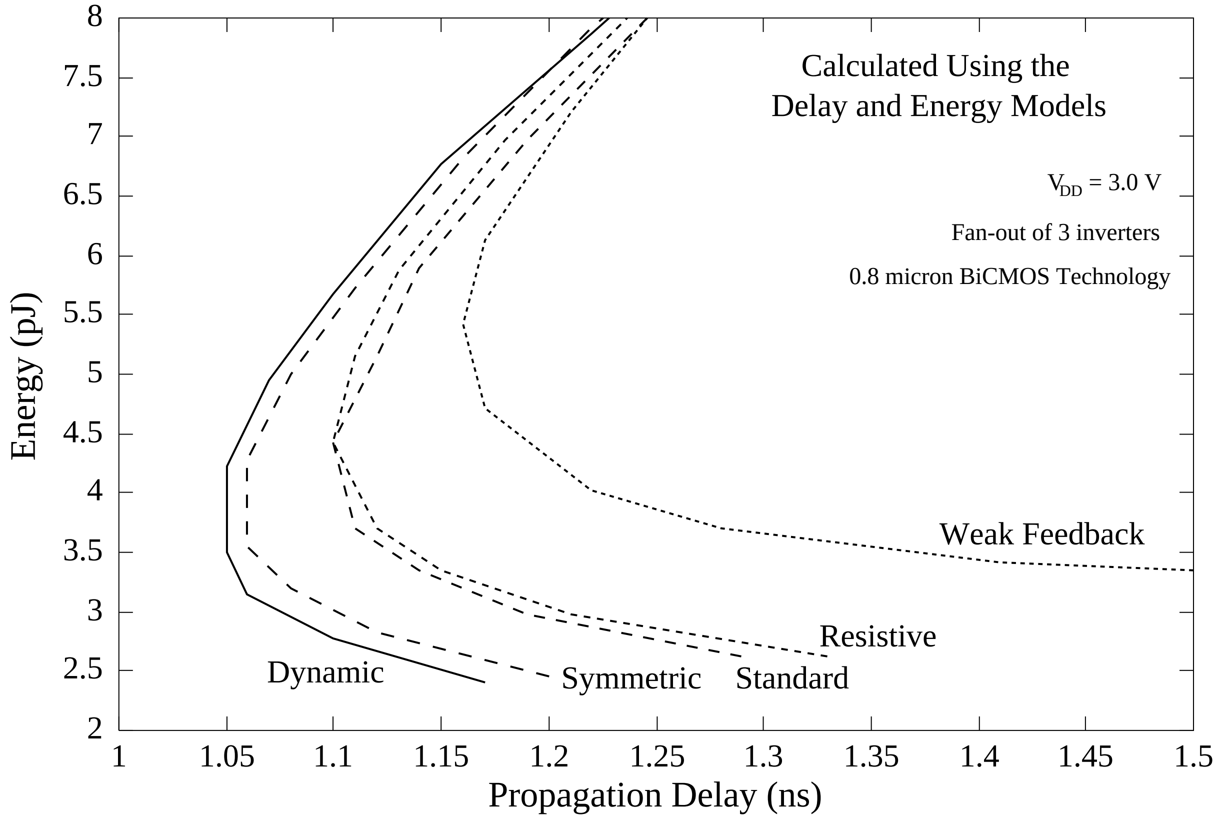


Figure 4.9: Energy-Delay graph of the C-element implementations in the first test environment based on the model.

point where the resistive implementation curve crosses over the standard implementation curve has been captured by the model. Some disagreements in the values of the energies and delays arise from the fact that the simulation results are plotted for the "worst-case" delays. More complicated phenomena affecting the delay are not included in our model, such as the body effect and the impact of the initial conditions at the internal nodes.

A more detailed comparison is found in Table 4.1. In the first column, the basic transistor width, W, is listed. The comparisons are made for a range of relatively (i.e., with respect to the load) small to large widths. Columns 2 to 4, list the average energy, the maximum delay, minimum delay, and average delay (of six cases) obtained through simulations. Notice that the range of the delays for a particular W can be quite large. For example at $W = 3$ μm for the weak feedback implementation, the minimum delay is almost half the maximum delay. Also considerable is that the "best" over all maximum delays (obtained by the symmetric

Table 4.1: Comparing SPICE simulation results with those of the analytical model for the C-element implementations.

W (μm)	SPICE SIMULATIONS AveE (pJ)	MaxD (ns)	MinD (ns)	AveD (ns)	MODEL E (pJ)	D (ns)	A (μm^2)	DIFFERENCE AveE, E %	AveD, D %
Dynamic Implementation									
3.00	2.56	1.28	1.06	1.16	2.49	1.17	182.70	5.88	-1.15
10.00	5.38	1.11	0.95	1.05	4.95	1.07	609.00	7.92	-1.75
25.00	11.28	1.37	1.15	1.28	10.40	1.37	1522.50	7.74	-7.14
Standard Implementation									
3.00	2.78	1.56	1.28	1.40	2.61	1.29	231.42	5.93	8.11
10.00	5.26	1.27	1.06	1.16	5.15	1.12	657.72	1.90	3.77
25.00	10.65	1.41	1.24	1.35	10.61	1.40	1571.22	0.35	-3.89
Weak Feedback Implementation									
3.00	3.55	2.47	1.33	1.87	3.22	1.69	198.94	9.26	9.48
10.00	5.44	1.34	1.06	1.18	5.41	1.16	625.24	0.38	1.67
25.00	10.81	1.47	1.24	1.36	10.82	1.41	1538.74	-0.18	-3.31
Resistive Implementation									
3.00	2.82	1.67	1.25	1.44	2.62	1.33	218.54	6.81	7.44
10.00	5.21	1.25	1.04	1.14	5.15	1.11	644.84	1.11	2.41
25.00	10.66	1.39	1.23	1.33	10.60	1.39	1558.34	0.56	-4.41
Symmetric Implementation									
3.00	2.55	1.36	1.20	1.27	2.45	1.20	198.94	3.58	5.51
10.00	5.03	1.15	1.06	1.11	5.00	1.08	625.24	0.53	3.03
25.00	10.37	1.38	1.29	1.33	10.45	1.37	1538.74	-0.82	-3.35

implementation) are larger than the "worst" over all minimum delays (obtained by the weak feedback implementation). The next 3 columns list the energy, delay, and silicon area calculated using the model. Finally, in the last two columns, we have the percentage difference between the results of the simulations and the model. It is interesting to note the calculated delay value is always within the delay range obtained by simulations, i.e., it is never larger than MaxD or smaller than MinD.

4.9 Second Test Environment

The second measurement setup is shown in Figure 4.10. The C-elements in the chain are all of the same size. This setup differs form the first setup in that the performance of the C-elements are now mutually dependent, which affects the overall performance of the chain. A bubble at the input of a C-element schematic means that an inverted version of that input must be used. This, of course, can be implemented using an inverter. Alternatively, if the input signal is coming from a gate which produces complementary outputs, the complement of the output can be directly used. For this test, the latter technique is chosen. The chain of the C-elements shown, without the inverters at the two ends, form the control circuit of an n-stage micropipeline. The inverters are added to make the micropipeline self-driven and oscillating. The signals at the nodes indicated by $r(\#)$, where "#" represents the stage number, can be interpreted as "request for the next stage". Initially, all nodes are set to low, and the only possible event is the rising of $r(0)$. This logic "one" then propagates through all the request nodes. Meanwhile, other transitions are produced at $r(0)$ which propagate toward the end, in turn. If not interfered, the oscillation of the nodes continues forever. A much simplified waveform is shown in the same figure. The parameters of interest in this test are the throughput and energy per throughput E. The frequency of oscillation of the micropipeline F, which is half its throughput, is the inverse of T. The energy dissipation during this period is

$$E = T\ P(V_{DD})$$

where $P(V_{DD})$ is the average power extracted from V_{DD}.

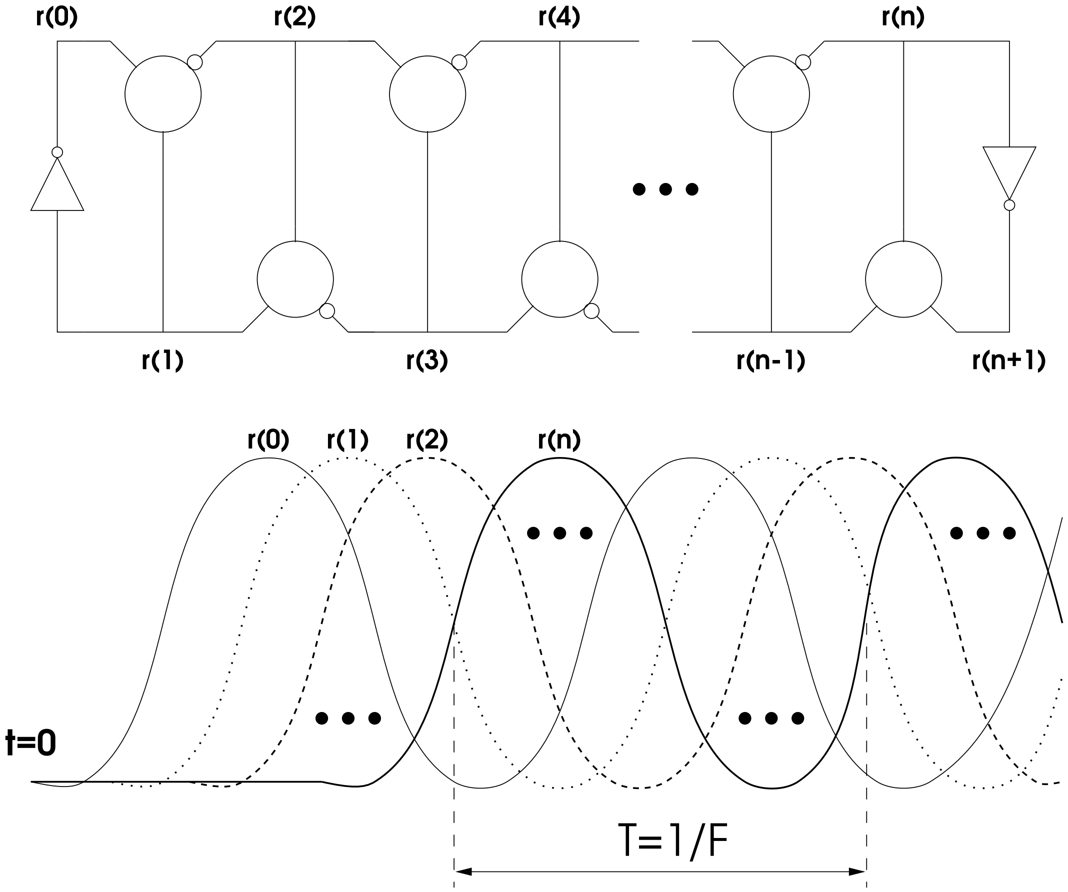


Figure 4.10: Second measurement setup: testing for optimal sizing of a chain structure in presence of feed-back.

Figure 4.11 illustrates the dynamic C-element in the micropipeline test environment. In order to maximize the frequency of operation of the micropipeline, the delay around the boldface loop, i.e. the period, must be minimized. The period can be expressed by

$$T = \delta\left(4\frac{U+W}{W} + \frac{W}{U}\right) + 5\delta' \tag{4.23}$$

and the optimum U obtained is

$$U_{\text{opt}} = \frac{W}{2} \tag{4.24}$$

Thus, the period is reduced to

$$T_{\text{opt}} = 8\delta + 5\delta' \tag{4.25}$$

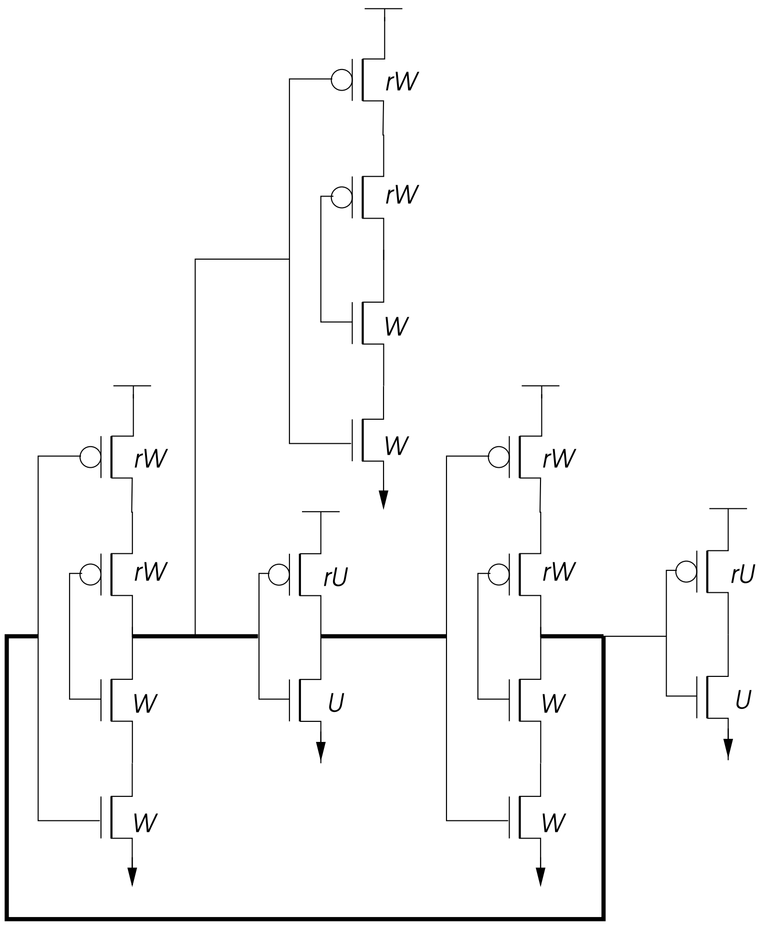


Figure 4.11: The dynamic C-element in the second test environment.

which is independent of W according to our model. However, as we shall see, simulations show that the period does depend on W for small W. This can be partly explained by the approximation made in (3.9). SPICE simulations have confirmed that the optimal value of U for the micropipeline test circuit is $0.5W$ as shown in Figure 4.12. Figure 4.12(left) illustrates the simulation results for $W = 25\mu$, $U = 0.4W$, as r varies. The maximum frequency is achieved around $r = 1.5$. If we want to design a circuit with symmetrical rise and fall times, we would choose r around 3, but simulations show that this increases the energy consumption by 60% and reduces the frequency by 10%. Figure 4.12(right) illustrates the simulation results for $W = 25\mu$, $r = 1.4$, as U varies. The frequency is at its maximum around $U = 0.5W$. Hence, the static C-elements are designed with $r = 1.5$ and $U = 0.5W$.

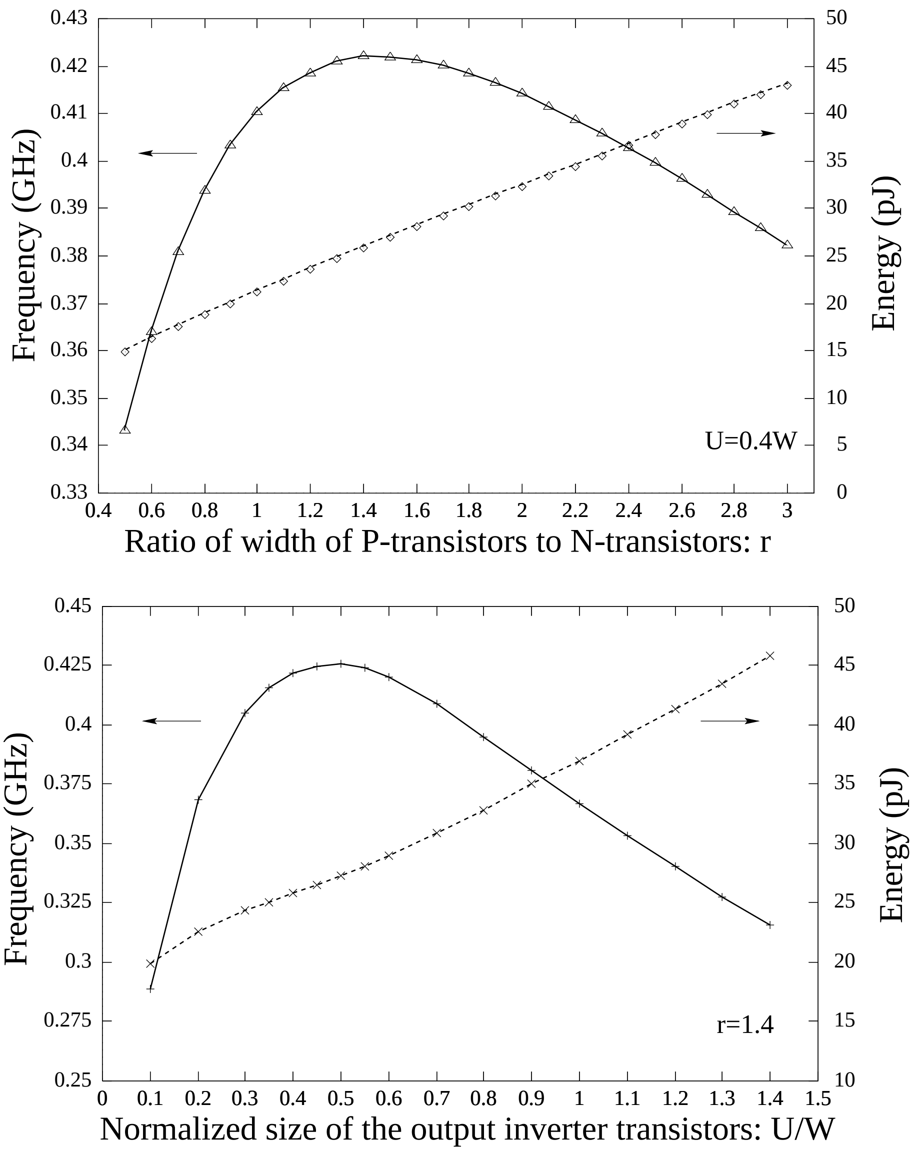


Figure 4.12: Optimization of the dynamic C-element in a micropipeline control circuit.

The results of the simulations for the second test environment are depicted in the form of an energy-frequency (E-F) graph in Figure 4.13. Similar to the E-D graph, in an E-F graph the outermost curve (here furthest from the origin) is the best alternative. For every implementation, each data point is taken at a different size. As we go from the left to the right along the curves, the size of the implementations increase, yielding higher frequencies at higher energy costs. The curve for the dynamic implementation is also shown in the figure to measure the deviation of the performance of each implementation from this ideal behaviour.

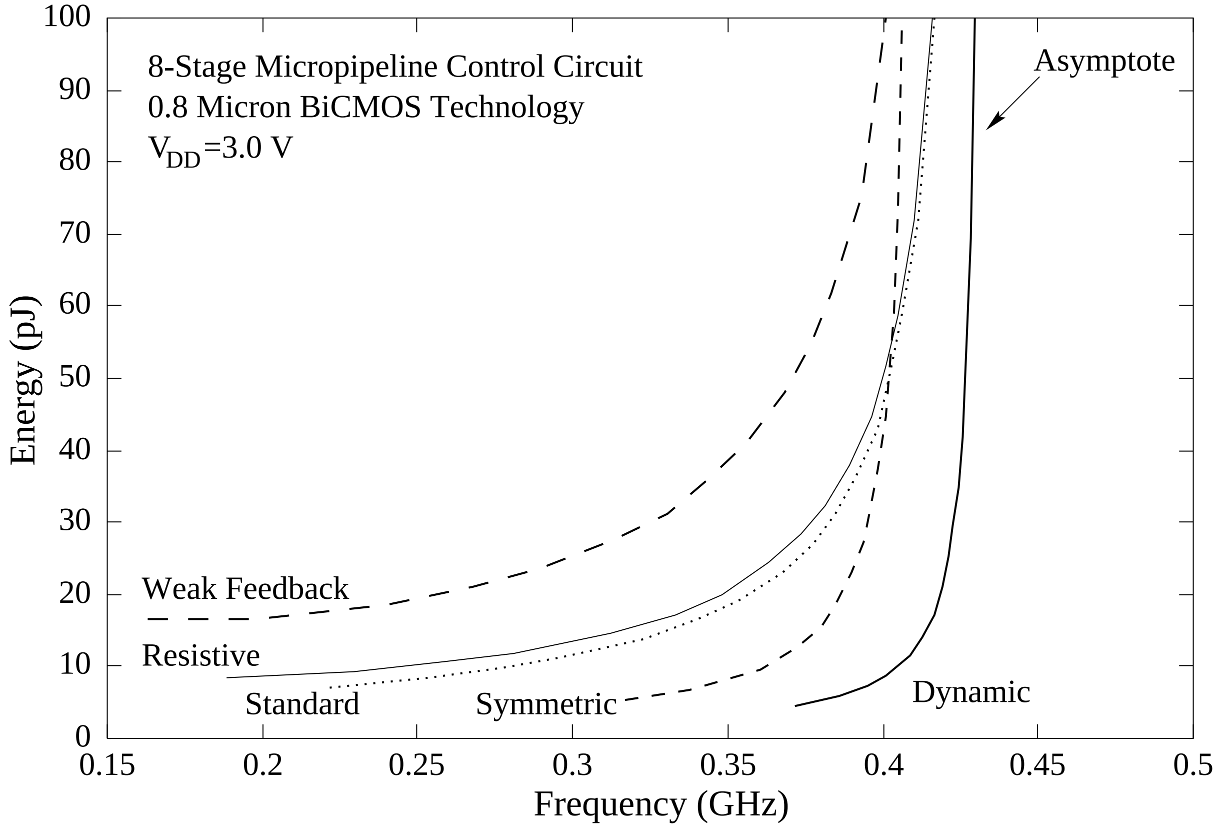


Figure 4.13: Energy-frequency graphs based on SPICE simulations for the single-rail C-element implementations under the second test.

The almost vertical line at 0.425 GHz, i.e., the highest frequency obtainable with the dynamic implementation is the asymptote for the performance of the single-rail implementations. This is 15% higher than the value predicted by our model, 0.362 GHz. The reason is that, according to the simulations, node c' has a voltage swing of around $0.8V_{DD}$ rather than V_{DD}. Considering the energy-frequency trade-offs, the symmetric implementation seems to be the circuit of choice among the static implementations for frequencies below 0.4 GHz. For instance, a frequency of 0.375 GHz can be obtained by the symmetric implementation at an energy of 14 pJ at $W = 9$ μm. The same frequency costs 27 pJ, $W = 17$ μm, with the standard (or a little more with the resistive) implementation and 54 pJ, $W = 35$ μm, with the weak feedback implementation; that is, extra energy by factors of 1.93, and 3.86, respectively. The dynamic implementation is capable of producing the same frequency for less than 5 pJ with $W = 3.5$ μm. We also observe that the curves for the standard, weak feedback, and resistive implementations are heading towards the asymptote, whereas that of the symmetric implementation saturates at a lower frequency. The reason is that these three implementations have exactly the same pull-up pull-down topology as the dynamic implementation does. The symmetric implementation differs, however, in that it has two parallel pull-up pull-down structures symmetric with respect to the input signals, and thus, it is unable to take advantage of the difference in arrival time of the two input signals as the other implementations would do. This was not predicted by our analytical model.

4.10 Concluding Remarks

A comparative study of the single-rail CMOS implementations of the C-element in terms of energy, delay, and area has been presented based on a first-order analysis and SPICE simulations. Four single-rail implementations from the literature have been considered. Special attention has been given to the energy efficiency of the implementations in order to identify the proper choice for an energy-efficient high-speed design environment. The techniques we discussed can be extended to compare different implementations of other primitives. The energy-delay and energy-frequency graphs illustrated in this chapter are a convenient way of presenting the trade-offs between energy and delay (or frequency).

The simple first-order model for the delay and energy of CMOS gates presented in Chapter 3 has been applied to the C-element implementations for optimization and comparison. Good agreement is observed between the model and simulation results. The C-elements were tested in two different environments using simulations. In the first one, a C-element of variable size and fan-out is driven by fixed-size inverters. The energy dissipation and delay of the different implementations were measured through SPICE simulations. In the second test environment, the circuits were tested for their performance in the presence of feed-back. For this purpose, a chain of eight C-elements in the form of a micropipeline control circuit was used. Using SPICE simulations, the energy dissipation per cycle and the frequency of oscillation of the control circuits were measured and reported. The results obtained through the first setup are useful when we want to insert a C-element in a spot where the input drive and output capacitance are known. The second setup, however, can give suggestions for the implementation of a subsystem in which the C-elements are mutually dependent. In our study we did not include any layout considerations. We have also ignored wire loads. Considering that wire loads play a more important role in future technologies, the results obtained here should be used cautiously.

Both first-order analysis and simulations in the two test environments identify the symmetric implementation as the right choice for energy-efficient, high-speed applications. In both tests, the shape of the energy-delay curve for the symmetric implementation is closest to that of the dynamic implementation of the C-element, because the symmetric implementation has the least overhead for maintaining the state of the output. The keepers in the symmetric implementation are only two minimum size transistors, which do not resist output switching. In the standard implementation of the C-element, the keepers are six minimum size transistors which do not resist output switching either. In the weak feedback implementation of C-element, however, the keepers are in the form of an inverter, which does resist output switching. Hence, the curve of the weak feedback implementation is farthest from that of the dynamic C-element. The resistive implementation is a modified version of the weak feedback implementation with two additional transistors to make it less resistive to output switching. The performance of the resistive implementation can be improved at the cost of higher noise sensitivity.

This comparison demonstrates that minimizing the number of transistors dedicated to latching (keepers) and avoiding topologies that resist output switching may result in significant energy savings. In a low-power environment, a designer can not afford to spend energy on transistors that do not contribute to output switching and are only dedicated to latching the output, even if they are made of smallest size. After all, in a low-power design, most transistors are made small, and the keeper transistors may have a large share in the total energy consumption. This study also demonstrates how the various implementations of the keepers change the performance of the very same circuit and affects the energy-delay trade-off.

Chapter 5

Differential CMOS Implementations of the C-Element

Differential logic circuits, also called double-rail circuits, usually require both the input signals and their complements. In return, they produce the output and its complement in a symmetrical manner. Although differential logic circuits usually double the wiring requirements, they may be beneficial in terms of speed, energy, and area, as reported for example in [105]. The most well-known differential CMOS logic families are Differential Cascade Voltage Switch Logic (DCVS) [42] and Complementary Pass-Transistor Logic (CPL) [105]. The C-element implementations in this chapter do not quite correspond to either logic families, because they use an inverter latch to hold the state of the outputs. Hence, we refer to their class as the Differential logic with Inverter Latch (DIL).

This chapter[1] first introduces the basic DIL implementation of the C-element. Then, a few modified versions of the DIL C-element are presented and their advantages and disadvantages are explained. After that, the problem of divergence of complementary outputs in a pipeline is addressed. Finally, we conclude with an overall comparison of the single-rail (including conventional) and double-rail (differential) implementations in the two typical environments discussed in the previous chapter. It is important to note that in this chapter evaluations and optimizations of the circuits are mainly based on HSPICE simulations. The reason is that the formulation introduced in Chapter 3 does not conveniently cover differential logic

[1]The content of this chapter has also appeared in [82].

gates. A unified model for delay estimation and optimization of conventional and differential logic gates will be introduced in the next two chapters. In fact, the exhaustive simulations required for optimizing the DIL implementations motivated us to develop the delay model of the next chapters.

5.1 Basic DIL Implementation of the C-element

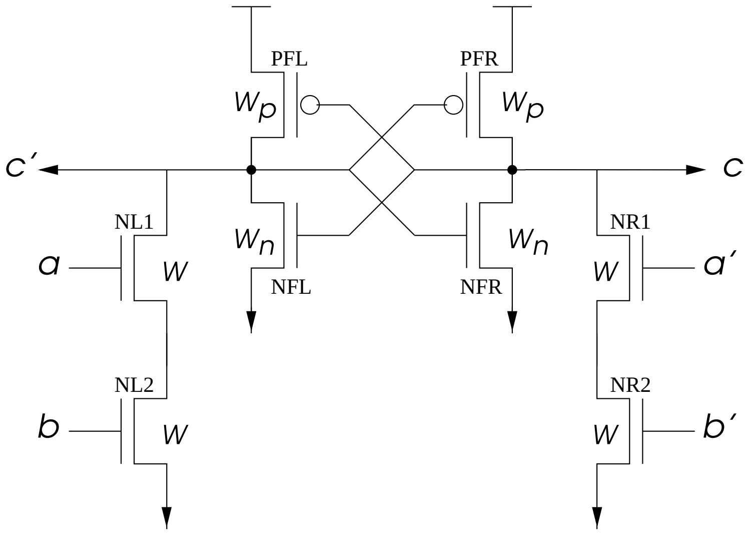


Figure 5.1: DIL implementation of the C-element.

The basic CMOS implementation that uses differential logic and an inverter latch (DIL) is shown in Figure 5.1. It consists of two pull-down trees of NMOS transistors and an inverter latch formed by the devices PFL, NFL, PFR, and NFR. If the inputs are both high, then c' goes low and c is pulled up through the P-device of the right hand side inverter of the latch, PFR. Similarly, if both inputs are low, then c goes low and c' rises through the P-device of the left inverter of the latch, PFL. If the inputs don't match, the inverter latch holds the previous values of the outputs.

The NMOS transistors of the inverter latch are inactive during switching and, thus, are of minimum size w to reduce their capacitive loading effect. The PMOS transistors of the inverter latch, on the other hand, are active during switching and their proper sizing is critical to delay optimization. For the circuit to function properly, the pull-down trees must be made less resistive than the latch P-devices. Accordingly, the following must hold: $W_p < 0.5\rho W$; where W_p is the width of the P-devices in the inverter latch and W is the width of the N-tree transistors, as shown in the figure. Although this gives an upper-bound for W_p, it leaves the designer with a wide range of widths to choose. However, we know that if W_p is made too small, the circuit will suffer from high rise-time delay. A large W_p, on the other hand, intensifies the race problem at the falling output node and results in both large falling and rising delays.

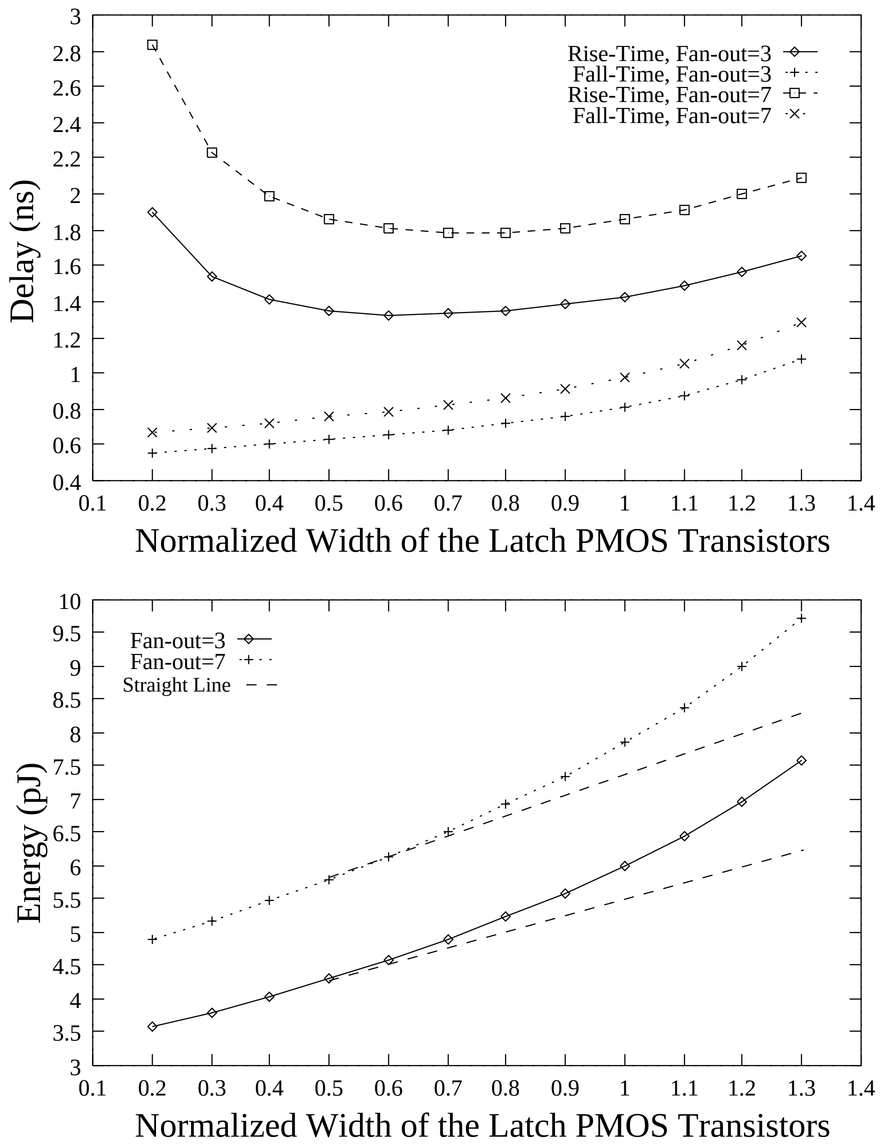


Figure 5.2: Delay and Energy of the DIL implementation for various sizes of the PMOS device in the latch.

Figure 5.2 shows the results of HSPICE simulations for the DIL implementation at various values of W_p/W. The simulations are done for fan-outs of 3 and 7 inverters. The minimum rising delay for a fan-out of 3 occurs at around $\frac{W_p}{W} = 0.6$. When the load is increased to a fan-out of 7, i.e. more than doubled, very little shift is observed for the minimum delay point along the horizontal axis to a value of 0.7. In our designs we have chosen $\frac{W_p}{W} = 0.6$ for the DIL implementation regardless of the load. The analysis of Chapter 7 indicates that for loads much larger than the input capacitance, minimum delay is achieved when $\frac{W_p}{W} \approx \frac{\rho}{4}$, which evaluates to about 0.6. The figure shows that the falling delay increases as W_p/W increases.

Moving our attention toward the energy graph of Figure 5.2, we realize that for both fan-outs the energy linearly increases as W_p/W increases up to a certain point, after which the increment occurs more rapidly. It is interesting that this point of deviation from linearity coincides with that of the minimum delay. That is, after this point, not only the delay increases, but also it costs more in terms of energy and obviously area. The more rapid change in the energy can be intuitively explained as follows. Starting near the origin, as W_p/W increases the parasitic capacitances also increase and so does the energy. However, as we pass the minimum delay point by further increasing W_p/W, the duration of the fighting increases and we end up spending extra energy for the race on top of the usual amount spent on the capacitances. The deviation from linearity, shown in the graph, indicates the energy cost for the race.

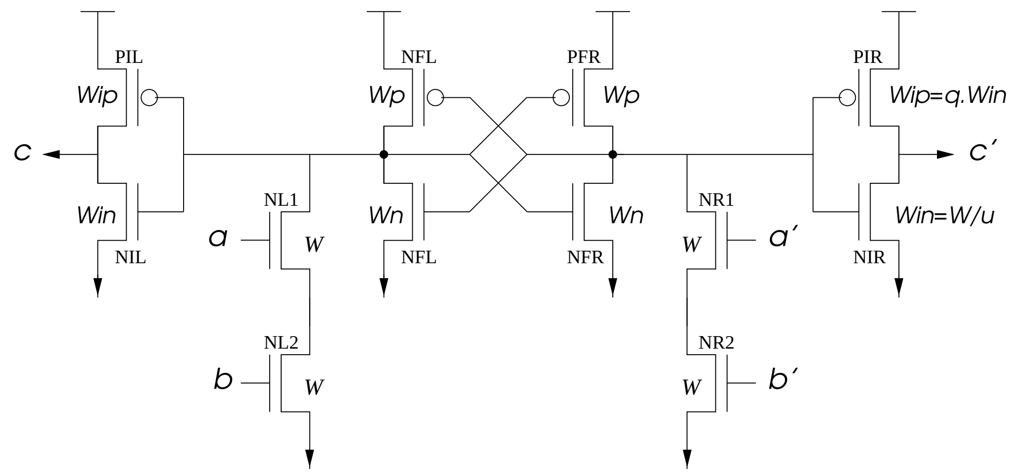


Figure 5.3: Modified DIL (MDIL) implementation of the C-element

Due to the fighting at one of the outputs during switching, the DIL implementation has large delay and energy sensitivities with respect to load. This means that as the load increases, the DIL implementation will have severe disadvantages compared to the single-rail implementations in terms of both delay and energy. One way of reducing the delay and energy sensitivities with respect to load is to insert a couple of inverters between the outputs of the DIL implementation and the loads, as illustrated in Figure 5.3. We refer to this implementation as the modified DIL (MDIL) implementation.

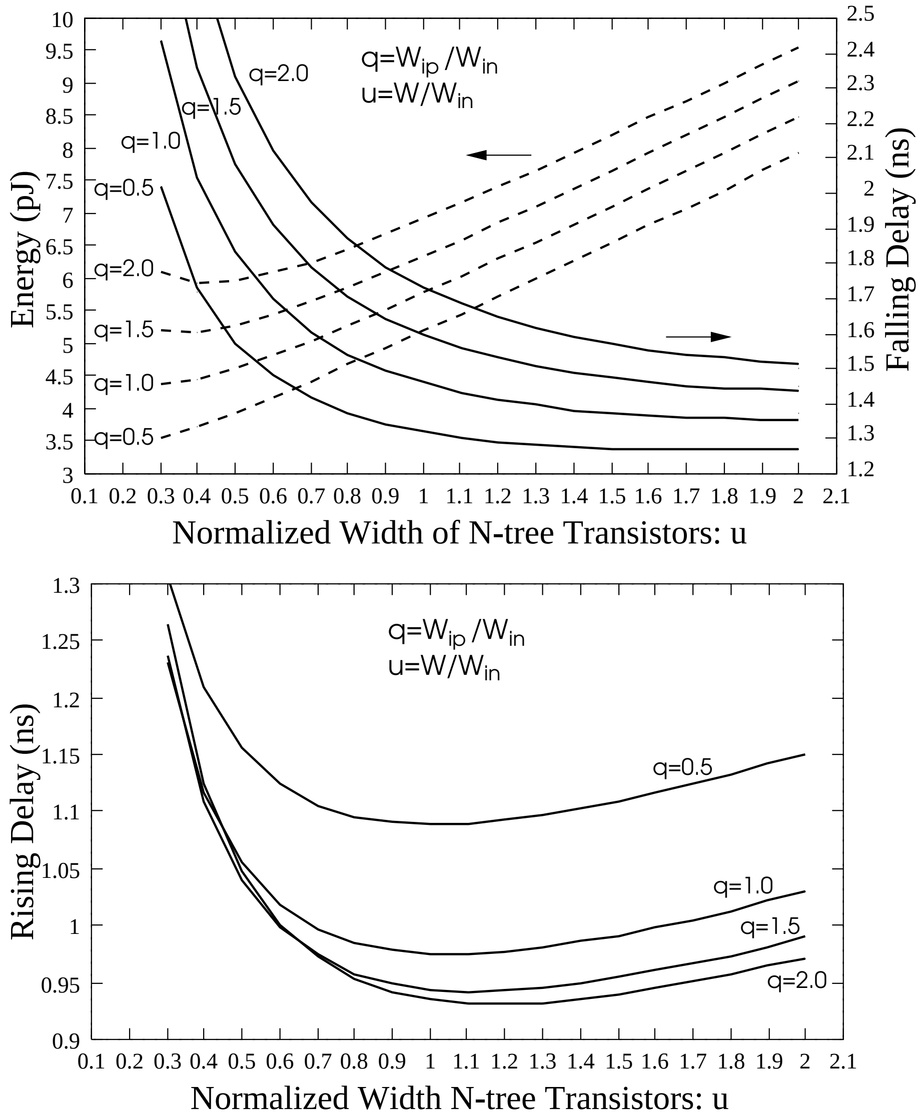


Figure 5.4: Delay and Energy of the MDIL implementation for various sizes of the output inverter (fan-out of three inverters)

The operation of this circuit is similar to the DIL implementation except that the worst-case delay is now characterized by the falling delay. With MDIL, the designer has to decide the width of the output transistors, namely W_{in} and W_{ip}. Let $q = W_{ip}/W_{in}$ and $u = W/W_{in}$. Figure 5.4 illustrates the results of HSPICE simulations for the MDIL implementation with various values of q and u. If $q = \rho$, the output inverters have a threshold of $V_{DD}/2$. This would be high for this circuit and results in poor fall time delay, because the original DIL implementation has a high rise time. As q is reduced, the threshold of the output inverters moves toward ground. However, if q is made too low, the circuit will suffer from poor rise time and looses sharpness in its low-to-high transitions. Larger q also means larger energy dissipation, because of higher parasitic capacitance. As u increases, the falling delay decreases, because W_p is also proportionally increased. The rising delay, however, first decreases and then increases as u increases, with the minimum being around $u = 1$.

Because of the above trade-offs, the optimum choice of q and u depends on the particular design requirements. In the next section we see how a design environment may restrict the value of q while we choose $u = 1$ to obtain the minimum rising delay.

5.2 Divergence of the Complementary Outputs

In differential circuits there is a time difference between the production of complementary outputs. In synchronous circuits this doesn't pose much of a problem, because of the presence of a clock signal and the absence of feed-backs. In asynchronous circuits, however, this time difference can be troublesome in two cases. One is when the faster output of a gate is fed back to the same gate and arrives at the gate before the other slower output is produced. Another case is when a series of differential gates form a long path and the divergence of the complementary outputs increases along the path until they become so much apart that at some stage one of the gates fails to respond to its inputs.

If the differential gates in a circuit are active high, i.e. activated only when some of their inputs become high, and their falling outputs are produced first, then the gates function properly no matter how long the production of the rising outputs take. On the contrary, if the gates are active high and the rising outputs are produced first, then the divergence of the complementary outputs could affect the operation of the gates. The DIL implementation, for example, belongs to the first category whereas the MDIL implementation is in the second category.

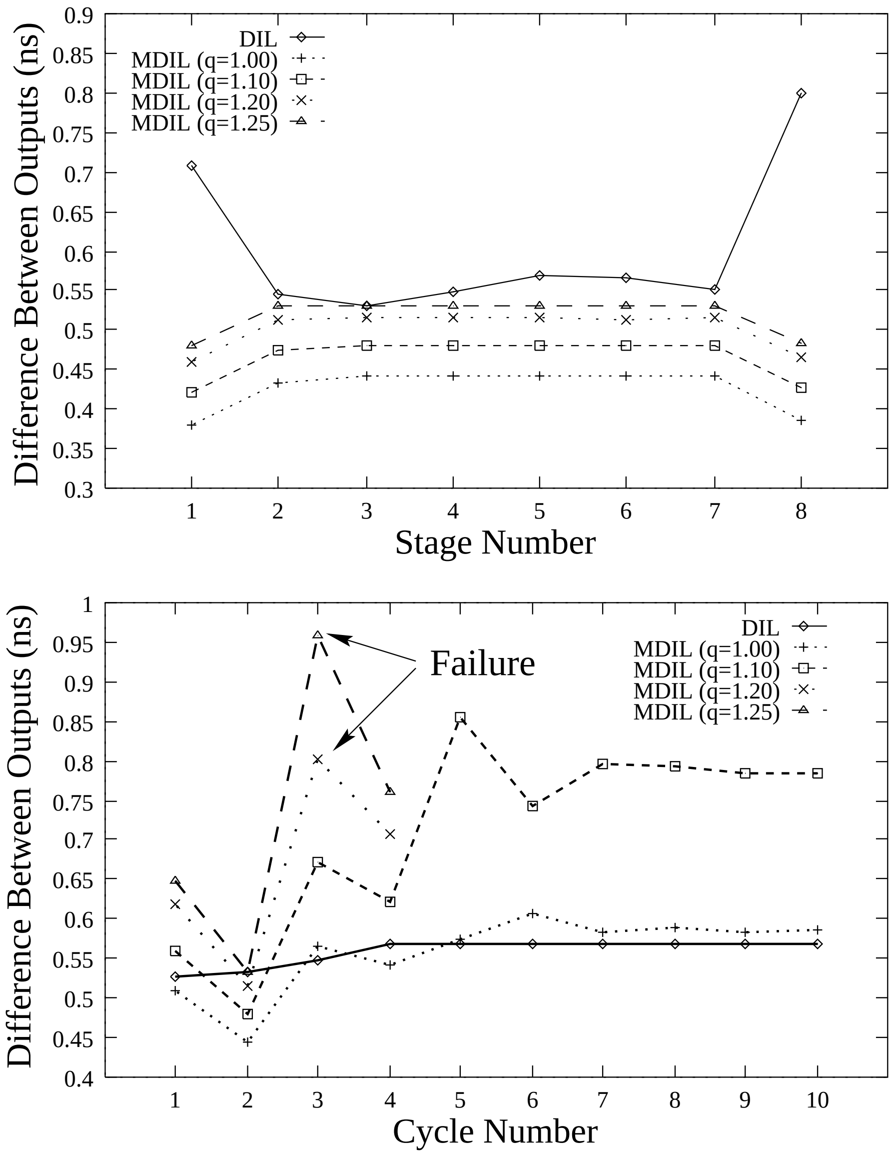


Figure 5.5: Divergence of the complementary outputs in a micropipeline control circuit: outputs of different stages at the first cycle (top), outputs of the same stage at different cycles (bottom).

Figure 5.5(top) shows the time difference between the complementary outputs of the C-elements at various stages of the micropipeline control circuit. The C-elements are implemented in DIL and MDIL with q ranging from 1 to 1.25. Obviously, as q increases the complementary outputs get further apart. Nevertheless, in all cases the time difference between the outputs remains almost constant along the pipeline except for the two ends. At the two ends the loads of the C-elements are higher and thus, in the case of DIL the outputs become further apart while in the case of MDIL the outputs get closer. Figure 5.5(bottom) shows the time difference between the complementary outputs of some C-element at the middle of the pipeline at different cycles. For the DIL implementation we can see that after 4 cycles the time difference between the outputs converges to around 0.57 ns. For the MDIL implementation, $q = 1.0$, after some fluctuations the time difference converges after the seventh cycle. If q is increased to 1.10, the fluctuations increase, but the time difference finally seems to converge. Further increase in q to 1.20, results in a large jump in the time difference between the second and third cycles, and the pipeline fails to oscillate after the third cycle. Of course, the problem becomes worse if q is made any larger. This test illustrates that if the MDIL implementation is to be used in such a micropipeline control circuit, q must be made less than or equal to unity. If the C-elements are driving some latches (load), q can be made larger, because loading decreases the time difference between the outputs of the MDIL implementation. Successive simulations show that for a micropipeline control circuit without any load, the frequency is maximized if $u = 0.7$ and $q = 1.2$ (we were able to set $q = 1.2$ by reducing u from 1 to 0.7) . These are the values used for the MDIL implementation in the comparative study made for the second test environment. For the first test environment, however, q and u are both set to unity.

5.3 DILP and DILN Implementations

Two possible modifications of the DIL implementation are shown in Figure 5.6. They are both intended to reduce the rising delay which is the main draw-back of the DIL implementation. In Figure 5.6(left), two pull-up trees of PMOS transistors have been added to the original DIL implementation. We refer to this implementation as the DILP C-element. When both inputs are low or both are high, one of the outputs falls and the other one rises almost simultaneously. Thus, the outputs switch independently, in contrast to the original DIL implementation. In this case, the inverter latch is merely used for maintaining the states

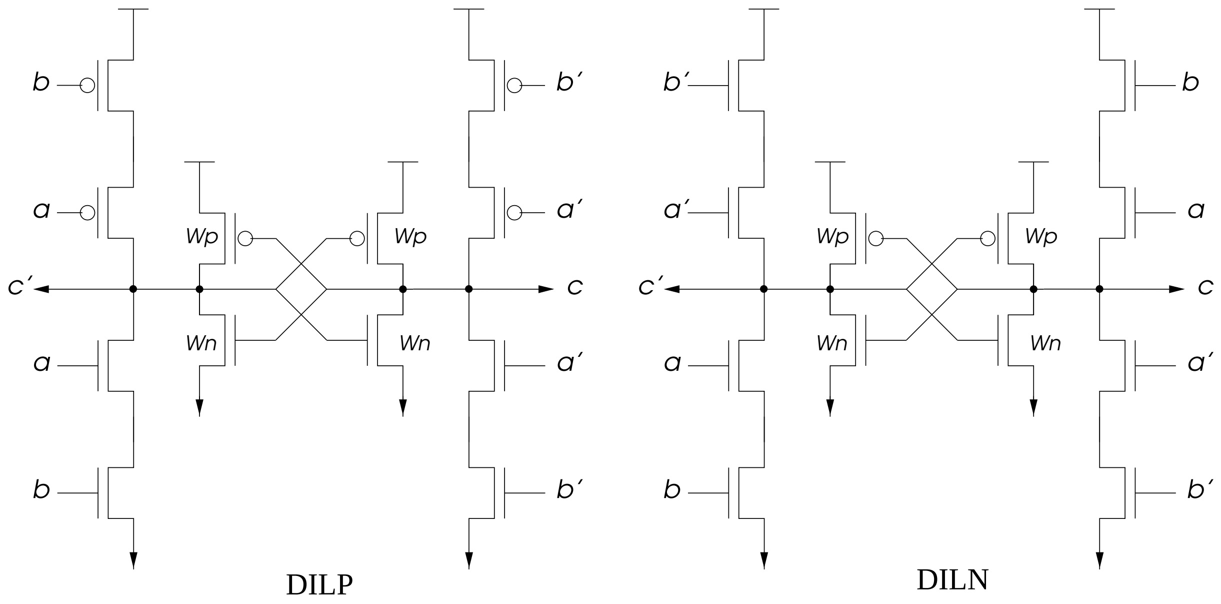


Figure 5.6: Schematics of the DILP and DILN C-element implementations.

of the outputs and, hence, have minimum size P- and N-devices. Later, we will see that the minimum size latch could be troublesome in the presence of feed-back. This implementation has less delay and energy than the DIL implementation, and less delay but higher energy than the MDIL implementation. The reason for the relatively high energy consumption is its high input capacitance. The input capacitance can be reduced by replacing the P-tree with an N-tree as shown in Figure 5.6(right). We refer to this implementation as the DILN C-element. In the DILN implementation, during switching, one of the outputs falls from V_{DD} to ground, while the other output simultaneously rises from ground to $V_{DD} - V_{TN}$. Then, it is boosted to V_{DD} with the help of the inverter latch. Thus, the rise and fall times are not symmetric, as in the case of the DILP implementation. However, the amount of symmetry is greatly enhanced over the DIL implementation.

In a micropipeline, because of the presence of feed-back, the divergence of the complementary outputs in the DILP implementation could be troublesome. Even if all of the complementary inputs of a DILP implementation arrive simultaneously, there is no guarantee that the complementary outputs would be simultaneously produced. The reasons are that, firstly, the P-tree and N-tree exhibit different resistances depending on the load and the state of the output voltages. Secondly, if the minimum size latch is used, then during

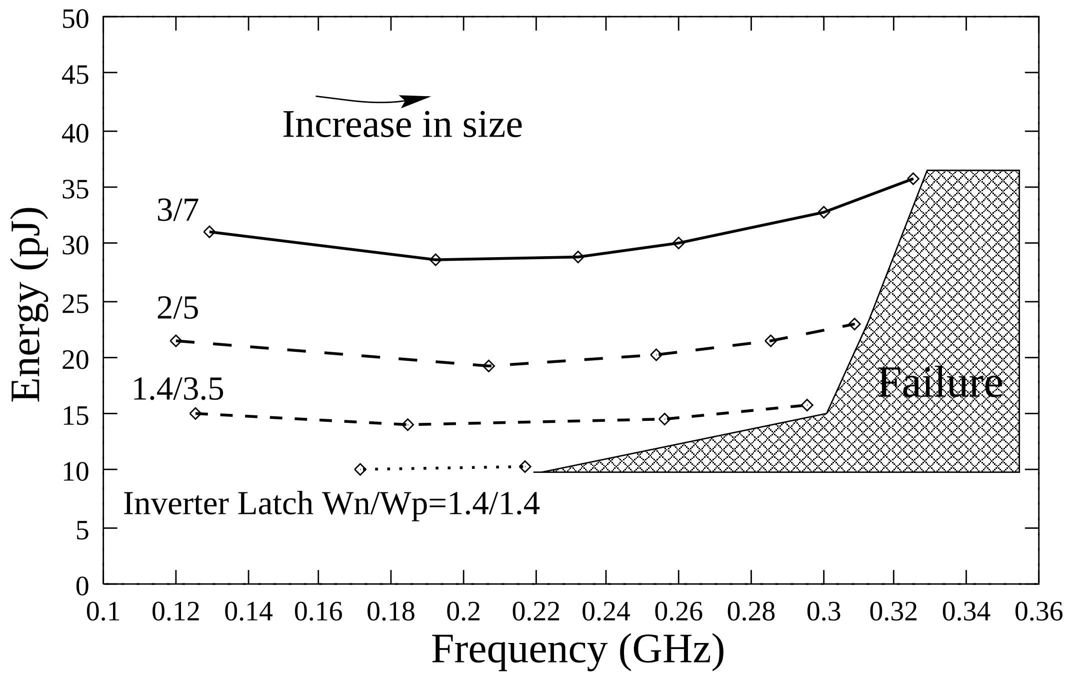


Figure 5.7: HSPICE results for the DILP C-element implementation.

switching the P-tree has to fight a resistance proportional to w, whereas the N-tree has to fight a resistance proportional to w/ρ. The second factor could be overcome by using a latch in which the P-devices are ρ times larger than the N-devices. The first factor could be compensated for by increasing the overall size of the latch as the loading increases.

Figure 5.7 shows the results of HSPICE simulations for the DILP implementation. The graph shows the frequency of operation of the micropipeline control circuit as the size of the C-element is increased. With inverter latches of $W_n/W_p = w/w = 1.4/1.4$, a frequency of 0.22 GHz is obtained at a gate area of about 25 μm^2. Further increase in the sizes of the C-elements results in failure. If we use symmetric latches with inverters of $W_n/W_p = 1.4/3.5$, we can achieve a wider range of frequencies up to 0.3 GHz, after which increasing the sizes of the C-elements leads to failure. To get higher frequencies we have to use larger symmetric latches. For example, by using latches with $W_n/W_p = 3/7$ a frequency of 0.32 GHz is also achievable. The draw-back is that each time we use a larger latch, we get to a higher level of energy consumption, as shown in the graph, because we are increasing the amount of parasitic capacitance.

The DILN implementation is much more robust than the DILP implementation in a

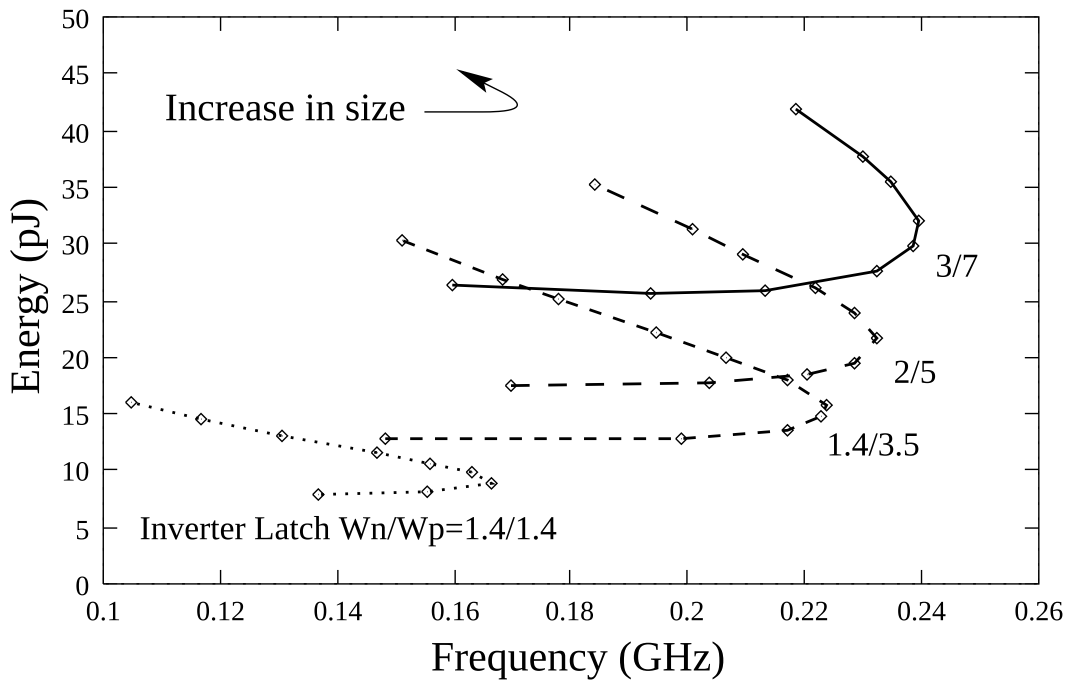


Figure 5.8: HSPICE results for the DILN C-element implementation.

micropipeline environment and rarely fails, because, like the basic DIL implementation, it is active-high and its rise-time delay is larger than its fall-time delay. The frequency versus gate area and energy versus frequency graphs for the DILN implementation are shown in Figure 5.8. With 1.4/1.4 latches, the complementary outputs are far apart and the performance is very poor. Using 1.4/3.5 latches helps to increase the frequency up to 0.22 GHz at an area of around 30 μm^2. Further increase of the area is counter productive and reduces the frequency, because the feed-back signals become so fast that they slow down the full generation of the forward signals. Increasing the size of the latch is of some help, since it increases the race and slows down the feed-back signals. The energy increases a lot by using larger latches without much gain in frequency. In order to avoid counter-productive interaction of signals in a micropipeline control circuit using the DILN implementations one can add a couple of inverters to the outputs of each DILN implementation. We call this implementation the MDILN implementation. The MDILN implementation for which the simulation results are reported in the next section is designed with a 1.4/3.5 latch and output inverters with N- and P-transistors 2.5 times larger than the N-tree transistors (i.e. $u = 0.4$ and $q = 1$).

5.4 Results and Discussion

This section presents the results of comparing the optimized single-rail and differential implementations of the C-element for the two typical environments introduced in the previous chapter.

Figure 5.9 depicts the energy-delay graphs under the first test (with fan-out of three inverters) based on simulation results for all of the C-element implementations considered. The best delay obtained by a conventional implementation is 1.15 ns and belongs to the symmetric C-element. The same delay can be obtained using the DILP or DILN implementations with 20% less energy consumption. Delays below 1.15 ns to 1.0 ns could only be obtained with the differential implementations. For a lower number of fan-outs, the results are even more in favour of the DILP and DILN implementations. The DILP and DILN C-elements for this environment are implemented with minimum size latches.

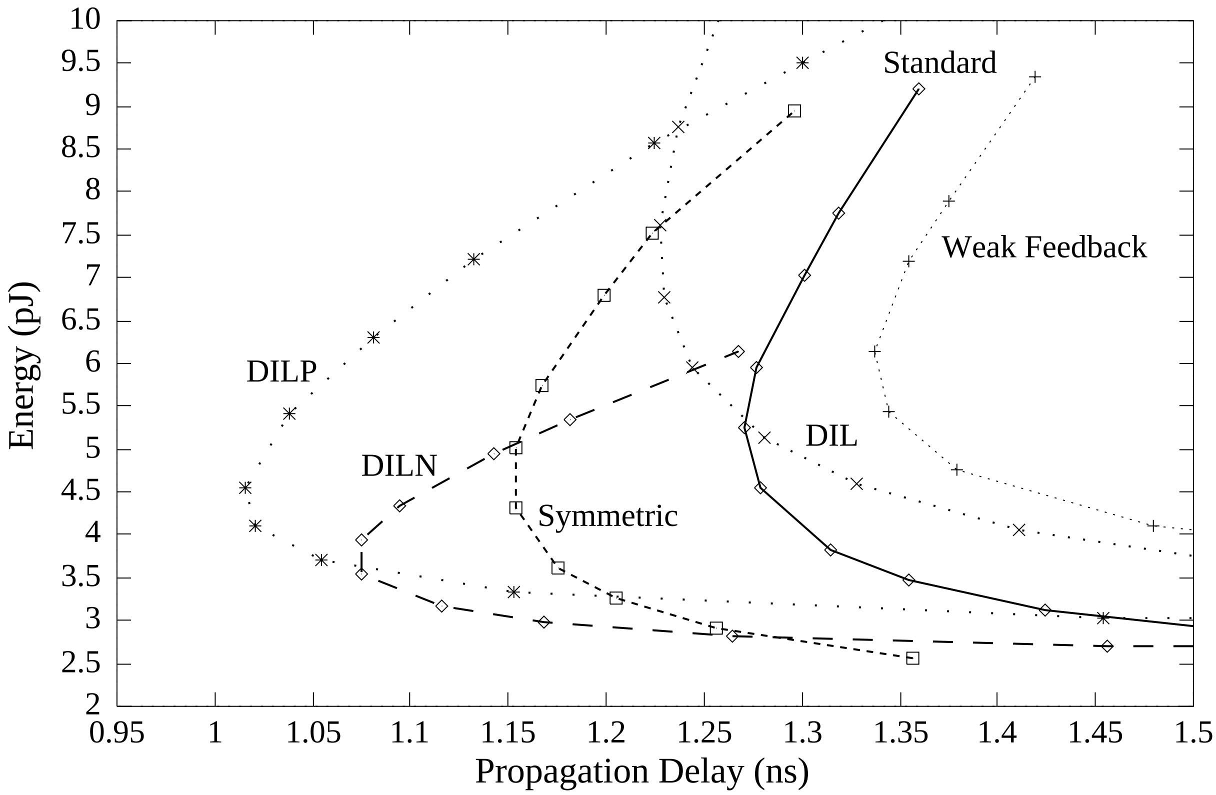


Figure 5.9: HSPICE results for the single-rail and double-rail C-element implementations under the first test.

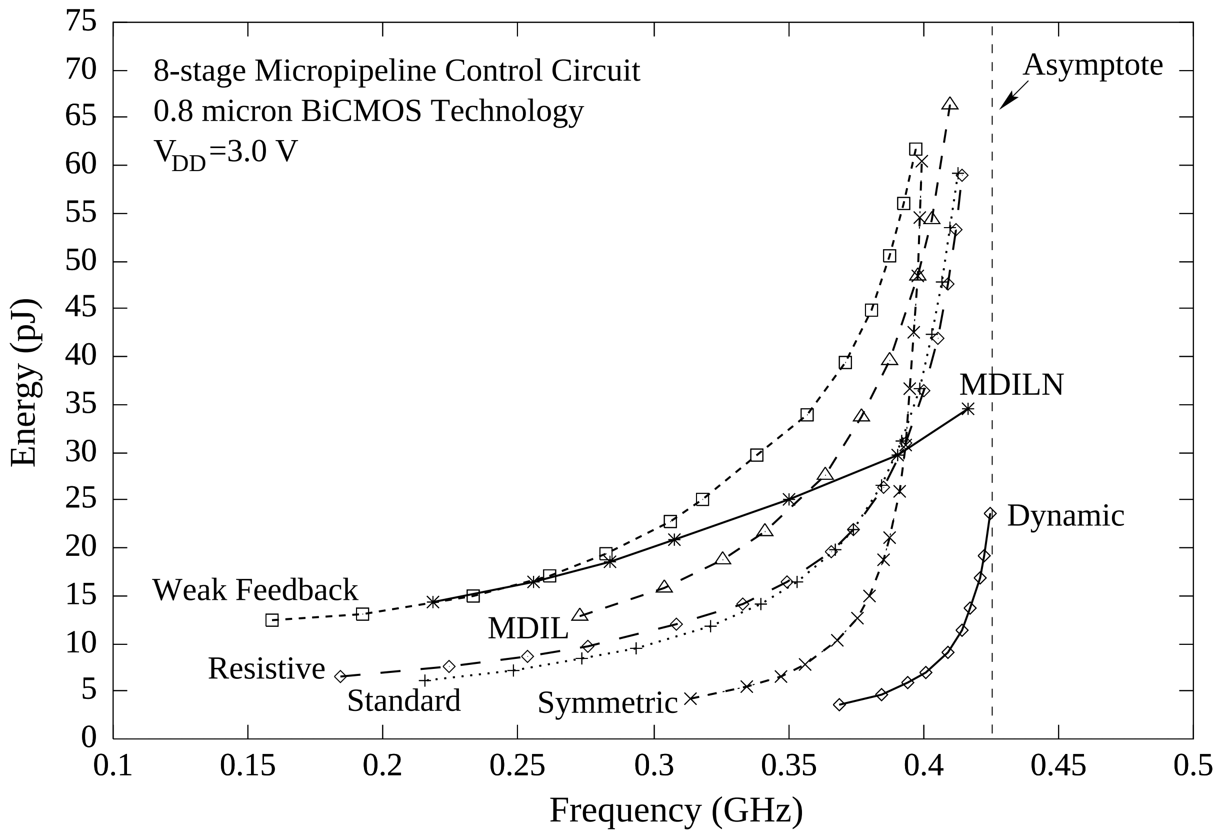


Figure 5.10: Simulation results for the single-rail and double-rail CMOS implementations of the C-element in a micropipeline environment.

The overall results of the simulations for the single-rail and double-rail implementations of the C-element under the second test are depicted in Figure 5.10. It shows the energy-frequency curves for the four single-rail implementations and of the MDIL and MDILN implementations. The curves for the other double-rail implementations are omitted, because of poor performance. (The curves for the DILP and the DILN implementations are depicted separately in Figure 5.7 and Figure 5.8 respectively.) The performance of each implementation has been optimized according to the preceding discussions. For every implementation, each data point is taken at a different size. As we go from the left to the right along the curves, the size of the implementations increase, yielding higher frequencies at higher energy costs. Of course, in an energy-frequency graph like this one, the outer curves indicate better energy-frequency trade-offs. The curve for the dynamic implementation is also shown in the figure as a representative of an almost ideal case, and to measure the deviation of the performance of each implementation from this more ideal behaviour. The dashed vertical line at 0.45 GHz, i.e. the highest frequency obtainable with the dynamic implementation, is the asymptote for the performance of the single-rail implementations. The double-rail implementations seem not to be able to cross that line either. The MDILN implementation comes very close, but fails afterwards, because of the divergence of the complementary outputs. In terms of energy and frequency, the symmetric C-element outperforms the other single-rail implementations. However, frequencies over 400 MHz are only obtainable with the double-rail implementations. The MDILN implementation offers the best energy-frequency trade-off followed by the MDIL implementation.

5.5 Concluding Remarks

We have presented a comparison of the optimized CMOS implementations of the C-element in terms of performance and energy. Beside the dynamic implementation of the C-element, we have addressed eight static implementations, four of which are single rail and the rest are double rail. The single-rail implementations are the standard, weak feedback, resistive, and symmetric C-elements introduced in the previous chapter. The double-rail implementations are the DIL implementation of Figure 5.1, the MDIL implementation of Figure 5.3, and the DILP and DILN implementations of Figure 5.6. We also briefly mentioned the MDILN implementation, which is basically the DILN implementation with a couple of output inverters. Ways to optimize the performance of the various implementations have been discussed. One should be very cautious when using the differential logic implementations in pipelines including feedback loops, as we have observed some divergence phenomena with these circuits leading to failure. Although we have explained methods for reducing the problem, further study is needed to fully explain this behaviour.

Chapter 6

Delay Modeling at the Device and Switch Levels

This chapter presents a new, unified, and simple model for estimating the charging and discharging delays of an MOS transistor, whether of NMOS or PMOS type. In other words, this model covers the delay of an MOS transistor for transferring both a logic 1 and a logic 0. Armed with such a model, it is possible to predict the delay of a CMOS logic gate implemented in any style. In conventional CMOS logic gates, NMOS transistors are solely used to discharge the output capacitance, i.e. to transfer a logic 0 to the output. Correspondingly, PMOS transistors are dedicated to charging the output capacitance, i.e. to transfer a logic 1 to the output. In general, both types of transistors may be used to carry a logic 1 or a logic 0, such as in a CMOS transmission gate. This idea has led to a number of innovative CMOS logic styles, such as complementary pass-transistor logic (CPL) [105] and the differential cascode voltage switch logic (DCVSL) [42]. NMOS transistors are also widely used in latches and flip-flops for passing both 1's and 0's. For developing a delay model which covers the various combinations of the transistors in a circuit, we basically need to predict accurately the current through NMOS and PMOS transistors during both charging and discharging scenarios.

First, we introduce a new and simple model for the saturation current in MOS transistors and demonstrate that it is more accurate than the model proposed in the literature with the same level of complexity. Then, we argue that the same model can be applied to predict the current in an NMOS transistor when charging a node and, similarly, to a PMOS transistor when discharging a node. We are not aware of any such model in the literature, except for the very rough techniques discussed in the text books like [72]. Next, we discuss the effect of finite input signal slope. This is followed by the extension of a ramp delay model to circuit scenarios involving overlapping and opposing currents. After that, we further extend the delay model to cover a chain of series-connected transistors. Finally, we present a procedure for extracting the empirical parameters of our model with HSPICE simulation.

Note that starting from this chapter, we use a 0.5 μm CMOS technology for modeling and HSPICE simulations, instead of the 0.8 μm BiCMOS technology used in the previous chapters.

Since we deal with two types of transistors and two types of delays for each transistor type, we need to establish a notation. We use the subscripts "n" and "p" to indicate correspondence of a parameter to NMOS or PMOS devices, respectively. We also use the accents " $\acute{}$ " and " $\grave{}$ " to indicate correspondence of a parameter to rising or falling delay, respectively.

6.1 On Short-Channel Effects

Shockley's square law over-estimates the saturation current in short-channel devices. The major reasons for this are the *velocity saturation* and the *mobility degradation* effects, which are not accounted for in the square-law. While the former effect is attributed to the component of the electric field along the channel, the latter effect is attributed to the component of the electric field perpendicular to the channel.

The velocity of the carriers v is proportional to the applied electric field E as long as E is less than $E_{sat} \approx 10^5$ V/cm. The constant of proportionality is the carrier mobility. For larger values of E, the carrier velocity tends to *saturate* at around 10^7 cm/s. This effect severely limits the saturation current in MOS devices.

On the other hand, since the oxide thickness is also scaled down besides the channel length, the effective carrier mobility in short-channel devices *degrades* due to the vertical component of the electric field. It is known that the effective surface mobility depends on the vertical electric field according to the following approximation [62]

$$\mu_s = \frac{\mu_0}{1 + \theta(V_{GS} - V_T)} \tag{6.1}$$

Where μ_0 is the low-field surface mobility and θ is an empirical factor with a typical value of around 0.2. Then, taking into account the effects of the lateral field and velocity saturation, the effective mobility, as used in SPICE Level 3, can be approximated by [62]

$$\mu = \frac{\mu_s}{1 \ + \ \mu_s \ [V_{DS}/(v_{max} L)]} \tag{6.2}$$

where v_{max} is the maximum drift velocity of the carriers and, in saturation, V_{DS} is replaced by its saturation value.

6.2 MOSFET Delay and Current

The delay D of charging or discharging a capacitance C to $V_{DD}/2$ through a transistor of width W is governed by the following general delay expression.

$$D = \frac{C \ V_{DD}}{2 \ I_D} = \nu \ \frac{C}{W} \tag{6.3}$$

We prefer to combine all supply voltage and technology dependent parameters in ν and write the delay expression as above. This form also clearly shows the relation between the delay and the width of the transistor.

Before proceeding any further with the details of MOSFET current models, let's see which mode of operation of the MOSFET is of interest to us. Figure 6.1 illustrates the current-voltage behaviour of a 0.5 μm NMOS transistor. The horizontal arrow in the figure shows the trace of the value of the current through the transistor when transferring a logic 0 half-way through. That is, from the time the output is at $V_{DD} = 3$ V until it is discharged to half V_{DD}. At the beginning of a logic 0 transfer, V_{DS} and V_{GS} both equal V_{DD}. At the end of the transfer, V_{DS} equals $0.5V_{DD}$, while V_{GS} remains equal to V_{DD}. In short-channel devices, thanks to their relatively extended saturation region, this process takes place almost

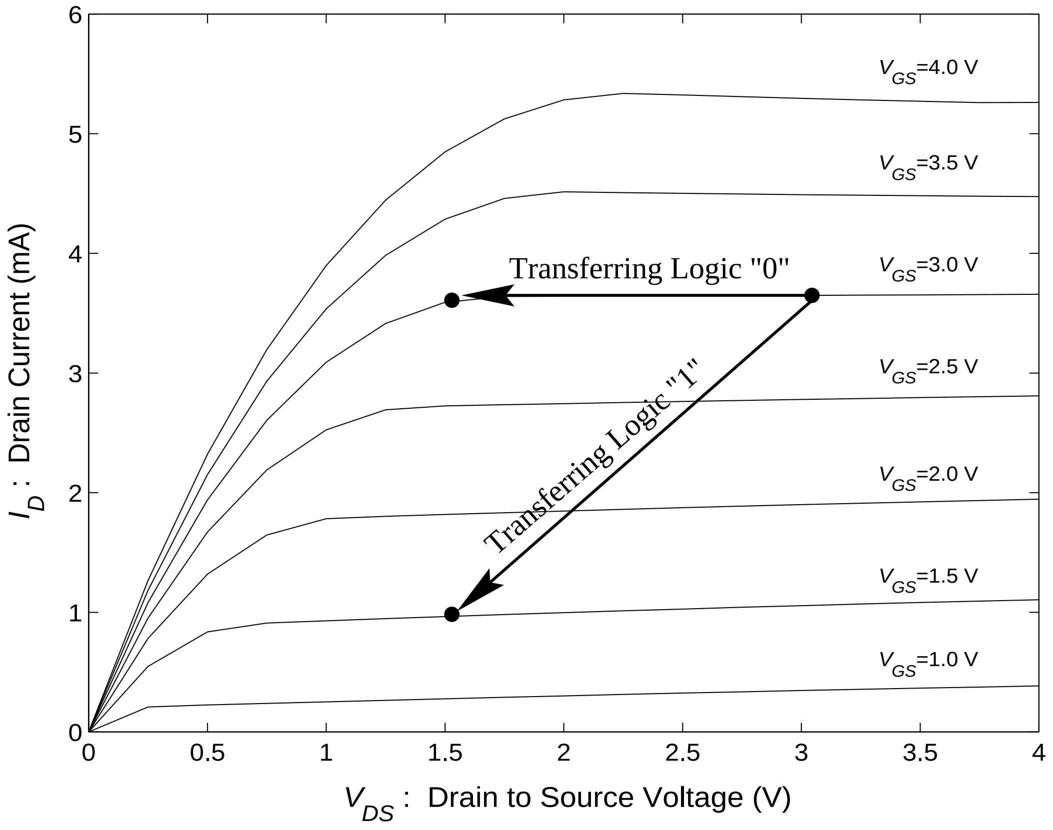


Figure 6.1: IV-characteristic curves for a 0.5 μm NMOS transistor showing the change in current during transferring a logic 1 and a logic 0.

entirely in the saturation mode and, hence, the value of the current remains constant, as shown by the arrow. The oblique arrow in the figure shows the trace of the value of the current when transferring a logic 1 half-way through. That is, from the time the output is at zero volts until it is charged to half V_{DD}. The initial condition at the beginning of a logic 1 transfer is exactly similar to that of a logic 0 transfer. At the end of a logic 1 transfer, however, V_{DS} and V_{GS} both reduce to half V_{DD}. In this case, although the value of the current decreases, the device still remains in the saturation mode, as shown by the arrow. Note that the presentation in this figure is only superficially true, because it doesn't show the body effect.

A PMOS transistor has a similar behaviour. Therefore, whether it is a rising delay or a falling delay that we want to calculate, the current in the general delay expression (6.3) is some average saturation drain current I_D. Hence, we may limit our search for a simple and accurate MOSFET model to the saturation region.

6.3 A New Model for MOSFET Saturation Current

Shockley's model stipulates that the saturation current is proportional to $(V_{GS} - V_T)^2$, that's why it is also known as the square law. The popular α-power law proposed in [76–78] argues that due to the velocity saturation effects, the saturation current in modern devices follows

$$I_D = \frac{W}{L}\ P_C\ (V_{GS} - V_T)^{\alpha} \tag{6.4}$$

Where P_c is a technology-related parameter and α is a number between 1 and 2 called the *velocity saturation index*. For long-channel devices $\alpha = 2$ and the model is reduced to that of Shockley's square law. For short-channel devices under velocity saturation effect α is closer to 1. Similar models in very recent publications claim that α equals 1.25 [14] or 1.3 [13] for deep sub-micron devices. Studies presented in [76] for devices as narrow as 0.5 μm show that the α-power approximation is generally good.

Figure 6.2 depicts the relation between the drain current and the gate voltage in the saturation mode for an NMOS transistor. The solid line in the figure is obtained by HSPICE simulation Level 13. The dashed lines with and without dots are obtained using the α-power law. Since the α-power law has two parameters, namely P_C and α, you can obtain their values by fitting the curve into the simulation data at two different points. The dashed line is the result of this fitting at $V_{GS} = 3$ V and $V_{GS} = 4$ V which leads to $\alpha = 1.03$. With this value of α the model loses its accuracy as V_{GS} decreases and approaches V_{TN}. It is clear that at $V_{GS} = 1$ V the error is over 50% compared to simulation. If we try to fit the model at lower values of V_{GS}, for example at $V_{GS} = 1$ V and $V_{GS} = 2$ V, we end-up with the dashed line with dots. We have to increase α from 1.03 to 1.46. Now the model becomes inaccurate towards higher values of V_{GS} and, for instance, over-estimates the value of current as much as 33% at $V_{GS} = 4$ V. Hence, α clearly needs to be decreased as V_{GS} increases.

As V_{GS} increases, it has an incremental and, at the same time, a decremental effect on I_D. Increasing V_{GS}, on the one hand increases the number of the mobile charges in the channel which boosts the current; and on the other hand, it reduces the effective mobility

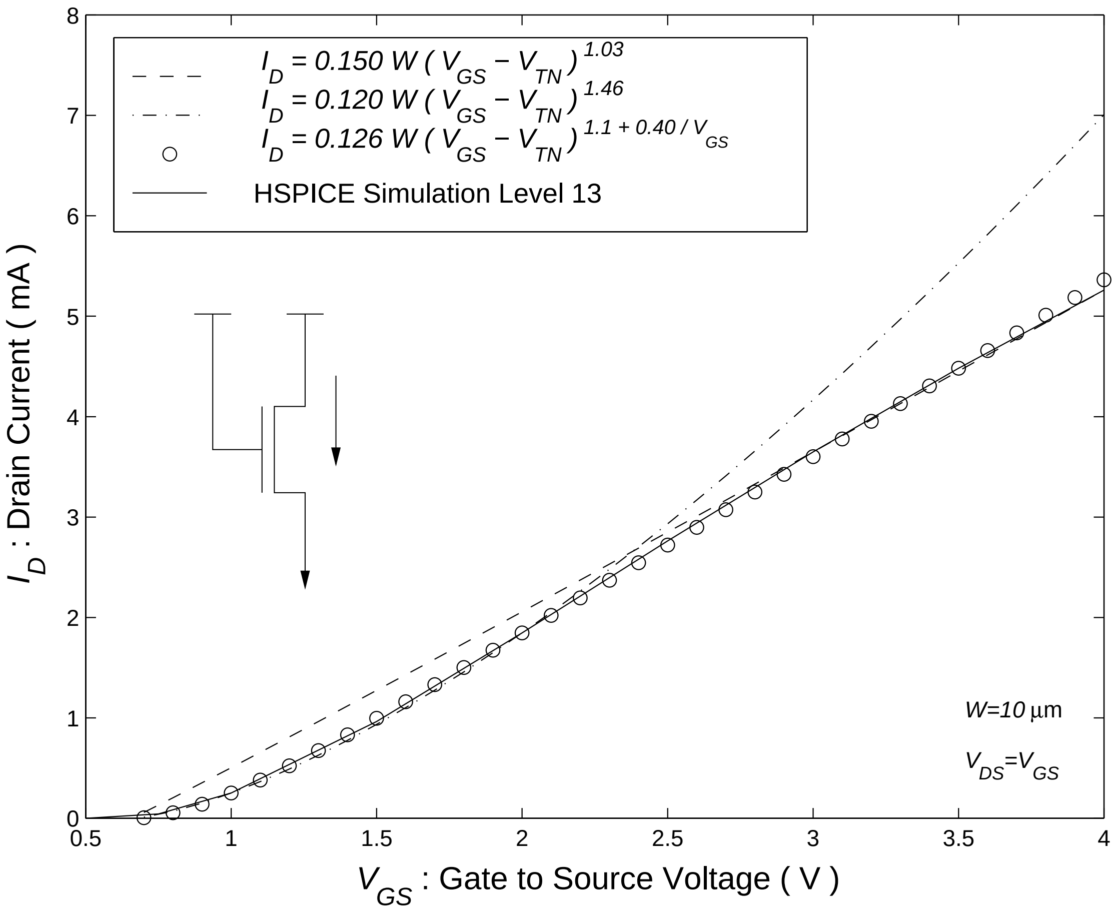


Figure 6.2: Simulated and calculated values of I_D as a function of V_{GS} using the α-power law and the new model. The threshold voltage of the device $V_{TN} \approx 0.66$ V.

of the carriers which limits the current. The combination of these two physical phenomena do not seem to be accurately reproduced by the α-power law. The α-power law only models the incremental effect of V_{GS} on I_D.

As an extension of Shockley's square-law and Sakuai's α-power law, we are proposing the following model for the saturation drain current of a MOSFET.

$$I_D = \kappa\ W\ (V_{GS} - V_T)^{\ \xi + \frac{\vartheta}{V_{GS}}} \tag{6.5}$$

Where α has been replaced by the V_{GS} dependent index $\xi + \vartheta / V_{GS}$. Now the index combines both short channel effects, velocity saturation and mobility degradation. The velocity saturation effect is captured by ξ and the mobility degradation effect is captured by ϑ. The

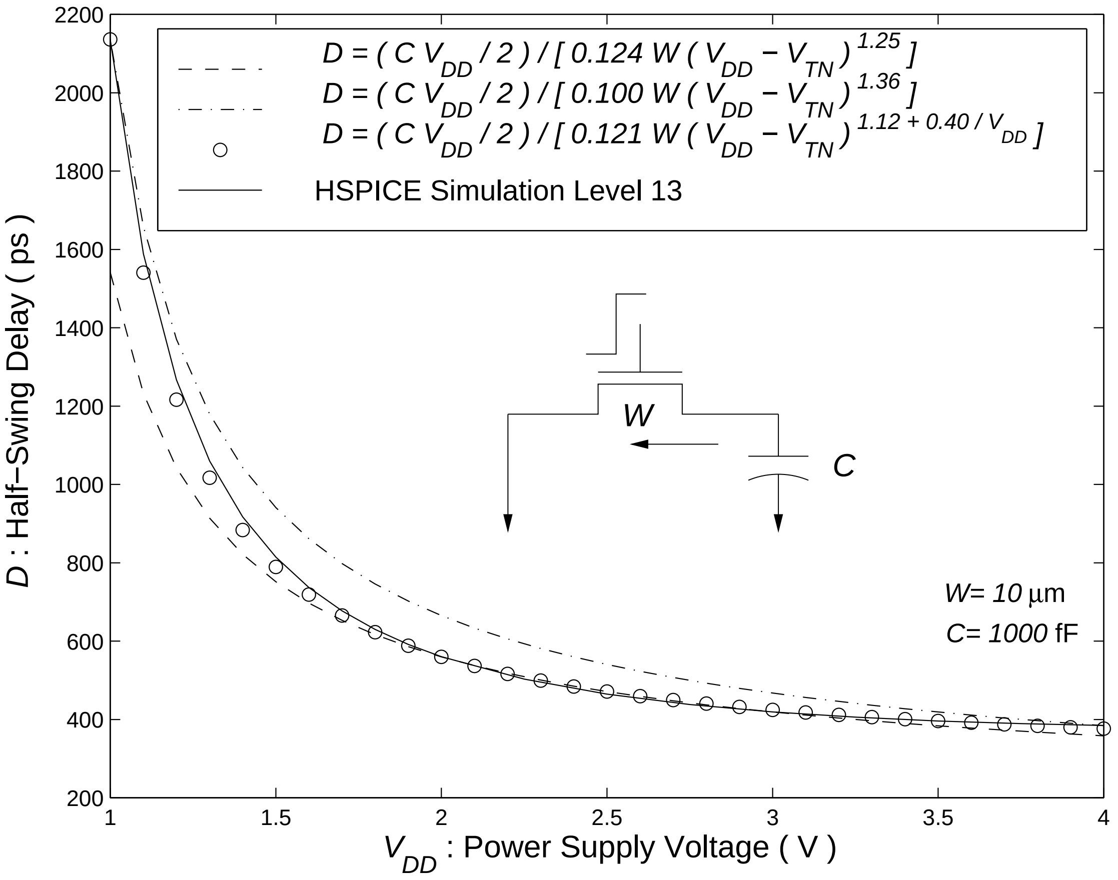


Figure 6.3: Step delay of an NMOS transistor discharging an output capacitance as V_{DD} changes. HSPICE simulation results are compared with the results obtained by the α-power law and the results obtained by the new model represented by small circles.

parameter κ depends on the technology and the effective channel length. We've hidden the channel length inside κ, because digital designers rarely use transistors with channel lengths larger than the minimum feature length specified by the technology. This model has about the same complexity as the previous models using a constant index, but it is more accurate. Since this model has an additional parameter, we need three points for curve fitting. The curve represented by small circles in Figure 6.2 is obtained by fitting the model at V_{GS} equal to 1, 2, and 3.5 volts. The model nicely reproduces the saturation region characteristics of an MOS device.

Figure 6.3 shows a comparison between the HSPICE simulation, the α-power model, and the new model in predicting the delay of discharging an output capacitance by an NMOS transistor at different power supply voltages. Although we had the values of the parameters α, P_C, ξ, ϑ, and κ from the previous experiment, we repeated the curve-fitting process here to obtain better matches. The minor differences in the values of the extracted parameters compared to the previous experiment are due to the overshooting that appears in the output voltage signal through the coupling capacitances. The simple delay equation does not include this effect. The α-power law curve is done for two sets of voltages, different from the previous experiment. The dashed line represents the delay obtained by the α-power law fitted at the points of V_{DD} equal to 2 V and 3 V, while the dashed line with dots represents the delay obtained by the same model fitted at the middle points of V_{DD} equal to 1 V and 4 V. For our model, the fitting points are the same as in the previous experiment, i.e. V_{DD} equals to 1 V, 2 V, and 3.5 V. Note that the values of α obtained here are very close to those suggested in [14] and [13]. This experiment confirms the superiority of the new model compared to the α-power law in predicting the delay over a wide range of the supply voltage.

6.4 Generalization of the Model

In reference to Figure 6.1 we mentioned that even if an NMOS transistor is used to transfer a logic 1 instead of a logic 0, our concern would still be the saturation current. This suggests that we might be able to use the same model as well. What follows explains our attempt in extending the model to predict the delay of an NMOS transistor when charging a capacitance.

Figure 6.4 illustrates the simulation and our modeling results for this case a variety of supply voltages. The solid line is, as usual, obtained by HSPICE simulations. At the initial state of the discharging process, $V_{GS} = V_{DS} = V_{DD}$. While the capacitance is being discharged, both V_{GS} and V_{DS} decrease, but remain equal. At the end of the process, the output reaches $V_{DD}/2 = V_{GS} = V_{DS}$. Note that the minimum V_{DD} applied here is 1.5 V rather than 1 V, because V_{DD} has to be at least $2V_{TN}$ for the output to reach $V_{DD}/2$. As our first attempt, we use the same equation as the falling delay, but replace V_{DD} with 3/4 V_{DD}, i.e. the value of V_{GS} at the middle of the charging process. The result of this technique is shown with the dashed line. It loses its accuracy very fast as it approaches the lower end of the V_{DD} scale. One important factor that we have ignored is the body effect, which dynamically increases V_{TN} during the charging process. Hence, in the next attempt, the

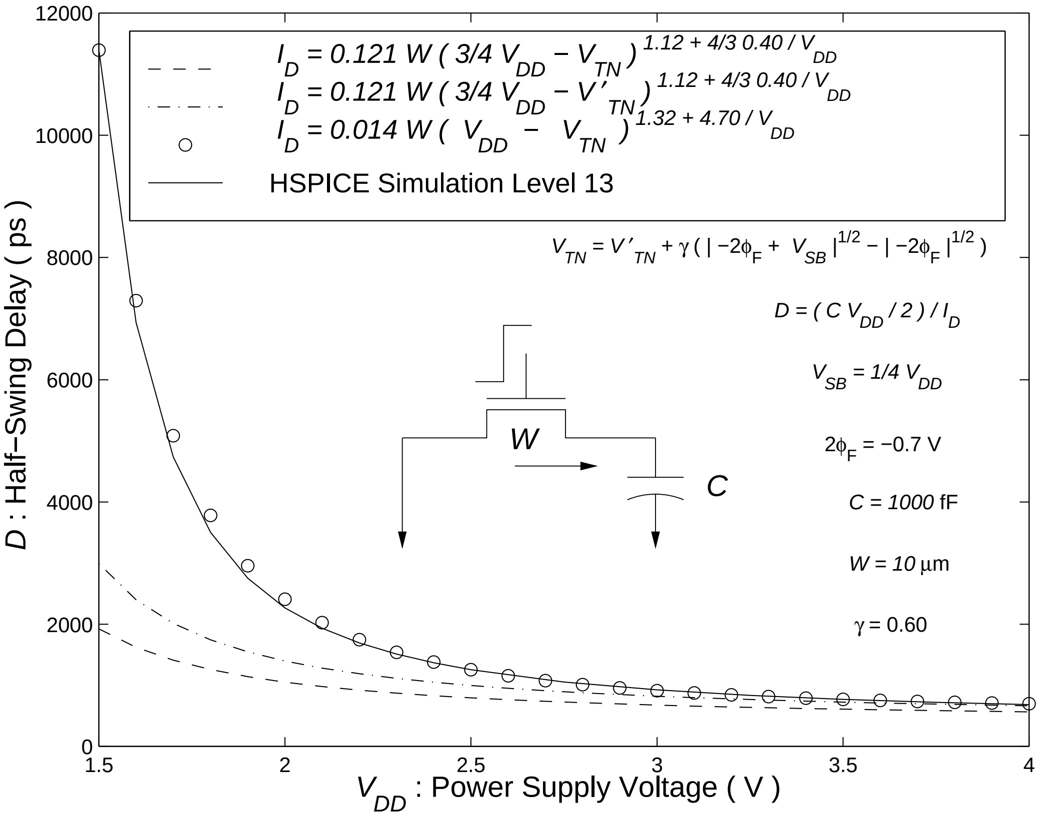


Figure 6.4: Step delay of an NMOS transistor charging an output capacitance as V_{DD} changes. HSPICE simulation results are compared with the results obtained by the new model.

result of which is shown by the dashed line with dots, we replace V_{TN} by V'_{TN} to include the body effect. The following, generally known [72], expression is used.

$$V'_T = V_T + \gamma\ (\sqrt{|-2\phi_F + V_{SB}|} - \sqrt{-2\phi_F}) \tag{6.6}$$

Where γ is the body effect coefficient, ϕ_F is the Fermi potential, and V_{SB} is the source to bulk potential. Midway in the process, $V_{SB} = 1/4\ V_{DD}$. This technique almost solves the problem for higher values of V_{DD}, but the current behaviour seems to be so non-linear that the technique obviously fails at lower values of V_{DD}. Observing that the shape of the delay curve resembles that of the discharging case, the final choice is to use direct fitting of the model into the simulation data. Thus, we define a separate set of parameters κ, ξ and ϑ for the case of the rising delay. This method is highly successful as depicted in the figure by

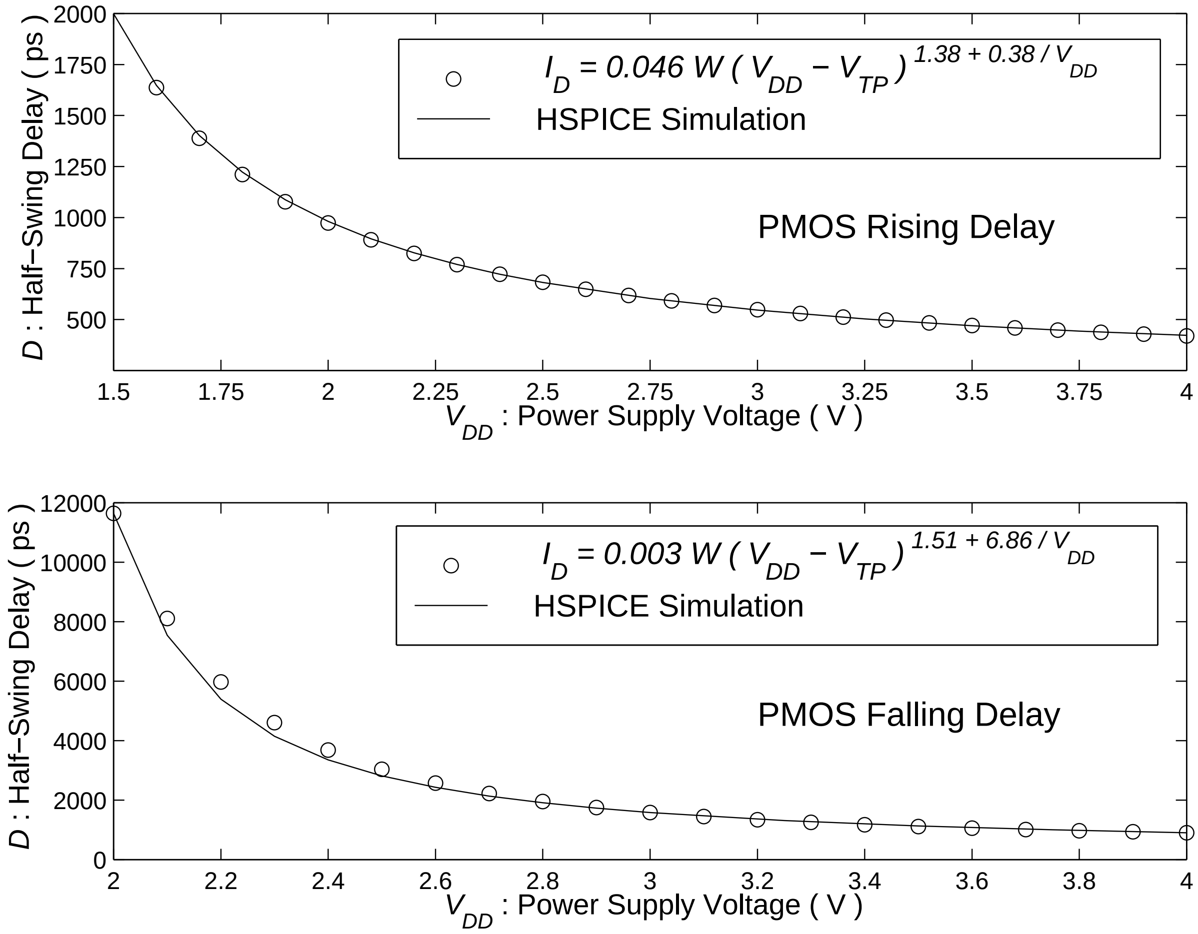


Figure 6.5: Charging and discharging delays of PMOS transistor as obtained by simulations and using the model.

the small circles. Notice that, comparing to the cases of the falling delay, ξ is only slightly larger, whereas ϑ is larger by an order of magnitude. This shows that the NMOS rising delay is much more sensitive to V_{DD} changes than its falling delay.

Figure 6.5 shows that the model is equally valid for PMOS transistors in the charging and discharging cases. Note that ξ for the PMOS rising delay is larger than ξ for the NMOS falling delay; this indicates that the velocity saturation effect is less severe in the PMOS transistors of this technology. We should mention that while ξ and ϑ of the NMOS falling delay and PMOS rising delay are directly related to the corresponding saturation currents and are associated with some physical phenomena, these parameters for NMOS rising delay and PMOS falling delay are not directly related to the same phenomena.

6.5 Effect of Input Waveform Slope on Delay

The simplest way of accounting for the effect of the input signal slope on the delay, is to use an approximation of the form [40]

$$D_T = D + S_\tau \, \tau \tag{6.7}$$

This technique adds a fraction S of the input rise or fall time τ to the step delay D to obtain the ramp delay D_T. It has been mentioned that this approximation is valid if the input slope exceeds one-third of the output slope [40]. On the ground that this rule is usually true in VLSI circuits, one may use the above formula. Otherwise, the input slope effect is much more complicated as analyzed for example in [7, 31, 49, 78]. In fact, if the input changes too slowly, the delay may be negative. When applying the above formula to a CMOS inverter, an additional source of error is the short-circuit current, which increases as the input slope decreases. Using his α-power law, Sakurai has derived the following expression for S_τ [76].

$$S_\tau = \frac{1}{2} - \frac{1 \; - \; V_T/V_{DD}}{1 + \alpha} \tag{6.8}$$

This is similar to the expression derived in [40] for long channel devices, where $\alpha = 2$. We use this expression in our work. However, instead of having an α value for NMOS devices and one for PMOS devices, we introduce four values for α corresponding to the four types of the delays. Figure 6.6 illustrates the relation between the four delay types and the input signal slope. The figure compares the simulation results with the above expression for the four cases at two different V_{DD} settings. The value of α for each case is stated in the figure's caption. This experiment demonstrates that with the right values of α, the same delay model as (6.7) can be used for charging and discharging cases through NMOS and PMOS transistors.

For delay calculations of cascaded gates, one should relate the input transition time τ of a gate to the step delay of the preceding, i.e. driving, gate. Although [76] has suggested a formula for this purpose, our simulations show that a step delay is approximately equivalent to half of the corresponding transition time. Hence, the delay of a gate i can be expressed in terms of its own step delay D_i and the step delay of it driving gate D_{i-1} as follows.

$$D_{Ti} = D_i + S D_{i-1} \tag{6.9}$$

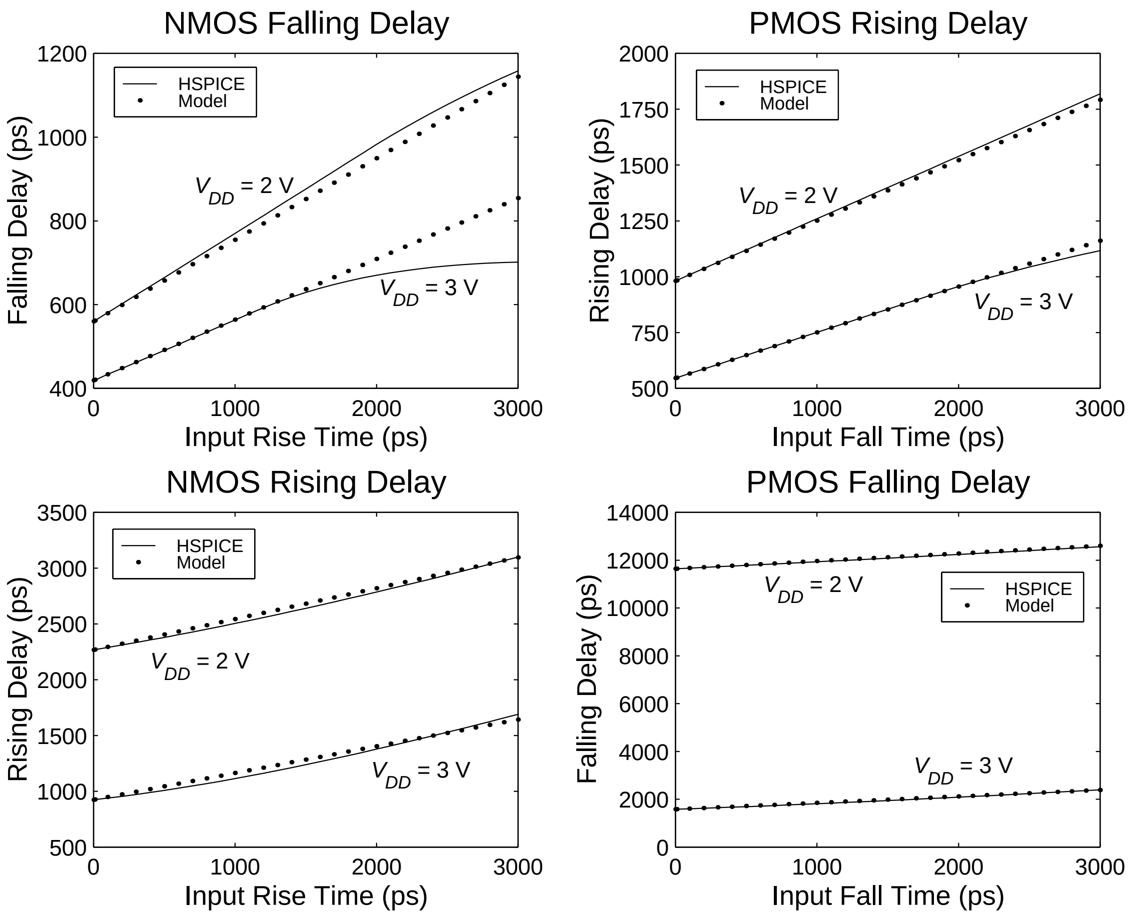


Figure 6.6: A comparison between the results of HSPICE simulations and the delay model for the four delay types: NMOS Falling ($\alpha = 1.2$), PMOS Rising ($\alpha = 1.35$), NMOS Rising ($\alpha = 2$), and PMOS Falling ($\alpha = 2$).

Where $S = 2S_\tau$ is called the slope factor. We define a slope factor for each of the four delay types.

6.6 Overlapping and Opposing Currents

In this section we discuss modifications of the delay expression for two cases which are common in digital CMOS circuits, namely, the cases of overlapping and opposing currents. A prominent example of the former is a CMOS transmission gate which is widely used in memory structures and multiplexers. Examples of the latter include level resorters, latches,

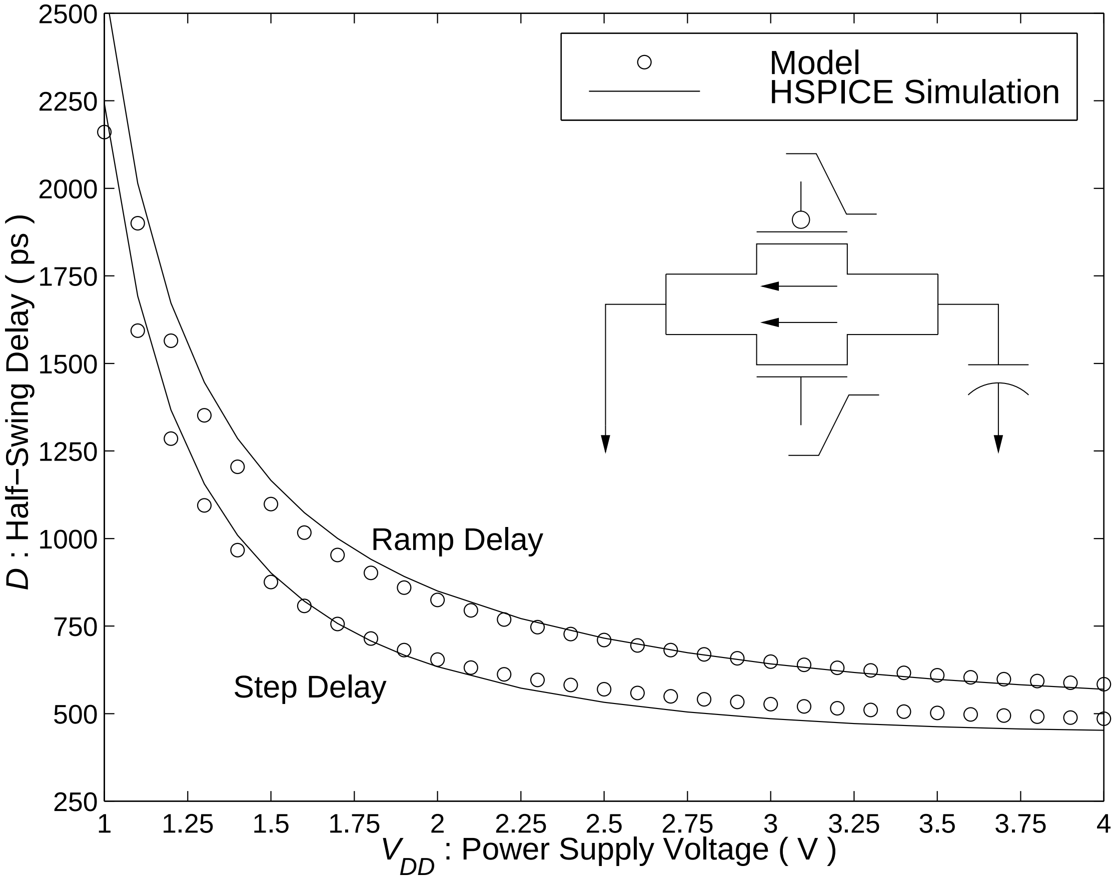


Figure 6.7: Variations of the step and ramp ($\tau = 1$ ns) delays of a CMOS transmission gate ($W_n = 10$ μm and $W_p = 20$ μm) discharging an output load ($C = 1$ pF).

and DCVSL gates.

Let's consider a CMOS transmission gate with a PMOS transistor of width W_p an NMOS transistor of width W_n, and an output load C. We develop an expression for the falling delay. A similar argument can be made for the rising delay. Since both transistors cooperate to discharge the output, the step delay can be simply written as

$$\grave{D} = \frac{C\ V_{DD}/2}{\grave{I}_n + \grave{I}_p} = \frac{\grave{\nu}_p \grave{\nu}_n}{W_p \grave{\nu}_n + W_n \grave{\nu}_p}\ C \tag{6.10}$$

Next, we consider the effect of the input signal slope. Predicting the exact effect of the input slope is very complicated, because in practice the arrival time and the slope of each input signal may be different from the other. We can, however, make reasonable assumptions. For all the cases when the difference between the arrival times of the two input signals is much smaller than the delay, we may assume that both transistors are activated at the same time with a similar rise and fall time τ. This is the ideal case of operation of a transmission gate. At the other extreme, for all the cases when the input signals are far apart relative to the delay, we may assume that only one of the transistors has contributed to the delay. This situation is avoided, because it defies the purpose of having both a PMOS transistor and an NMOS transistor in the circuit. For the ideal case, based on intuition, we propose that the slope factor S of each transistor should be weighted by its share of the total current. Hence, the overall slope factor is approximated by

$$\grave{S}_T = \frac{\grave{S}_n \, \grave{I}_n \; + \; \grave{S}_p \, \grave{I}_p}{\grave{I}_n \; + \; \grave{I}_p} = \frac{\grave{S}_n \, W_n/\grave{\nu}_n \; + \; \grave{S}_p \, W_p/\grave{\nu}_p}{W_n/\grave{\nu}_n \; + \; W_p/\grave{\nu}_p} \tag{6.11}$$

Thus, the value of the total slope factor is between the values of the individual slope factors. Note that this is valid independent of the formulation used for calculating the individual slope factors. Therefore, the total falling delay is given by

$$\grave{D}_T = \grave{D} \; + \; \frac{\grave{S}_T}{2} \, \tau \tag{6.12}$$

Figure 6.7 illustrates the close match achieved between this model and the results of HSPICE simulations for both step delay and ramp delay with $\tau = 1$ ns at different supply voltages.

Now we turn to the case of opposing currents. In this case, a network of one or more transistors tries to charge (discharge) a node while, simultaneously, the node is being discharged (charged) by, usually, a single transistor. The opposing transistor is often already active when the network is activated by an arriving input signal. For example, assume a capacitance C which is already charged by a PMOS transistor of width W_p and is about to discharge through an NMOS transistor of width W_n while the PMOS transistor remains active. For the step delay, we simply subtract the two currents and obtain

$$\grave{D} = \frac{\grave{\nu}_p \grave{\nu}_n}{W_n \grave{\nu}_p \; - \; W_p \grave{\nu}_n} \, C \tag{6.13}$$

For the ramp delay, we take a similar approach as in the case of the transmission gate. Since the input is only applied to the NMOS transistor, we weigh its slope factor by its share of

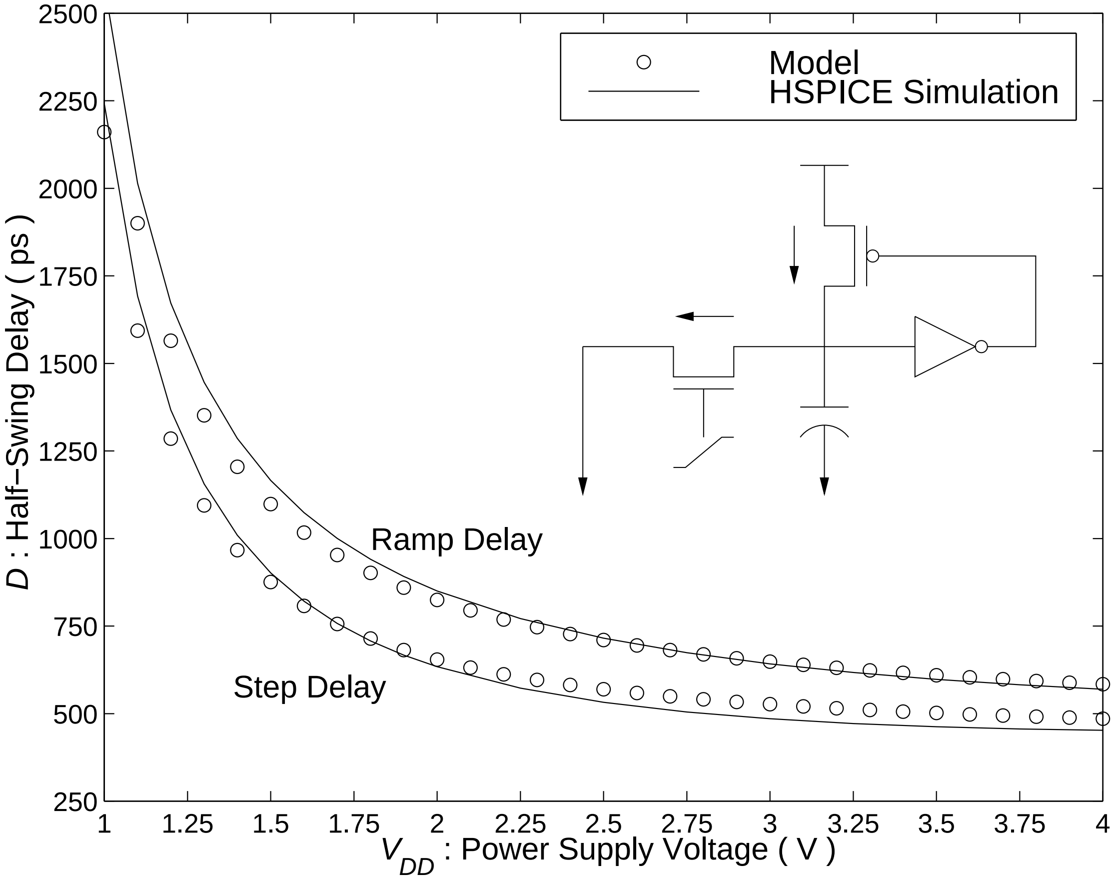


Figure 6.8: Variations of the step and ramp ($\tau = 1$ ns) falling delays of a CMOS structure involving opposing currents ($W_n = 10$ μm, $W_p = 20$ μm, and $C = 1$ pF).

the total current, and calculate the total slope factor by the following expression. The total delay can, then, be calculated using 6.12.

$$\grave{S}_T = \frac{\grave{S}_n \; W_n/\grave{\nu}_n}{W_n/\grave{\nu}_n \; - \; W_p/\grave{\nu}_p} \tag{6.14}$$

Therefore, the slope factor for the delay through a transistor influenced by opposing currents is higher than the delay through the same transistor operating free of such an influence. Figure 6.8 shows the match achieved between this model and the results of HSPICE simulations for both step delay and ramp delay with $\tau = 1$ ns at different supply voltages.

The technique described in these two examples can be applied to a variety of situations involving two types of currents. The important point is the ability to accurately predict the amount of each current. This is facilitated by using the current model presented in the previous sections.

6.7 MOS Transistors Connected in Series

One of the complications involved in modeling transistor circuits is delay estimation through a chain of series-connected devices. In such cases, usually, an equivalent RC network is used for modeling the delay. We assume a uniform size for the series-connected transistors in a gate. Another choice is the so called graduated [89] or progressive [72] sizing of cascaded transistors. An analytical procedure for progressive MOSFET sizing is presented in [8] and used in [100]. Progressive transistor sizing seems particularly useful in designing dynamic CMOS logic gates [89, 100, 104]. For today's short-channel devices, however, the literature suggests only marginal improvement in the delay by using progressive transistor sizing over uniform transistor sizing. In [45], it is reported that for a 3-input NAND gate, the graduated version shows less than 4% and 2% delay improvements over the uniform version for in 1.5 μm and 1 μm technologies, respectively. Hence, this delay improvement should even be less for deep submicron technologies. Besides, progressive transistor sizing increases the area and energy consumption of a gate.

Consider the RC network of Figure 6.9. Assume that initially all nodes are precharged to V_{DD}. In 1948, Elmore [35] showed that the voltage drop at node n, after a zero-going step is applied at the input, is governed by the following dominating time constant.

$$\tau_N = \sum_{i=1}^{N} C_i \sum_{j=1}^{i} R_j \tag{6.15}$$

Rubinstein, Penfield, and Horowitz generalized Elmore's theory for an RC tree. They also provided precise lower and upper bounds on the voltage waveforms in an RC tree [46, 74]. Many computer-aided timing analyzers have incorporated these theories.

Elmore's approximation is very useful, provided that the values of all resistances and capacitances are known. A major obstacle, however, is the determination of these value, especially those of the resistances. Sakurai and Newton have shown that short-channel devices do not obey the RC models used to predict the delay in series-connected long-channel

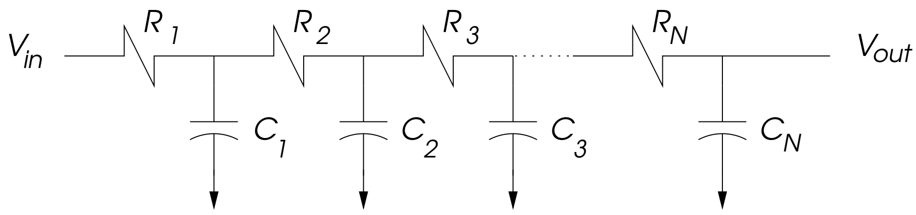


Figure 6.9: A general RC chain.

MOSFETs [77]. In a pull-up or pull-down chain of transistors, usually the transistor closest to the output operates in the saturation mode and the rest operate in the linear mode [77]. We use this assumption, combined with Elmore's theory, to simplify the delay modeling. This is reflected in Figure 6.10, where a chain of series connected transistors are replaced by an RC chain, and R_l stands for the equivalent resistance of a transistor in the linear mode, R stands for the equivalent resistance of a transistor in the saturation mode, C stands for the diffusion capacitance per transistor, and C_L stands for the load capacitance. Applying

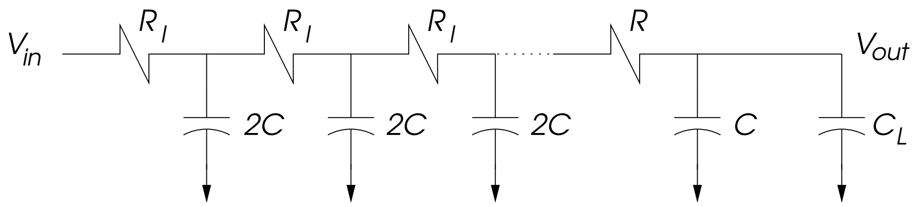


Figure 6.10: An equivalent RC chain for series-connected MOS transistors.

Elmore's approximation gives

$$\begin{aligned} \tau &= R_l\ 2C\ [1+2+3+...+(N-1)] + [R_l(N-1)\ +\ R]\ (C\ +\ C_L) && (6.16) \\ &= C\ [R_l(N^2-1)+R]\ +\ C_L[R_l(N-1)+R] && (6.17) \end{aligned}$$

which results in the following, if C_L is much larger than C.

$$\tau = C_L[R_l(N-1)+R] = [y(N-1)+1]RC_L \tag{6.18}$$

where $y = R_l/R$. On the other hand, for very large C_L, simulations show that the delay of n series-connected transistors is proportional to the delay of a single transistor with the same load through the voltage-dependent delay degradation factor Y.

$$Y = 1 + (N-1)(a\ -\ b\frac{V_{DD}}{V_T}) \tag{6.19}$$

Where a and b are empirical parameters with the typical values of 1.2 and 0.1, respectively. Comparing (6.19) with (6.18) shows that

$$y = a - b\frac{V_{DD}}{V_T} \tag{6.20}$$

Replacing R_l with yR in (6.17) yields

$$\tau = R\ (XC + YC_L) \tag{6.21}$$

Where $X = y(N^2 - 1) + 1$ and $Y = y(N - 1) + 1$. Hence, using our formulation, we may express the delay through N series-connected transistors by

$$D_N = \frac{\nu}{W}\ (XC + YC_L) \tag{6.22}$$

The values of a and b are obtained through simulations for each case. When C_L dominates, X can be set to zero. For fast hand-analysis, $Y = N$ is a good approximation.

Before concluding this section, we should mention an important point deduced from (6.22). Note that the diffusion capacitance C is proportional to the transistor width W. Hence, the first term in (6.22), that is, the term related to the internal nodes, is a constant and does not depend on W. Therefore, in pull-up and pull-down transistor chains, only the load capacitance determines the optimal transistor sizing for minimizing the delay, and the internal nodes can be ignored in that regard. We rely on this finding in dealing with optimization of CMOS logic circuits in the next chapter.

6.8 Extraction of Delay Parameters

To fully characterize the delays in a particular technology, we should extract four sets of parameters corresponding to four types of currents or delays. The symbols for these parameters and their values for our technology are listed in Table 6.1 and Table 6.2.

The following steps summarize the parameter extraction procedure. These steps have to be repeated for each of the four delay cases.

Table 6.1: Current Model Parameters, Symbols, and Values for a 0.5 μm CMOS Technology.

Delay Type	κ (**mA/μm/V**$^{-\xi-\vartheta/V_{DD}}$)	ξ	ϑ **(V)**
$\grave{D}_n$: NMOS Falling	$\grave{\kappa}_n = 0.12065$	$\grave{\xi}_n = 1.1265$	$\grave{\vartheta}_n = 0.4061$
$\acute{D}_p$: PMOS Rising	$\acute{\kappa}_p = 0.04564$	$\acute{\xi}_p = 1.3765$	$\acute{\vartheta}_p = 0.3756$
$\grave{D}_n$: NMOS Rising	$\acute{\kappa}_n = 0.01408$	$\acute{\xi}_n = 1.3203$	$\acute{\vartheta}_n = 4.6958$
$\acute{D}_p$: PMOS Falling	$\grave{\kappa}_p = 0.00295$	$\grave{\xi}_p = 1.5065$	$\grave{\vartheta}_p = 6.8598$

Table 6.2: Delay Degradation Parameters, Symbols, and Values for a 0.5 μm CMOS Technology.

Delay Type	α	a	b
$\grave{D}_n$: NMOS Falling	$\grave{\alpha}_n = 1.20$	$\grave{a}_n = 1.12$	$\grave{b}_n = 0.10$
$\acute{D}_p$: PMOS Rising	$\acute{\alpha}_n = 1.35$	$\acute{a}_p = 1.45$	$\acute{b}_p = 0.14$
$\grave{D}_n$: NMOS Rising	$\acute{\alpha}_p = 2.00$	$\acute{a}_n = 1.25$	$\acute{b}_n = 0.09$
$\acute{D}_p$: PMOS Falling	$\grave{\alpha}_p = 2.00$	$\grave{a}_p = 1.45$	$\grave{b}_p = 0.09$

1. Select a relatively large output capacitance C (e.g. 1 pF) and a reasonable transistor size W (e.g. 10 μm).

2. Measure the step delays D_1, D_2, and D_3 at three different supply voltage settings V_1, V_2, and V_3, respectively.

3. Calculate ξ and ϑ from the following set of equations.

$$\begin{aligned}
\xi \ \log\left(\frac{V_2 - V_T}{V_1 - V_T}\right) \ &+ \ \vartheta \ \log\left[\frac{(V_2-V_T)^{\frac{1}{V_2}}}{(V_1-V_T)^{\frac{1}{V_1}}}\right] \ &= \ \log\left(\frac{D_1}{D_2}\right) - \log\left(\frac{V_1}{V_2}\right) \\
\xi \ \log\left(\frac{V_3 - V_T}{V_2 - V_T}\right) \ &+ \ \vartheta \ \log\left[\frac{(V_3-V_T)^{\frac{1}{V_3}}}{(V_2-V_T)^{\frac{1}{V_2}}}\right] \ &= \ \log\left(\frac{D_2}{D_3}\right) - \log\left(\frac{V_2}{V_3}\right)
\end{aligned}$$

4. Calculate κ from

$$\kappa = \frac{C \ V_2}{2 \ D_2 \ W \ (V_2 - V_T) \ ^{\xi+\vartheta/V_2}}$$

5. At $V_{DD} = V_2$ apply a ramp signal with rise time or fall time τ (e.g. 1 ns) and measure the delay D_T. Calculate α from

$$\alpha = \left(1 - \frac{V_T}{V_2}\right) / \left(\frac{1}{2} - \frac{D_T - D_2}{\tau}\right)$$

6. At V_1 measure the step delay D_{11} due to two identical transistors of width W connected in series to C. Similarly, at V_2 measure the step delay D_{22} due to two identical transistors of width W connected in series to C. Calculate a and b from the following set of equations.

$$\begin{aligned} \frac{D_{11}}{D_1} &= 1 + a - b\frac{V_1}{V_T} \\ \frac{D_{22}}{D_2} &= 1 + a - b\frac{V_2}{V_T} \end{aligned}$$

6.9 Extraction of MOSFET Capacitances

The gate and diffusion capacitances in MOS devices are both voltage dependent and, thus, have different average values during the rising delay and the falling delay. We use $\acute{g}_p$, $\grave{g}_p$, $\acute{g}_n$, and $\grave{g}_n$ corresponding to gate capacitance per unit width for PMOS rising delay, PMOS falling delay, NMOS rising delay, and NMOS falling delay, respectively. Similarly, $\acute{d}_p$, $\grave{d}_p$, $\acute{d}_n$, and $\grave{d}_n$ denote diffusion capacitances per unit width for the above cases, respectively. The values of these parameters for our technology are listed in Table 6.3.

Table 6.3: Gate and Diffusion Capacitances per Unit Width: Symbols and Values for a 0.5 μm CMOS Technology.

Type / Transition	g **(fF/μm)**	d **(fF/μm)**
NMOS / Falling	$\grave{g}_n = 2.00$	$\grave{d}_n = 1.25$
PMOS / Rising	$\acute{g}_p = 2.25$	$\acute{d}_p = 2.50$
NMOS / Rising	$\acute{g}_n = 1.55$	$\acute{d}_n = 1.75$
PMOS / Falling	$\grave{g}_p = 1.35$	$\grave{d}_p = 3.00$

It is important to note that for energy calculations the average gate and diffusion capacitances over one switching cycle should be used. The average PMOS and NMOS gate capacitances per unit width are given by

$$g_p = \frac{\acute{g}_p + \grave{g}_p}{2} \qquad g_n = \frac{\acute{g}_n + \grave{g}_n}{2} \tag{6.23}$$

respectively. Similarly, the average PMOS and NMOS diffusion capacitances per unit width are given by

$$d_p = \frac{\acute{d}_p + \grave{d}_p}{2} \qquad d_n = \frac{\acute{d}_n + \grave{d}_n}{2} \tag{6.24}$$

respectively.

For quick estimation of the delay and energy, one may use the following values for a transistor gate capacitance per unit width and a transistor diffusion capacitance per unit width, respectively.

$$g = \frac{g_p + g_n}{2} \qquad d = \frac{d_n + d_p}{2} \tag{6.25}$$

This is the approximation used in the formulation of Chapter 3. For the 0.5 μm CMOS technology, according to Table 6.3, $g = 1.85$ fF/μm and $d = 2.13$ fF/μm. For example, if the load of a logic gate entirely consists of gates of NMOS transistors, this approximation gives about 20% error in estimating the rising step delay due to the load. Therefore, the approximation is, particularly, not suitable for PTL circuits.

Consider the inverter chain of Figure 6.11. The rising and falling propagation delay between the input and the output of the cell are given by

$$\begin{aligned} \acute{D}_{dc} &= \frac{\acute{\nu}_p}{W_p}(W_{Lp}\,\acute{g}_p + W_{Ln}\,\acute{g}_n + W_p\,\acute{d}_p + W_n\,\acute{d}_n) \\ &\quad + \acute{S}_p\frac{\grave{\nu}_n}{W_{Dn}}(W_p\,\grave{g}_p + W_n\,\grave{g}_n + W_{Dp}\,\grave{d}_p + W_{Dn}\,\grave{d}_n) \end{aligned} \tag{6.26}$$

$$\begin{aligned} \grave{D}_{dc} &= \frac{\grave{\nu}_n}{W_n}(W_{Lp}\,\grave{g}_p + W_{Ln}\,\grave{g}_n + W_p\,\grave{d}_p + W_n\,\grave{d}_n) \\ &\quad + \grave{S}_n\frac{\acute{\nu}_p}{W_{Dp}}(W_p\,\acute{g}_p + W_n\,\acute{g}_n + W_{Dp}\,\acute{d}_p + W_{Dn}\,\acute{d}_n) \end{aligned} \tag{6.27}$$

The following procedure may be used to extract the values of the MOS capacitances.

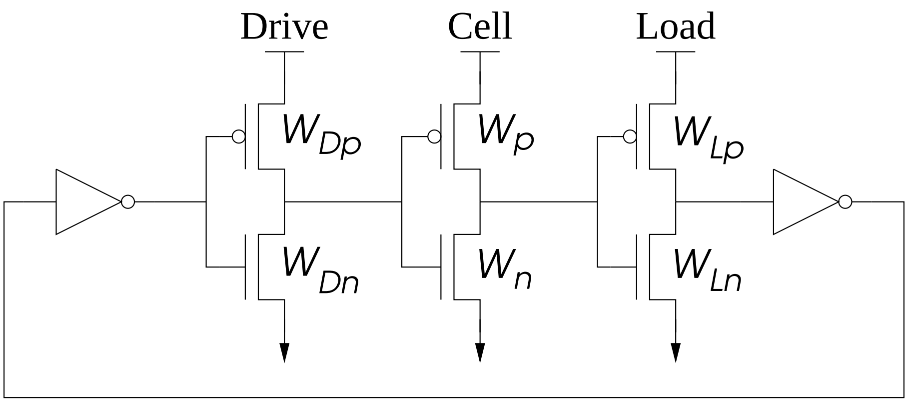


Figure 6.11: A CMOS inverter chain.

1. Initially, let $W_p = W_{Dp} = W_{Lp} = W_{1p}(= 20\mu\text{m})$ and $W_n = W_{Dn} = W_{Ln} = W_{1n}(= 10\mu\text{m})$. Measure $\acute{D}_1 = \acute{D}_{dc}$ and $\grave{D}_1 = \grave{D}_{dc}$.

2. Let $W_{Lp} = W_{2p}(= 30\mu\text{m})$. Measure $\acute{D}_2$ and $\grave{D}_2$. Then, calculate the following.

$$\acute{g}_p = \frac{W_{1p}}{\acute{\nu}_p} \frac{\acute{D}_1 - \acute{D}_2}{W_{1p} - W_{2p}}$$

$$\grave{g}_p = \frac{W_{1n}}{\grave{\nu}_n} \frac{\grave{D}_1 - \grave{D}_2}{W_{1p} - W_{2p}}$$

3. Reset all transistor sizes back to Step 1. Let $W_{Ln} = W_{3n}(= 20\mu\text{m})$. Measure $\acute{D}_3$ and $\grave{D}_3$. Then, calculate the following.

$$\acute{g}_n = \frac{W_{1p}}{\acute{\nu}_p} \frac{\grave{D}_1 - \grave{D}_3}{W_{1n} - W_{3n}}$$

$$\grave{g}_n = \frac{W_{1n}}{\grave{\nu}_n} \frac{\grave{D}_1 - \grave{D}_3}{W_{1n} - W_{3n}}$$

4. Reset all transistor sizes back to Step 1. Let $W_p = W_{4p}(= 30\mu\text{m})$. Measure $\grave{D}_4$. Then, calculate the following.

$$\grave{d}_p = \frac{W_{1n}}{\grave{\nu}_n} \left(\frac{\grave{D}_1 - \grave{D}_4}{W_{1p} - W_{4p}} - \grave{S}_n \frac{\acute{\nu}_p}{W_{1p}} \acute{g}_p \right)$$

5. Reset all transistor sizes back to Step 1. Let $W_n = W_{5n}(= 20\mu\text{m})$. Measure $\acute{D}_5$. Then, calculate the following.

$$\acute{d}_n = \frac{W_{1p}}{\acute{\nu}_p} \left(\frac{\acute{D}_1 - \acute{D}_5}{W_{1n} - W_{5n}} - \acute{S}_p \frac{\grave{\nu}_n}{W_{1n}} \grave{g}_n \right)$$

6. Reset all transistor sizes back to Step 1. The only unknown parameters are $\acute{d}_p$ and $\grave{d}_n$. Change the width ratio of PMOS transistor to NMOS transistor in the inverters from 2, as in Step 1, to 1.5. Measure $\acute{D}_6$ and $\grave{D}_6$. Express $\acute{D}_1 - \acute{D}_6$ and $\grave{D}_1 - \grave{D}_6$. Solve these two equations for the two unknowns.

A diffusion junction capacitance is a non-linear function of the voltage across the junction. It is possible to obtain an average value for C_j over the voltage range of interest, V_1 to V_2, using the following.

$$C_j = \frac{C_{j0}}{V_2 - V_1} \int_{V_1}^{V_2} \left[1 - \frac{V}{V_b}\right]^{-m} dV. \tag{6.28}$$

Where C_{j0} is the value of the capacitance at zero potential, V_b is the junction's built-in potential, and m is the grading coefficient. The above expression yields:

$$C_j = C_{j0} \frac{V_b^m \left[(V_b + V_2)^{(1-m)} - (V_b + V_1)^{(1-m)}\right]}{(m-1)(V_1 - V_2)}. \tag{6.29}$$

For energy calculations we are interested in the value of the junction capacitance over the range of full voltage swing C_{jFs}.

$$C_{jFs} = C_{j0} \frac{V_b^m \left[(V_b + V_{DD})^{(1-m)} - V_b^{(1-m)}\right]}{(m-1)V_{DD}} \tag{6.30}$$

For delay calculations, however, we are interested in the value of the junction capacitance over the range of half voltage swing. Hence, we need to have two expressions: one for the rising delay

$$C_{jHsRt} = 2C_{j0} \frac{V_b^m \left[(V_b + \frac{1}{2}V_{DD})^{(1-m)} - V_b^{(1-m)}\right]}{(1-m)V_{DD}} \tag{6.31}$$

and one for the falling delay

$$C_{jHsFt} = 2C_{j0} \frac{V_b^m \left[(V_b + V_{DD})^{(1-m)} - (V_b + \frac{1}{2}V_{DD})^{(1-m)}\right]}{(1-m)V_{DD}}. \tag{6.32}$$

We presented a method of extracting the diffusion capacitances for a reference supply voltage through simulations. Comparing the calculated and extracted values for the reference V_{DD} (3 V in our case), gives us an adjustment factor. For a different V_{DD}, first we calculate the diffusion capacitances using the above formulas. Then, we multiply the adjustment factor with the calculated results to obtain the values of the diffusion capacitances for using in our model.

6.10 Concluding Remarks

In this chapter, we have presented a discussion on the components of a new, unified delay model for CMOS circuits, which is applicable to pull-up and pull-down chains of both PMOS and NMOS transistors. This is especially useful in pass-transistor circuits, where NMOS transistors usually are used to transfer a logic 0 as well as a logic 1. The delay components include the current, the gate and diffusion capacitances, the input slope factor, and the series-connection delay degradation factor. We have also offered procedures for extracting the values of these components through circuit-level simulations.

Chapter 7

Modeling and Optimization of CMOS Logic Styles

A method of enhancing performance and saving energy in digital CMOS circuits is to use a combination of conventional and non-conventional logic styles, such as DCVSL and PTL, when appropriate. Unfortunately, the literature has little to say about delay and energy calculations in unconventional CMOS styles in general. This chapter applies the unified delay model of the previous chapter to CMOS gates implemented in the conventional, DCVSL, and PTL styles. It also presents closed-form formulas for optimal transistor sizing in each style. These formulas are simplified further to rules of thumb for quick optimization of CMOS logic circuits.

Since the delay of a gate is characterized by its driving gate and its load, we write the delay expression for three cascaded gates. We refer to the gate at the middle as the cell. The advantage of expressing the delay for three cascaded gates is that one can obtain closed-form formula for optimum transistor sizing of the cell with respect to its drive and its load. These formulas can then be used to optimize a critical path of an arbitrary number of gates. For convenience, we deal with the gates at an abstract level. That is, we concern ourselves only with the critical path in a gate and assume that all series-connected transistors along that path have the same size.

The formulation of Chapter 3 neglects the signal slope effect, has a long-channel view of series-connected transistors, assumes a uniform PMOS to NMOS size ratio for all logic gates, uses a uniform gate capacitance value per unit width for PMOS and NMOS transistors, uses a uniform diffusion capacitance value per unit width for PMOS and NMOS transistors, uses similar capacitance values for rising and falling transitions, and, most importantly, is confined to the conventional CMOS logic style. In this chapter, we remove all of these limitations.

The previous chapter showed that the internal node capacitances in a chain of similarly sized, series-connected transistors do not affect optimal transistor sizing. Hence, for simplification, the delay expressions in this chapter, which are used to derive the optimal transistor sizing formulas, do not include those capacitances. Their contributions to the delay, however, have been taken into account for producing the delay curves, as explained in the previous chapter.

We use the familiar XOR gate as an example to verify our technique. The XOR gate, known as the MERGE in delay-insensitive circuits [34], is widely used in asynchronous design. This chapter also includes a comparison between various CMOS implementations of the XOR gate. We assume that the XOR gate is operating in an asynchronous environment as a MERGE. This assumption imposes the following.

- Gate optimization concerns the total delay over one switching cycle.
- One input may change at a time followed by an output transition.

7.1 Conventional CMOS Style

The most widely practiced CMOS logic style is the conventional one, which is well introduced in text books such as [51, 72, 102]. The conventional CMOS logic style usually offers a good trade-off between the performance and energy consumption.

Figure 7.1 depicts the schematic of a CMOS cell implemented in the conventional pull-up pull-down style with its drive and its load. The input of the drive is assumed to be a step voltage function. Capacitance C_d includes the interconnect capacitances and any other gate and diffusion capacitances at the output of the drive. Similarly, Capacitance C_l includes the interconnect capacitances and any other gate and diffusion capacitances at the output

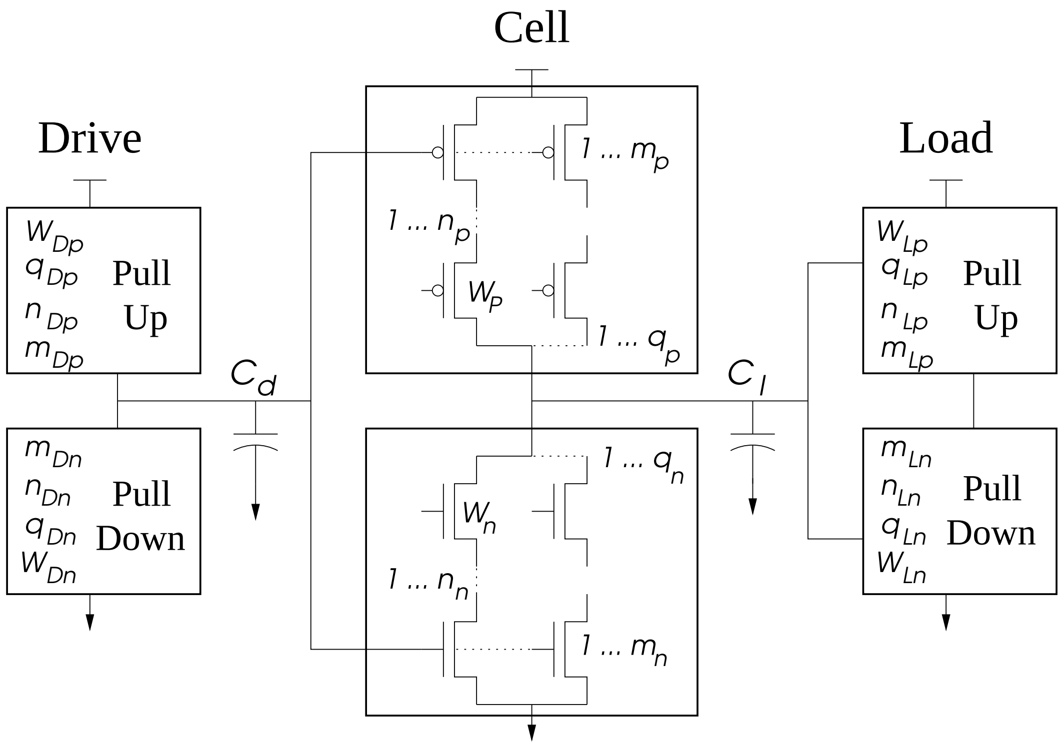


Figure 7.1: A conventional CMOS cell between a driving gate and output load.

of the cell. Hence, the values for C_d and C_l may be different values for a rising delay and for a falling delay. The critical path in the pull-up section of the cell consists of n_p PMOS transistors of which, usually, the closest one to V_{DD} is driven by the drive. This scenario produces the worst-case delay if the transition time of the input signal to the cell is not very large [77]. Each of the series-connected transistors in the critical path have width W_p. Due to symmetry, the drive may also be connected to some other transistors of the same width within the pull-up section of the cell. Let m_p indicate the number of these transistors. At the output of the cell, again due to symmetry, there are q_p drain diffusions of width W_p. The pull-down section of the cell can be described in a similar fashion by replacing W_p with W_n, n_p with n_n, m_p with m_n, and q_p with q_n. Each pull-up or pull-down section in the drive or the load is also characterized by some W, n, m, and q. Therefore, we are considering a general case.

As an example consider the carry generation circuit shown in Figure 7.2 [72]. Assume that signal b is along the critical path. Considering the pull-up section, for instance, symmetry may force one to optimize both branches involving b simultaneously. Hence, $m_p = 2$,

$n_p = 2$, and $q_p = 2$. Symmetry has to be usually observed in designing cells for a library of logic gates. In most other circumstances, however, only one of the branches determines the worst-case delay. In such cases, the symmetry is void and $m_p = 1$, $n_p = 2$, and $q_p = 1$.

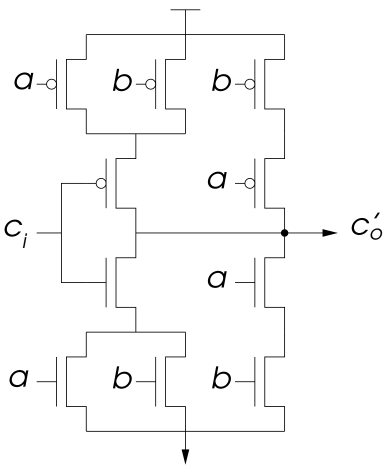


Figure 7.2: Carry generation circuit in a mirror adder.

Turning back to Figure 7.1, let's combine all capacitances at the output of the drive except those related to W_p into one component C_D as follows.

$$\acute{C}_D = \acute{C}_d + q_{Dp}W_{Dp}\,\acute{d}_p + W_{Dn}\,q_{Dn}\,\acute{d}_n \tag{7.1}$$

$$\grave{C}_D = \grave{C}_d + q_{Dp}W_{Dp}\,\grave{d}_p + W_{Dn}\,q_{Dn}\,\grave{d}_n \tag{7.2}$$

Similarly, let C_L represent all capacitances at the output of the cell except those related to W_p.

$$\acute{C}_L = \acute{C}_l + m_{Lp}W_{Lp}\,\acute{g}_p + W_{Ln}\,m_{Ln}\,\acute{g}_n \tag{7.3}$$

$$\grave{C}_L = \grave{C}_l + m_{Lp}W_{Lp}\,\grave{g}_p + W_{Ln}\,m_{Ln}\,\grave{g}_n \tag{7.4}$$

The rising delay between the input of the drive to the output of the cell can be expressed as

$$\acute{D} = \acute{D}_C + (1 + \acute{S}_p)\,\grave{D}_D \tag{7.5}$$

where $\acute{D}_C$ and $\grave{D}_D$ are the step delays of the cell and the drive, respectively.

$$\acute{D}_C = \frac{\acute{\nu}_p}{W_p}\,\acute{Y}_p\,(q_nW_n\,\acute{d}_n + q_pW_p\,\acute{d}_p + \acute{C}_L) \tag{7.6}$$

$$\grave{D}_D = \frac{\grave{\nu}_n}{W_{Dn}}\,\grave{Y}_{Dn}(m_nW_n\,\grave{g}_n + m_pW_p\,\grave{g}_p + \grave{C}_D) \tag{7.7}$$

By solving $\frac{\partial \acute{D}}{\partial W_p} = 0$ one may obtain the optimum W_p for minimizing the rising delay.

$$\acute{W}_p = \sqrt{W_{Dn} \frac{\acute{C}_L + q_n W_n \acute{d}_n}{m_p \grave{g}_p} \frac{\acute{\nu}_p \acute{Y}_p}{\grave{\nu}_n \grave{Y}_{Dn}(1 + \acute{S}_p)}} \tag{7.8}$$

Rearranging the terms in the above expression leads to

$$\frac{\grave{\nu}_n}{W_{Dn}} \grave{Y}_{Dn} m_p \acute{W}_p \grave{g}_p (1 + \acute{S}_p) = \frac{\acute{\nu}_p}{\acute{W}_p} \acute{Y}_p \left(q_n W_n \acute{d}_n + \acute{C}_L\right) \tag{7.9}$$

The term at the left is the falling delay at the input of the cell due to only W_p, and the term at the right is the rising delay at the output of the cell through W_p due to the load excluding W_p. Thus, the optimum W_p for rising delay must satisfy the following condition.

Input falling delay due to W_p as load= Output rising delay through W_p excluding its own load

This expression may be modified into a better form by observing the addition of the term

$$\frac{\acute{\nu}_p}{\acute{W}_p} \acute{Y}_p q_p \acute{W}_p \acute{d}_p$$

to both sides of (7.9). By doing so, the right hand side will become the total delay in producing a rising output transition due to W_p as load and the left hand side will become the total delay in producing that output through W_p as a drive . Hence, the optimum sizing condition may be restated in this form.

Delay due to W_p as load = Delay through W_p as drive

Now we try to impose some approximations to simplify (7.8). Note that, in general, $\acute{W}_p$ depends on W_n. If the output load is much larger than the diffusion capacitance related to W_n, then $q_n W_n \acute{d}_n$ may be ignored in favour of $\acute{C}_L$ in (7.8). By using this approximation and replacing $\acute{\nu}_p / \grave{\nu}_n$ with Λ, (7.8) takes the following form[1].

$$\acute{W}_p = \sqrt{\Lambda W_{Dn} \frac{\acute{C}_L}{m_p \grave{g}_p} \frac{\acute{Y}_p}{\grave{Y}_{Dn}(1 + \acute{S}_p)}} \tag{7.10}$$

We need to perform further approximations to obtain a simple rule of thumb for circuit designers in optimizing the rising delay. A circuit designer doesn't know the exact value of

[1] Λ is equivalent to ρ defined in Chpetr 3.

$\acute{C}_L$ before examining the layout. However, since the fanout gates are known, $\acute{C}_L$ may be replaced by the total width of the fanout W_{Lt} times the average gate capacitance per unit width g. Using this approximation for $\acute{C}_L$, $\acute{Y}_p \approx n_p$, $\grave{Y}_{Dn} \approx n_{Dn}$, and neglecting the input slope effect in (7.10) yields the following rule of thumb for optimizing the rise time.

$$\acute{W}_p = \sqrt{\Lambda W_{Dn} W_{Lt} \frac{n_p}{m_p n_{Dn}}} \tag{7.11}$$

An advantage of (7.11) is that its parameters are physical and technology independent, except Λ. For Λ, there is evidence that in the future deep submicron technologies it has a constant value of around 2.2 [13, 14].

Similar to $\acute{W}_p$, there is an optimum $W_n = \grave{W}_n$ for the falling delay. The falling delay from the input of the drive to the output of the cell can be expressed as

$$\grave{D} = \acute{D}_C + (1 + \grave{S}_n)\, \grave{D}_D \tag{7.12}$$

where

$$\grave{D}_C = \frac{\grave{\nu}_n}{W_n} \grave{Y}_n (q_n W_n \grave{d}_n + q_p W_p \grave{d}_p + \grave{C}_L) \tag{7.13}$$

$$\acute{D}_D = \frac{\acute{\nu}_p}{W_{Dp}} \acute{Y}_{Dp} (m_n W_n \acute{g}_n + m_p W_p \acute{g}_p + \acute{C}_D) \tag{7.14}$$

The optimum W_n for the falling delay is then given by the following exact and approximated formulas. In the approximated form, the output load is assumed to be much larger than the diffusion capacitance related to W_p.

$$\grave{W}_n = \sqrt{W_{Dp} \frac{\grave{C}_L + q_p W_p \grave{d}_p}{m_n \acute{g}_n} \frac{\grave{\nu}_n \grave{Y}_n}{\acute{\nu}_p \acute{Y}_{Dp}(1 + \grave{S}_n)}} \approx \sqrt{W_{Dp} \frac{\grave{C}_L}{m_n \acute{g}_n} \frac{\grave{\nu}_n \grave{Y}_n}{\acute{\nu}_p \acute{Y}_{Dp}(1 + \grave{S}_n)}} \tag{7.15}$$

By some rearrangements the following condition now appears.

Input rising delay due to W_n as load= Output falling delay through W_n excluding its own load

This may be modified to

Delay due to W_n as load = Delay through W_n drive

Applying the same approximations as for $\acute{W}_p$, results in the following rule of thumb for minimizing the falling delay.

$$\grave{W}_n = \sqrt{\frac{W_{Dp} W_{Lt}}{\Lambda} \frac{n_n}{m_n n_{Dp}}} \tag{7.16}$$

Note that there are no optimum W_p and W_n for the falling and the rising delays, respectively. Summing the rising and falling delays gives the total delay.

$$\hat{D} = \acute{D} + \grave{D} \tag{7.17}$$

From the total delay we can obtain a set of equations for the optimum sizing by setting $\partial\hat{D}/\partial W_p = \partial\hat{D}/\partial W_n = 0$. However, we may as well use a short-cut method by extrapolating the rules obtained in the optimization of the rising and falling delays. We speculate that optimized values of W_p and W_n for the total delay would satisfy the following set of conditions.

$$\textit{Total Delay due to } W_p \textit{ as load} = \textit{Total Delay through } W_p$$
$$\textit{Total Delay due to } W_n \textit{ as load} = \textit{Total Delay through } W_n$$

Consider the first condition. If the similar terms are cancelled out from both sides of the equality, the left hand side has three components: one component due to the gate capacitance during the input rising transition, one due to the gate capacitance during the input falling transition, and one due to the diffusion capacitance during the output falling transition. The right hand side has only one component, since W_p is only active during the output rising transition. Writing these components in the same order, the above speculation translates into

$$\begin{gathered}(1+\acute{S}_p)m_p\hat{W}_p\,\grave{g}_p\tfrac{\grave{\nu}_n}{W_{Dn}}\,\grave{Y}_{Dn} + (1+\grave{S}_n)m_p\hat{W}_p\,\acute{g}_p\tfrac{\acute{\nu}_p}{W_{Dp}}\,\acute{Y}_{Dp} + q_p\hat{W}_p\,\grave{d}_p\tfrac{\grave{\nu}_n}{W_n}\,\grave{Y}_n \\ = (\acute{C}_L + q_nW_n\,\acute{d}_n)\tfrac{\acute{\nu}_p}{\hat{W}_p}\,\acute{Y}_p\end{gathered}$$

Similarly, the second condition for the optimum W_n translates into

$$\begin{gathered}(1+\grave{S}_n)m_n\hat{W}_n\,\acute{g}_n\tfrac{\acute{\nu}_p}{W_{Dp}}\,\acute{Y}_{Dp} + (1+\acute{S}_p)m_n\hat{W}_n\,\grave{g}_n\tfrac{\grave{\nu}_n}{W_{Dn}}\,\grave{Y}_{Dn} + q_n\hat{W}_n\,\acute{d}_n\tfrac{\acute{\nu}_p}{W_p}\,\acute{Y}_p \\ = (\grave{C}_L + q_pW_p\,\grave{d}_p)\tfrac{\grave{\nu}_n}{\hat{W}_n}\,\grave{Y}_n\end{gathered}$$

Combining the above two conditions, gives the following set of formulas for simultaneous optimization of the pull-up and pull-down sections of the cell.

$$\hat{W}_p = \sqrt{\frac{(\acute{C}_L + q_nW_n\,\acute{d}_n)\,\acute{\nu}_p\,\acute{Y}_p}{m_p\,\grave{g}_p\tfrac{\grave{\nu}_n}{W_{Dn}}\,\grave{Y}_{Dn}(1+\acute{S}_p) + m_p\,\acute{g}_p\tfrac{\acute{\nu}_p}{W_{Dp}}\,\acute{Y}_{Dp}(1+\grave{S}_n) + q_p\,\grave{d}_p\tfrac{\grave{\nu}_n}{W_n}\,\grave{Y}_n}} \tag{7.18}$$

$$\approx \sqrt{\frac{\acute{C}_L\ \acute{\nu}_p\ \acute{Y}_p}{m_p\ \grave{g}_p \frac{\grave{\nu}_n}{W_{Dn}}\ \grave{Y}_{Dn}(1+\ \acute{S}_p)+m_p\ \acute{g}_p \frac{\acute{\nu}_p}{W_{Dp}}\ \acute{Y}_{Dp}(1+\ \grave{S}_n)}} \tag{7.19}$$

$$\hat{W}_n = \sqrt{\frac{(\ \grave{C}_L + q_p W_p\ \grave{d}_p)\ \grave{\nu}_n\ \grave{Y}_n}{m_n\ \acute{g}_n \frac{\acute{\nu}_p}{W_{Dp}}\ \acute{Y}_{Dp}(1+\ \grave{S}_n)+m_n\ \grave{g}_n \frac{\grave{\nu}_n}{W_{Dn}}\ \grave{Y}_{Dn}(1+\ \acute{S}_p)+q_n\ \acute{d}_n \frac{\acute{\nu}_p}{W_p}\ \acute{Y}_p}} \tag{7.20}$$

$$\approx \sqrt{\frac{\grave{C}_L\ \grave{\nu}_n\ \grave{Y}_n}{m_n\ \acute{g}_n \frac{\acute{\nu}_p}{W_{Dp}}\ \acute{Y}_{Dp}(1+\ \grave{S}_n)+m_n\ \grave{g}_n \frac{\grave{\nu}_n}{W_{Dn}}\ \grave{Y}_{Dn}(1+\ \acute{S}_p)}} \tag{7.21}$$

Solving $\partial\hat{D}/\partial W_p = \partial\hat{D}/\partial W_n = 0$ confirms our speculations, as it leads exactly to the same results. If the approximated formulas are used, then they are independent and may be solved separately. Otherwise, the two equations can be solved together by iteration. The value obtained from the one can be used in the other and so forth until the changes in the values are very small. Usually it takes a few iterations to get the final results. The initial values of $\hat{W}_p$ and $\hat{W}_n$ for starting the iteration process may be obtained by the approximated set of formulas. A point worth mentioning is that the optimization is supply voltage dependent mainly through $\acute{\nu}_p$ and $\grave{\nu}_n$. Using further approximations, as for the cases of the rising and falling delays, leads to the following rules of thumb for optimization of the total delay.

$$\hat{W}_p = \sqrt{\Lambda\ \frac{n_p}{m_p}\ \frac{W_{Dp}W_{Dn}W_{Lt}}{n_{Dn}W_{Dp}+\Lambda n_{Dp}W_{Dn}}} \tag{7.22}$$

$$\hat{W}_n = \sqrt{\frac{n_n}{m_n}\ \frac{W_{Dp}W_{Dn}W_{Lt}}{n_{Dn}W_{Dp}+\Lambda n_{Dp}W_{Dn}}} = \frac{\hat{W}_p}{\hat{\Gamma}} \tag{7.23}$$

Where $\hat{\Gamma}$ is the optimum width ratio of PMOS to NMOS transistors for minimizing total delay given by[2]

$$\hat{\Gamma} = \sqrt{\Lambda \frac{m_n n_p}{m_p n_n}} \tag{7.24}$$

which equals $\sqrt{\Lambda}$ if the cell is an inverter. This is an interesting result, because it is independent of the driving gate.

As far as the optimization of conventional CMOS gates is concerned, there is enough evidence, as demonstrated, to claim the theory shown in Figure 7.3, where the term "due to

[2] Γ is equivalent to r defined in Capter 3.

Given a conventional CMOS logic gate producing the output signal OUT, the size of a transistor MOS along the critical path of the gate is optimal,
if
the signal delay in producing OUT due to the transistor MOS as the load
equals
the signal delay in producing OUT through the transistor MOS as the drive.

Figure 7.3: Informal claim regarding optimization of conventional CMOS circuits. The delay may be a rising delay, falling delay, or average delay.

the transistor MOS" includes the transistor MOS itself and those related to this transistor by symmetry, if any, as previously explained. This theory may find applications in developing CAD optimization tools. It also saves time in quick hand-analysis and optimization by just writing the relevant delay terms rather than the whole delay expressions.

7.1.1 Conventional XOR

Figure 7.4 depicts the schematic of a conventional XOR gate. Each input comes from a driving gate, in this case an inverter. The output capacitance C_L includes the loads due to the fanout and interconnects. One of the driving inverters, one of the pull-up (pull-down) chains, and C_L constitute the critical path. Here is a typical statement of the problem.

- Given are the following data on the environment.
  1. The driver's $\Gamma_D = \frac{W_{Dp}}{W_{Dn}} = \frac{20}{10}$.
  2. The load $C_L = 200$ fF and $V_{DD} = 3$ V.
- Find the following.
  1. The optimal transistor sizing of the cell such that the delay over one cycle is minimum.
  2. The minimum delay and its corresponding energy dissipation.
  3. The behaviour of the minimum delay and its energy cost as V_{DD} is scaled.

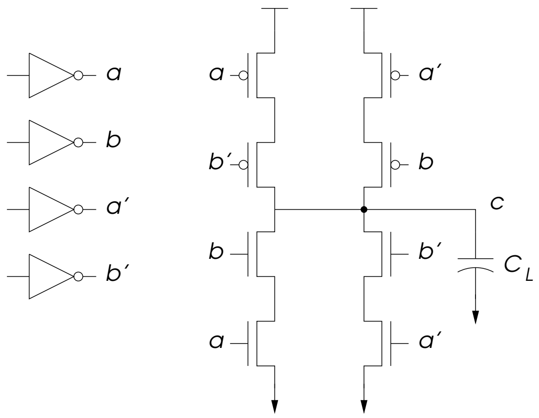


Figure 7.4: Conventional CMOS XOR Implementation.

This section only answers the first part of the problem and leaves the rest for a later section when CMOS implementations of the XOR gate are compared. From the schematic we obtain $m_p = 1$, $m_n = 1$, $n_p = 2$, $n_n = 2$, $q_p = 2$, and $q_n = 2$.

Table 7.1: Optimal transistor sizing for the conventional XOR gate.

Applied Formula Set	$\hat{W}_p$ (μm)	$\hat{W}_n$ (μm)	$\hat{\Gamma}$
Exact: 7.18 and 7.20	41	28	1.46
Approximation: 7.19 and 7.21	44	25	1.76
Rule of Thumb: 7.22 and 7.23	50	31	1.61

Solving the problem calls for applying the optimal transistor sizing formulas of the total delay. There are three choices: the exact set of formulas (7.18 and 7.20), the approximate set of formulas (7.19 and 7.21), and finally the rules of thumb (7.22 and 7.23). (The term exact here implies preciseness with respect to the delay model, which of course, may be different from SPICE and the real world.) The result of applying these pairs of formulas are summarized in Table 7.1. The values obtained from the three formulas are fairly close to each other. Considering the fact that the delay usually has a broad minimum with respect to transistor sizing, all three pairs of transistor sizes are acceptable. Actually, HSPICE simulations show that the difference in the total delay using these pairs of transistor sizes is less that 2%. Recall, however, that we are dealing with a particular situation and are unable to generalize the difference in the results of the three formulas. For instance, if $C_L = 0$, the approximate and the rule-of-thumb formulas both return null answers. Regarding the energy dissipation, HSPICE simulations show that the transistor sizing pair obtained by the rules of thumb results in 10% more energy dissipation than the transistor sizing pairs obtained by the exact and approximate formulas. Figure 7.5 shows the results obtained by the exact

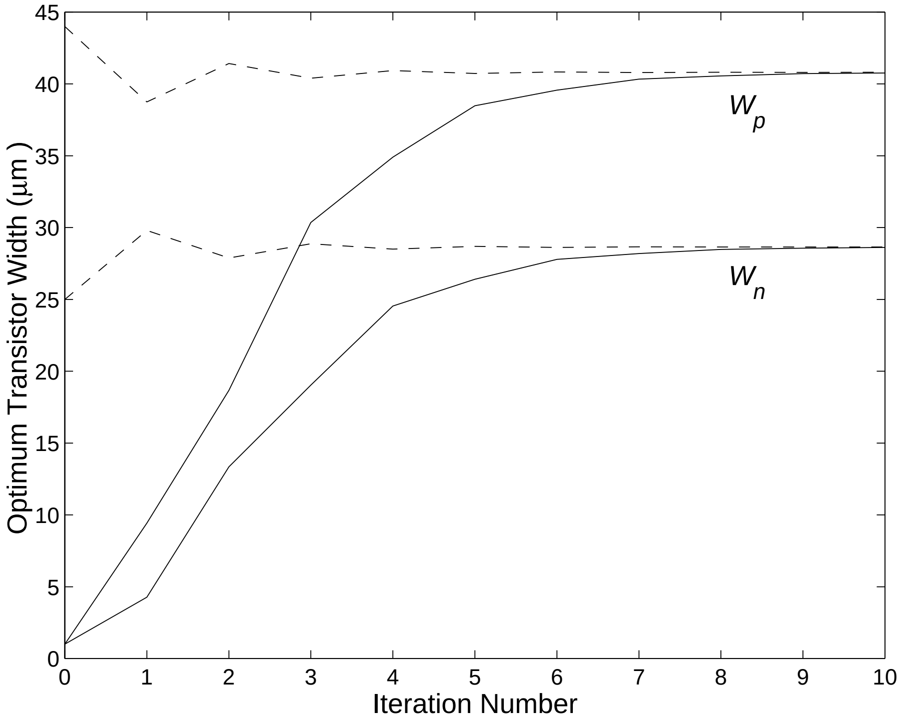


Figure 7.5: Optimal transistor sizing of the conventional XOR gate using the derived formulas. The solid lines are obtained with the initial values of $W_n = W_p = w = 1$ μm. The dashed lines are obtained with the initial values calculated from the approximated formulas.

formulas (7.18 and 7.20) through iteration. If the minimum transistor width $w = 1$ μm is used as the initial value for both W_p and W_n, it takes about 10 iterations to establish the final results. However, the number of required iterations reduces by half, if the initial values are calculated from the approximate formulas (7.19 and 7.21). Note that this is, in fact, a constrained, non-linear optimization problem. Hence, a natural alternative approach in obtaining the optimal sizes is to use a non-linear optimization technique. This has been the standard method in dealing with the problem of gate sizing. In order to confirm our results and evaluate the efficiency of our iterative technique, we also solved the problem using MATLAB's constrained optimization package. The problem was defined for MATLAB as

- Minimize $\hat{D}$ as given by (7.17)
  Subject to $W_p \geq w$ and $W_n \geq w$

As Figure 7.6 illustrates, MATLAB gave exactly the same results after about 90 iterations. That is, about 10 times the number of iterations required by the pair of formulas. Realizing that each iteration in MATLAB's program takes much longer than merely evaluating a pair of formulas, our iterative technique is considerably more efficient in terms of computation cost.

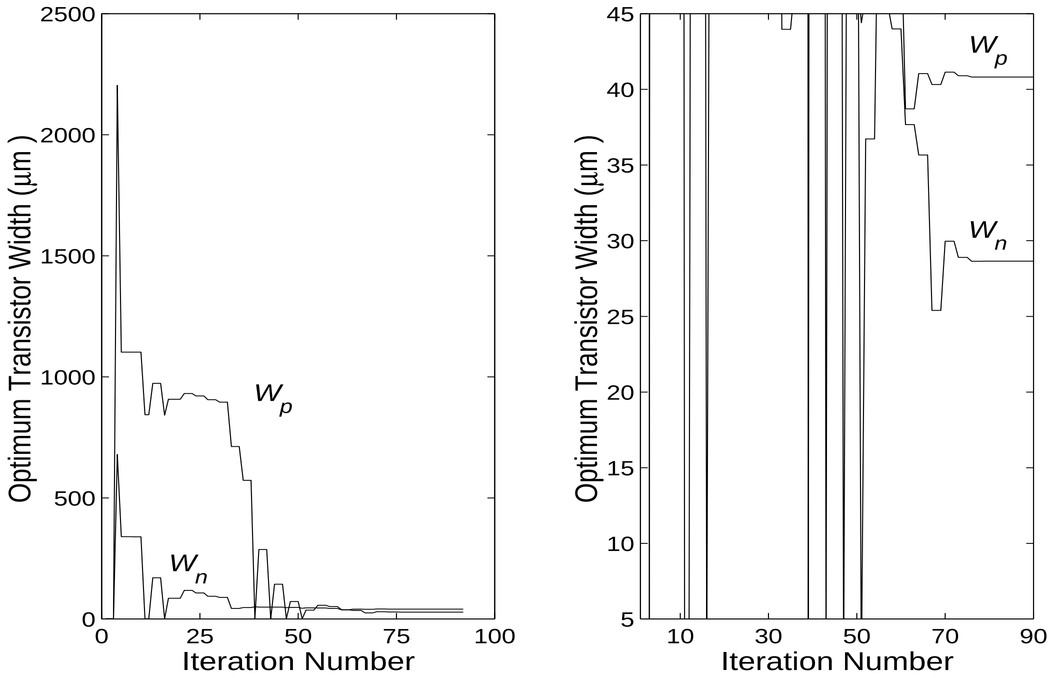


Figure 7.6: Optimal transistor sizing of the conventional XOR gate using the delay models and optimization package of MATLAB.

It is difficult to confirm the optimal sizing results with HSPICE simulations, because exhaustive simulations would be needed. A partial confirmation may be obtained by fixing for example W_p at $\hat{W}_p = 41$ μm and checking whether the optimal W_n found by HSPICE corresponds to $\hat{W}_n$ found by the model. The outcome of this experiment is shown in Figure 7.7, which includes the average, rising, and falling delays as obtained by HSPICE simulations as well as the model. One can make a few observations based on this figure.

1. The delay has a broad minimum, such that spotting the exact minimum point on the graph is hard. Therefore, we may consider an optimal range of W_n between 24 μm and 33 μm for which the delay is very close to its minimum.

2. The delay drops sharply as W_n increases from w to the beginning of this optimal range. Hence, the process of optimal transistor sizing is, indeed, necessary.

3. As W_n increases beyond the optimal range, however, the delay increases rather slowly. Yet, this does not justify using arbitrarily large transistors, because the energy dissipation increases linearly as the transistors become larger. Thus, over-sizing the transistors is a waste of energy. This is another solid reason in support of optimal transistor sizing.

4. The optimum W_n found by the model $\hat{W}_n = 28\ \mu$m not only is well within the optimal range, but also exactly matches the optimum W_n offered by HSPICE simulations. This is confirmed by inspecting the delay values obtained by the simulations.

5. Finally, very good agreement is achieved between the results of the model and those of HSPICE simulations for the average, rising, and falling delays.

7.2 DCVSL CMOS Style

DCVSL was first introduced in [42]. Later, [16] described a design procedure for DCVSL circuits and [17] presented a comparison between conventional and DCVSL full-adder circuits. DCVSL is, especially, very efficient in designing full-adders. For instance, [73] reports the design of fast asynchronous full-adder structures implemented in DCVSL. DCVSL gates are used in asynchronous circuits as completion signal detectors [61]. The operation of DIL gates is similar to DCVSL gates, except that DIL gates have memory, which is made possible by two additional minimum size NMOS transistors. Since these two transistors do not interfere with the switching activities of the gate, the arguments of this section applies to the DIL style as well.

Figure 7.8 illustrates the schematics of a general DCVSL cell, its drive, and its load. A DCVSL gate, produces the output signal on one side and the complement of the output on

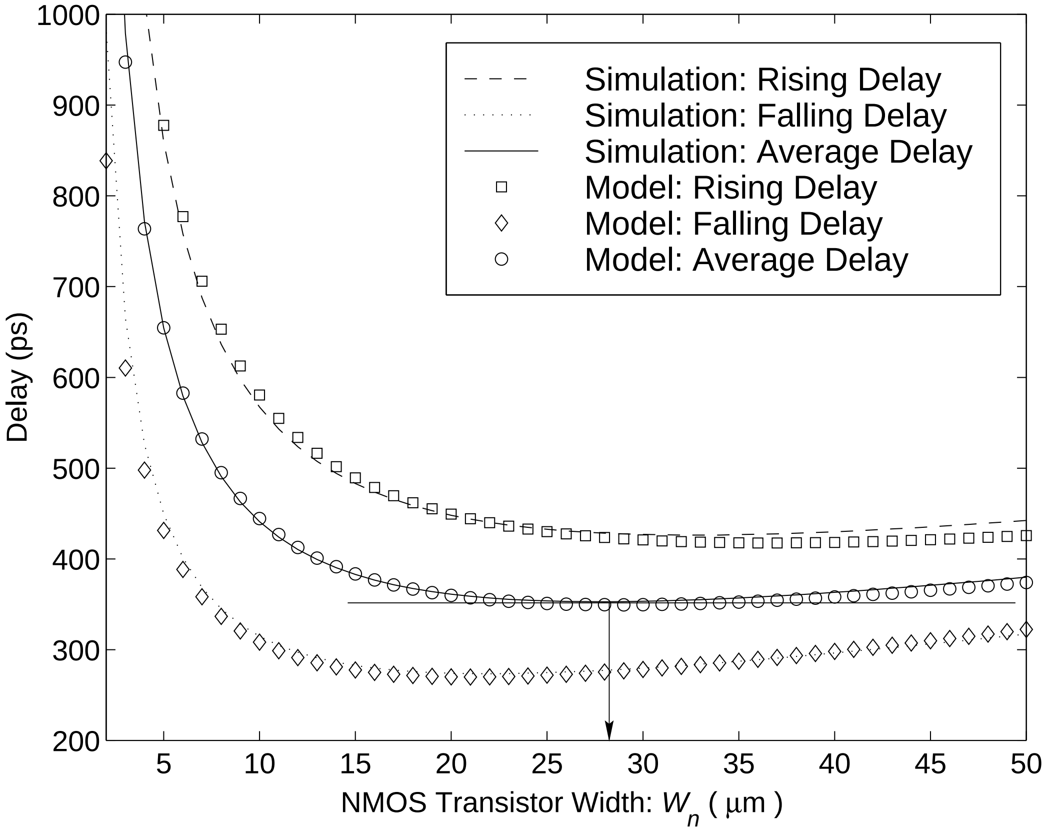


Figure 7.7: Delay estimation for the conventional XOR gate using simulations and the model. The width of PMOS transistor W_p is fixed at $\hat{W}_p = 41$ μm obtained by the model.

the other side. The assumption is that, by symmetry, the loads at both outputs are equal to C_L, which includes the fanout and the interconnects capacitances. The drive, as illustrated in the figure, may be a conventional CMOS gate or another DCVSL gate. The difference in the abstract representation of a conventional gate and a DCVSL gate is that in a DCVSL gate the pull-up section is always characterized by $m_p = 0$, $\acute{Y}_p = n_p = 1$, and $q_p = 1$. Hence, it is more convenient to replace m_n with m, n_n with n, and q_n with q, as shown in the figure. The pull-down network of a DCVSL gate has two parts, often interconnected, corresponding to the two outputs. These two parts are related to each other by symmetry and, thus, optimal transistor sizing of one automatically dictates the transistor sizing in the other one.

In DCVSL gates, the rising output is actually produced by the falling output and, hence,

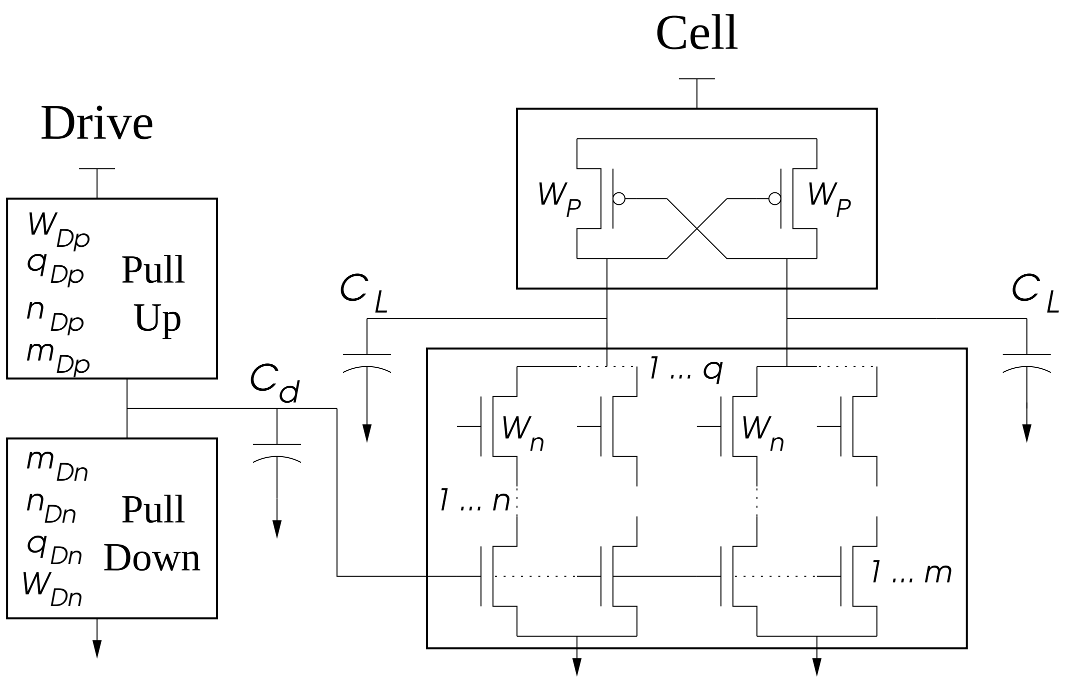


Figure 7.8: A DCVSL CMOS cell between a driving gate and output load.

the worst-case delay is always represented by the rising delay. Considering this fact, the optimization process only concerns the rising delay. The rising and the falling delays between the input of the drive and the output of the cell are expressed by

$$\acute{D} = (1 + \grave{S}_{Tn})\,\acute{D}_D + (1 + \acute{S}_p)\,\grave{D}_C + \acute{D}_C \tag{7.25}$$

$$\grave{D} = (1 + \grave{S}_{Tn})\,\acute{D}_D + \grave{D}_C \tag{7.26}$$

respectively. Where $\grave{D}_C$ is the cell's falling step delay and $\acute{D}_C$ is the cell's rising step delay. Note that $\acute{D}_C$ is not obtained by applying a step rising voltage to the input of the cell, but rather by applying an imaginary, internal step falling voltage to the gate of the PMOS transistor producing the rising output. Also note that in the above equations only the rising step delay of the drive is used, as in this model, the falling delay of the drive does not affect the output delay of the cell. In reality, however, poor falling delay of the drive might prolong the production of the output signal by generating short-circuit current. The rising delay of the cell depends on the falling step delay through $(1+ \acute{S}_p)\,\grave{D}_C$. The falling delay experiences a case of opposing currents and, hence, its slope factor is calculated according to

$$\grave{S}_{Tn} = \frac{\grave{S}_n\, W_n/(\,\grave{\nu}_n\, \grave{Y}_n)}{W_n/(\,\grave{\nu}_n\, \grave{Y}_n)\; - \;W_p/\,\acute{\nu}_p} \tag{7.27}$$

Using this term complicates the derivation of the optimal sizing formulas. Since in functional DCVSL gates the pull-down network must dominate over the pull-up transistor, we approximate $\grave{S}_{Tn}$ by $\grave{S}_n$ in deriving those formulas.

The current for the falling step delay $\grave{D}_C$ is the current through the pull-down chain minus the current through the PMOS transistor. During the falling transition, the PMOS current changes from zero to its saturation value. Therefore, the average PMOS current, which is considered in delay calculations, equals half of its saturation value. Following are the expressions for the step delays involved in the total rising and falling delay equations.

$$\begin{aligned}
\acute{D}_D &= \frac{\acute{\nu}_p}{W_{Dp}}\,\acute{Y}_{Dp}(m\,\acute{g}_n W_n + \,\acute{C}_D) \\
\grave{D}_C &= \frac{\acute{\nu}_p\,\grave{\nu}_n\,\grave{Y}_n}{W_n\,\acute{\nu}_p - 0.5 W_p\,\grave{\nu}_n\,\grave{Y}_n}[(\,\grave{g}_p + \,\grave{d}_p)W_p + q\,\grave{d}_n W_n + \,\grave{C}_L] \\
\acute{D}_C &= \frac{\acute{\nu}_p}{W_p}[(\,\acute{g}_p + \,\acute{d}_p)W_p + q\,\acute{d}_n W_n + \,\acute{C}_L]
\end{aligned}$$

Solving $\frac{\partial\,\acute{D}}{\partial W_p} = \frac{\partial\,\acute{D}}{\partial W_n} = 0$ yields the following set of formulas for the optimal W_p and W_n minimizing the rising delay.

$$\acute{W}_n = \sqrt{A\frac{(1+\,\acute{S}_p)\,[\,\grave{C}_L + W_p(\,\grave{d}_p + \,\grave{g}_p) + 0.5AW_p\,\grave{d}_n]}{(1+\,\grave{S}_n)\;\acute{Y}_{Dp} m\,\acute{g}_n/W_{Dp}\; + \;q\,\acute{d}_n/W_p}} \; + \; \frac{1}{2}AW_p \tag{7.28}$$

$$\frac{1}{\acute{\Gamma}} = \frac{\acute{W}_n}{\acute{W}_p} = \sqrt{(1+\,\acute{S}_p)\left[\frac{1}{2}A^2\frac{\grave{C}_L + qW_n\,\grave{d}_n}{\acute{C}_L + qW_n\,\acute{d}_n} + A\frac{W_n(\,\grave{d}_p + \,\grave{g}_p)}{\acute{C}_L + qW_n\,\acute{d}_n}\right]} \; + \; \frac{1}{2}A \tag{7.29}$$

with

$$A = \frac{\grave{\nu}_n}{\acute{\nu}_p}\,\grave{Y}_n = \frac{\grave{Y}_n}{\Lambda}$$

and $\acute{\Gamma}$ being the optimum width ratio of the PMOS transistor over that of the NMOS transistor for minimizing the rising delay. The above shows that $\acute{\Gamma}$ increases as the output load increases. For DIL implementations, one should add $w_n(\acute{d}_p + \acute{g}_p)$ to $\acute{C}_L$ and $w_n(\grave{d}_p + \grave{g}_p)$ to $\grave{C}_L$ in the above formulas to account for the NMOS transistors of the inverter latch. Here, w_n denotes the smallest allowable width for an NMOS transistor. Usually, the smallest allowable width for a PMOS transistor, w_p, and that for an NMOS transistor, w_n, are equal. Therefore, we can denote both by w.

If the cell's own output capacitances are ignored in favour of the fanout and interconnect capacitances, the above set of formulas can be approximated to

$$\acute{W}_p = \sqrt{\Lambda \frac{2\,\grave{C}_L W_{Dp}}{m\,\acute{Y}_{Dp}\,\grave{Y}_n\,\acute{g}_n\,(1+\grave{S}_n)}} \tag{7.30}$$

$$\acute{\Gamma} = \frac{\acute{W}_p}{\acute{W}_n} = \frac{\Lambda}{\grave{Y}_n}\,\frac{2}{1+\sqrt{2(1+\acute{S}_p)}} \tag{7.31}$$

The rules of thumb are obtained by further approximations as follows.

$$\acute{W}_p = \sqrt{\Lambda \frac{2\,W_{Dp} W_L}{m\,n\,n_{Dp}}} \tag{7.32}$$

$$\acute{\Gamma} = \frac{\acute{W}_p}{\acute{W}_n} = \frac{3}{4\,n}\,\Lambda \tag{7.33}$$

Where $2/[1+\sqrt{2(1+\acute{S}_p)}]$ ranges between 0.75 to 0.78 for the typical values of $\acute{S}_p$ between 0.4 to 0.2, respectively. For a DCVSL inverter, where $n = 1$, $\acute{\Gamma} = 0.75\Lambda$, which is about 2 at $V_{DD} = 3$ V in our 0.5 μm CMOS technology. As n increases, the pull-down chain becomes weaker and, hence, $\acute{\Gamma}$ decreases to maintain the functionality of the DCVSL gate. Note that because of the race problem that exists in the operation of DCVSL gates, the rules found for the optimization of conventional CMOS circuits are not *directly* applicable to DCVSL circuits.

The dynamic energy dissipation in CMOS gates with an output capacitance C is usually calculated from $E = V_{DD} V_{swing} C$, if short-circuit current and the internal capacitances are neglected. This formula can be readily applied to the conventional and PTL gate styles. In DCVSL gates, however, there is an additional component to the energy due to the case of opposing currents, which cannot be neglected. Therefore, we may express the energy consumption in a DCVSL gate as the sum of two energies, a useful component used to charge and discharge the output node and a wasted component.

$$E = E_{\text{useful}} + E_{\text{wasted}}$$

The useful energy component per cycle is calculated from

$$E_{\text{useful}} = 2\ V_{DD}^2[C_D + C_L + W_n(m_n g_n + q_n d_n) + W_p(g_p + d_p)]$$

The wasted energy component per cycle is governed by

$$\begin{aligned}
E_{\text{wasted}} &= 2 \times \text{supply voltage} \times \text{wasted current} \times \text{duration of wasting energy} \\
&= 2\ V_{DD}\ \frac{I_p}{2}\ \acute{D}_{CC} \qquad (7.34) \\
&= 2\ V_{DD}\ \ (\frac{1}{2}\ \frac{W_p}{2\ \acute{\nu}_p}\ V_{DD})\ \ [\ \grave{D}_c(1+\ \acute{S}_p) +\ \acute{D}_c] \\
&= \frac{1}{2}\ V_{DD}^2\ \frac{W_p}{\acute{\nu}_p}\ [\ \grave{D}_c(1+\ \acute{S}_p) +\ \acute{D}_c])
\end{aligned}$$

Where the wasted current is the current supplied by the PMOS transistor while it is fighting the output switching. As for the delay model, this current is estimated as half of the PMOS transistor saturation current I_p. The duration of wasting energy is, in fact, the duration of the fight, which is estimated as the time when the gate is activated until the rising output is produced $\acute{D}_{CC}$.

We should mention that the theory of delay optimization, as mentioned for the conventional logic style, does not precisely apply to DCVSL style. The reason is the presence of overlapping currents in DCVSL gates.

7.2.1 DCVSL XOR

This section applies the DCVSL delay and energy models to the XOR gate whose schematic is depicted in Figure 7.9.

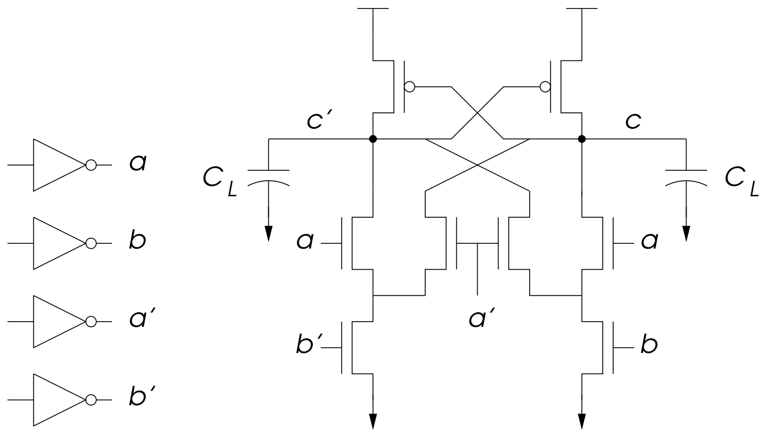


Figure 7.9: Schematic of DCVSL XOR gate.

Table 7.2: Optimal transistor sizing for the DCVSL XOR gate.

Applied Formula Set	$\acute{W}_p$ (μm)	$\acute{W}_n$ (μm)	$\acute{\Gamma}$
Exact: 7.28 and 7.29	27.5	35	0.78
Approximation: 7.30 and 7.31	41	35	1.17
Rule of Thumb: 7.32 and 7.33	41	42	0.97

The specifications for the drives and the load are similar to those defined for the conventional XOR gate, i.e. $W_{Dp}/W_{Dn} = 20/10$. However, the total 200 fF output capacitive load is equally divided between the two outputs of the DCVSL XOR gate, i.e. $C_L = 100$ fF on each side of the gate. The optimization problem we want to address is also similar to the problem statement made for the conventional XOR gate. In that case it was required to optimize the delay over one cycle, i.e. the total or average delay. For a DCVSL gate, the delay over one cycle is twice the rising delay. Hence, the optimal transistor sizing should target the rising delay. Since input a is driving two transistors, whereas input b is driving only one, the critical path is more likely to involve a rather than b. Therefore, the schematic identifies $m = 2$, $n = 2$, and $q = 2$. Table 7.2 summarizes the results of the optimization using the three formula sets: the exact set of formulas (7.28 and 7.29), the approximate set of formulas (7.30 and 7.31), and finally the rules of thumb (7.32 and 7.33). HSPICE simulations show that using the transistor sizing listed in Table 7.2, the rising delay is 381 ps for the exact formula set, 420 ps for the approximated formula set, and 401 ps for the rules of thumb. The energy dissipations for the three sizing sets are 12.3 pJ, 15.8 pJ, and 16.0 pJ, in the same order. Therefore, there is a maximum difference of about 10% among the minimum delays achieved by the three formula sets, with the lowest delay produced by the exact formulas. This percentage of difference is much more than the 2% for the conventional XOR gate. We leave the reason for a later paragraph.

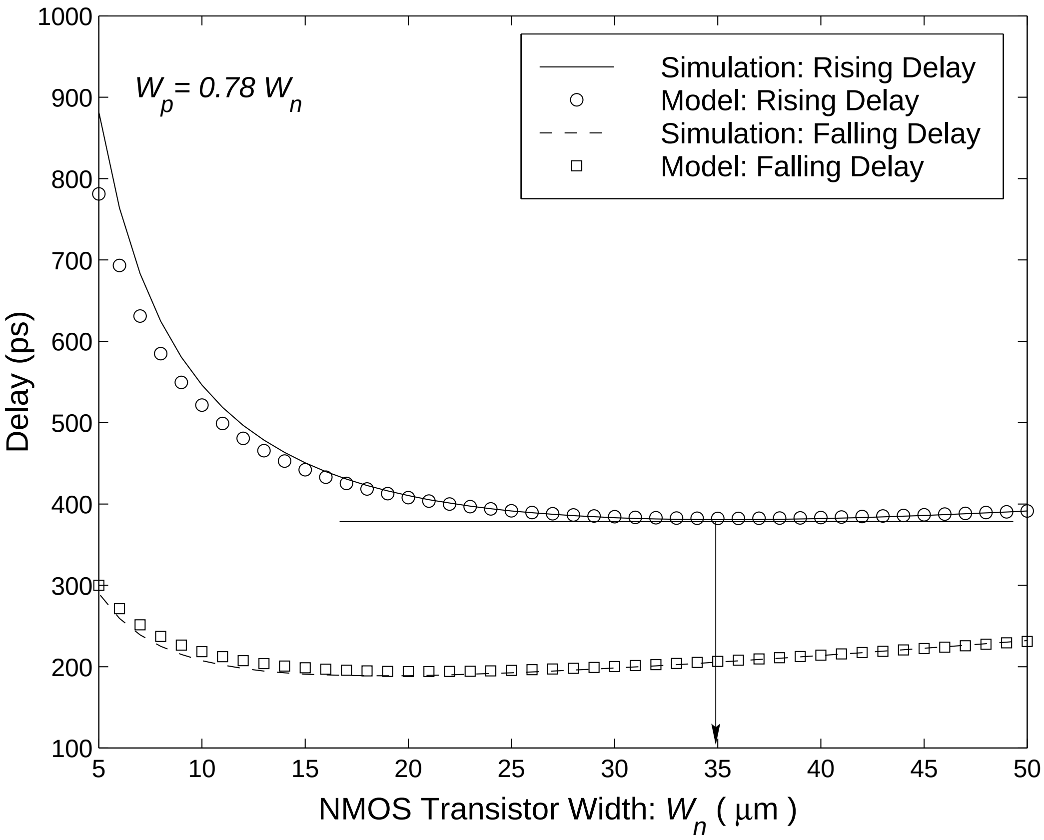


Figure 7.10: Delay estimation and optimization for the DCVSL XOR gate using simulations and the model. The ratio W_p/W_n is kept constant to find the optimum W_n.

In order to confirm the results of the model with HSPICE simulations, we report two sets of simulations. In the first set of simulations, as shown in Figure 7.10, W_p/Wn is kept constant at its optimal value 0.78 found by the model while W_n is changed from 5 μm to 50 μm. These simulations confirm the value of 35 μm for the optimal W_n obtained by the model. There is also a good agreement between the rising and falling delays predicted by the model and simulations. The rising delay has a broad minimum covering a wide range of

W_n values. The value of W_n at which the falling delay is minimum is far less than the value required to minimize the rising delay.

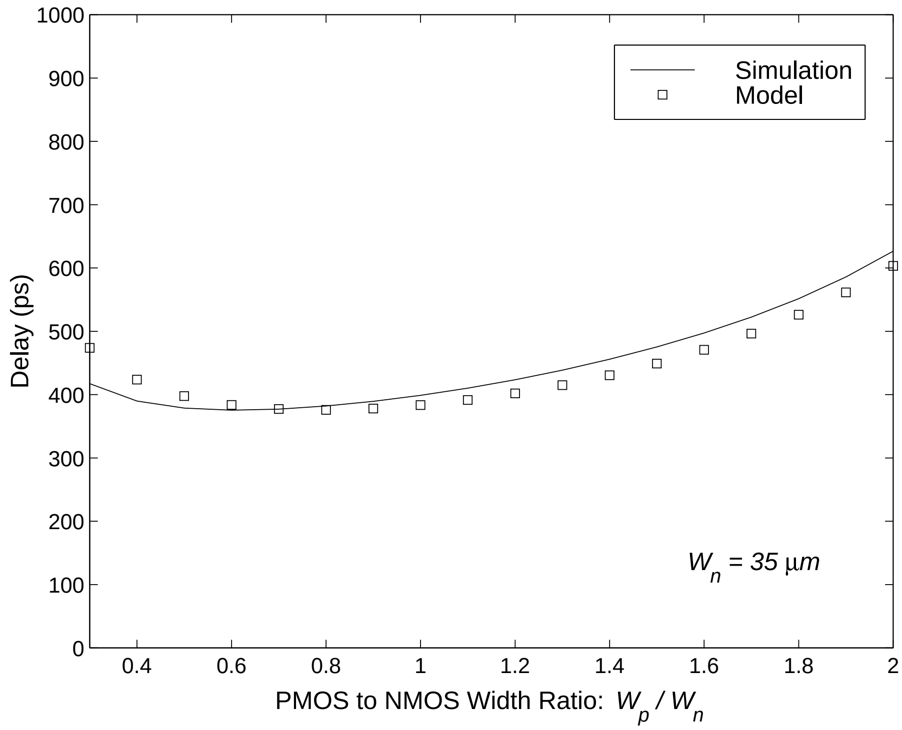


Figure 7.11: Delay estimation and optimization for the conventional XOR gate using simulations and the model. W_n is kept constant to find the optimum W_p/W_n.

In the second set of HSPICE simulations, W_n is kept constant at its optimal value while changing W_p/W_n. As illustrated in Figure 7.11, simulations approve the optimum W_p/W_n determined by the model. It is noticeable that the delay does not exhibit such a broad minimum with respect to W_p/W_n as it does with respect to W_n. In fact, the delay of a DCVSL gate is more sensitive towards W_p/W_n when compared to a conventional CMOS gate. For instance, the average delay of a CMOS inverter has a broad minimum between $W_p/W_n = 1.3$ to 3, approximately. Whereas this range for the delay of the DCVSL XOR is only between $W_p/W_n = 0.5$ to 0.9, i.e. four times smaller. The higher significance of W_p/W_n in the case of a DCVSL gate is that this ratio, on the one hand, determines the fate of the fighting during the output switching and, on the other hand, determines the delay. If W_p/W_n is made too large, the battle is lost, and if it is made too small the delay is poor. Looking back at Table 7.2, we realize that the values of $\acute{W}_n$ predicted by the three formula sets are close to each other, but the same is not true for the values of $\acute{\Gamma}$. This justifies the reason behind the differences in the resultant minimum delays from the three sets of formulas.

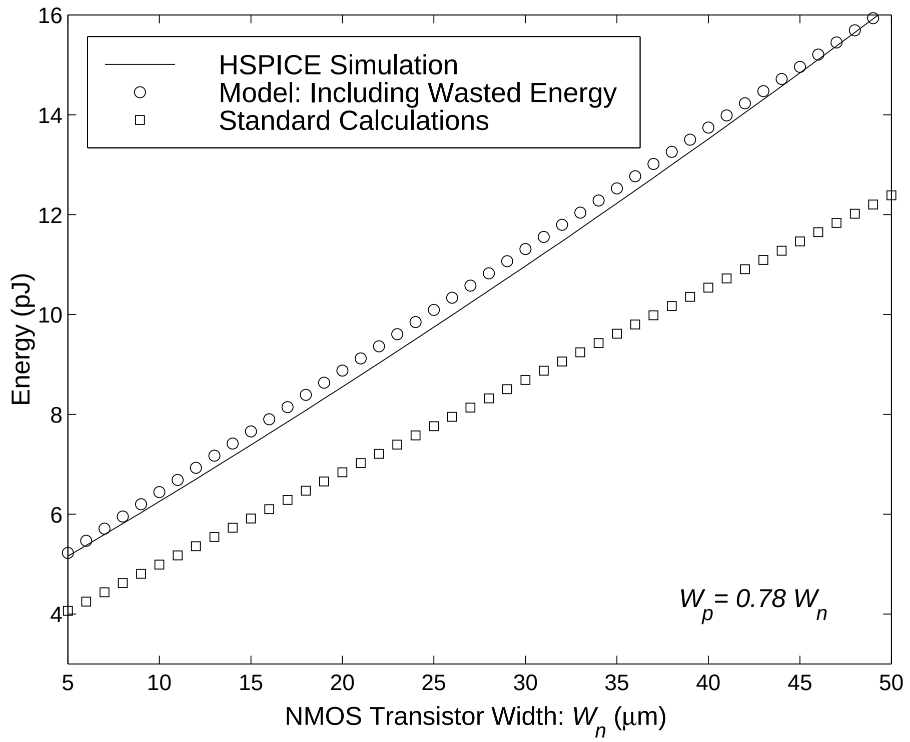


Figure 7.12: Energy estimation for the conventional XOR gate using simulations and the model.

Finally, Figure 7.12 shows the the energy dissipation of the DCVSL XOR as a function of W_n. The figure compares the results of HSPICE simulations with hand calculations. The circles represent calculations including the wasted energy based on the model of expression (7.35). The squares represent calculations neglecting the wasted energy. The former are in good agreement with simulations. In this case, the energy wasted on fighting constitutes around 25% of the total energy consumption. Note that the total energy includes the energy dissipated by the output load and the diffusion capacitances of the drive besides the the energy dissipated by the cell. Hence, the wasted energy is a major part of the DCVSL cell's energy consumption, which cannot be neglected.

7.3 PTL CMOS Styles

By pass-transistor logic (PTL), we refer to the general category of CMOS circuits in which signals, not necessarily V_{DD} or ground, are passed from their inputs to their outputs through a chain of MOS transistors. There are different styles of PTL circuits. Complementary Pass-transistor Logic (CPL), introduced in [105], is the most widely used PTL style. Design issues regarding CPL circuits are discussed in [69]. Double Pass-transistor Logic (DPL) is another PTL style which, compared to CPL, uses double the number of transistors to achieve higher performance [94]. According to our definition, the DILN C-element is a PTL circuit. PTL usually uses fewer transistors than DCVSL and conventional CMOS style for implementing the same function.

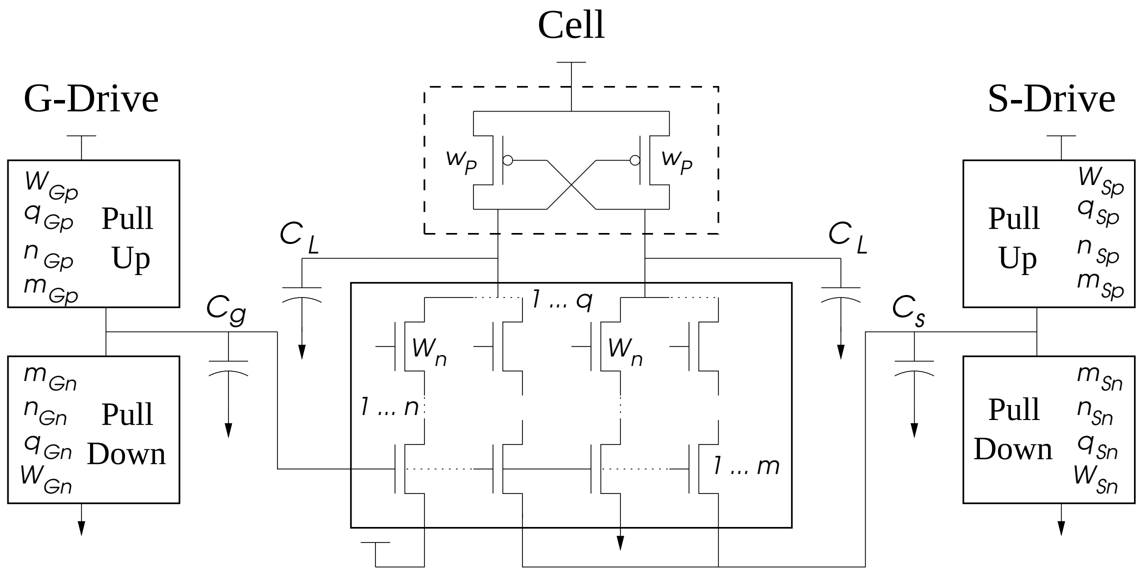


Figure 7.13: A PTL CMOS cell between driving gates and output load.

Figure 7.13 illustrates the schematics of a PTL cell, its drives, and its load. A PMOS latch is usually used at the output of PTL circuits to reduce short-circuit energy dissipation. The latch transistors have the minimum PMOS width allowed by the technology w_p. Because of the small size of the latch and the fact that both sides of the latch switch almost simultaneously, the latch does not interfere in the operation of the circuit. This is in contrast to the case of the PMOS latch in a DCVSL circuit. The inputs to the transistors' sources in CPL are only the variable signals and in DPL are combinations of the variable signals, ground, and V_{DD}. In general, the drive for the gates of the transistors and the drive for the sources of the transistors may be different, as shown in the figure. To distinguish between the two, we refer to the former type of driving circuits as G-drives and refer to the latter type of diving circuits as S-drives. If a drive is connected to both the gates and sources of the transistors, for convenience, we still categorize it as an S-drive. The varieties in the structure of PTL circuits create a number of situations which need to be considered for the delay calculation and optimization. In PTL circuits the rising delay is the worst-case delay and the subject of optimization, because NMOS transistors are slower in transferring a logic 1 than transferring a logic 0. The critical path in PTL circuits, however, may extend from a G-drive to the output or from an S-drive to the output. The critical path may also involve both a G-drive and an S-drive, as the upcoming discussions clarify. Therefore, there are three different cases, which are addressed in the following sub-sections.

7.3.1 Critical Path Involving G-Drive

In this case, the NMOS chain along the critical path producing the rising output is connected to V_{DD} and, by symmetry, the NMOS chain producing the falling output is connected to ground. Although this scenario is rare in PTL circuits, it should be covered as a possibility in the modeling of PTL circuits. This is also the least complicated case among the three. An example of this scenario is the DILN C-element. The following arguments are with reference to Figure 7.13, if the S-drive is ignored.

The rising and falling delays as functions of the step delays are given by

$$\acute{D} = (1 + \acute{S}_n)\, \acute{D}_G + \acute{D}_C \tag{7.35}$$

$$\grave{D} = (1 + \grave{S}_n)\, \acute{D}_G + \grave{D}_C \tag{7.36}$$

where the step delays are calculated from these expressions.

$$\begin{aligned}
\acute{D}_G &= \frac{\acute{\nu}_p}{W_{Gp}}\, \acute{Y}_{Gp}(m_n W_n\, \grave{g}_n + \acute{C}_G) \\
\acute{D}_C &= \frac{\acute{\nu}_n}{W_n}\, \acute{Y}_n(\, \acute{g}_p w_p + \acute{d}_p w_p + q_n W_n\, \acute{d}_n + \acute{C}_L) \\
\grave{D}_C &= \frac{\grave{\nu}_n}{W_n}\, \grave{Y}_n(\, \grave{g}_p w_p + \grave{d}_p w_p + q_n W_n\, \grave{d}_n + \grave{C}_L)
\end{aligned}$$

Where $\acute{C}_G$ includes all capacitances at the output of the G-drive , except the capacitances related to the cell. The above results in the following expression for the optimum W_n for the rising delay.

$$\acute{W}_n = \sqrt{\frac{\acute{\nu}_n\ \acute{Y}_n}{\acute{\nu}_p\ \acute{Y}_{Gp}} \frac{W_{Gp}[\ \acute{C}_L + (\ \acute{g}_p +\ \acute{d}_p)w_p]}{m\ \acute{g}_n(1+\ \acute{S}_n)}} \approx \sqrt{\frac{\acute{\nu}_n\ \acute{Y}_n}{\acute{\nu}_p\ \acute{Y}_{Gp}} \frac{\acute{C}_L W_{Gp}}{m\ \acute{g}_n(1+\ \acute{S}_n)}} \tag{7.37}$$

Where the approximation is almost always valid, because of the small size of the PMOS latch. As before, the rule of thumb is obtained by replacing C_L with a total loading gate-oxide capacitance $g\ W_{Lt}$.

$$\acute{W}_n = \sqrt{\frac{\acute{\nu}_n\ n}{\acute{\nu}_p\ m\ n_{Gp}}\ W_{Gp} W_{Lt}} \tag{7.38}$$

Inspection of (7.37) reveals that the theory of delay optimization stated for the conventional style, also applies to a PTL gate whose delay is governed by a G-drive. It doesn't apply, however, to a PTL gate whose delay is governed by an S-drive or by both drives. In fact, the theory applies to any CMOS topology that does not include overlapping and opposing currents.

7.3.2 Critical Path Involving S-Drive

This seems to be the dominant case in PTL circuits. MOST CPL gates, like the OR/NOR gate shown in Figure 7.14, resemble this structure. In this scenario, as depicted in Figure 7.14, the NMOS chain along the critical path is connected to the output of another circuit, an S-drive. The S-drive also controls the gates of m transistors of width W_n in the cell, where m is greater than or equal to zero. If $m = 0$, then the assumption is that the transistors along the chain have already been turned on by a G-drive. There are q_o diffusions of width W_n at the output and q_i such diffusions at the input, as shown in the figure.

The rising falling delay expression is as follows.

$$\acute{D} = \left(\frac{\acute{\nu}_p}{W_{Sp}}\ \acute{Y}_{Sp} + \frac{\acute{\nu}_n}{W_n}\ \acute{Y}_n\right)[\ \acute{C}_L + q_o W_n\ \acute{d}_n + (\ \acute{g}_p +\ \acute{d}_p)w_p] + (1+\ \acute{S}_n)\ \acute{D}_S \tag{7.39}$$

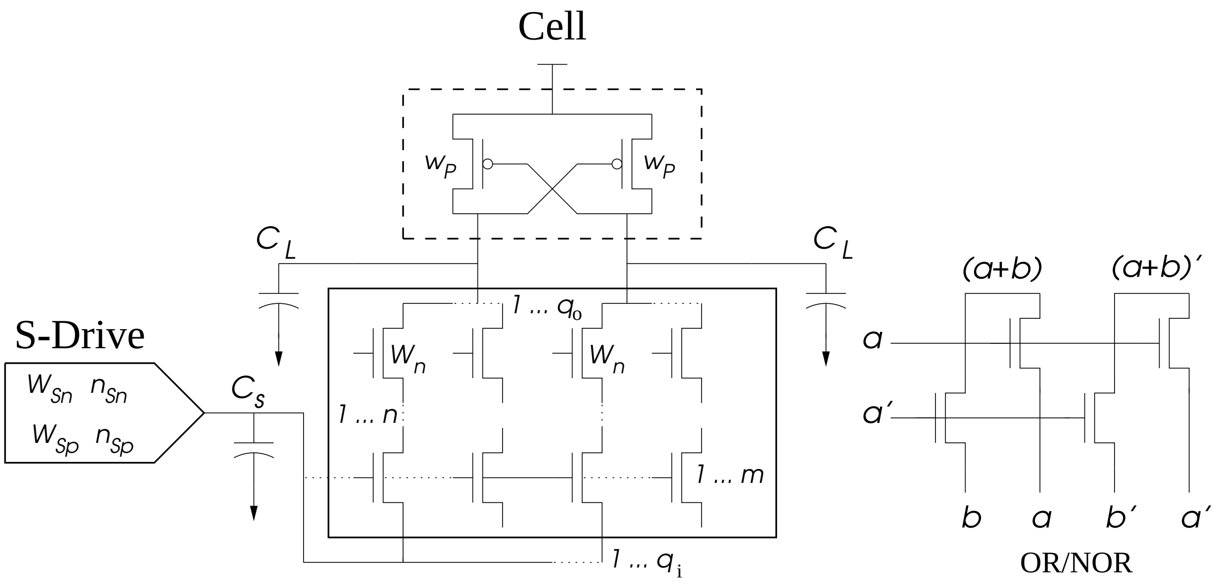


Figure 7.14: A PTL CMOS cell connected to an S-drive along the critical path. A CPL OR/NOR gate is also shown as an example.

Where the first term represents the step rising delay of the cell. In this term, the parentheses enclose the total resistance of the critical path, which is the sum of the S-drive's pull-up resistance and the resistance of the NMOS chain. A similar expression governs the falling delay. The S-drive step rising delay is given by

$$\acute{D}_S = \frac{\acute{\nu}_p}{W_{Sp}} \acute{Y}_{Sp}(\acute{C}_S + q_i W_n \acute{d}_n + m W_n \acute{g}_n) \tag{7.40}$$

where $\acute{C}_S$ includes all capacitances at the output of the S-drive, except the capacitances related to the cell. The optimal transistor sizing for the rising delay can be calculated using the following formula

$$\begin{aligned} \acute{W}_n &= \sqrt{\frac{\acute{\nu}_n \acute{Y}_n}{\acute{\nu}_p \acute{Y}_{Sp}} \frac{W_{Sp}[\acute{C}_L + (\acute{g}_p + \acute{d}_p) w_p]}{(m \acute{g}_n + q_i \acute{d}_n)(1 + \acute{S}_n) + q_o \acute{d}_n}} \\ &\approx \sqrt{\frac{\acute{\nu}_n \acute{Y}_n}{\acute{\nu}_p \acute{Y}_{Sp}} \frac{\acute{C}_L W_{Sp}}{(m \acute{g}_n + q_i \acute{d}_n)(1 + \acute{S}_n)}} \end{aligned} \tag{7.41}$$

which leads to this rule of thumb.

$$\acute{W}_n = \sqrt{\frac{\acute{\nu}_n \, n}{\acute{\nu}_p \, n_{Sp} \, (m + q_i)} W_{Sp} W_{Lt}} \tag{7.42}$$

7.3.3 Critical Path Involving Both G-Drive and S-Drive

In this case, as illustrated in Figure 7.15, the assumption is that the input signal, which is the output of an S-drive, has already arrived at the source of the NMOS transistors chain. Once the chain is turned on by the G-drive, the output of the S-drive is transferred to the output of the cell. Therefore, both drives influence the delay. The CPL XOR/XNOR gate shown in Figure 7.15 is an example of a circuit subject to the scenario discussed here.

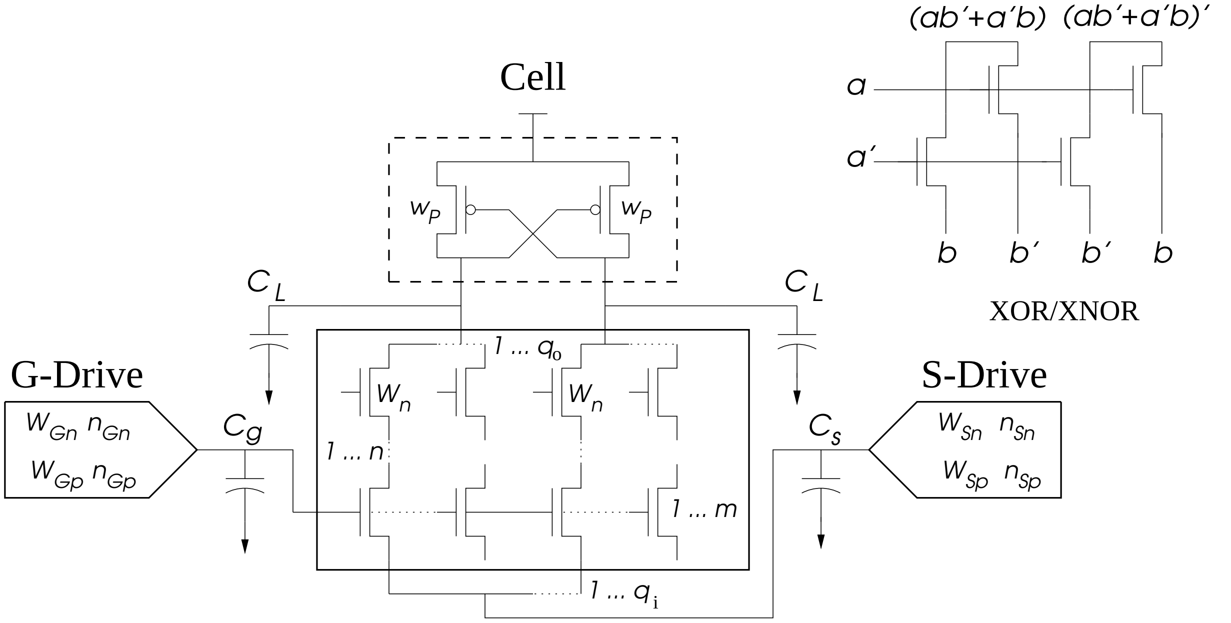


Figure 7.15: A PTL CMOS cell connected to an S-drive along the critical path controlled by a G-drive. A CPL XOR/XNOR gate is also shown as an example.

Because of the complications involved in the delay modeling for this case, we need to use some approximations, which we shall justify using an RC network model. Consider the RC network of Figure 7.16. Initially, C_1 is fully charged, but C_2 has no charges stored. We are interested in estimating the delay from the time the switch is closed to the time when the potential across C_2 reaches half of the value of the supply voltage, i.e. until C_2 is halfly charged. This network with its initial condition can be easily solved by the standard methods of circuit theory. The result, however, is complicated and involves many terms. Using Elmore's theory [35], if C_1 had no initial charges, the delay would be

$$\acute{D}_{RC} = 0.69\ [C_2(R_1 + R_2) + C_1 R_1]$$

Since C_1 is already charged, the term $C_1 R_1$ should be spared. However, by doing so, the influence of C_1 on the delay is totally ignored, which is a good approximation if C_1 is much smaller than C_2, but may result in large errors otherwise. On the other hand, if C_1 is much larger than C_2, the delay does not depend on R_1, because C_1 supplies C_2 with all the necessary charges. Hence, the following approximation of the delay is valid, at least, in these two extreme cases.

$$\acute{D}_{RC} = 0.69\ C_2(R_2 + R_1 \frac{C_2}{2C_1 + C_2})$$

Where the factor 2 appears in the denominator to account for the fact that C_1 is fully charged, as opposed to the halfly charged state of C_2 we are seeking for the delay estimation. Figure 7.16 shows how this approximation compares to the simulation results for a range of C_2/C_1. As expected, the approximation works well when C_2 is much larger or much smaller than C_1. In between these two cases, the accuracy of the approximation depends on the relative values of R_1 and R_2. The error seems to be larger when $R_2 > R1$, but this is very unlikely in our application.

Turning back to the PTL circuit of Figure 7.15, C_1 resembles the total capacitance at the input of the cell C_I, i.e. output of the S-drive, and C_2 resembles the total output capacitance of the cell C_O.

$$\acute{C}_I = \acute{C}_B + q_i W_n\ \acute{d}_n \tag{7.43}$$

$$\acute{C}_O = \acute{C}_L + q_o W_n\ \acute{d}_n + (\ \acute{g}_p +\ \acute{d}_p) w_p \tag{7.44}$$

Using the RC network approximation yields the following expressions for the rising and falling delays.

$$\acute{D} = (1 +\ \acute{S}_n)\ \acute{D}_G + \frac{\acute{\nu}_n}{W_n}\ \acute{Y}_n\ \acute{C}_O + \frac{\acute{\nu}_p}{W_{Sp}}\ \acute{Y}_{Sp} \left(\ \acute{C}_O \frac{\acute{C}_O}{\acute{C}_O + 2\ \acute{C}_I}\right) \tag{7.45}$$

$$\grave{D} = (1 +\ \grave{S}_n)\ \acute{D}_G + \frac{\grave{\nu}_n}{W_n}\ \grave{Y}_n\ \grave{C}_O + \frac{\grave{\nu}_p}{W_{Sn}}\ \grave{Y}_{Sp} \left(\ \grave{C}_O \frac{\grave{C}_O}{\grave{C}_O + 2\ \grave{C}_I}\right) \tag{7.46}$$

Assuming C_O is much larger than C_1 results in

$$\acute{W}_n = \sqrt{\frac{\acute{\nu}_n}{\acute{\nu}_p}\ \acute{Y}_n \frac{W_{Gp} W_{Sp} [\ \acute{C}_L + (\ \acute{g}_p +\ \acute{d}_p) w_p]}{(1 +\ \acute{S}_n) W_{Sp}\ \acute{Y}_{Gp}\ m\ \acute{g}_n + W_{Gp}\ \acute{Y}_{Sp} q_o\ \acute{d}_n}} \tag{7.47}$$

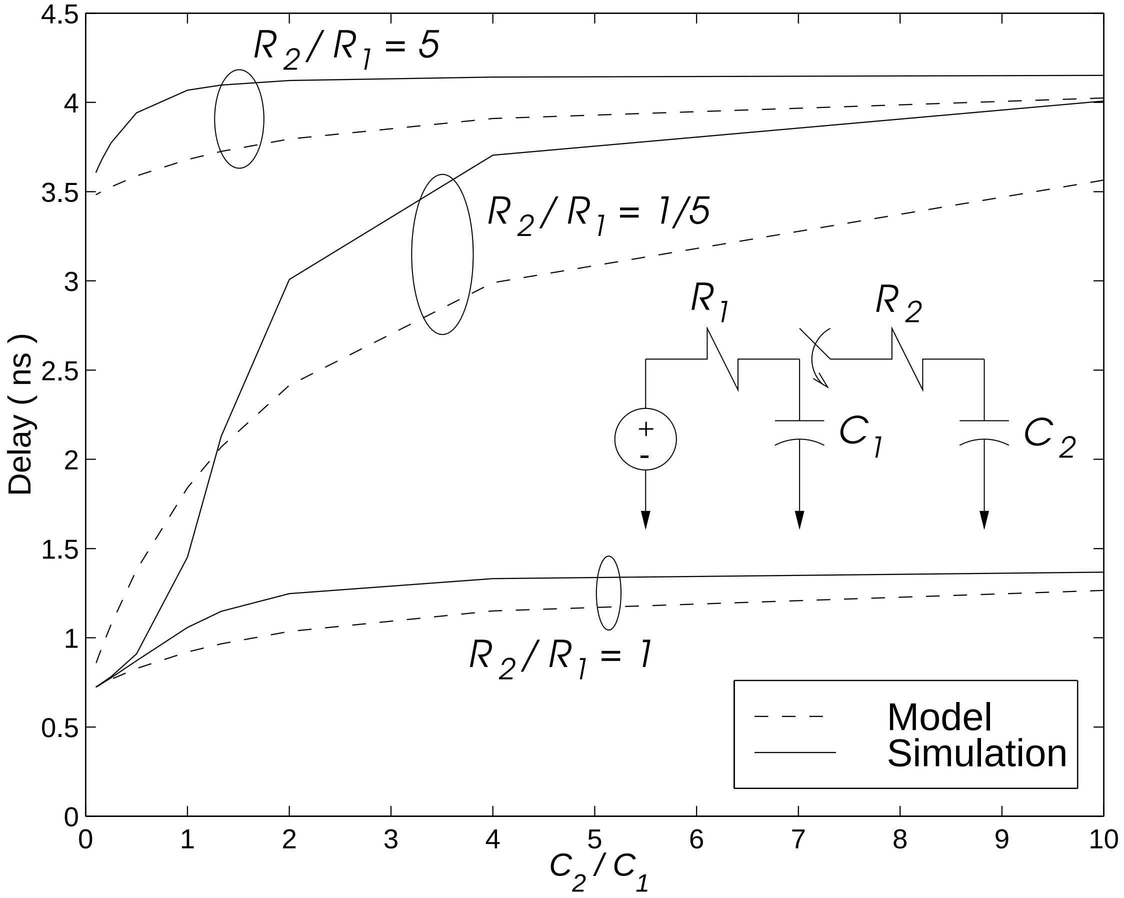


Figure 7.16: An RC network for modeling the delay in PTL circuits involving both a G-drive and an S-drive

and, consequently, the following rule of thumb for optimal transistor sizing.

$$\acute{W}_n = \sqrt{\frac{\acute{\nu}_n}{\acute{\nu}_p} n \frac{W_{Gp}W_{Sp}W_{Lt}}{W_{Sp}n_{Gp}\ m + W_{Gp}n_{Sp}q_o}} \tag{7.48}$$

If C_I is much large than C_O, then the S-drive may be ignored, as if the source of the NMOS chain is connected to V_{DD}. That is, the scenario simplifies to the first case discussed in Section 7.3.1, and the optimal sizing formula becomes

$$\acute{W}_n = \sqrt{\frac{\acute{\nu}_n}{\acute{\nu}_p}\ \acute{Y}_n \frac{W_{Gp}[\ \acute{C}_L + (\ \acute{g}_p + \ \acute{d}_p)w_p]}{(1+\ \acute{S}_n)\ \acute{Y}_{Gp}\ m\ \acute{g}_n}} \tag{7.49}$$

7.3.4 CPL XOR

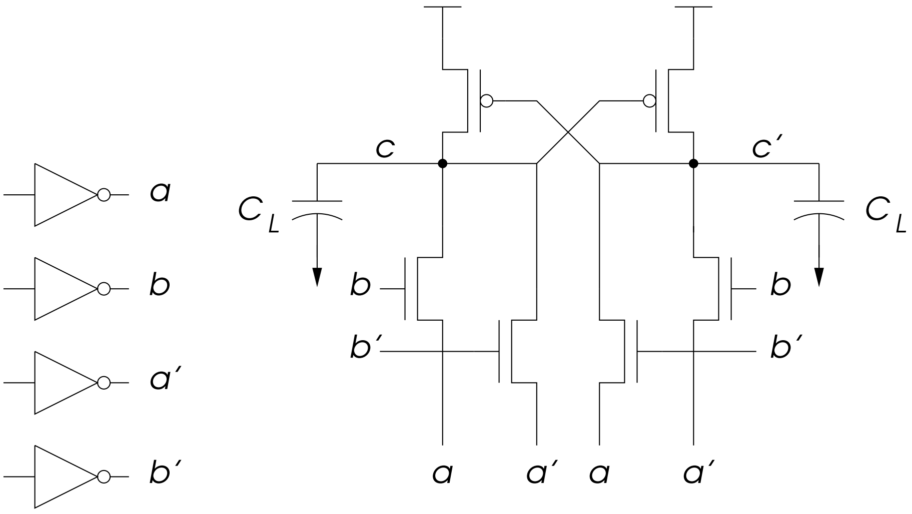


Figure 7.17: Schematic of CPL XOR gate.

The CPL implementation of the XOR gate is depicted in Figure 7.17. As mentioned before, the worst-case delay is more likely to involve both the driving gate for b and the driving gate for a. The sizes of the driving inverters and the output load are similar to those previously defined for the conventional and DCVSL XOR gates, i.e. $W_{Dp}/W_{Dn} = 20/10$ and $C_L = 100$ fF on each side. The schematic reveals that $m = 2$, $n = 1$, and $q_o = 2$. Applying (7.47) results in $\acute{W}_n = 14.70$ μm, whereas (7.48) yields $\acute{W}_n = 16.40$ μm. Both results are acceptable and within the range of W_n producing a minimum delay, as confirmed by HSPICE simulation results shown in Figure 7.18. Figure 7.18 also demonstrates the good agreement between the model and the simulations in predicting the rising and falling delays of the CPL XOR gate. Note that the CPL XOR gate involving both drives, as examined here, represents the most complicated scenario among the three cases presented for the delay estimation in a PTL gate. As explained, we have incorporated some additional approximations in the delay modeling for this particular case and, naturally, expect some degree of disagreement with HSPICE results.

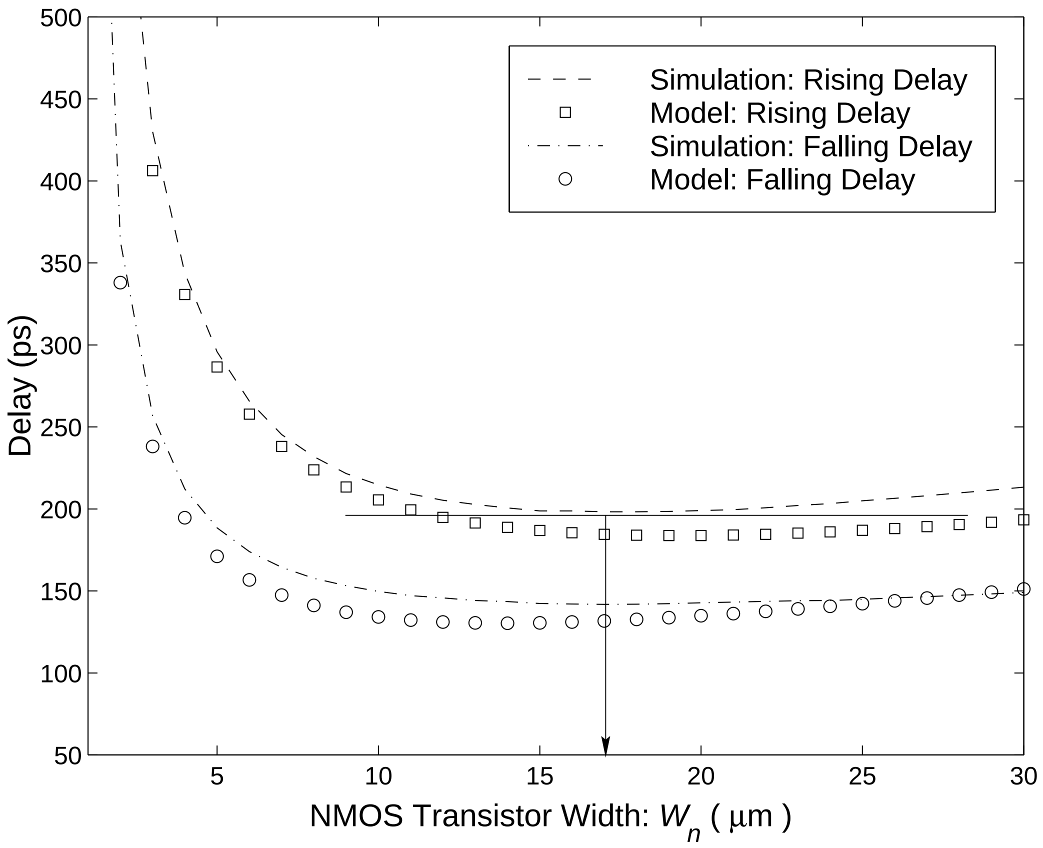


Figure 7.18: Delay estimation for the conventional XOR gate using simulations and the model.

7.4 Comparing CMOS Implementations of XOR

Being able to optimize the performance of the gates implemented in various logic styles, we are now ready to perform a fair comparison between the CMOS implementations of the XOR gate. When the gates are implemented in different logic styles, there is a high possibility that their performances are not equally affected by the supply voltage scaling. Hence, for the XOR gate, we are adding another dimension to our comparison technique developed for the case of the C-element. That is, the performance and the energy of the gates are evaluated at different supply voltages. At each value of the supply voltage, the gates are sized to deliver the minimum worst-case delay over one switching cycle. The optimization is performed using the formulas derived in this chapter.

Figure 7.19 shows the behaviour of the minimum delays and their corresponding energies for the conventional, DCVSL, and CPL XOR gates at different values of the supply voltage. The results are obtained by the model and HSPICE simulations. Very good agreement is observed between the two for the delays and energies of all three styles. The outcome of the comparison is very much in favour of the CPL implementation in terms of delay and energy. However, it is noticeable that the performance of the CPL XOR gate degrades faster than the conventional and DCVSL XOR gates as the supply voltage is scaled down. For instance, at $V_{DD} = 1.5$ V, the conventional and DCVSL XOR gates both outperform the CPL XOR gate. In this study, we have not considered the behaviour of the gates in relation to the difference in the arrival times of the complementary input signals and the possible effects of the charge sharing phenomenon.

7.5 Concluding Remarks

In this chapter, we have derived macromodels for the main CMOS logic styles by applying the delay model developed in the previous chapter. The CMOS logic styles studied include conventional, DCVSL, and PTL. Based on the macro-models, we have derived exact and approximate closed-form formulas for the optimal transistor sizing of the various CMOS logic styles in terms of the driving gates and the loads. In addition, further approximations were applied to come up with a set of simple formulas to be used as rules of thumb for the optimization of each CMOS logic style. The macro-models and the optimal transistor sizing formulas were compared with HSPICE simulations for different CMOS implementations of the XOR gate. Finally, the optimally-sized XOR gates were compared in terms of the delay and energy at various values of the supply voltage.

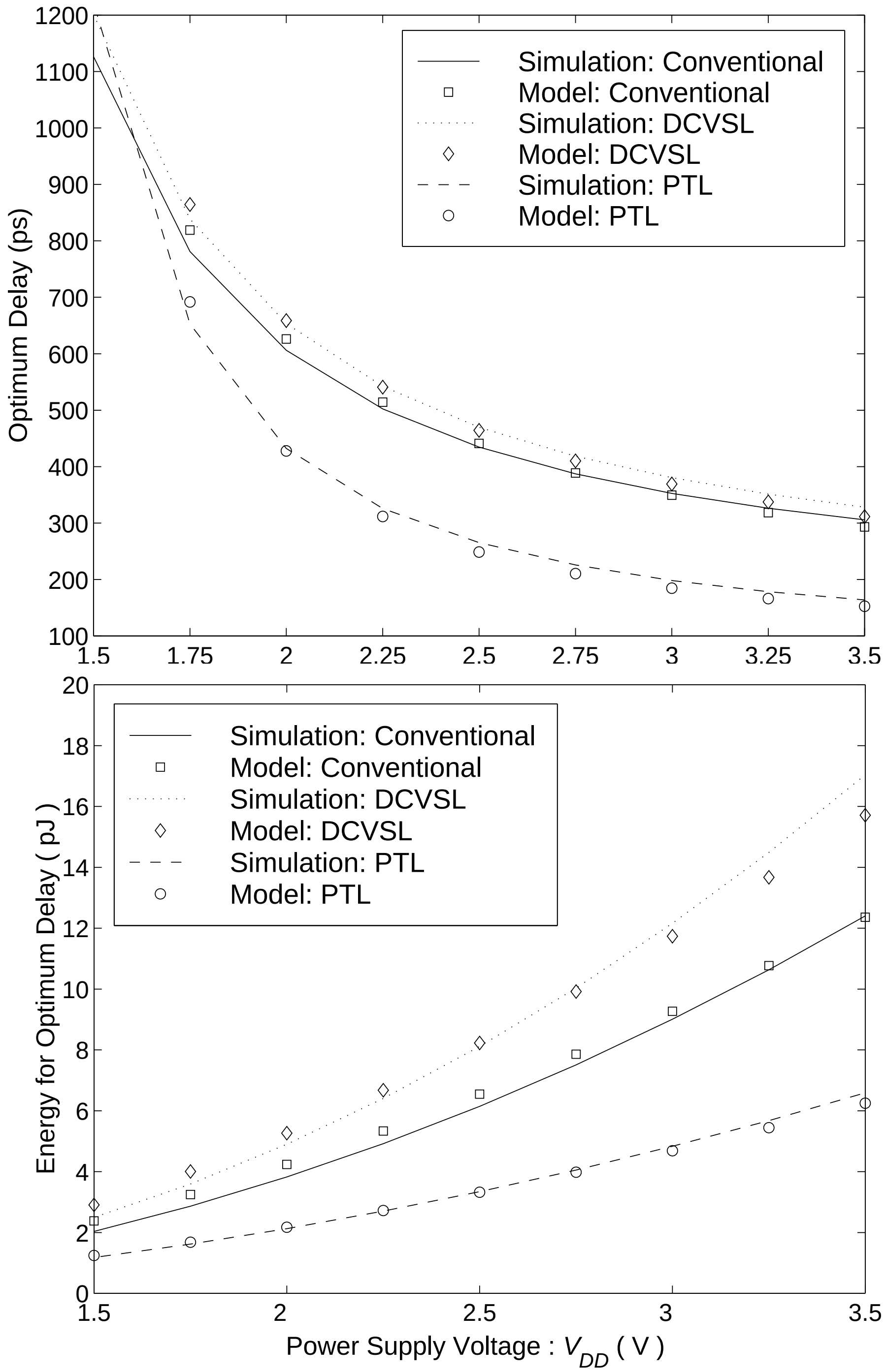


Figure 7.19: Delay and energy dissipation versus V_{DD} for the optimized CMOS implementations of the conventional (standard), DCVSL, and PTL (CPL) XOR gates. The drive's $W_{Dp}/W_{Dn} = 20/10$ and total $C_L = 200$ fF.

Chapter 8

Delay Estimation and Optimization at the Module Level

This chapter demonstrates how our findings in the previous chapters can be used to estimate and optimize the delay in a structure consisting of a number of logic gates. Such an structure is usually called a functional block or a module. One section in this chapter is devoted to delay estimation and optimization of conventional CMOS logic circuits. Another section is devoted to delay estimation and optimization of mixed logic style CMOS circuits. Finally, the chapter concludes with some remarks. The following two paragraphs present a quick summary of the previous two chapters and shows how they are related to the content of this chapter.

At the device level, we presented a unified model for the average current in an MOS device that covers a logic 1 transfer and a logic 0 transfer. Using this model at the switch level, we formulated the PMOS rising delay, NMOS falling delay, PMOS falling delay, and NMOS rising delay. The swich-level delay models accommodate the signal slope effect and the short-channel behaviour of series-connected MOS transistors. At the logic level, we applied the switch-level delay model to derive delay macromodels for different CMOS logic styles including conventional, DCVSL, and PTL. The macromodel for a logic gate is expressed in terms of the size and the topology of the gate itself, the size and the topology of the driving gate, the size of the loading gates, and any spurious capacitances. In the delay and energy macromodels for a DCVSL gate, we included the effect of the race between the PMOS latch and the NMOS network during output switching. We also derived three macromodels for a general PTL cell based on whether the critical path involves a gate drive, a source drive, or both. At the module level, to estimate the delay of a path containing mixed logic styles, we simply add the delays of the gates along the path using the developed logic-level delay macromodels.

As far as delay optimization is concerned, at the logic level, we derived exact and approximate optimal transistor sizing formulas for minimizing the delay of different CMOS logic styles. We also developed a theory for delay optimization of conventional CMOS logic gates. At the module level, we use these findings to address the problem of delay optimization. Given a circuit with an arbitrary number of logic gates, it is possible to use the formulation presented in the previous chapter to obtain the worst-case delay in the circuit and minimize it by iteration. The worst-case delay is the maximum delay over all of the paths in the circuit extending from the input to the output. Sometimes it is possible to identify the critical path in a circuit by inspection. If in doubt, the paths' delays may be re-estimated after the optimization process.

8.1 Conventional CMOS Logic Circuits

The previous chapter showed that the size of a transistor in a conventional CMOS gate is optimal if the delay due to that transistor (and any other related transistor due to symmetry) as a load equals the delay through that transistor as a drive. We mentioned that, since a CMOS gate may have more than one branch in its pull-up and its pull-down networks, any possible symmetry between the sizes of the transistors in these branches should be considered. For this purpose, with reference to Figure 7.1, we denoted the number of transistor gates related by symmetry with m and the number of transistor diffusions related by symmetry with q. The symmetry is often important in designing cell libraries and optimizing the delay of individual gates for a given input drive and an output load.

In a circuit consisting of a number of conventional CMOS gates, however, usually only one of the pull-up branches and one of the pull-down branches in a gate determine the worst-case delay of that gate along the critical path of the circuit. Therefore, observing symmetry in optimizing the gates is not necessary. We call the combination of such a pull-up branch and such a pull-down branch within a gate a stage. For a stage, usually $m = 1$ and $q = 1$. In a module or a path, we need to optimize these stages along the path. Hence, we may state that the delay of a circuit consisting of conventional CMOS logic gates is optimal if for each stage along the critical path of the circuit the following holds: *the delay due to that stage as a load equals the delay through that stage as a drive.* According to the derivation of Section 7.1, spurious capacitances along the path, signal slope factor, and series-connected transistors do not void this theory of delay optimization in conventional CMOS logic circuits.

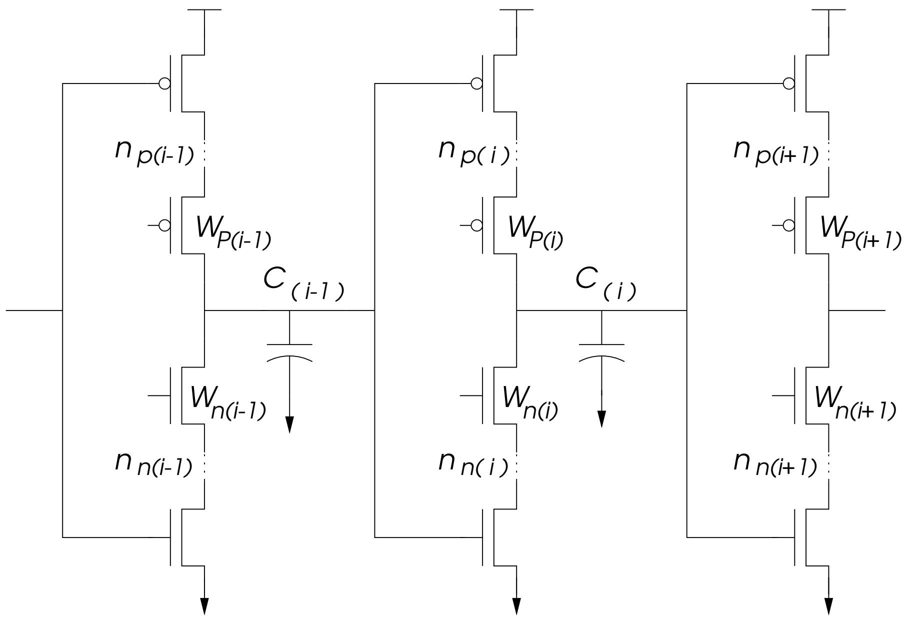


Figure 8.1: Three stages along the critical path of a conventional CMOS logic circuit.

Figure 8.1 illustrates three stages along the critical path of a CMOS logic circuits consisting of conventional gates. Using the stated theory, for optimizing the rising delay of stage i we have

$$\begin{aligned} \frac{\grave{\nu}_n}{W_{n(i-1)}} \; \grave{Y}_{n(i-1)} \acute{W}_{p(i)} \; \grave{g}_p (1 + \; \acute{S}_p) \quad &= \quad \frac{\acute{\nu}_p}{\acute{W}_{p(i)}} \; \acute{Y}_{p(i)} \; < 1 + \; \grave{S}_n > \\ &\times \quad (W_{n(i)} \; \acute{d}_n + \; \acute{C}_{(i)} + W_{p(i+1)} \; \acute{g}_p + W_{n(i+1)} \; \acute{g}_n) \quad (8.1) \end{aligned}$$

where the left hand side represents the delay of the stage as a load and the right hand side represents the delay through the stage as a drive. Note that the delay slope factor of stage i must be accounted for if i is not the last stage to be optimized. This is indicated by $<1+\grave{S}_n>$ in the above equation. Also note that, in order to express the complete delays, we should add the following delay term due to the internal capacitances to both sides of (8.1).

$$\acute{D}_{int} = \frac{\acute{\nu}_p}{\acute{W}_{p(i)}} \; \acute{X}_{p(i)} \acute{W}_{p(i)} \; \acute{d}_p = \; \acute{\nu}_p \; \acute{X}_{p(i)} \; \acute{d}_p$$

This term, however, cancels out from both sides. The reason for adding this term on both sides is that it is a delay term in which the pull-up network appears as both a load and a drive. For quick checking of optimization results, a designer may use the following approximation of (8.1).

$$\frac{\grave{\nu}_n}{W_{n(i-1)}} \, n_{n(i-1)} \acute{W}_{p(i)} \; g = \frac{\acute{\nu}_p}{\acute{W}_{p(i)}} \, n_{p(i)} \; [W_{n(i)} \; d + C_{(i)} + (W_{p(i+1)} + W_{n(i+1)}) \; g] \tag{8.2}$$

Where the signal slope factors have been ignored, the series-connection factor Y has been replaced by n, the PMOS gate capacitances per unit width for rising and falling transitions $\acute{g}_p$ and $\grave{g}_p$ have been replaced by the average gate capacitance per unit width g, the NMOS diffusing capacitance per unit width for rising delay $\acute{d}_n$ has been replaced by the average diffusion capacitance d and, similarly, the rising delay load $\acute{C}_{(i)}$ has been replaced by $C_{(i)}$.

As far as the total or average delay is concerned, according to the theory, the following set of equations must hold for each stage i on the path.

$$\left\{ \begin{array}{l} \grave{\nu}_n/(W_{n(i-1)} \; \grave{Y}_{n(i-1)}) \; \hat{W}_{p(i)} \; \grave{g}_p \; (1+ \; \acute{S}_p) \\ + \; \acute{\nu}_p/(W_{p(i-1)} \; \acute{Y}_{p(i-1)}) \; \hat{W}_{p(i)} \; \acute{g}_p \; (1+ \; \grave{S}_n) \\ + \; \grave{\nu}_n/(\hat{W}_{n(i)} \; \grave{Y}_{n(i)}) \; \hat{W}_{p(i)} \; \grave{d}_p \; <1+ \; \acute{S}_p> \quad = \\ \acute{\nu}_p/(\hat{W}_{p(i)} \; \acute{Y}_{p(i)}) \; (\hat{W}_{n(i)} \; \acute{d}_n + \; \acute{C}_{(i)} + W_{p(i+1)} \; \acute{g}_p + W_{n(i+1)} \; \acute{g}_n) \; <1+ \; \grave{S}_n> \\ \\ \grave{\nu}_n/(W_{n(i-1)} \; \grave{Y}_{n(i-1)}) \; \hat{W}_{n(i)} \; \grave{g}_n \; (1+ \; \acute{S}_p) \\ + \; \acute{\nu}_p/(W_{p(i-1)} \; \acute{Y}_{p(i-1)}) \; \hat{W}_{n(i)} \; \acute{g}_n \; (1+ \; \grave{S}_n) \\ + \; \acute{\nu}_p/(\hat{W}_{p(i)} \; \acute{Y}_{p(i)}) \; \hat{W}_{n(i)} \; \acute{d}_n \; <1+ \; \grave{S}_n> \quad = \\ \grave{\nu}_n/(\hat{W}_{n(i)} \; \grave{Y}_{n(i)}) \; (\hat{W}_{p(i)} \; \grave{d}_p + \; \grave{C}_{(i)} + W_{p(i+1)} \; \grave{g}_p + W_{n(i+1)} \; \grave{g}_n) \; <1+ \; \acute{S}_p> \end{array} \right. \tag{8.3}$$

Similarly, the following must hold for optimizing the falling delay of stage i

$$\begin{aligned} \frac{\acute{\nu}_p}{W_{p(i-1)}} \acute{Y}_{p(i-1)} \grave{W}_{n(i)} \acute{g}_n (1 + \grave{S}_n) &= \frac{\grave{\nu}_n}{\grave{W}_{n(i)}} \grave{Y}_{n(i)} < 1 + \acute{S}_p > \\ &\times (W_{p(i)} \grave{d}_p + \grave{C}_{(i)} + W_{p(i+1)} \grave{g}_p + W_{n(i+1)} \grave{g}_n) \end{aligned} \tag{8.4}$$

Consider a critical path with N logic stages. For optimizing the total delay, there are $2N$ equations and $2N$ unknowns. This system of non-linear equations may be solved by iteration with the minimum allowable transistor width w as the initial value for all transistors.

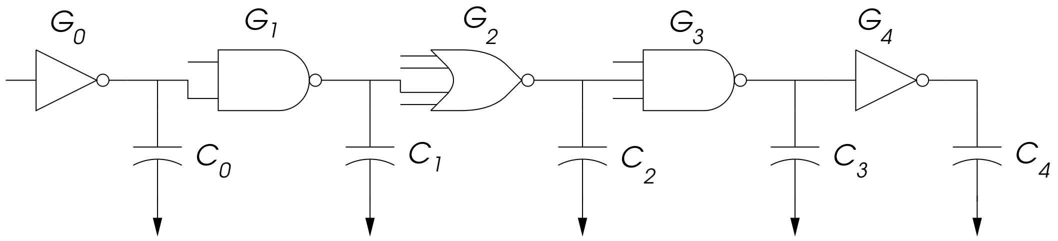


Figure 8.2: An example of a critical path in a conventional CMOS logic circuit.

For example, the critical path of Figure 8.2 includes four gates G_1 through G_4. The driving gate G_0 has $W_{p0} = 20\ \mu$m and $W_{n0} = 10\ \mu$m. The spurious load capacitances are $C_0 = 300$ fF, $C_1 = 150$ fF, $C_2 = 100$ fF, $C_3 = 200$ fF, and $C_4 = 250$ fF. Using (8.3) gives the results listed in Table 8.1 for optimizing the total delay in the path. With a circuit of this size, it is almost impossible to verify the optimization results by a circuit level simulator such as HSPICE.

Table 8.1: Optimal transistor sizing for the critical path of Figure 8.2.

W_{p1} / W_{n1}	W_{p2} / W_{n2}	W_{p3} / W_{n3}	W_{p4} / W_{n4}
52 μm / 56 μm	180 μm / 50 μm	65 μm / 118 μm	69 μm / 57 μm

Although the conditions for optimizing the rising, falling, and total delays in conventional CMOS logic circuits may be turned into closed-form formulas, similar to those of the previous chapter, the message we want to convey in this section is that one may easily derive these formulas by knowing the developed theory of delay optimization. This is important for a quick hand analysis of the optimization results. Let us check the results of Table 8.1 by examining, for example, W_{p2}. From the theory, we know that the total loading effect of $\hat{W}_{p2}$ on the delay must equal its driving effect on the delay. Therefore, the following must approximately hold for $\hat{W}_{p2}$.

$$2\ \frac{\grave{\nu}_n}{W_{n1}}\ \hat{W}_{p2}\ g + 1\ \frac{\acute{\nu}_p}{W_{p1}}\ \hat{W}_{p2}\ g + 1\ \frac{\grave{\nu}_n}{\hat{W}_{n2}}\ \hat{W}_{p2}\ d \overset{?}{=}\ 4\ \frac{\acute{\nu}_p}{\hat{W}_{p2}}\ [\hat{W}_{n2}\ d + C_2 + (W_{p3} + W_{n3})\ g]$$

Where factors 2, 1, 1, and 4 are the numbers of series-connected transistors in the NMOS network of G_1, PMOS network of G_1, NMOS network of G_2, and PMOS network of G_2, respectively. Substituting the values for the parameters yields

$$180\left(2\frac{4.15}{56}\ 1.85 + \frac{10.95}{52}\ 1.85 + \frac{4.15}{50}\ 2.13\right) \overset{?}{=} 4\frac{10.95}{180}\ [50\quad 2.13 + 100 + (65 + 118)1.85]$$

The left hand side evaluates to 151 ps and the right hand side evaluates to 132 ps. Considering that we have ignored a number of effects, the difference of about 10% between the two sides is quite acceptable. On the other hand, if we calculate $\hat{W}_{p2}$ from

$$\hat{W}_{p2}\left(2\frac{4.15}{56}\ 1.85 + \frac{10.95}{52}\ 1.85 + \frac{4.15}{50}\ 2.13\right) = 4\frac{10.95}{\hat{W}_{p2}}\ [50\quad 2.13 + 100 + (65 + 118)1.85]$$

the result is 169 μm, which is very close to the value listed in Table 8.1. Such a quick check gives the designer some degree of assurance that the optimal transistor sizing produced by the employed CAD tool are indeed valid.

8.2 Mixed Logic Style CMOS Circuits

This section discusses optimization of digital CMOS circuits involving mixed logic styles. Table 8.2 summarizes the optimal transistor sizing formulas for different CMOS logic styles. The formulas are similar to those of Chpater 7, except for the terms enclosed within “< >”. Such a term accounts for the signal slope effect of the stage being optimized on the next stage. Therefore, they should not be included if the stage being optimized is the last stage on the path. In this regard, the formulas given in Chapter 7 as rules of thumb remain unchanged for optimizing a path, except $\acute{\Gamma}$ for DCVSL, which changes from $(3/4n)\Lambda$ to $(4/5n)\Lambda$. As

Table 8.2: Optimal transistor sizing of CMOS logic styles. Terms enclosed within “< >” should not be included if the stage being optimized is the last stage. Notation: subscripts n (NMOS), p (PMOS), D, G, S (drives), and L (load); Accents: “ ´ ” (rising transition) and “ ` ” (falling transition); $\Lambda = \acute{\nu}_p / \grave{\nu}_n$. The parameters are defined in Chapter 6 and Chpater 7 with reference to Figure 7.1 for conventional CMOS, Figure 7.8 for DCVSl, and Figure 7.13 for PTL.

CONVENTIONAL CMOS LOGIC STYLE

Rising Delay:

$$\acute{W}_p = \sqrt{W_{Dn} \frac{\acute{C}_L + q_n W_n \acute{d}_n}{m_p \grave{g}_p} \frac{\acute{\nu}_p \acute{Y}_p <1+ \grave{S}_n>}{\grave{\nu}_n \grave{Y}_{Dn}(1+ \acute{S}_p)}}$$

Falling Delay:

$$\grave{W}_n = \sqrt{W_{Dp} \frac{\grave{C}_L + q_p W_p \grave{d}_p}{m_n \acute{g}_n} \frac{\grave{\nu}_n \grave{Y}_n <1+ \acute{S}_p>}{\acute{\nu}_p \acute{Y}_{Dp}(1+ \grave{S}_n)}}$$

Average or Total Delay:

$$\hat{W}_p = \sqrt{\frac{(\acute{C}_L + q_n W_n \acute{d}_n) \acute{\nu}_p \acute{Y}_p <1+ \grave{S}_n>}{m_p \grave{g}_p \frac{\grave{\nu}_n}{W_{Dn}} \grave{Y}_{Dn}(1+ \acute{S}_p) + m_p \acute{g}_p \frac{\acute{\nu}_p}{W_{Dp}} \acute{Y}_{Dp}(1+ \grave{S}_n) + q_p \grave{d}_p \frac{\grave{\nu}_n}{W_n} \grave{Y}_n <1+ \acute{S}_p>}}$$

$$\hat{W}_n = \sqrt{\frac{(\grave{C}_L + q_p W_p \grave{d}_p) \grave{\nu}_n \grave{Y}_n <1+ \acute{S}_p>}{m_n \acute{g}_n \frac{\acute{\nu}_p}{W_{Dp}} \acute{Y}_{Dp}(1+ \grave{S}_n) + m_n \grave{g}_n \frac{\grave{\nu}_n}{W_{Dn}} \grave{Y}_{Dn}(1+ \acute{S}_p) + q_n \acute{d}_n \frac{\acute{\nu}_p}{W_p} \acute{Y}_p <1+ \grave{S}_n>}}$$

DCVSL and DIL STYLES

$$\frac{\acute{W}_n}{\acute{W}_p} = \sqrt{\frac{(1+ \acute{S}_p)}{<1+ \grave{S}_n>} \left[\frac{1}{2} \left(\frac{\grave{Y}_n}{\Lambda} \right)^2 \frac{\grave{C}_L + q W_n \grave{d}_n}{\acute{C}_L + q W_n \acute{d}_n} + \frac{\grave{Y}_n}{\Lambda} \frac{W_n(\grave{d}_p + \grave{g}_p)}{\acute{C}_L + q W_n \acute{d}_n} \right]} + \frac{1}{2} \frac{\grave{Y}_n}{\Lambda}$$

$$\acute{W}_n = \sqrt{\frac{\grave{Y}_n}{\Lambda} \frac{(1+ \acute{S}_p) [\grave{C}_L + W_p(\grave{d}_p + \grave{g}_p) + 0.5 \frac{\grave{Y}_n}{\Lambda} W_p \grave{d}_n]}{(1+ \grave{S}_n) \acute{Y}_{Dp} m \acute{g}_n / W_{Dp} + q \acute{d}_n / W_p}} + \frac{1}{2} \frac{\grave{Y}_n}{\Lambda} W_p$$

PTL STYLES (CPL, DPL, DILN, DILP)

Gate Drive:

$$\acute{W}_n = \sqrt{\frac{\acute{\nu}_n \acute{Y}_n}{\acute{\nu}_p \acute{Y}_{Gp}} \frac{W_{Gp}[\acute{C}_L + (\acute{g}_p + \acute{d}_p) w_p] <1+ \acute{S}_n>}{m \acute{g}_n (1+ \acute{S}_n)}}$$

Source Drive:

$$\acute{W}_n = \sqrt{\frac{\acute{\nu}_n \acute{Y}_n}{\acute{\nu}_p \acute{Y}_{Sp}} \frac{W_{Sp}[\acute{C}_L + (\acute{g}_p + \acute{d}_p) w_p] <1+ \acute{S}_n>}{(m \acute{g}_n + q_i \acute{d}_n)(1+ \acute{S}_n) + q_o \acute{d}_n}}$$

Both Drives:

$$\acute{W}_n = \sqrt{\frac{\acute{\nu}_n}{\acute{\nu}_p} \acute{Y}_n \frac{W_{Gp} W_{Sp}[\acute{C}_L + (\acute{g}_p + \acute{d}_p) w_p] <1+ \acute{S}_n>}{(1+ \acute{S}_n) W_{Sp} \acute{Y}_{Gp} m \acute{g}_n + W_{Gp} \acute{Y}_{Sp} q_o \acute{d}_n}}$$

we mentioned in the previous section, the symmetry related parameters m and q for the conventional logic stages usually equal unity. This, however, is not true for the differential logic styles. Due to the inherent symmetrical structures of DCVSL and PTL styles, the symmetry related parameters are often larger than unity for these logic styles. The reader is reminded that, for DIL implementations, one should include $w(\acute{d}_p + \acute{g}_p)$ in $\acute{C}_L$ and $w(\grave{d}_p + \grave{g}_p)$ in $\grave{C}_L$ in the corresponding formulas listed in Table 8.2. This is to account for the NMOS transistors of the inverter latch, which are not active during switching. We should also remind that the parameters used in formulas of Table 8.2 are defined in Chapter 6 and Chpater 7 with reference to Figure 7.1 for the conventional CMOS logic style, Figure 7.8 for DCVSl, and Figure 7.13 for PTL.

We have implemented the optimal transistor sizing formulas of Table 8.2 as functions in a programming library. Each set of formulas corresponding to a logic style is represented by a function of the following pattern.

$$\begin{aligned}(W_p, W_n) \quad = \quad & \textit{LogicStyle-opt}(V_{DD}, \{<\text{ Drive Info }> W_{Dp}, n_{Dp}, W_{Dn}, n_{Dn}\}, \\ & \{<\text{ Cell Info }> W_p, m_p, n_p, q_p, W_n, m_n, n_n, q_n\}, \\ & \{<\text{ Load Info }> \text{Gate and Diffusion Capacitances}\}) \qquad (8.5)\end{aligned}$$

Where *LogicStyle* is *conve* for a conventional CMOS implementation, *dcvsl* for a DCVSl implementation, *ptlgd* for a PTL implementation whose worst-case delay is governed by a G-drive, *ptlsd* for a PTL implementation whose worst-case delay is governed by an S-drive, and *ptlbd* for a PTL implementation whose worst-case delay is governed by both a G-drive and an S-drive. Note that V_{DD} has to be specified, because some of the parameters, such as ν, Y, S, and the capacitances are supply voltage dependent. Separate functions in the library calculate each of these parameters for NMOS and PMOS transistors and, for each transistor type, for rising and falling transitions. The calculations are based on the models of Chapter 6. In addition to the optimization functions, the library includes functions for estimating the delays of the CMOS logic styles. These functions are based on the delay macromodels developed in Chapter 7. The rest of this section presents an example for delay optimization and estimation in a CMOS circuit including mixed logic styles.

Chapter 2 explained that, in asynchronous circuits, there are two common signaling protocols for communicating data between a sender and a receiver: the four-phase and

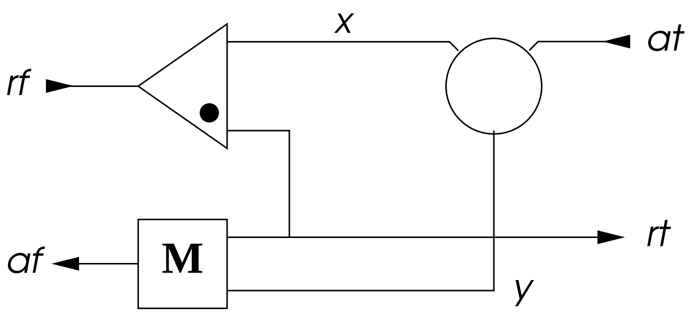


Figure 8.3: A four-to-two phase converter [34].

the two-phase protocol. If the sender is using the four-phase protocol and the receiver is using the two-phase protocol, then a module called four-to-two phase converter provides the interface between the sender and the receiver. A delay-insensitive implementation of the four-to-two phase converter is depicted in Figure 8.3 [34]. This module uses a TOGGLE, a JOIN (C-element), and a MERGE (XOR gate). In the figure, rf denotes the request signal from the four-phase sender, af denotes the acknowledgment signal to the four-phase sender, rt denotes the request signal to the two-phase receiver, and at denotes the acknowledgment signal from the two-phase receiver. An operation cycle of the four-to-two phase converter starts with a transition on rf and ends with a transition on af. In between these two transitions, there are two sets of events that may take place in parallel. One set of events consists of a transition on af followed by a transition on rf. The other set of events consists of a transition on rt followed by a transition on at.

Assume that a designer has decided to implement this four-to-two phase module with a conventional CMOS TOGGLE, DIL C-element, and CPL XOR gate. The DIL C-element and CPL XOR gate were introduced in the previous chapters. The behaviors of the outputs of the TOGGLE, b and c, can be expressed in terms of the input a and the previous states of the outputs, $\hat{b}$ and $\hat{c}$, by the following Boolean functions.

$$b = a\,c' + a'\,\hat{b}$$
$$c = a'\,b + a\,\hat{c}$$

A conventional implementation of the TOGGLE derived from these expressions is illustrated in Figure 8.4. A similar implementation with an additional initialization signal is presented in [37] under the name Yantchev TOGGLE. In the TOGGLE implementation of Figure 8.4,

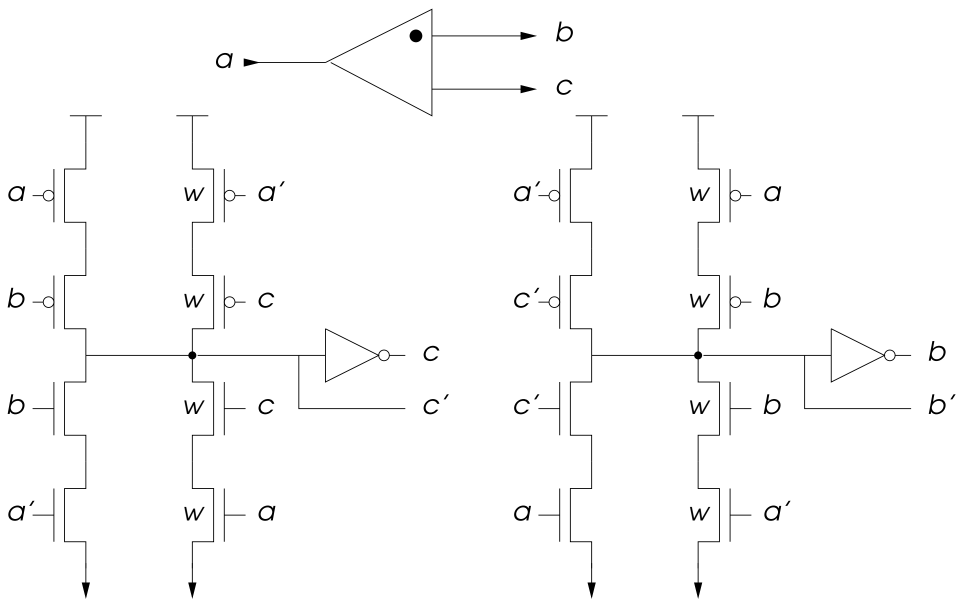


Figure 8.4: Schematic of a conventional implementation of the TOGGLE.

some of the transistors are not active during output switching and are only dedicated to maintaining the state of an output. These transistors are assigned the minimum width w.

Figure 8.5 depicts the transistor-level schematic of the four-to-two phase converter. Each input and each output of the module is connected to a buffer with a PMOS transistor of width $W_{pb} = 15\ \mu$m and an NMOS transistor of width $W_{nb} = 10\ \mu$m. The transistor sizing parameters to be determined for the TOGGLE include W_{p1}, W_{n1}, W_{p2}, W_{n2}, W_{p3}, W_{n3}, W_{p4}, and W_{n4}. Similarly, the transistor sizing parameters to be determined for the C-element include W_{p5}, and W_{n5}. The XOR gate has only one transistor sizing parameter, W_{n6}. The rest of the transistors in the circuit have the minimum width w, as indicated. One way of optimizing this circuit is to target minimal delay in conveying the request signal to the receiver, i.e. producing rt, and minimal delay in producing the last event of an operation cycle of the module, i.e. producing the second transition on af. This requires calling the following functions, which are evaluated in a number of iterations.

$$
\begin{aligned}
(W_{p1}, W_{n1}) &= \textit{conve-opt} \quad (3\ V, \{W_{pb}, 1, W_{nb}, 1\}, \{W_{p1}, 1, 2, 1, W_{n1}, 1, 2, 1\}, \\
&\qquad \{(W_{p2} + W_{pb})g_p + (W_{n2} + W_{nb} + 2W_{n6})g_n + w\ d_n + w\ d_p\}\ ) \\
(W_{p2}, W_{n2}) &= \textit{conve-opt} \quad (3\ V, \{W_{p1}, 2, W_{n1}, 2\}, \{W_{p2}, 1, 1, 1, W_{n2}, 1, 1, 1\},
\end{aligned}
$$

$$
\begin{aligned}
& && \{(W_{p3}+W_{pb}+w)g_p+(W_{nb}+W_{n3}+w+2W_{n6})g_n\}) \\
(W_{p3},W_{n3}) &= \textit{conve-opt} && (3\ V, \{W_{pb},1,W_{nb},1\}, \{W_{p3},1,2,1,W_{n3},1,2,1\}, \\
& && \{(W_{p1}+W_{p4})g_p+(W_{n1}+W_{n4}+W_{n5})g_n+w\ d_n+w\ d_p\}\ ) \\
(W_{p4},W_{n4}) &= \textit{conve-opt} && (3\ V, \{W_{p3},2,W_{n3},2\}, \{W_{p4},1,1,1,W_{n4},1,1,1\}, \\
& && \{w\ g_p+(w+W_{n5})g_n\}\ ) \\
(W_{p5},W_{n5}) &= \textit{dcvsl-opt} && (3\ V, \{W_{p4},1,W_{n4},1\}, \{W_{p5},0,1,1,W_{n5},1,2,1\}, \\
& && \{w\ g_p+w\ g_n+(2W_{n6}+w)d_n+w\ d_p\}\ ) \\
(W_{p6},W_{n6}) &= \textit{dptsd-opt} && (3\ V, \{W_{p5},1,W_{n5},1\}, \{w,0,1,1,W_{n6},0,1,2\}, \\
& && \{W_{pb}\ g_p+W_{nb}\ g_n\}\ )
\end{aligned}
$$

The reader may verify our procedure by inspecting Figure 8.5 and the general pattern for the optimization functions given in (8.5). Table 8.3 lists the results of the optimization process. To evaluate the delay along a path, we add the delays of the logic stages along that path. Using the transistor sizings suggested in Table 8.3, the estimated average delay between receiving the first transition on rf and producing the following transition on rt is 210 ps, and the estimated average delay between receiving the second transition on rf and producing the following transition on af is 750 ps.

Table 8.3: Optimal transistor sizing for the four-to-two phase converter of Figure 8.5.

W_{p1} / W_{n1}	W_{p2} / W_{n2}	W_{p3} / W_{n3}
46 μm / 32 μm	43 μm / 33 μm	52 μm / 46 μm
W_{p4} / W_{n4}	W_{p5} / W_{n5}	W_{p6} / W_{n6}
27 μm / 21 μm	32 μm / 49 μm	w μm / 13 μm

8.3 Concluding Remarks

This chapter has demonstrated a method of delay optimization in conventional and mixed logic CMOS circuits by using the theory of delay optimization for conventional CMOS circuits and the logic-level delay macromodels developed in the previous chapter. The delay

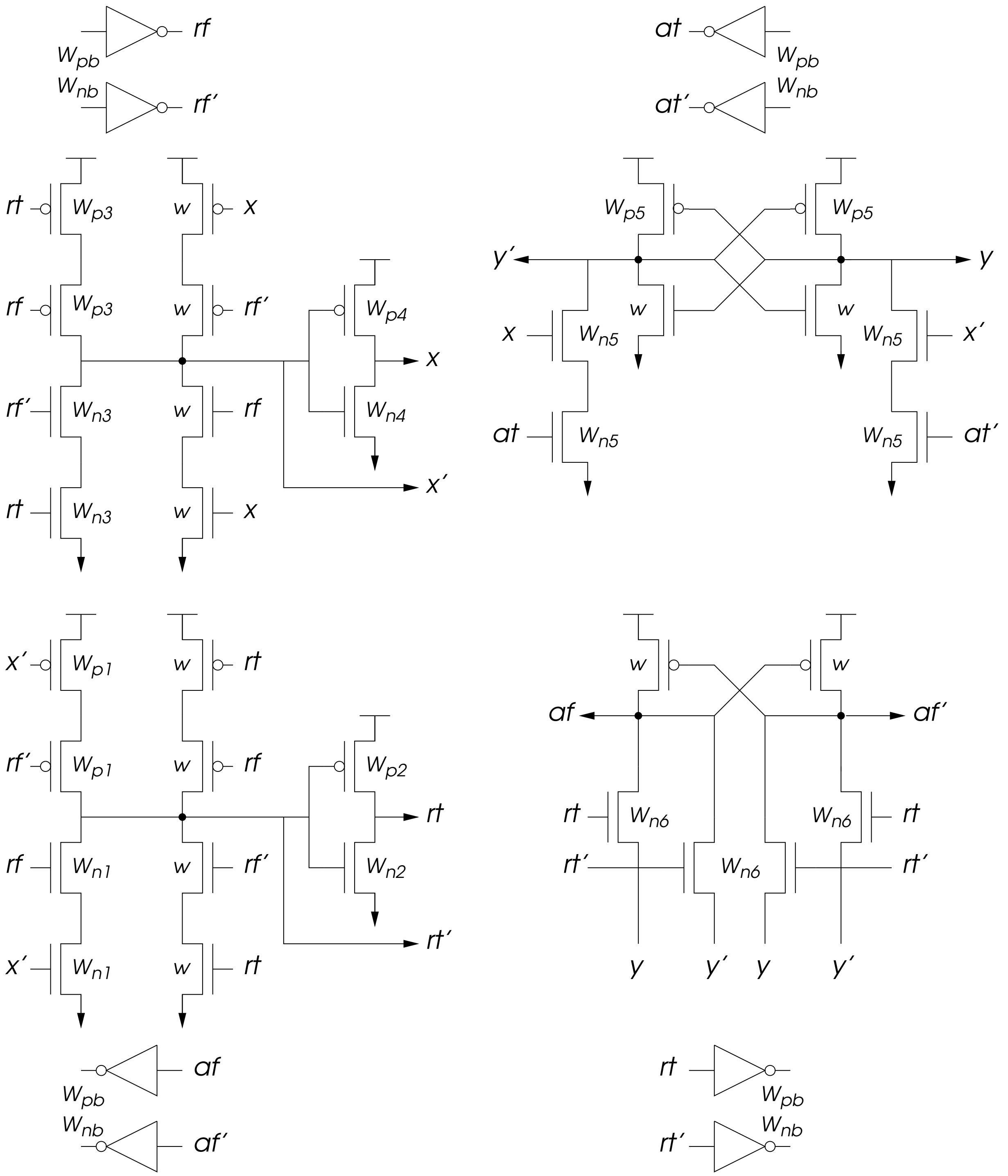


Figure 8.5: Transistor-level schematic of an asynchronous four-to-two phase converter.

macromodels may also be used to optimize a circuit under multiple constraints of delay, energy, area, and supply voltage. This issue, however, is not part of this thesis. Another possible work is delay optimization at the system level. System level optimization requires a separate study, because optimizing the delay along a path may actually increase the delay of another path which, in turn, may increase the overall worst-case delay of the system.

Chapter 9

Conclusion

In this thesis, we have presented a technique for modeling, evaluation, and optimization of digital CMOS circuits. Based on this technique, we have derived delay models and closed-form optimal transistor sizing formulas for several conventional and differential CMOS logic styles. The scope of our technique covers the device, switch, logic, and module levels of abstraction. The technique evolves from the idea of explicit formulation of the delay of a logic gate in terms of its own size and the sizes of its driving and loading gates. We have applied the developed delay models and optimization formulas to a number of asynchronous circuits primitives and modules.

One of the contributions of this thesis is the theory of delay optimization in CMOS logic circuits. The theory states that *the delay in a circuit consisting of conventional CMOS logic gates is minimal if for each stage along the critical path of the circuit, the delay due to that stage (as a load) equals the delay through that stage (as a drive).* In other words, if the loading effect of each stage equals its driving effect, then the delay along the path is minimal. This theory also generally holds for logic gates which experience no cases of overlapping and opposing currents. We have shown that the theory is valid even when the input slope factor and the effect of serial connection of transistors are taken into account. Moreover, presence of branches and spurious capacitances along the path does not void the theory. This theory simplifies the process of delay optimization into expressing and solving a system of non-linear equations by iteration. The theory is also convenient for checking the optimal transistor sizing results of a CAD tool.

Another contribution of this thesis is the derivation of the optimal transistor sizing formulas for both conventional and unconventional CMOS logic styles. These optimal transistor sizing formulas enable optimization of mixed logic-style CMOS circuits. The logic styles covered include conventional, DCVSL, and PTL. We have derived three optimal sizing formulas for PTL style, based on the position of the driving gate or gates. We have demonstrated optimal transistor sizing of a mixed logic style module by using these formulas.

A third contribution of this thesis which, in fact, has led to the previous two contributions, is the development of a unified delay model for CMOS logic styles. The delay model includes an expression for the saturation current of short-channel MOS transistors that is equally valid for rising and falling transitions. The model captures the effect of input signal slope and characterizes the behaviour of MOS transistors connected in series. This model shows that the internal diffusion capacitances of a chain of similarly-sized series-connected transistors do not affect optimal sizing of the chain.

These three contributions are likely to have impact on developing logic simulators and optimization tools that support mixed CMOS logic styles in future. The rest of this chapter reviews the results of this thesis and highlights directions for future work.

9.1 Review

The following is a review of the material presented in this thesis divided into subsections on modeling, optimization, and applications. This summary is intended to be self-contained.

9.1.1 Delay Modeling

The scope of the delay modeling technique presented in this thesis covers a number of abstraction levels. At each level, the reliability of the model has been verified by HSPICE simulations. In general, the proposed model exhibits very good agreement with HSPICE simulations. We have followed a bottom-up approach in delay modeling.

Starting at the device level, we have proposed the following expression for evaluating the saturation current of submicron MOS devices.

$$I_D = \kappa \; W \; (V_{GS} - V_T)^{\;\xi + \frac{\vartheta}{V_{GS}}}$$

Where W is the effective width of the device, parameter κ depends on the technology and the effective channel length, index ξ captures the velocity saturation effect, and index ϑ captures the mobility degradation effect. This model is considerably more accurate than the popular α-power law, which replaces $\xi + \vartheta/V_{GS}$ with a constant index α. With proper parameters, the above expression is also used to represent the saturation current in an NMOS device transferring a logic 1, rather than 0, and in a PMOS device transferring a logic 0, rather than 1. Overall, four sets of parameters κ, ξ, and ϑ are required to characterize the four types of currents. Procedures for extracting these parameters through circuit simulations have been provided.

At the switch level, we recognize four types of delays: PMOS rising delay, NMOS falling delay, NMOS rising delay, and PMOS falling delay. These delays are derived using the corresponding four types of currents. To capture the effect of input signal slope, we have extended a previously reported technique to the four delay types. This technique adds a fraction of the input transition time to the step delay. Moreover, we have offered a semi-empirical model that characterizes the delay behaviour through a chain of MOS transistors connected in series. According to our model, the general expression for the delay through N similarly-sized transistors connected in series is given by

$$D = \frac{\nu}{W} \; [X d \; W + Y(g \; W_{Lg} + d \; W_{Ld} + C_l)] \; + \; S \; D_0$$

Where D_0 is the step-delay of the driving gate, S is the input slope factor, and X and Y are the degradation factors related to N. Parameter $\nu = WV_{DD}/(2I_D)$, W_{Lg} is the total gate-oxide width of the load, W_{Ld} is the total diffusion width of the load, g is the gate-oxide capacitance per unit width, d is the diffusion capacitance per unit width, and C_l is the load capacitance due to interconnects. Note that different X, Y, and S are defined for each of the four delay types. Also, g and d are different for NMOS and PMOS transistors as well as for the rising and falling transitions. We have included expressions for X, Y, and S and methodologies for extracting d and g through circuit simulations. From the above expression for the delay, it is clear that neglecting the diffusion capacitances does not affect optimal transistor sizing. Other topics studied at this level include the influence of overlapping and opposing currents on the delay. We have suggested intuitive expressions to cover these two cases.

At the logic level, we have applied the switch-level delay model to formulate delay macromodels for different CMOS logic styles including conventional, DCVSL, and PTL, as depicted in Figure 9.1. Defining and accommodating the four types of delays at the switch level has proven an innovative move that enabled us to treat CMOS gates implemented in a variety of logic styles. The proposed macromodel for a logic gate is expressed in terms of the size and the topology of the gate itself, the size and the topology of the driving gate, the size of the loading gate or gates, and the interconnect capacitances. At this level, a pull-up or pull-down network along the critical path of a gate is represented by parameters n, m, and q, as illustrated in Figure 9.1. In the delay and energy macromodels for a DCVSL gate, we have included the effect of the race between the PMOS latch and the NMOS network during output switching. Neglecting the effect of opposing currents may result in considerable underestimation of the delay and energy. We have derived three macromodels for a general PTL cell based on whether the critical path involves a gate drive, a source drive, or both, as illustrated in the figure. At the module level, to estimate the delay of a path containing mixed logic styles, we add the delays of the gates along the path using the developed delay macromodels.

9.1.2 Delay Optimization

Using the logic-level delay macromodels, we have derived closed-form formulas for the optimization of CMOS logic styles. For each logic style, there is an expression for the optimal sizing of the pull-up network and another one for the pull-down network. In their exact forms, these two expressions are not independent and, hence, are solved by a few iterations. However, we have also approximated these expressions into simpler formulas for quick optimizations, as listed in Table 9.1. In these formulas, all parameters, except ν, are physical and identifiable through the schematics. Using the optimal sizing formulas, we have demonstrated that it is feasible to optimize a circuit involving mixed CMOS logic styles.

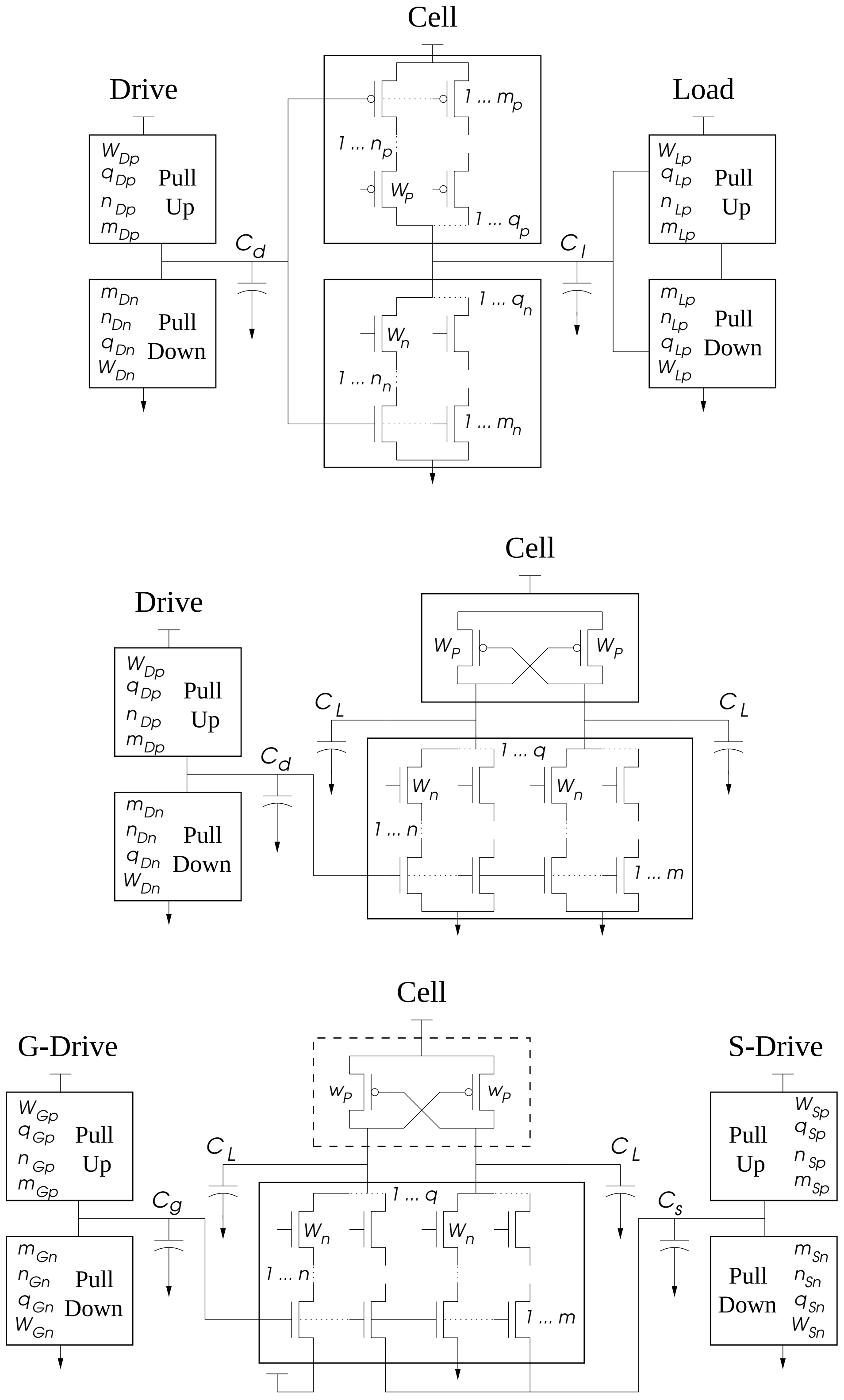


Figure 9.1: General schematics of a conventional (top), DCVSL (middle), and PTL (bottom) gate.

Table 9.1: Optimal transistor sizing in CMOS logic styles for minimizing the delay over one cycle. Notation: subscripts n (NMOS), p (PMOS), t (total NMOS+PMOS), D, G, S (drives), and L (load); Accents: ´ (rising transition) and ` (falling transition); $\Lambda = \acute{\nu}_p / \grave{\nu}_n$.

Style	W_n	W_p
Conv.	$\sqrt{\frac{n_n}{m_n} \frac{W_{Dp}\ W_{Dn}\ W_{Lt}}{n_{Dn}\ W_{Dp} + \Lambda\ n_{Dp}\ W_{Dn}}}$	$W_n \sqrt{\Lambda \frac{m_n n_p}{m_p n_n}}$
DCVSL	$\frac{4}{3}\sqrt{\frac{2n}{\Lambda} \frac{W_{Dp}\ W_{Lt}}{m\ n_{Dp}}}$	$W_n\ \frac{3}{4\ n}\ \Lambda$
PTL	$\sqrt{n \frac{\acute{\nu}_n}{\acute{\nu}_p} \frac{W_{Lt}\ W_{Gp}\ W_{Sp}}{W_{Sp}\ n_{Gp}\ m + W_{Gp}\ n_{Sp}\ q}}$	$W_{\min}$

For the special case of inverter circuits with a uniform PMOS to NMOS width ratio Γ, we have shown that the following optimization formulas are valid for minimizing the total delay, rising delay, and falling delay, respectively.

$$\hat{W} = \sqrt{W_D W_L} \qquad \acute{W} = \sqrt{W_D W_L \frac{\Lambda}{\Gamma}} \qquad \grave{W} = \sqrt{W_D W_L \frac{\Gamma}{\Lambda}}$$

Where W, W_D, and W_L are the widths of the NMOS transistor in a reference inverter, its driving inverter, and its loading inverter, respectively. The technology dependent parameter Λ is the PMOS-to-NMOS driveability ratio. In addition, we have been able to derive the following relation for minimizing the energy-delay product, which is a design criterion in some VLSI circuits.

$$\Gamma = \frac{1}{4}(\sqrt{\Lambda^2 + 8\Lambda} - \Lambda)$$

For typical values of Λ such as 2.5, the above evaluates to 0.65. On the other hand, Our formulation confirms that the total delay is minimum when $\hat{\Gamma} = \sqrt{\Lambda}$, which is around 1.5. The formulation also clearly shows that in designing a chain of buffers for driving a large load, tapering the buffers by a constant factor is a necessary condition rather than an arbitrary assumption.

9.1.3 Circuits and Applications

A considerable part of this work has been devoted to comparing different CMOS implementations of logic gates. We have developed a fair method for this purpose. Based on this method we have proposed that two rules should be observed. The first rule is that before evaluating different implementations of a gate, each of them should be optimized for the particular environment. The second rule is that the performance of the implementations should be compared in light of their cost in terms of energy dissipation. Our optimal transistor sizing formulas facilitate fulfilling the first rule. To comply with the second rule, we have introduced the idea of using energy-delay and energy-frequency graphs. Our methodology has been exemplified by applying it to the implementations of the C-element and XOR gate.

We have studied the performance and energy consumption of eight CMOS implementations of the C-element. Four out of the eight implementations belong to the single-rail family of logic styles, and all have been used in practical circuits. The other four are based on a differential logic style introduced in this thesis under the acronym DIL. DIL resembles DCVSL in structure and operation but, unlike DCVSL, has a static memory. This property of DIL makes it suitable for implementing primitives like the C-element and the TOGGLE. We have compared the performance and energy dissipation of the C-element implementations in two typical environments. The first environment evaluates a C-element operating in isolation, while the second environment evaluates a group of C-elements that are mutually dependent. In both environments, the C-elements were optimized for their best performance. Results show that among the single-rail implementations, the symmetric C-element by Van Berkel offers a better balance between performance and energy in both test environments. The reason is that, compared to the other single-rail implementations, the symmetric one has the least overhead for maintaining the state of the output. This is realized by first, having a topology that does not resist output switching and, second, having fewer keeper transistors. These studies demonstrated that minimizing the number of transistors dedicated to latching (keepers) and avoiding topologies that resist output switching may result in significant energy savings. The symmetric implementation, however, is outperformed in the first test environment by two modified versions of the DIL implementation, namely DILP and DILN. In the second test environment, the symmetric C-element is still preferred, because the differential implementations may fail in structures with feedback loops due to the problem of divergence of complementary outputs. These studies demonstrated that minimizing the number of transistors dedicated to latching (keepers) and avoiding topologies that resist output switching may result in significant energy savings.

We have also compared the conventional, DCVSL, and CPL implementations of the XOR gate. Since the performance and optimal transistor sizing of a gate are both functions of the supply voltage, we have added another dimension to our comparison technique for the case of the XOR gate. We have evaluated the optimized performance and the corresponding energy of the implementations for a range of power supply voltages. The results of the comparison are in favour of the CPL implementation followed by the conventional implementation of the XOR gate.

9.2 Directions for Future Research

Considering the scope of this work, it may be extended in a number of directions. This section outlines some relevant potential future work.

We have paved the way for developing a CAD tool for logic simulation and optimization of digital CMOS circuits. This is the most obvious extension of our work. The tool would support circuits including mixed CMOS logic styles. We are not aware of any such tool currently available. The results of our work indicate that the CAD tool would be a fast and accurate one. Resorting to a CAD tool is inevitable for delay estimation and optimization at the system level. System level optimization requires special considerations, because optimizing one path of the system may increase the delay in another one.

The applications of this work may be extended to cover additional asynchronous and synchronous primitives. The most important elements on the priority list include latches and flip-flops [28, 106]. This opens the door for delay estimation and optimization of sequential synchronous and asynchronous circuits. For example, it would then be possible to optimize a complete micropipeline structure, which consists of a control circuit and a data path. Another interesting application for this work is the area of dynamic CMOS circuits.

The delay models developed in this thesis may be used to optimize mixed logic-style digital CMOS circuits under multi-constraints on the delay, energy, area, and power supply voltage. This requires an investigation into finding or developing efficient non-linear optimization algorithms for this purpose. The literature seems helpful in this regard [25, 45, 68, 79].

Another interesting work would be to relate the parameters of our MOSFET current model to the basic technology factors such as the carrier mobilities and doping densities. An investigation into progressive transistor sizing of cascaded MOS transistors also seems interesting [8, 88].

9.3 Publications That Arose from the Thesis

We have already published some of the early results of this work presented in chapters 2 through 5. The major results presented in chapters 6, 7, and 8, however, remain to be prepared for publication.

- M. Shams, M. Elmasry, "A Formulation for Quick Evaluation and Optimization of Digital CMOS Circuits," to appear in *IEEE International Symposium on Circuits and Systems*, ISCAS-99, June 1999.
- M. Shams, J. Ebergen, M. Elmasry, "Modeling and Comparing CMOS Implementations of the C-Element," *IEEE Transactions on VLSI Systems*, Special Issue on Low-Power Electronics and Design, pp 563–567, December 1998.
- M. Shams, J. Ebergen, M. Elmasry, "Asynchronous Circuits," in *Encyclopedia of Electrical and Electronics Engineering*, Editor: J. Webster, John Wiley, pp 716–725, March 1999.
- M. Shams, J. Ebergen, M. Elmasry, "Optimizing CMOS Implementations of the C-Element," in *IEEE International Conference on Computer Design*, ICCD-97, pp 700–705, October 1997.
- M. Shams, J. Ebergen, M. Elmasry, "Comparing CMOS Implementations of an Asynchronous Circuits Primitive: the C-Element," in *IEEE International Symposium on Low-Power Electronics and Design*, pp 93–96, August 1996.

Bibliography

[1] A. J. Al-Khalili, Y. Zhu, and D. Al-Khalili, "A module generator for optimized CMOS buffers," *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 1028–1046, Oct. 1990.

[2] K. v. Berkel, "Beware the isochronic fork," *Integration, the VLSI journal*, vol. 13, pp. 103–128, June 1992.

[3] K. v. Berkel, *Handshake Circuits: an Asynchronous Architecture for VLSI Programming*, vol. 5 of *International Series on Parallel Computation*. Cambridge University Press, 1993.

[4] K. v. Berkel, R. Burgess, J. Kessels, A. Peeters, M. Roncken, and F. Schalij, "A fully-asynchronous low-power error corrector for the DCC player," *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 1429–1439, Dec. 1994.

[5] K. v. Berkel, R. Burgess, J. Kessels, A. Peeters, M. Roncken, and F. Schalij, "A fully-asynchronous low-power error corrector for the DCC player," in *International Solid State Circuits Conference*, pp. 88–89, Feb. 1994.

[6] K. v. Berkel and M. Rem, "VLSI programming of asynchronous circuits for low power," in *Asynchronous Digital Circuit Design* (G. Birtwistle and A. Davis, eds.), Workshops in Computing, pp. 152–210, Springer-Verlag, 1995.

[7] L. Bisdounis, S. Nikolaidis, and O. Koufopavlou, "Analytical transient response and propagation delay evaluation of the CMOS inverter for short-channel devices," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 302–306, Feb. 1998.

[8] S. S. Bizzan, G. A. Jullien, and W. C. Miller, "Analytical approach to siaing nFET chians," *Electronics Letters*, vol. 28, pp. 1334–1335, July 1992.

[9] E. Brunvand and R. F. Sproull, "Translating concurrent programs into delay-insensitive circuits," in *Proc. International Conf. Computer-Aided Design (ICCAD)*, pp. 262–265, IEEE Computer Society Press, Nov. 1989.

[10] J. A. Brzozowski and C.-J. H. Seger, *Asynchronous Circuits.* Springer-Verlag, 1995.

[11] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 473–484, Apr. 1992.

[12] T. J. Chaney and C. E. Molnar, "Anomalous behavior of synchronizer and arbiter circuits," *IEEE Transactions on Computers*, vol. C-22, pp. 421–422, Apr. 1973.

[13] K. Chen and C. Hu, "Performance and V_{dd} scaling in deep submicron CMOS," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 1586–1589, Oct. 1998.

[14] K. Chen, C. Hu, P. Fang, M. R. Lin, and D. L. Wollesen, "Predicting CMOS speed with gate oxide and volatge scaling and interconnect loading effects," *IEEE Transactions on Electron Devices*, vol. 44, pp. 1951–1957, Nov. 1997.

[15] K. Choi, K. Lee, and J.-W. Kang, "A self-timed divider using RSD number system," in *Proc. International Conf. Computer Design (ICCD)*, IEEE Computer Society Press, Oct. 1994.

[16] K. Chu and D. Pulfrey, "Design procedures for differential cascode voltage switch circuits," *IEEE Journal of Solid-State Circuits*, vol. 21, pp. 1082–1087, Dec. 1986.

[17] K. Chu and D. Pulfrey, "A comparison of CMOS circuit techniques: Differential cascode voltage switch logic versus conventional logic," *IEEE Journal of Solid-State Circuits*, vol. 22, pp. 528–532, Aug. 1987.

[18] T.-A. Chu, *Synthesis of Self-Timed VLSI Circuits from Graph-Theoretic Specifications.* PhD thesis, MIT Laboratory for Computer Science, June 1987.

[19] M. A. Cirit, "Transistor sizing in CMOS circuits," in *Proc. ACM/IEEE Design Automation Conference*, pp. 121–124, ACM, 1987.

[20] W. A. Clark, "Macromodular computer systems," in *AFIPS Conference Proceedings: 1967 Spring Joint Computer Conference*, vol. 30, (Atlantic City, NJ), pp. 335–336, Academic Press, 1967.

[21] W. A. Clark and C. E. Molnar, "Macromodular computer systems," in *Computers in Biomedical Research* (R. W. Stacy and B. D. Waxman, eds.), vol. IV, ch. 3, pp. 45–85, Academic Press, 1974.

[22] B. Coates, A. Davis, and K. Stevens, "The Post Office experience: Designing a large asynchronous chip," *Integration, the VLSI journal*, vol. 15, pp. 341–366, Oct. 1993.

[23] P. Cocchini, G. Piccinini, and M. Zamboni, "A comprehensive submicrometer MOST delay model and its application to CMOS buffer," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 1254–1262, Aug. 1997.

[24] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the lambert W function," Tech. Rep. CS-93-03, University of Waterloo, Mar. 1993.

[25] O. Coudert, "Gate sizing for constrained delay/power/area optimization," *IEEE Transactions on VLSI Systems*, vol. 5, pp. 465–472, Dec. 1997.

[26] A. Davis, "Synthesizing asynchronous circuits: Practice and experience," in *Asynchronous Digital Circuit Design* (G. Birtwistle and A. Davis, eds.), Workshops in Computing, pp. 104–150, Springer-Verlag, 1995.

[27] A. Davis and S. M. Nowick, "Asynchronous circuit design: Motivation, background, and methods," in *Asynchronous Digital Circuit Design* (G. Birtwistle and A. Davis, eds.), Workshops in Computing, pp. 1–49, Springer-Verlag, 1995.

[28] P. Day and J. V. Woods, "Investigation into micropipeline latch design styles," *IEEE Transactions on VLSI Systems*, vol. 3, pp. 264–272, June 1995.

[29] D. L. Dill, *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. ACM Distinguished Dissertations, MIT Press, 1989.

[30] D. W. Dobberpuhl and et. al., "A 200-mhz 64-b dual-issue cmos microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 1555–1568, Nov. 1992.

[31] S. Dutta, S. S. Mahant, and S. L. Lusky, "A comprehensive delay model for CMOS inverters," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 864–871, Aug. 1995.

[32] J. Ebergen and S. Gingras, "A verifier for network decompositions of command-based specifications," in *Proc. Hawaii International Conf. System Sciences*, vol. I, IEEE Computer Society Press, Jan. 1993.

[33] J. C. Ebergen, *Translating Programs into Delay-Insensitive Circuits*, vol. 56 of *CWI Tract*. Centre for Mathematics and Computer Science, 1989.

[34] J. C. Ebergen, J. Segers, and I. Benko, "Parallel program and asynchronous circuit design," in *Asynchronous Digital Circuit Design* (G. Birtwistle and A. Davis, eds.), Workshops in Computing, pp. 51–103, Springer-Verlag, 1995.

[35] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifires," *Journal of Applied Physics*, vol. 19, pp. 55–63, Jan. 1948.

[36] J. P. Fishburn and A. E. Dunlop, "Tilos: A posynomial programming approach to transistor sizing," in *Proc. International Conf. Computer-Aided Design (ICCAD)*, pp. 326–328, IEEE Computer Society Press, Nov. 1985.

[37] S. Furber, "Computing without clocks: Micropipelining the ARM processor," in *Asynchronous Digital Circuit Design* (G. Birtwistle and A. Davis, eds.), Workshops in Computing, pp. 211–262, Springer-Verlag, 1995.

[38] J. Garside, "The Asynchronous Logic Homepage." `http://www.cs.man.ac.uk/amulet/async/`.

[39] S. Hauck, "Asynchronous design methodologies: An overview," *Proceedings of the IEEE*, vol. 83, Jan. 1995.

[40] N. Hedenstierna and K. O. Jeppson, "CMOS circuit speed and buffer optimization," *IEEE Transactions on Computer-Aided Design*, vol. CAD-6, pp. 270–281, Mar. 1987.

[41] K. S. Hedlund, "Aesop: A tool for automated transistor sizing," in *Proc. ACM/IEEE Design Automation Conference*, pp. 114–120, ACM, 1987.

[42] L. G. Heller, W. R. Griffin, J. W. Davis, and N. G. Thoma, "Cascode voltage switching logic: A differential CMOS logic family," in *International Solid State Circuits Conference*, pp. 16–17, 1984.

[43] C. A. R. Hoare, *Communicating Sequential Processes.* Prentice-Hall, 1985.

[44] L. A. Hollaar, "Direct implementation of asynchronous control units," *IEEE Transactions on Computers*, vol. C-31, pp. 1133–1141, Dec. 1982.

[45] B. Hoppe, G. Neuendorf, D. Schmitt-Landsiedel, and W. Specks, "Optimization of high-speed cmos logic circuits with analytical models for signal delay, chip area, and dynamic power dissipation," *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 236–247, Mar. 1990.

[46] M. Horowitz, "Timing models for mos pass networks," in *Proc. International Symposium on Circuits and Systems*, pp. 198–201, IEEE, 1983.

[47] C. Hu, "Device and technology impact on low power electronics," in *Low-Power Design Methodologies* (J. M. Rabay and M. Pedram, eds.), pp. 317–322, Kluwer Academic Publishers, 1996.

[48] D. A. Huffman, "The synthesis of sequential switching circuits," *IRE Transactions on Electronic Computers*, vol. 257, no. 3 & 4, 1954.

[49] K. O. Jeppson, "Modeling the influence of the transistor gain ratio and the input-to-output coupling capacitance on the CMOS inverter delay," *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 646–654, June 1994.

[50] M. B. Josephs and J. T. Udding, "An overview of DI algebra," in *Proc. Hawaii International Conf. System Sciences*, vol. I, IEEE Computer Society Press, Jan. 1993.

[51] S.-M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits Analysis and Design.* McGraw-Hill, 1996.

[52] U. Ko, P. T. Balsara, and W. Lee, "Low-power design techniques for high-performance CMOS adders," *IEEE Transactions on VLSI Systems*, vol. 3, pp. 327–332, June 1995.

[53] N. C. Li, G. L. Haviland, and A. A. Tuszynski, "CMOS tapered buffer," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 1005–1008, Aug. 1990.

[54] L. R. Marino, "General theory of metastable operation," *IEEE Transactions on Computers*, vol. C-30, pp. 107–115, Feb. 1981.

[55] A. J. Martin, "Formal program transformations for VLSI circuit synthesis," in *Formal Development of Programs and Proofs* (E. W. Dijkstra, ed.), UT Year of Programming Series, pp. 59–80, Addison-Wesley, 1989.

[56] A. J. Martin, "Programming in VLSI: From communicating processes to delay-insensitive circuits," in *Developments in Concurrency and Communication* (C. A. R. Hoare, ed.), UT Year of Programming Series, pp. 1–64, Addison-Wesley, 1990.

[57] A. J. Martin, S. M. Burns, T. K. Lee, D. Borkovic, and P. J. Hazewindus, "The design of an asynchronous microprocessor," in *Advanced Research in VLSI: Proceedings of the Decennial Caltech Conference on VLSI* (C. L. Seitz, ed.), pp. 351–373, MIT Press, 1989.

[58] E. J. McCluskey, "Fundamental mode and pulse mode sequential circuits," in *Proc. of IFIP Congress 62*, pp. 725–730, North-Holland, 1963.

[59] R. Mehrotra, M. Pedram, and X. Wu, "Comparison betwen nMOS pass transistor logic styles vs. cmos complementary cells," in *Proc. International Conf. Computer Design (ICCD)*, pp. 130–135, IEEE Computer Society Press, Oct. 1997.

[60] T. H.-Y. Meng, *Asynchronous Design for Digital Signal Processing Architectures*. PhD thesis, UC Berkely, 1988.

[61] T. H.-Y. Meng, R. W. Brodersen, and D. G. Messerschmitt, "Automatic synthesis of asynchronous circuits from high-level specifications," *IEEE Transactions on Computer-Aided Design*, vol. 8, pp. 1185–1205, Nov. 1989.

[62] G. Merkel, J. Borel, and N. Z. Cupcea, "An accurate large signal MOS transistor model for use in computer-aided design," *IEEE Transactions on Electron Devices*, 1972.

[63] R. E. Miller, *Sequential Circuits and Machines*, vol. 2 of *Switching Theory*. John Wiley & Sons, 1965.

[64] C. E. Molnar, I. W. Jones, B. Coates, and J. Lexau, "A FIFO ring oscillator performance experiment," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE Computer Society Press, Apr. 1997.

[65] D. E. Muller and W. S. Bartky, "A theory of asynchronous circuits," in *Proceedings of an International Symposium on the Theory of Switching*, pp. 204–243, Harvard University Press, Apr. 1959.

[66] A. Nabavi-Lishi and N. C. Rumin, "Inverter models of CMOS gates for supply current and delay evaluation," *IEEE Transactions on Computer-Aided Design*, vol. 13, pp. 1271–1279, Oct. 1994.

[67] S. M. Nowick and D. L. Dill, "Automatic synthesis of locally-clocked asynchronous state machines," in *Proc. International Conf. Computer-Aided Design (ICCAD)*, pp. 318–321, IEEE Computer Society Press, Nov. 1991.

[68] P. Pant, V. K. De, and A. Chatterjee, "Simultaneous power supply, threshold voltage, and transistor size optimization for low-power operation of CMOS circuits," *IEEE Transactions on VLSI Systems*, vol. 6, pp. 538–545, Dec. 1998.

[69] J. H. Pasternak and C. A. T. Salama, "Design of submicron CMOS differential pass-transistor logic circuits," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 1249–1258, Sept. 1991.

[70] A. Peeters, "The 'Asynchronous' Bibliography (BibTeX) database file `async.bib`." `ftp://ftp.win.tue.nl/pub/tex/async.bib.Z`. Corresponding e-mail address: `async-bib@win.tue.nl`.

[71] J. L. Peterson, "Petri nets," *Computing Surveys*, vol. 9, pp. 223–252, Sept. 1977.

[72] J. M. Rabaey, *Digital Integrated Circuits*. Prentice-Hall, 1996.

[73] M. Renaudin and B. E. Hassan, "The design of fast asynchronous adder structures and their implementation using DCVS logic," in *Proc. International Symposium on Circuits and Systems*, 1994.

[74] J. Rubinstein, P. Penfield, and M. Horowitz, "Signal delay in RC tree networks," *IEEE Transactions on Computer-Aided Design*, vol. 2, pp. 202–211, July 1983.

[75] A. E. Ruehli, P. K. Wolff, and G. Goertzel, "Analytical power/timing opimization technique for digital system," in *Proc. ACM/IEEE Design Automation Conference*, pp. 142–146, ACM, 1977.

[76] T. Sakurai and A. R. Newton, "Alpha-power law mosfet model and its applications to CMOS inverter delay and other formulas," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 584–594, Apr. 1990.

[77] T. Sakurai and A. R. Newton, "Delay analysis of series-connected mosfet circuits," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 122–131, Feb. 1991.

[78] T. Sakurai and A. R. Newton, "A simple mosfet model for circuit analysis," *IEEE Transactions on Electron Devices*, vol. 38, pp. 887–894, Apr. 1991.

[79] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S.-M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *IEEE Transactions on Computer-Aided Design*, vol. 12, pp. 1621–1634, Nov. 1993.

[80] C. L. Seitz, "System timing," in *Introduction to VLSI Systems* (C. A. Mead and L. A. Conway, eds.), ch. 7, Addison-Wesley, 1980.

[81] M. Shams, J. Ebergen, and M. Elmasry, "A comparison of CMOS implementations of an asynchronous circuits primitive: the C-element," in *International Symposium on Low Power Electronics and Design*, pp. 93–96, Aug. 1996.

[82] M. Shams, J. C. Ebergen, and M. I. Elmasry, "Optimizing CMOS implementations of C-element," in *Proc. International Conf. Computer Design (ICCD)*, pp. 700–705, Oct. 1997.

[83] M. Shams, J. C. Ebergen, and M. I. Elmasry, "Modeling and comparing CMOS implementations of the C-element," *IEEE Transactions on VLSI Systems*, vol. 6, pp. 563–567, Dec. 1998.

[84] M. Shams, J. C. Ebergen, and M. I. Elmasry, "Asynchronous circuits," in *Encyclopedia of Electrical and Electronics Engineering* (J. Webster, ed.), John Wiley & Sons, 1999.

[85] M. Shams and M. I. Elmasry, "A formulation for quick evaluation and optimization of digital CMOS circuits," in *Proc. International Symposium on Circuits and Systems*, p. (To appear), May 1999.

[86] B. J. Sheu, D. L. Scharfetter, P.-K. Ko, and M.-C. Jeng, "Bsim: Berkeley short-channel igfet model for mos transistors," *IEEE Journal of Solid-State Circuits*, vol. sc-22, pp. 558–563, Aug. 1987.

[87] W. Shockley, "A unipolar field effect transistor," *Proc. IRE*, vol. 40, pp. 1365–1376, Nov. 1952.

[88] M. Shoji, "FET scaling in domino CMOS gates," *IEEE Journal of Solid-State Circuits*, vol. 20, pp. 1067–1071, Oct. 1985.

[89] M. Shoji, *CMOS Digital Circuit Technology.* Prentice-Hall, 1988.

[90] J.-M. Shyu, A. Sangiovanni-Vincentelli, J. P. Fishburn, and A. E. Dunlop, "Optimization-based transistor sizing," *IEEE Journal of Solid-State Circuits*, vol. 23, pp. 400–409, Apr. 1988.

[91] R. F. Sproull and I. E. Sutherland, *Asynchronous Systems.* Palo Alto: Sutherland, Sproull and Associates, 1986. Vol. I: Introduction, Vol. II: Logical effort and asynchronous modules, Vol. III: Case studies.

[92] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, pp. 720–738, June 1989.

[93] I. E. Sutherland and R. F. Sproull, "Logical effort: Desiging for speed on the back of an envelope," in *Advanced Research in VLSI*, pp. 1–16, Sept. 1991.

[94] M. Suzuki, N. Ohkubo, T. Shinbo, T. Yamanaki, A. Shimizu, K. Sasaki, and Y. Nakagome, "A 1.5-ns 32-b CMOS ALU in double pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 28, pp. 1145–1151, Nov. 1993.

[95] K.-Y. Toh, P.-K. Ko, and R. G. Meyer, "An engineering model for short-channel MOS devices," *IEEE Journal of Solid-State Circuits*, vol. 23, pp. 950–958, Aug. 1988.

[96] J. T. Udding, *Classification and Composition of Delay-Insensitive Circuits*. PhD thesis, Dept. of Math. and C.S., Eindhoven Univ. of Technology, 1984.

[97] S. H. Unger, *Asynchronous Sequential Switching Circuits*. New York: Wiley-Interscience, John Wiley & Sons, Inc., 1969.

[98] S. R. Vemuru and A. R. Thorbjornsen, "Variable-taper CMOS buffer," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 1265–1269, Sept. 1991.

[99] T. Verhoeff, *A Theory of Delay-Insensitive Systems*. PhD thesis, Dept. of Math. and C.S., Eindhoven Univ. of Technology, May 1994.

[100] Z. Wang, G. A. Jullien, W. C. Miller, J. Wang, and S. S. Bizzan, "Fast adders using enhanced multiple-output domino logic," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 206–214, Feb. 1997.

[101] S. Weber, B. Bloom, and G. Brown, "Compiling Joy to silicon," in *Proceedings of Brown/MIT Conference on Advanced Research in VLSI and Parallel Systems* (T. Knight and J. Savage, eds.), pp. 79–98, MIT Press, Mar. 1992.

[102] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*. Addison-Wesley, 1994.

[103] T. E. Williams and M. A. Horowitz, "A zero-overhead self-timed 160ns 54b CMOS divider," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 1651–1661, Nov. 1991.

[104] L. T. Wurtz, "An efficient procedure for domino CMOS logic," *IEEE Journal of Solid-State Circuits*, vol. 28, pp. 979–982, Sept. 1993.

[105] K. Yano, T. Yamanka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8-ns CMOS 16×16-b multiplier using complementary pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 388–395, Apr. 1990.

[106] J. Yuan and C. Svensson, "New single-clock CMOS latches and flipflops with improved speed and power savings," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 62–69, Jan. 1997.

[107] K. Y. Yun, P. A. Beerel, and J. Arceo, "High-performance asynchronous pipeline circuits," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE Computer Society Press, Mar. 1996.

[108] K. Y. Yun and D. L. Dill, "Automatic synthesis of 3D asynchronous state machines," in *Proc. International Conf. Computer-Aided Design (ICCAD)*, pp. 576–580, IEEE Computer Society Press, Nov. 1992.

[109] R. Zimmermann and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 1097–1090, July 1997.