DESIGN AND OPTIMIZATION OF MOS CURRENT-MODE LOGIC CIRCUITS

by

Osman Bakri Musa Abdulkarim

A thesis
Submitted to Carleton University
in fulfillment of the requirements for the degree of
MASTER OF APPLIED SCIENCE

Carleton University, Ottawa, Canada

Your file Votre référence
ISBN: 978-0-494-23323-8
Our file Notre référence
ISBN: 978-0-494-23323-8

Abstract

MOS Current-Mode Logic (MCML) is a low-noise alternative to CMOS logic for mixed-signal applications. If properly designed, MCML circuits can achieve significant power reduction compared to their CMOS counterparts at frequencies as low as 300MHz. MCML logic has, however, fallen out of favor because of its high design complexity and the lack of automated design and optimization tools.

In this work, simple and accurate propagation-delay models for MCML circuits, that are suitable for mathematical programming, have been developed and verified. The models are based on a modified version of the differential-pair MCML universal gate. The modified universal-gate performance has been compared to the standard universal gate topology. Simulations have shown that the modified universal gate has better DC symmetry, lower switching noise and higher operation frequency.

When compared to simulation results, the proposed delay model has an average error of about 3.7% and a maximum error of 12%. The proposed model has significantly reduced the complexity of the MCML universal-gate optimization problem. When compared to the most recent work, the proposed model has reduced the number of optimization variables from $7N+1$ to $N+1$, where N is the number of logic gates in the optimization problem. The optimization problem constraints have also been reduced from $5N$ to only one constraint. The model has been successfully implemented to optimize a 4-bit ripple-carry adder and an 8-bit decoder. Numerical tests show that the proposed optimization program produces the global solution regardless of the initial guess.

Acknowledgements

" Proclaim! (or Read!) in the name of thy Lord and Cherisher Who created. Created man out of a (mere) Leech-Like clot; Proclaim! And thy Lord is Most Bountiful." Quran (96:1-3)

This work would have not been possible without the support of many. First, it is my duty to thank God Almighty for making the completion of this research possible. Next, I would like to express my gratitude to my parents for their continued support.

I would like to thank my supervisor Dr. Maitham Shams for the invaluable guidance and support. Many thanks to the faculty and staff of the Department of Electronics at Carleton University. I would like to mention in particular Dr. Garry Tarr for his encouragement to pursue graduate studies, Dr. John Knight for his valuable feedback and Dr. Calvin Plett for his help and his work ethic which inspired me and many others.

I would like to extend my appreciation to Ziad El Khatib and Atif Shamim for providing mentorship and advice, Duha Jakhabanji and the VLSI group at Carleton University for their valuable feedback and Dr. Mohamed Abdeen for his encouragement. Last but not least, I would like to thank my friends in Ottawa and wish them all the best in life.

To my parents

Table of Contents

List of Tables

List of Figures

List of Symbols

α Velocity saturation index

χ Data switching activity

ΔV MCML voltage swing

ΔV_{min} Minimum input voltage swing to completely switch MCML tail-current

λ channel length modulation parameter

τ_{pHL} High to Low propagation delay

τ_{pLH} Low to High propagation delay

A_v Small-signal gain

C Total capacitance seen at the output including the load and driver intrinsic capacitance

C_j Junction capacitance per unit area

C_{db} Drain to bulk junction capacitance

C_{int} Logic gate's intrinsic capacitance

C_{jsw} Junction side wall capacitance per unit perimeter

C_L Output load capacitance

C_{ox} Gate oxide capacitance per unit area

D Low-current region propagation delay

D Propagation delay

D_H High-current region propagation delay

f_c Clock frequency

f_{unity} Unity gain frequency

g_m Transconductance

$Gain_{DC}$ Large-signal gain

I_{off} MCML off branch current

I_{SS} MCML tail-current

j fan-out number

k_n process transconductance parameter

L MOSFET channel length

L_{min} Minimum transistor length allowable by the technology

N Number of logic gates in the design

NM Noise Margin

P_{max} Maximum Power Dissipation

S A feasible set

V_T MOSFET Threshold Voltage

V_{DD} Supply Voltage

V_{DS} Drain-to-source voltage

V_d Voltage at node d in the MCML universal gate

V_{GS} Gate to source voltage

V_H Logic High voltage

V_L Logic Low voltage

V_x Voltage at node x in the MCML universal gate

VSR Voltage Swing Ratio

W MOSFET channel width

W_{min} Minimum transistor width allowable by the technology

x^* Optimization problem solution

z Drain region lateral extension

CORDIC Coordinate Rotation Digital Computer

DSP Digital Signal Processing

DUT Device Under Test

DyCML Dynamic Current-Mode Logic

ITRS International Technology Road map for Semiconductors

MCML MOS Current-Mode Logic

MUX Multiplexer

PFSCL Positive Feedback Source-Coupled Logic

RF Radio Frequency

SoC System-on-Chip

UG Universal Gate

Chapter 1

Introduction

1.1 Thesis Motivation

A major problem that hinders the advancement towards complete System-on-Chip integration is the switching noise that is generated by digital circuitry. The International Technology Road map for Semiconductors (ITRS) has stated in its 2005 report on Radio Frequency and Analog /Mixed-Signal Technologies for Wireless Communications that "As the integration density and the operation frequency increase, protection of noise sensitive analog circuits from noisy digital circuits will become increasingly difficult" [1].

MOS Current-Mode Logic (MCML) is a promising alternative to conventional CMOS for mixed-signal applications. Many efforts were exhausted to realize the potential of MCML [2] [3] [4] [5] [6] [7] [8]. Even though MCML has been shown to dissipate less power than CMOS at operation frequencies of more than 300 MHz [7], designers were reluctant to exchange MCML for CMOS. The high complexity of MCML and the lack of automation tools made it impossible to produce robust and power efficient designs while maintaining low cost and reasonable time-to-market. To entertain this problem, a number of attempts to automate MCML design and optimization were carried with varying success [3] [5]. In conventional CMOS logic, robustness is an inherent characteristic and for most applications, the major objectives in any optimization procedure are the delay, area and power dissipation, which can be expressed in simple forms in terms of transistor dimensions and process parameters. The case for MCML design is different, since the gate

has an analog topology, where robustness is hard to achieve and requires imposing tight constraints. The constraints include internal node voltages and currents, which do not only degrade the accuracy of any model, but also complicate the model and put a burden on the optimization tools. This is as far as equation-based tools are concerned. Simulation-based tools will not fare any better as this type of optimization is only suitable for small scale designs [9].

On the same topic of robustness, the MCML universal gate, which is a popular MCML topology due to its versatility and relative small size, has an asymmetric topology. The asymmetry of the gate degrades the overall circuit performance and increases the complexity of any potential MCML model for use in design automation.

1.2 Thesis Objectives

The purpose of this work is to develop a feasible automation method that would allow designers to conveniently implement designs in MCML with time-to-market comparable to conventional CMOS. This allows the designers to explore different implementation options, and thus, accommodate different performance requirements.

1.3 Thesis Organization

In the next chapter, MCML is introduced. The discussion includes the general operation, performance requirements, problems associated with the logic and a brief review of the various MCML topologies.

Chapter 3 discusses the different approaches in VLSI optimization and an introduction to the fundamentals of numerical optimization. The chapter also defines some terminologies and metrics used in mathematical optimization. In Chapter 4, the universal gate operation and performance are assessed to develop a better understanding of the logic gate. A modified topology of the standard universal gate is then examined and compared to the standard universal gate in terms of robustness and performance.

Chapter 5 objectives are to develop and verify a simplified delay model by exploiting the symmetry and the performance constraints of MCML gates. In Chapter 6, the delay model developed in Chapter 5 is modified to be used in an efficient mathematical program suitable for the automatic design and optimization of large MCML designs. The program is constructed and tested in a design with 144 transistors. Chapter 7 provides a conclusion of the thesis, suggestions for improvement and ideas for future work.

Chapter 2

Background and Theory

2.1 MCML Basic Operation

MOS Current-Mode Logic is a digital implementation of the differential amplifier. As illustrated in Figure 2.1, the logic is realized by completely switching the current from one branch to the other. In contrast to the analog differential amplifier, MCML is intended to work in the nonlinear region.

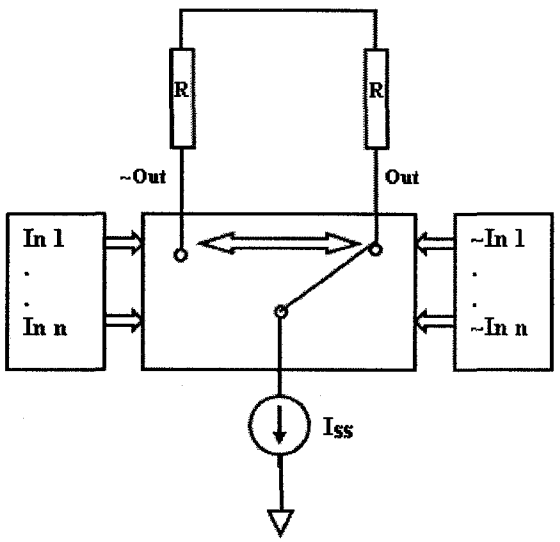

Figure 2.1: Basic MCML operation

The circuit is composed of a logic realization network, usually a set of differential-pair or differential-pairs organized in a certain configuration to steer the current in one branch and switch off the other based on the input combination applied. The output voltage levels are $V_H = V_{DD}$ at logic High and $V_L = V_{DD} - \Delta V$ at logic Low where $\Delta V = I_{ss} \times R$ is the logic swing. The load resistance can be replaced by an active load such as a PMOS transistor operating in the linear region. The tail-current I_{ss} is controlled by a transistor operating in the saturation mode.

2.2 MCML Advantages

All MCML advantages are attributed to its differential nature: Firstly, a differential topology has high immunity to common mode noise. Secondly, differential signaling doubles the effective voltage swing which has a direct relationship with the noise margin. The improvement in noise immunity and noise margin allows designers to trade excess noise margin for voltage swing. Reducing the voltage swing directly improves the delay according to

$$D = \frac{C\Delta V}{I_{SS}} \tag{2.2.1}$$

where C is the capacitance seen at the output. A major source of switching noise is the sudden and drastic current change in the supply lines causing effects such as ground bounce and charge injection into the substrate. Contrary to CMOS, the current sink in MCML gates provides a steady current regardless of the switching activity making MCML a mixed-signal environment friendly alternative.

2.3 MCML Disadvantages

Ironically, MCML's most important advantage happens to be its worst disadvantage. The static DC current which is responsible for the quiet operation of MCML causes the gate to bleed excess power even when the gate is idle. The picture is still bright, however. The only source of power dissipation in MCML gates is the static power and is expressed as

$P_{MCML} = I_{ss} \times V_{DD}$. CMOS, on the other hand, dissipates static power, due to charge leakage, and dynamic power during switching. As the feature sizes become smaller, the static leakage becomes more problematic. The dynamic power dissipation is due to the charging and discharging of the output capacitance. The dynamic power dissipation in CMOS is expressed as

$$P_{CMOS} = \chi f_c C_L V_{DD}^2 \tag{2.3.1}$$

where χ is the switching activity, f_c is the clock frequency and C_L is the capacitance seen at the output node. The power dissipation is directly proportional to the operation frequency. Figure 2.2 shows the power dissipation against frequency for MCML and conventional CMOS.

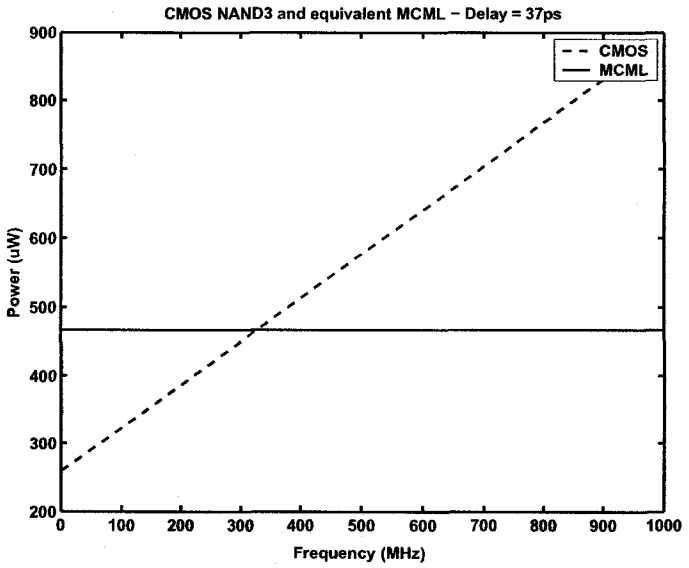

Figure 2.2: MCML and CMOS power dissipation versus frequency [7]

The conventional knowledge is that the frequency at which MCML gates dissipate less power than their CMOS counterparts is in the GHz range. It is reported in [7] that the MCML implementation of a pipelined CORDIC DSP unit shows 30% less power dissipation than its CMOS counterpart implementation at 300 MHz. Thus, by properly designing

every MCML gate in the design, the power dissipation may be cut considerably. The lack of automated design and optimization tools for MCML circuits, however, means that the optimization must be done using time consuming simulations. This is a very costly process, especially for large circuits.

2.4 MOSFET Models

2.4.1 Threshold Voltage

Threshold voltage marks the point at which strong inversion occurs in the MOSFET channel. In NMOS devices, strong inversion occurs when the surface of the p-type semiconductor under the gate oxide inverts to n-type due to high accumulation of electrons under the influence of the positive potential on the gate terminal. Threshold voltage is a function of the potential across the device terminals and several material related parameters. The threshold voltage can be expressed as [10]

$$V_T = V_{T0} + \gamma(\sqrt{|(-2)|\,\phi_F + V_{SB}} - \sqrt{|2\phi_F|}) \tag{2.4.1}$$

where V_{T0} is the threshold voltage when the source-bulk voltage $V_{SB} = 0$, ϕ_F is the Fermi potential and γ is a parameter that expresses the effect of the change of V_{SB} on V_T .

2.4.2 DC Current

The MOSFET DC current in the linear region of operation is expressed as [10]

$$I_D = 2k_n\frac{W}{L}\left[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}\right] \tag{2.4.2}$$

where W and L are the MOSFET channel width and length respectively, V_{GS} is the voltage drop between the gate and the source terminals and V_{DS} is the drain-to-source voltage. The parameter k_n is the process transconductance parameter and is given by

$$k_n = \frac{\mu_n c_{ox}}{2} \tag{2.4.3}$$

In the saturation region, the MOSFET static current is given by [10]

$$I_D = k_n \frac{W}{L}(V_{GS} - V_T)^2(1 + \lambda V_{DS}) \tag{2.4.4}$$

where λ is the channel length modulation parameter and is related to velocity saturation in short channel devices.

To account for velocity saturation and mobility degradation in today's short channel devices, the alpha power-law has been proposed in [11] as

$$I_D = k_n \frac{W}{L}(V_{GS} - V_T)^{\alpha}(1 + \lambda V_{DS}) \tag{2.4.5}$$

The power index α is called the velocity saturation index and is about 1.3 for most new short channel devices.

2.4.3 MOSFET Capacitance

There are two main groups of parasitic capacitance in the MOS transistor. The gate capacitance, which is due to the gate oxide, and the junction capacitances in the depletion region between the drain region and the bulk, between the source region and the bulk, and between the channel and the bulk. Table 2.1 lists the average distribution of the MOSFET gate parasitic capacitance in the different regions of operation. The capacitances are expressed as functions of the transistor gate width W, length L, gate oxide capacitance C_{ox} in F/m^2 and the gate to drain overlap capacitance in F/m [10].

Table 2.1: Average distribution of MOS gate capacitances for different operation regions [9]

Region	C_{GCB}	C_{GCS}	C_{GCD}	C_{GC}	C_G
Cutoff	$C_{ox}WL$	0	0	$C_{ox}WL$	$C_{ox}WL + 2C_oW$
Linear	0	$C_{ox}WL/2$	$C_{ox}WL/2$	$C_{ox}WL$	$C_{ox}WL + 2C_oW$
Saturation	0	$(2/3)C_{ox}WL$	0	$(2/3)C_{ox}WL$	$(2/3)C_{ox}WL + 2C_oW$

The junction capacitances which were not listed in Table 2.1 are the drain-to-bulk and

source-to-bulk junction capacitance. The drain to bulk capacitance is given by $C_{db} = C_j zW + C_{jsw}(2z + W)$ [10], where C_j is the junction capacitance per unit area, C_{jsw} is the side wall junction capacitance per unit perimeter and z is the drain extension. The source-to-bulk capacitance has a similar expression. The junction capacitance is a strong function of the bias across the depletion region.

2.5 Performance Metrics

In this section, all performance metrics will be quantified by assuming the square-law model from equation 2.4.4 is used.

2.5.1 Gate Delay

The logic gate propagation delay is defined as the duration between 50% of the input signal to 50% of the output. The propagation delay is usually estimated as

$$D = \frac{\tau_{pLH} + \tau_{pHL}}{2} \tag{2.5.1}$$

where τ_{pLH} and τ_{pHL} are the rising and falling propagation delays. The rising propagation delay τ_{pLH} is illustrated in Figure 2.3 .

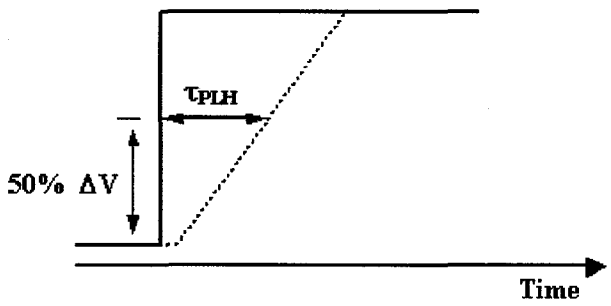

Figure 2.3: Definition of rising propagation delay

To calculate the propagation delays, digital networks are usually modeled as first-order RC circuits like the one shown in Figure 2.4 . When a step input is applied the circuit transient response is an exponential function given by [12]

$$V_{out} = V_{\infty}(1 - e^{-t/\tau}) \tag{2.5.2}$$

where V_{∞} is the output steady state voltage after a long time and τ is the circuit time constant RC. The time for the output to reach the 50% point is $t = \tau \times ln(2) = 0.69\tau$.

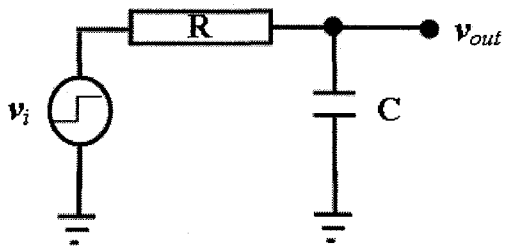

Figure 2.4: First-order RC network

2.5.2 AC Gain

The AC or small-signal gain is an important measure of the circuit behavior during transition and has a direct effect on the gate noise margin.

For the MCML inverter in Figure 2.5, the small-signal gain is [13]

$$A_v = g_m R \tag{2.5.3}$$

where g_m is the MOSFET small-signal transconductance. The small-signal gain must be high enough to produce a healthy noise margin. The noise margin of MCML circuits is discussed in Section 2.5.4 .

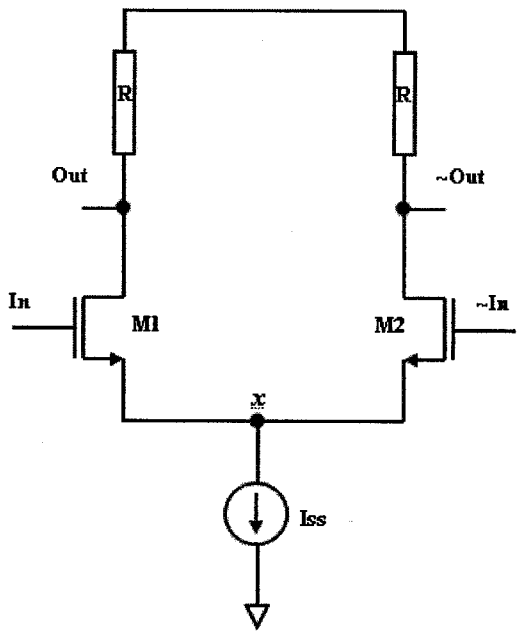

Figure 2.5: MCML inverter

2.5.3 DC Gain

The DC or large-signal gain gauges the gate's ability to propagate the logic value and preserve its voltage levels. In analog applications, decoupling capacitors are used to insure that the DC level of the output of one stage does not affect the performance of the next stages, that is as long as the small-signal behavior is consistent. In MCML, the output of one stage is directly connected to the input of the next.

Assuming a square law model is used for the saturation current, the DC gain of an MCML inverter is expressed as [12]

$$Gain_{DC} = R\sqrt{\frac{2I_{SS}k_nW}{L}} \tag{2.5.4}$$

Equation 2.5.4 suggests that the relationship between the gate's DC gain, delay and power dissipation is a competitive one. Increasing the gain requires either increasing the tail-current, the load resistance or both. Any of these options will also increase the voltage swing. This leads to the conclusion that the DC gain is directly proportional to the logic gate voltage swing. This is the reason why DC gain is designed to be slightly higher than one, that is, just enough to guarantee correct operation.

2.5.4 Noise Margin

The noise margin is an important metric of the logic gate robustness. Noise margin is the amount of noise that can be sustained at the input without causing the output to switch. For an MCML inverter, the noise margin is expressed as [14]

$$NM = \Delta V\left[1 - \frac{\sqrt{2}}{A_V}\sqrt{1 - \frac{1}{\sqrt{2}A_V}}\right] \tag{2.5.5}$$

A noise margin of $0.4\Delta V$ or higher is desirable and can be achieved by having a small-signal gain of 2 or higher. Figure 2.6 shows the noise margin as a percentage of the voltage swing versus the small-signal gain.

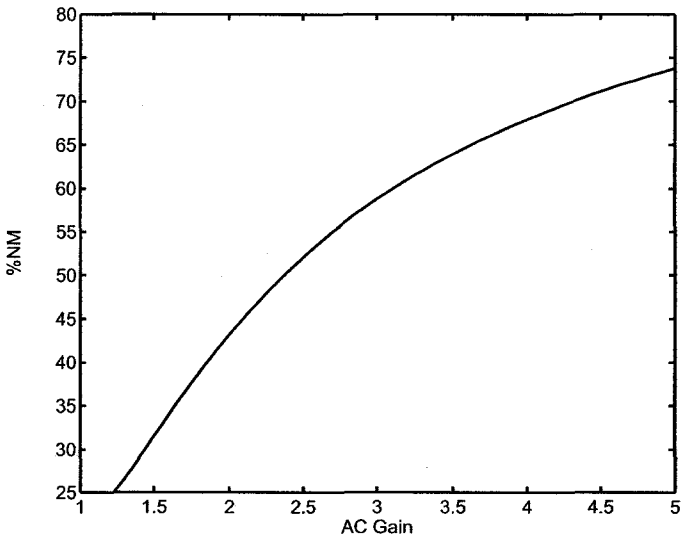

Figure 2.6: Noise Margin versus small-signal gain

2.5.5 Voltage Swing Ratio

Ideally, MCML NMOS transistors act as perfect switches steering all the DC current from one branch to the other. In reality, however, only a portion of the tail-current is switched back and forth leaving some current to flow in the 'off' branch. The Voltage Swing Ratio (VSR) is the ratio between the current in the 'on' branch to the tail-current I_{ss} . A 100% VSR ratio is desirable, but achieving such a high ratio requires large transistors widths. This in turn causes a substantial drop in speed. A small VSR on the other hand degrades the gate DC gain and causes the signal to fade as it passes through the stages.

A VSR of 95% guarantees healthy operation with reasonable circuit speed [3]. If the square-law model is used to estimate the transistor DC saturation current, then

$$VSR = \frac{k\frac{W}{L}(V_{GS} - V_T)^2(1 + \lambda V_{DS})}{I_{SS}} \tag{2.5.6}$$

where V_{DS} is the drain-source voltage of the differential-pair.

2.6 MCML Universal Gate Topologies

The MCML universal gate can realize all the basic logic operations (AND/NAND/OR/NOR) simply by interchanging the positive and negative inputs and outputs. For example, an AND gate can be made into a NAND gate by taking the negative output as the positive one and vice-versa. This section provides a quick survey of the common topologies used to construct a universal gate.

2.6.1 Differential-pair Universal Gate

Figure 2.7 shows the standard MCML universal gate, the most common universal gate topology. This circuit has a simple structure and has good delay and signal waveform in low to mid-range frequencies. This logic gate, however, has an asymmetric topology. The right branch in Figure 2.7 has only one transistor between the output node and the current sink, while the other two paths, M3-M1 and M4-M1, have two transistors in series. This is the main reason why this gate fails at high frequencies. To solve the symmetry problem, a transistor is added between the right branch output and M2. Chapter 4 is devoted to assessing the benefits of this modification to the differential-pair universal gate.

2.6.2 Non-Differential Universal Gate

The Non-differential universal gate is shown in Figure 2.8. This topology has close resemblance to Pseudo NMOS logic, except it is differential. The term Non-differential here is not intended to describe the signaling mode, but the topology of the gate. Conventionally, the complementary signals are connected to source-coupled transistors as in the differential-pair universal gate in Figure 2.7 and the MUX-based universal gate in Figure 2.9. The drawbacks of this topology include the asymmetry between the two branches and the low noise-immunity compared to the differential-pair topology [15]. The topology has less immunity to noise because of the asymmetry of the gate, which in tern degrades its Common-Mode-Rejection.

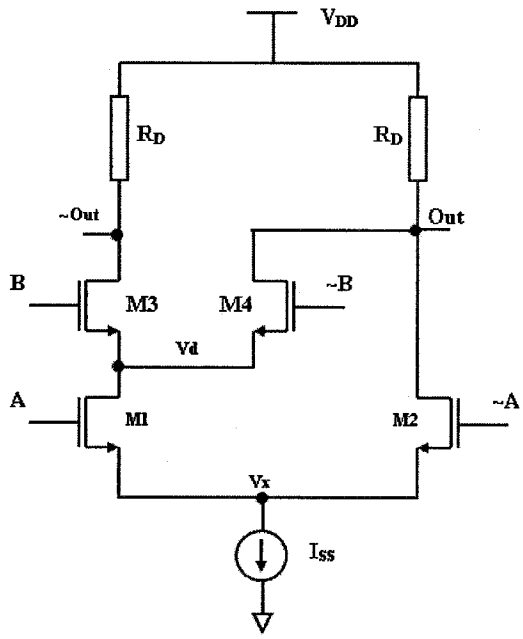

Figure 2.7: MCML Differential-pair universal gate

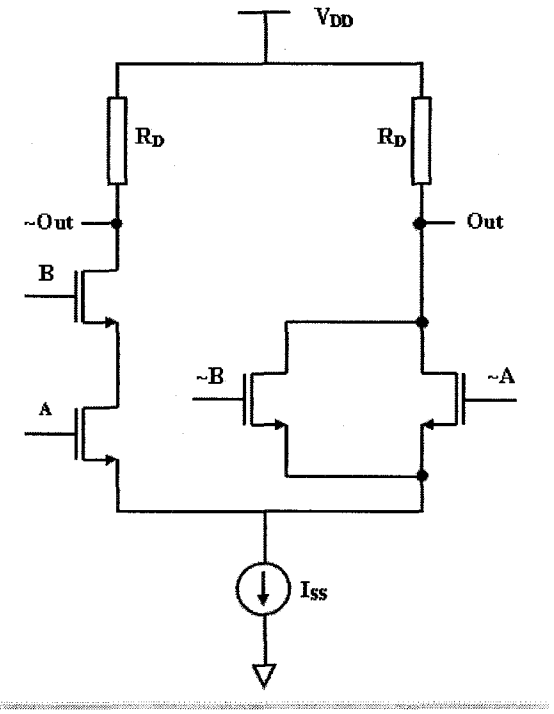

Figure 2.8: MCML Non-Differential-pair universal gate

2.6.3 MUX-based MCML Universal Gate

The MUX-based universal gate is shown in Figure 2.9 . Depending on the inputs configuration, this circuit, originally a multiplexer, can realize the universal gates functions. The MUX based universal gate has good symmetry but is prone to undesired leakage current through the path M5-M2 when the input signal slopes are finite. The gate is also slower than the previous ones because of the increased capacitance, both at the input and the output. The input drives two transistors instead of one. Chapter 4 provides a detailed comparison between the various universal gate topologies.

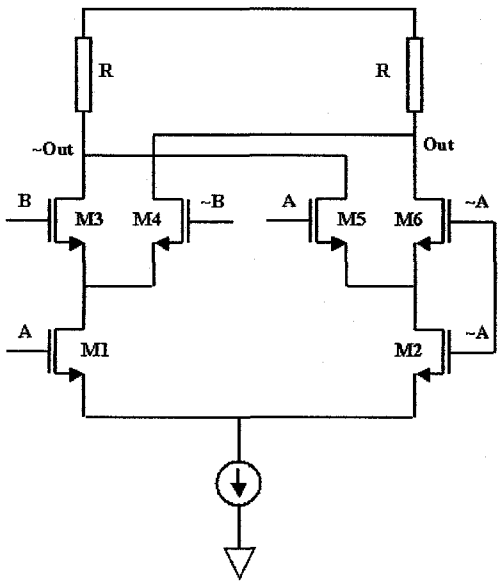

Figure 2.9: MUX-based MCML universal gate

2.7 Other MCML Topologies

In this section, some of the alternate topologies that were proposed to address specific issues in MCML are presented.

2.7.1 Dynamic CML

Dynamic Current-Mode Logic (DyCML) has been proposed in [4] to solve the static power dissipation problem in standard MCML. DyCML is a dynamic logic implementation of MCML. Figure 2.10 shows a DyCML gate. The operation of DyCML is as follows. In the pre-charge phase, the clock is low and thus, transistors M2, M3 and M4 are turned on. Thus, causing the outputs to rise to V_{DD} and capacitor C1 to discharge to ground. In the evaluation phase, the clock is high, and M2, M3 and M4 are turned off, and hence, closing the path from C1 to the ground. Depending on the input combination, chargees will flow from one of the output nodes to the capacitor C1 which acts as a current sink. The PMOS transistors, M5 and M6, act as a latch to preserve the logic value. It has been shown in [4] that DyCML can achieve significant power reduction over the standard MCML. Problems with this topology include increased design complexity and switching noise.

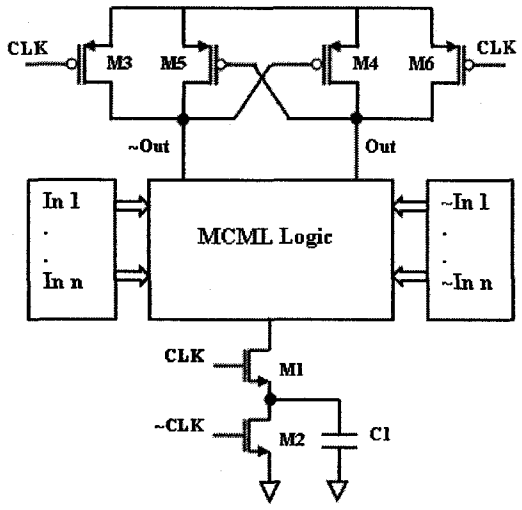

Figure 2.10: Dynamic CML style

2.7.2 Positive Feedback Source-Coupled Logic (PFSCL)

Figure 2.11 shows a PFSCL gate. PFSCL is realized by connecting a single-ended MCML gate in a feedback configuration.

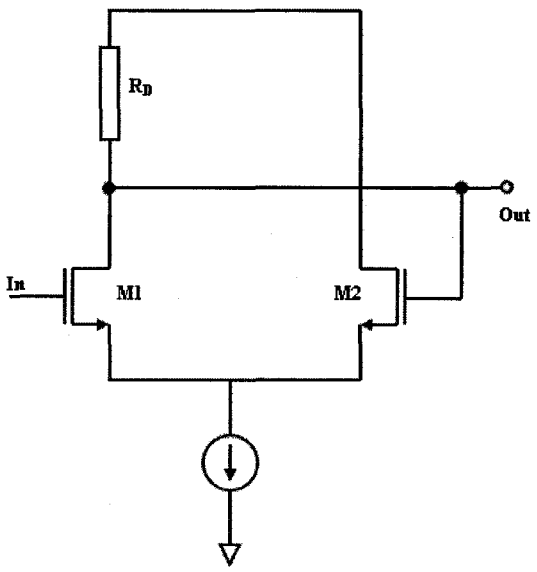

Figure 2.11: Positive Feedback Source-Coupled Logic

The gate is shown to achieve better delay for the same power dissipation when compared to standard MCML [16]. PFSCL is, however, operated as a single ended gate which degrades its noise immunity in comparison to the differential circuits.

Chapter 3

Optimization

3.1 VLSI Optimization

In this section, we briefly discuss some of the techniques used to optimize high performance digital circuits. One can achieve maximum reward by exploiting the capabilities of the optimization tools. This can be done by carefully crafting the circuit models and constraints, taking into account optimization theory and procedures. Depending on the constraints on accuracy and the available resources, one can choose between two approaches: An accurate model that is hard to solve, or a heuristic model with poor accuracy but is easier to solve. The first approach describes what is known as dynamic tuning.

Dynamic tuning uses a feedback loop with a time-domain circuit simulation in the middle. A typical dynamic tuning flow is shown in Figure 3.1 . The cycle starts with a simulation run with the tunable parameters set to initial values. The information from the simulation are then fed to a nonlinear optimizer which calculates the component values that will improve the circuit performance. The results from the optimizer are then used to reset the tunable parameters and the cycle is repeated. Convergence is measured by sufficient stationarity combined with sufficient feasibility [9]. The results are sufficiently stationary when the amount of change in the results from one iteration to the next is less than some specified value. Sufficient feasibility is achieved when all the design constraints are met within some tolerance margin. Dynamic optimization is accurate, but involves time consuming simulation runs. Careful specification of the test signals is also required.

JiffyTunes, a dynamic tuning tool developed by IBM, was reportedly able to tune logic circuits with 4218 tunable transistors [17].

Static tuning, on the other hand, is an equation-based approach. The circuits are represented by a set of relatively simple functions that express the circuit performance metrics. These functions are then used to construct a mathematical program. Based on the problem type and size, an appropriate solver is selected to optimize the circuit. Static tuning is faster than dynamic tuning, since it does not involve lengthy simulations, and hence, it is more suitable for optimizing large designs. The use of simpler models, however, reduces the accuracy of results. Statistics in the field of circuit optimization show that static tuning accuracy is within 24% [17]. The biggest task when applying static tuning is to develop a proper model with reasonable accuracy and yet as simple and compact as possible. Simplicity can be achieved by eliminating redundant relations and unnecessary constraints. Compactness may be possible by using algebraic manipulation and taking advantage of symmetric topologies and implicit relationships.

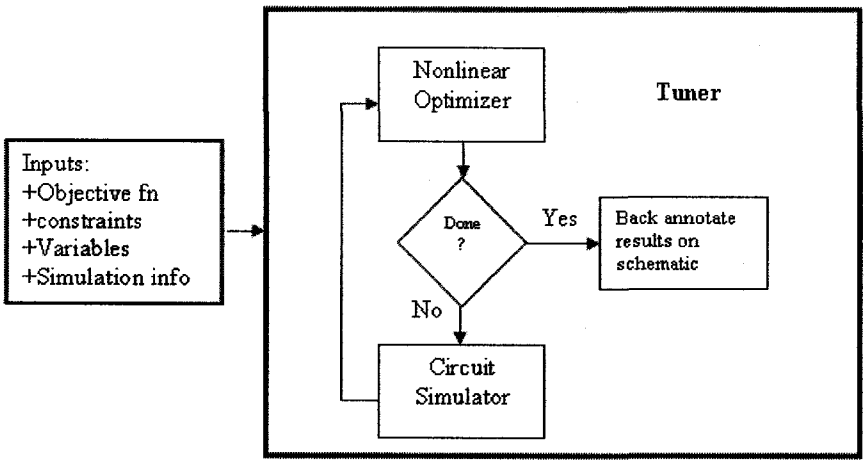

Figure 3.1: A template of a typical dynamic tuner

In some cases, the design is required to meet multiple objectives. Many techniques have been developed to accommodate multiple, often competing requirements. One technique is the use of a weighted sum objective function of the form [18]

$$Minimize : M = \sum W_i f_i(x_1, x_2, ...) \tag{3.1.1}$$

The weight W_i of the objective f_i is proportional to the design priorities. In a high-speed chip design, for example, more weight is assigned to the delay function than the area or power terms in the merit function M.

3.2 MCML Optimization

Despite the potential MCML holds, this logic style has not been exploited due to its design complexity and the lack of standard cell libraries and design automation tools. A few attempts have been made in the past to automate MCML design. In [19], a semi-automatic procedure was proposed. The procedure involves a closed simulation loop that is intended to find the proper transistor sizes and bias voltages that yields the required tail-current, voltage swing and noise margin for a single logic gate. The procedure, however, does not involve optimizing the gate specifications.

In [2], Adaptable MOS-Current-Mode Logic (AMCML) is used to implement a 15/16 dual-modulus prescaler for multi-band transceivers in the range of 800 MHz to 3 GHz. AMCML is an MCML implementation with a capability to adjust the tail-current and the voltage swing in real-time using control signals and feedback mechanisms. It is reported in [2] that 80% of the power required for operation at 3 GHz can be conserved when operating at 800 MHz.

The first attempts to use mathematical-programming to optimize MCML performance are due to [3] and [5]. The two papers proposed similar procedures but with slightly different design constraints. The procedure is to express the MCML gate performance and robustness metrics in terms of the tunable parameters. The performance metrics are mainly the gate propagation delay and the power dissipation. The robustness constraints

are required to insure correct propagation of the signal. Robustness constraints include the noise margin, small-signal gain, large-signal gain and the voltage swing ratio (VSR). The tunable parameters include the transistors sizes, the load resistance and the tail-current control voltage. These performance expressions and design constraints are then put in a mathematical-program form and solved by a mathematical solver. Equation 3.2.1 shows the mathematical program proposed in [5] for the differential-pair MCML universal gate.

$$
\begin{aligned}
&Minimize : Delay(w, I_{SS}, R) \\
&Subject : \\
&V_{DD} \times I_{SS} \leq P_{\max} \\
&\tfrac{I_{SS}}{2} - k\tfrac{W_1}{L}(V_{DD} - R\tfrac{I_{SS}}{2} - V_x - V_T)^{\alpha}(1 + \lambda(V_d - V_x)) = 0 \\
&\tfrac{I_{SS}}{2} - k\tfrac{W_2}{L}(V_{DD} - R\tfrac{I_{SS}}{2} - V_x - V_T)^{\alpha}(1 + \lambda(V_{DD} - R\tfrac{I_{SS}}{2} - V_x)) = 0 \\
&\tfrac{I_{SS}}{2} - k\tfrac{W_3}{L}(V_{DD} - R\tfrac{I_{SS}}{2} - V_d - V_T)^{\alpha}(1 + \lambda(V_{DD} - R\tfrac{I_{SS}}{2} - V_d)) = 0 \\
&Gain_{DC} = 351.2 \times \tfrac{I_{SS}W_1^5W_2^5R}{(I_{SS}W_1^5)^{5/6}W_2^5 + (I_{SS}W_2^5)^{5/6}W_1^5} \geq 1
\end{aligned}
\tag{3.2.1}
$$

According to this, the solver minimizes the propagation delay provided that the power dissipation remains below the maximum power dissipation P_{max}. The delay is approximated as $D = 0.69RC$. The DC current constraints insure that the voltage at node x remains constant during switching and hence reduce the output offset voltage and minimize the switching noise. It is reported that the mathematical program in [5] has an average error of 8.3% and a maximum error of 16%.

In [3], extra constraints for complete-switching were added to the set of constraints in equation 3.2.1. Complete switching occurs when the gate's voltage swing given by $I_{SS} \times R$ is larger than or equals to $V_{DD} - V_x - V_T$, where V_T is the threshold voltage. A detailed discussion on complete-switching and its importance to the gate robustness is provided in Section 5.1.

The programs discussed above provide a convenient way of optimizing MCML circuits. The problem, however, is that the expressions used for the objective function and the constraints are complicated. This is due to many factors. The program in equation 3.2.1,for

example, has two types of variables. The first type is the tunable variables. These are the variables that can be directly varied in the circuit. Examples of such variables are the tail-current I_{SS}, transistors sizes W and the load resistance R. The second type is the interdependent variables. Those variables are dependent on the tunable variables and can only be varied in the circuit by varying the tunable variables. Examples of interdependent variables are V_x, V_d, V_T and the voltage swing ΔV. This is in the circuit level. In the programming level, the solver sees all the variables as tunable. Moreover, some interdependent variables are functions of other interdependent variables. In short channel devices, for example, V_T is a function of V_x, V_d and the transistor width W. These interdependencies may be ignored, but the accuracy of the solution will be questionable. Including these interdependencies, on the other hand, introduces more nonlinear and tight constraints, and hence increases the problem complexity.

In [8], analytical expressions that relate performance metrics to the circuit electrical and physical parameters (current, transistor sizes, logic swing) are derived. The expressions are analyzed to shed light into various tradeoffs for different design objectives. This is done by dividing the MCML gate operation region into smaller regions where the delay may be expressed in terms of the current, the voltage swing and technology parameters. When compared to simulation results, the delay model has an average error of 11%. The study also reveals the varying effects of each transistor on the delay in various modes of operation (low power, high speed). This work, however, does not consider the potential that may be realized by exploiting mathematical programming as a means to reduce the complexity of MCML design.

3.3 Mathematical Programming

3.3.1 Feasibility

Consider the general optimization problem

$$\begin{aligned} &Minimize : f(x) \\ &Subject : \left| \begin{array}{l} g_i(x) \geq 0 \\ h_i(x) = 0 \end{array} \right. \end{aligned} \tag{3.3.1}$$

We are required to minimize the function $f(x)$, which is referred to as the objective function. $g_i(x)$ is a set of inequality constraints and $h_i(x)$ is a set of equality constraints. A point x that satisfies all the problem constraints is a feasible point. We call the set of all feasible points, a feasible region or a feasible set denoted by S. At any feasible point, an inequality constraint is said to be active if $g_i(x) = 0$, and inactive if $g_i(x) > 0$. The set of active inequality constraints at any feasible point is called an active set. Figure 3.2 illustrates the feasible region defined by the constraints

$$\begin{aligned} h(x) &= x_1 + 2x_2 + 3x_3 - 6 = 0 \\ g_1(x) &= x_1 \geq 0 \\ g_2(x) &= x_2 \geq 0 \\ g_3(x) &= x_3 \geq 0 \end{aligned} \tag{3.3.2}$$

The feasible set S in this example lies on the flat surface that represents the first constraint. At the point P(0,0,2), the fist two inequality constraints are active, while at the point P(1,1,1) the active set is empty.

We now define terms associated with optimality. A point x^* is a solution to the minimization problem if x^* is feasible and if $f(x^*) \leq f(x)$ for all x in S. Furthermore, x^* is a strict global minimizer if it satisfies $f(x^*) < f(x)$ for all x in S.

Most of the techniques we will use find a solution using Taylor series approximation around a point. This approximation is usually only valid around that point, meaning that these methods converge to local minima. Furthermore, providing an educated starting guess to the optimizer increases the chances of finding the desired global minimum. A point x^* is said to be a local minimizer of f, if $f(x^*) \leq f(x)$ for all x in S such that $|x - x^*| < \epsilon$, where ϵ is some small positive number. Figure 3.3 demonstrates graphically the maxima and minima definitions.

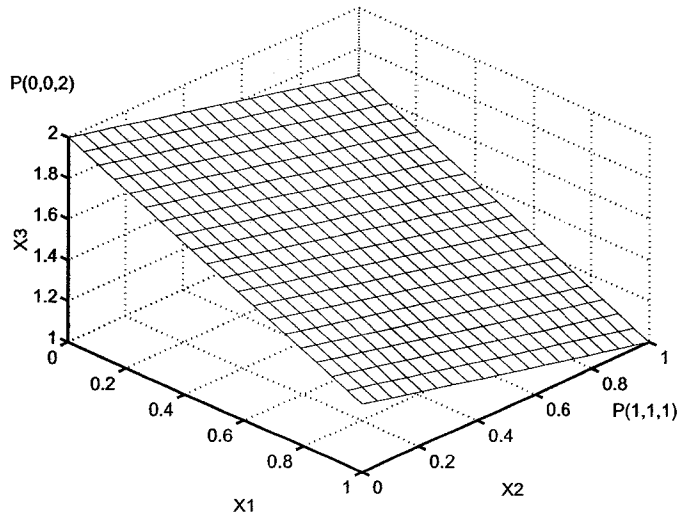

Figure 3.2: Example of a feasible set

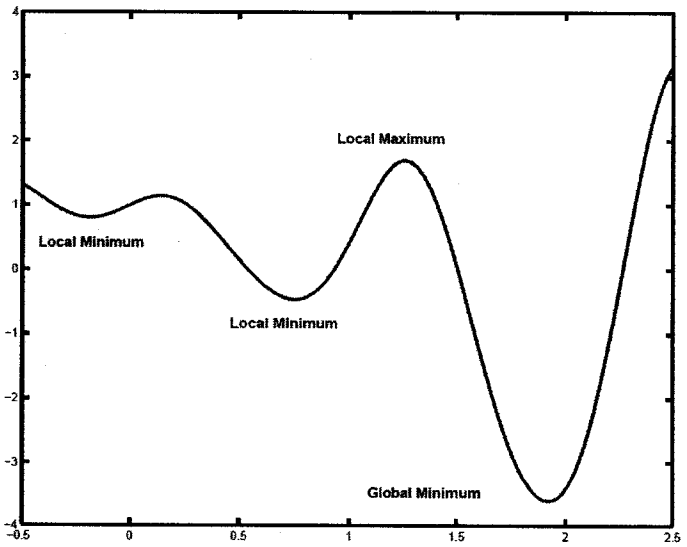

Figure 3.3: Examples of stationary points

3.3.2 Optimality Conditions

The solution to the unconstrained problem

$minimize f(x)$

must satisfy the following conditions [20]

$$\begin{aligned} \nabla f(x^*) &= 0 \\ \nabla^2 f(x^*) &\geq 0 \end{aligned} \tag{3.3.3}$$

The first equality is a first order necessary condition. It suggests that the point x^* is a stationary point. A stationary point could be a local minimizer, maximizer or a saddle point. This condition alone is not sufficient to determine a local minimizer. The second condition is referred to as a second order sufficient condition. It is sufficient to guarantee that the point x^* is a local minimizer. A local maximizer satisfies the conditions

$$\begin{aligned} \nabla f(x) &= 0 \\ \nabla^2 f(x) &\leq 0 \end{aligned} \tag{3.3.4}$$

3.3.3 Convexity

Convexity is an important characteristic in the science of mathematical programming. A set S is convex if, for all elements of S [20]

$$\theta p_1 + (1-\theta)p_2 \in S \tag{3.3.5}$$

where θ is a positive number between zero and one. This means that if points p_1 and p_2 are in S, then the line segment connecting p_1 and p_2 is also in S.

A function f is convex on a convex set S if [20]

$$f(\theta p_1 + (1-\theta)p_2) \leq \theta f(p_1) + (1-\theta) f(p_2) \tag{3.3.6}$$

In other words, f is convex if the line segment connecting the points $(p_1, f(p_1))$ and $(p_2, f(p_2)$ lies on or above the graph of the function. Figure 3.4 illustrates the definitions

of convexity. The function $f(x)$ in the Figure is convex, since any line connecting between any pair of points in the function is above the function graph.

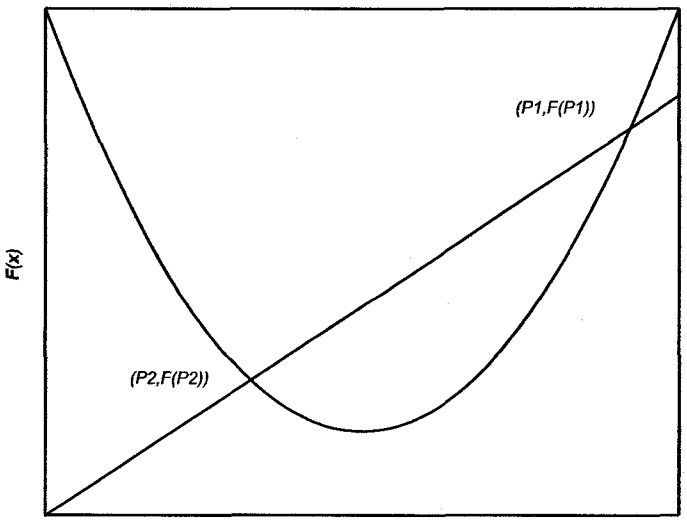

Figure 3.4: A convex function

A function f is concave on S if

$$f(\theta p_1 + (1-\theta)p_2) \geq \theta f(p_1) + (1-\theta)f(p_2) \tag{3.3.7}$$

The program

$$\begin{aligned} &Minime : f(x) \\ &Subject : g_i(x) \leq 0 \end{aligned} \tag{3.3.8}$$

is a convex mathematical program if the objective and the constraints are convex functions.

A local minimizer to a convex mathematical program is also a global minimizer to that program [20].

3.3.4 General Optimization Algorithm

Most of the general purpose optimization methods have the general form [21]

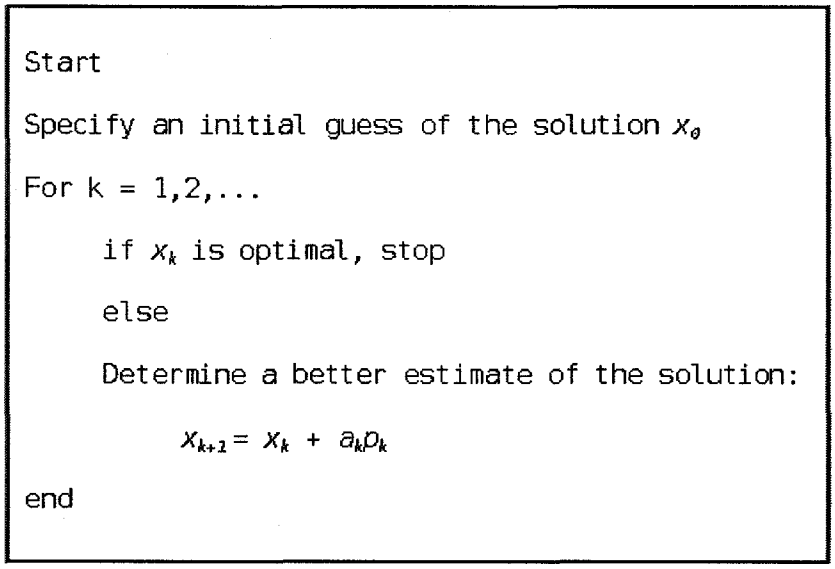

Figure 3.5: General optimization algorithm

where p_k is a direction pointing towards the general solution or at least towards a better estimate of the solution. The direction p_k is typically required to be a descent direction. That means that taking a small step in direction p_k will produce a reduction in the function value. The factor a_k is some positive scalar called the step length. In Newton's method, which is discussed in Section 3.3.6, the step length is always 1, that is equivalent to taking a full step in the direction p_k. A popular technique to calculate the step length is the line search method. The line search finds a solution to the problem [20]

$$\underset{a_k}{Minimize} : f(x_k + a_k p_k) \tag{3.3.9}$$

that is find some a_k that will minimize the objective function in the direction p_k.

3.3.5 Performance Metrics

There are various costs associated with optimization methods. Almost all optimization techniques involve an iterative procedure. At each iteration, a number of computations are performed to calculate first derivatives and sometimes the second derivative information,

the search direction and the step length. The number of arithmetic operations required by the method to achieve the solution is referred to as the computational complexity. There is also the cost of storing the results of the calculations. An N-dimensional objective function requires an $(N \times N)$ matrix or N^2 data locations to store the second order derivative information. Another performance metric is the rate of convergence. The rate of convergence is a measure of how fast the method converges to the solution [22].

3.3.6 Newton's Method for Root Finding

Most of the optimization algorithms used in general purpose solvers are derived in a manner similar to Newton's method. Newton's method solves a problem of the form

$$f(x) = 0 \tag{3.3.10}$$

There are many efficient ways to finding the roots for this problem. Newton's method is based on solving a sequence of linear approximations to the original problem [23]. The derivation of the Newton's method formula starts by writing the Taylor series approximation for the function f around the point x_k as

$$f(x_k + p) = f(x_k) + p_k f'(x_k) \tag{3.3.11}$$

At every iteration, we hope that $x_k + p$ is a better estimate of the solution x^*. If $f'(x_k)$ is not equal to 0, we can solve the equation

$$f(x^*) = f(x_k) + p_k f'(x_k) = 0 \tag{3.3.12}$$

for p_k to obtain

$$p_k = -f(x_k)/f'(x_k) \tag{3.3.13}$$

The new estimate of the solution now is

$$x_{k+1} = x_k - f(x_k)/f'(x_k) \tag{3.3.14}$$

Newton's method approximates the function by its tangent line at x_k. The intersection of the tangent with the x-axis is taken as the new estimate of the solution. This is illustrated in Figure 3.6.

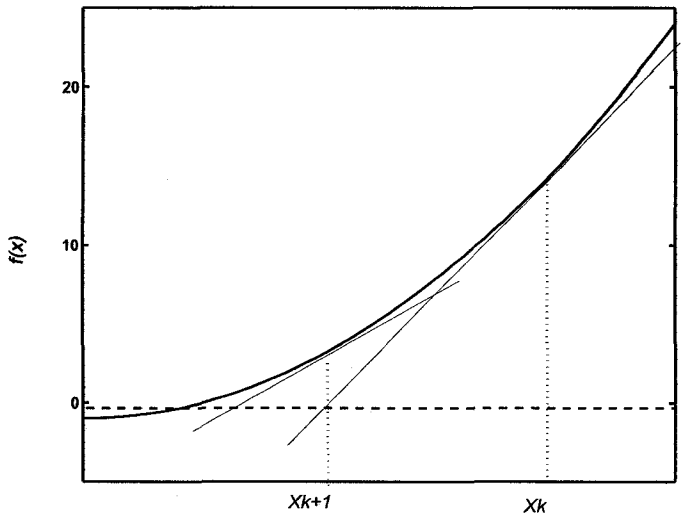

Figure 3.6: Illustration of Newton's method convergence mechanism

3.3.7 Newton's Method for Minimization

In this section we derive the formulae for Newton's method for minimization [20]. The procedure begins by approximating the objective function by Taylor series expansion

$$f(x_{k+1}) = f(x_k) + p_k \nabla f(x_k) \tag{3.3.15}$$

Taking the derivative of the approximation yields

$$\nabla f(x_{k+1}) = \nabla f(x_k) + p_k \nabla^2 f(x_k) \tag{3.3.16}$$

We expect the next point x_{k+1} to be the optimum solution. If we apply the first order

optimality condition $\nabla f(x_{k+1}) = 0$, then the search direction p_k may be expressed as

$$p_k = -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \tag{3.3.17}$$

This expression can be written in the form $x_{k+1} = x_k + p_k$, where p_k is the solution to the system

$$\left[\nabla^2 f(x_k)\right]^{-1} p_k = -\nabla f(x_k) \tag{3.3.18}$$

Analogous to the linear approximation used in Newton's method for $f(x) = 0$, the linear approximation for the case when $\nabla f(x) = 0$ is:

$$\nabla f(x_k + p_k) \approx \nabla f(x_k) + \nabla^2 f(x_k) p_k \tag{3.3.19}$$

This linear approximation is the gradient of the quadratic function

$$Q(p) \equiv f(x_x) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 \tag{3.3.20}$$

We can conclude that Newton's method for minimization approximates the nonlinear objective function by a quadratic function. A few comments about Newton's method convergence are in order.

It is expected that a quadratic rate of convergence to be achieved, which is a major advantage, except in some special cases, where the method may fail or even diverge. There is nothing in the method that would push the iterates towards a minimum. All that we have until now is a technique that moves towards the nearest critical point, be it minimum, maximum or a saddle point. Clearly, more conditions are required to convergence to a minimum point. A widely used set of constraints to test the solutions optimality are the Karush-Kuhn-Tucker conditions [24] [25].

Newton's method requires first and second order derivative information. The second order derivative is very costly both in terms of calculation and storage. If the derivative information was provided to the optimization tool, then this will not be a problem, but

usually this is not the case. There is also the cost of solving the Newton equations to find the Newton direction p. Many of the optimization algorithms based on Newton's methods implement creative techniques to reduce the costs associated with Newton's method while retaining its good convergence rate [20].

Chapter 4

Balancing the Act: A Symmetric MCML Universal Gate

This chapter compares the different MCML universal gate topologies that were introduced in chapter 2. The goal is to quantify the advantages and the drawbacks of balancing the differential-pair universal gate with the addition of a transistor, M5, between the right branch output node and M2. This topology is commonly used in industry, but its performance in comparison to other universal gate topologies has not be properly quantified in literature. The procedure to assess the feasibility of the proposed modification is as follows. A logical reasoning is outlined and followed by large-signal and small-signal analysis to gauge the benefits and drawbacks of the modified topology. Simulation results are then collected and analyzed to compare the modified topology to the standard universal gate and the MUX-based universal gate in terms of timing and signal integrity. The chapter concludes by summarizing the results and commenting on the feasibility of the modification.

4.1 Motivation

4.1.1 A Mathematical Programming Perspective

In order to automate the design of predefined circuit structures, a simple and an accurate model must be found to mimic the circuit operation. The analog nature of MCML has made

it hard to express its behavior and performance in a simple form suitable for automatic design and optimization. The asymmetric topology of the standard universal gate only adds extra hurdles to any automation attempts. In a symmetric topology, like the MUX gate for example, there are more possibilities for variable and constraint reduction. There is also a better chance of finding dependencies and eliminating redundancies to reduce model complexity. It will be shown in later chapters that using a symmetric structure greatly reduces the delay model complexity.

4.1.2 A Circuit Perspective

Standard Universal Gate

The asymmetric topology of the universal gate in Figure 4.1 introduces many problems. These problems are directly related to the mismatch in the propagation delay between the two differential outputs, and also to the DC output offset voltage. By definition, differential signals are exactly 180 degrees out of phase. If this phase angle is not preserved, the signal processing will not be accurate. This phase shift will increase as operation frequency increases, since the waveform period becomes smaller while the delay difference between the two paths remains constant for any specific gate. The DC output offset voltage adds another obstacle to the operation of the standard MCML universal gate. Contrary to analog design, where AC coupling techniques are used to isolate the effect of DC voltage offset in different design stages, MCML gates are directly affected by the DC voltage levels at their inputs. It is also worth noting that a well-balanced eye-shape waveform achieves the maximum theoretical noise margin for any given voltage swing and gain.

MUX-based Universal Gate

The notion that the MUX-based topology is the most balanced is not entirely true. Referring to Figure 4.2, by examining the electrical path from V_{DD} down through the load, transistors M5 and M2 to the sink, it is noted that this path is ideally off regardless of the inputs values. In practice however, the input signal slope is finite and there is a brief

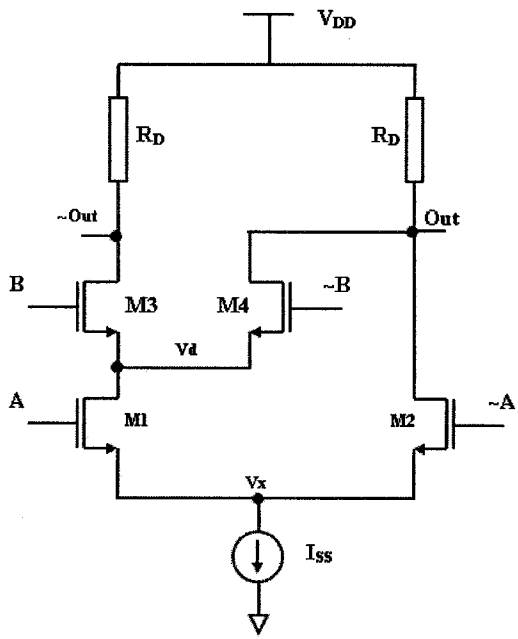

Figure 4.1: Standard MCML universal gate

period during transition when both transistors M4 and M2 are turned on causing undesired current flow. This current has significant small-signal effects.

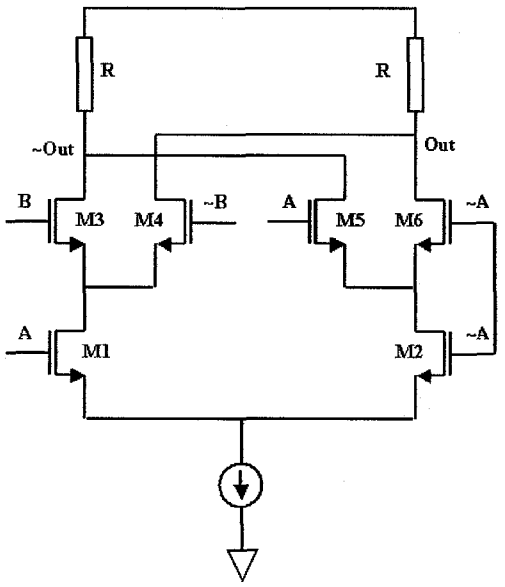

Figure 4.2: MUX-based MCML Universal Gate

Assuming B and -B are held at High and Low respectively, then the output should follow the input A. When A goes from High to Low, transistors M1 and M5 switch off and transistors M2 and M6 turn on causing the output node and the node at the drain of M2 to discharge. But the undesired current that flows through M5 will pour an extra charge in M2 drain node and hence increase the time required to discharge this node as well as the input node. On the other hand, during charging time, the stray current will help charge the output faster. The outcome is faster charging and slower discharging times. Table 4.1 shows rising and falling delays for the standard and the MUX-based universal gates when the input signal slope is 10,000 V/μs, which is a reasonable output slew rate for high-speed and low-power applications.

Figures 4.3 and 4.4 show output rising and falling times and stray currents through M5 for a chain of MUX-based universal gates. In Figure 4.3, the test input signal and the

Table 4.1: Rising and falling times for standard and MUX-based universal gates.

Topology	Rising Delay	Falling Delay
Standard	38.4ps	31.8ps
MUX-based	17.2ps	52ps

outputs of the first, second and third gates are plotted against time. The output charges rapidly but it takes much longer time to discharge. This can be explained by monitoring the current activity through transistor M5. Figure 4.4 supports the prediction that as the slope of the input signal becomes smaller than infinity, the average amount of stray current through M5 increases. The first gate input is an ideal clock signal with small rise/fall times and the path through M5-M2 is on for a short period of time. This time increases substantially for the following gates.

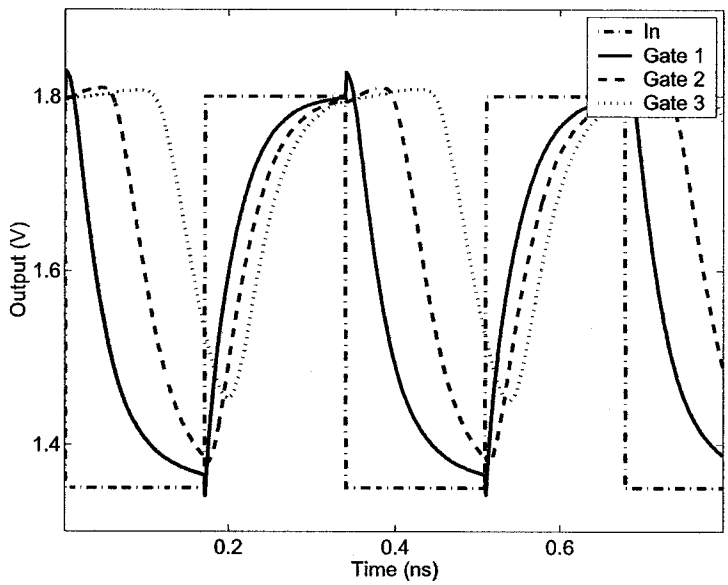

Figure 4.3: Output waveforms for a chain of MUX-based universal gates showing that the rising times are much smaller than the falling times

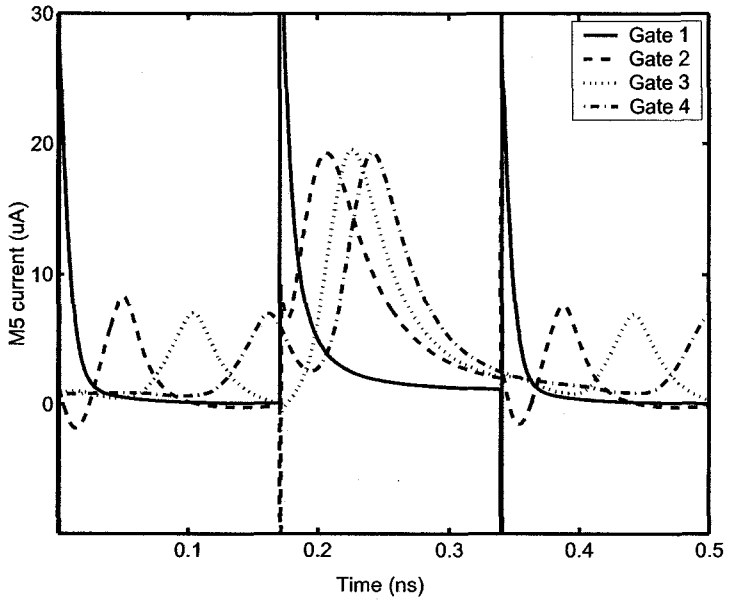

Figure 4.4: Waveforms of the current through M5 in the MUX-based universal gates

4.2 Analysis

Studying the large-signal and small-signal effects of the asymmetric topology on the gate design and performance will help aid in finding the main problems and developing alternate solutions. The following analysis is carried out on the standard universal gate in Figure 4.1.

For simplicity, it is assumed that transistors M1 and M2 have equal threshold voltages, $V_{T1} = V_{T2}$. The switching-off conditions in this case are

$$V_{GS1,2} < V_{T1,2} \tag{4.2.1}$$

Simulations show that when the input to transistor M1 changes from Low to High while M3 is on, transistor M1 is in saturation for most of the transition period and then settles in the triode region when the output level is settled at logic Low. The voltage at node x at this point is controlled mainly by M3. Hence we can write

$$I_{SS} = k\frac{W_3}{L}(V_{DD} - V_d - V_T)^{\alpha}(1 + \lambda V_{DS3}) \tag{4.2.2}$$

When M2 is on, we have

$$I_{SS} = k\frac{W_2}{L}(V_{DD} - V_x - V_T)^{\alpha}(1 + \lambda V_{DS2}) \tag{4.2.3}$$

Also note that

$$\begin{aligned} V_d &= V_x + V_{DS1} \\ V_{DS2} &= V_{DD} - \Delta V - V_x \\ V_{DS3} &= V_{DD} - \Delta V - V_d \end{aligned} \tag{4.2.4}$$

By making a number of substitutions, we obtain

$$\frac{W_3}{W_2} = \frac{(1 + \lambda V_T)\Delta V^{\alpha}}{(1 + \lambda(V_T - V_{DS1}))(\Delta V - V_{DS1})^{\alpha}} \tag{4.2.5}$$

For typical index and channel length modulation parameter values in 0.18 μm technology and a swing of 0.3 V, the ratio between W_3 and W_2 is in the range of 2.5. The increase in the upper-level transistors sizes increases the capacitive loading through the critical path and contributes to the already existent problem of delay mismatch between the two MCML branches.

Table 4.2 lists the values of process dependent parameters used in the proposed model for the 0.18 μm technology.

Table 4.2: Technology dependent parameters

Parameter	Value
W_{min}	0.22E-6 m
L_{min}	0.18E-6 m
k	68E-6
α	1.26
λ	0.36

4.3 The Modified Topology

The MUX-based MCML topology is an attempt to overcome the asymmetry problems in the standard universal gate. The improvement in the gate symmetry, however, comes at a cost. First, the total gate input capacitance is increased by a factor of 1.5. This leads to a reduction in the input signal slew rate by 1/1.5. Secondly, the gate internal capacitance is also higher. The increase in the gate capacitance has a direct effect on the gate power-delay-product. The goal is then to find a topology that will solve the asymmetry problem while preserving the gate's efficiency. Figure 4.5 shows a compromise circuit developed to satisfy the symmetry and efficiency requirements.

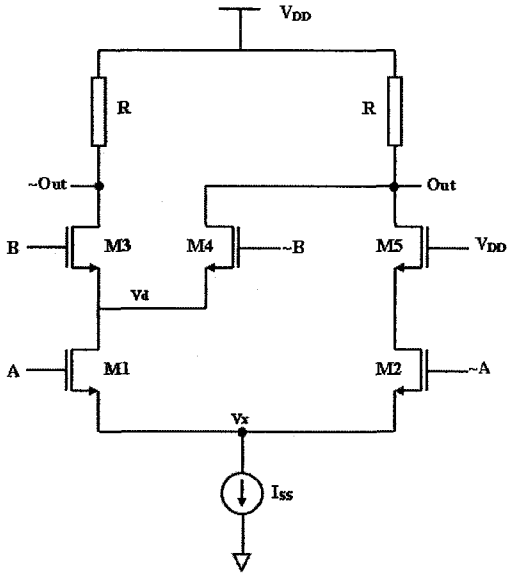

Figure 4.5: Modified MCML universal gate with added transistor M5

Considering the standard universal gate in Figure 4.1, when the right hand branch is at logic Low, depending on the values of inputs A and B, the current will flow through either transistor M2 down to the current sink or through transistors M4 and M1. This difference between the two paths is the main cause of the electrical and timing mismatch. If we add transistor M5 as in Figure 4.5, then in DC sense both branches will have the same

conditions since only one of either M3 or M4 is on at any given time. In small-signal sense, the critical-path total capacitance is increased slightly due to the diffusion capacitance of transistor M5. The gate terminal of M5 is connected to V_{DD} so that its gate capacitance does not load the input. From this point, the transistor names will be complemented with a subscript to avoid any confusion. $M5_{MDF}$ and $M5_{MUX}$ denote M5 in the modified topology and the MUX-based topology respectively.

When considering the MUX-topology, the gates of transistors $M2_{MUX}$ and $M6_{MUX}$ are connected to the same input and hence are redundant. If $M6_{MUX}$ is connected directly to V_{DD}, the logic gate will still function in the same way while the input capacitance will be dramatically reduced. Transistor $M5_{MUX}$ is always turned off (Ideally, since if $M2_{MUX}$ input is High, then $M5_{MUX}$ input is Low and vice versa) and, hence, it does not contribute to the logic realization mechanism. If $M6_{MUX}$ is connected to V_{DD}, then $M5_{MUX}$ may be removed without affecting the gate operation.

It might be tempting to bias $M5_{MDF}$ to $V_{DD} - \Delta V/2$ to achieve zero output offset voltage when all inputs are at mid-swing. However, since the aim is to design a digital cell, the focus should be on the DC symmetry of the circuit at the two logic levels '0' and '1'.

4.4 Simulation and Results

To draw the comparison between the standard universal gate and the modified one, a test environment is set up. The signal propagation through a chain of 5 gates is analyzed under different operation frequencies. The gates are sized to produce a swing of 0.75 V and a tail-current of 95 μA. For the standard universal gate, the delays are measured for two cases. The first case is when all the transistors are sized equally and the second case is when the transistors are sized to reduce the DC imbalance. The second measurement is intended to weigh the expense of balancing the standard topology by adjusting the transistors sizes according to the ratio in equation 4.2.5 to eliminate the DC offset voltage.

4.4.1 Before Resizing

Table 4.3 shows the transistors sizes of the standard and modified universal gates (UG) topologies. The standard gate was sized without any balancing attempts and all transistors are assumed to have typical sizes. The values for the modified universal gate were found by applying mathematical programming techniques. Chapter 6 includes a detailed description of the MCML optimization mathematical program.

Table 4.3: Transistor sizes for the test environment

Transistor Width (μm)	Standard UG	Modified UG
W_1	0.49	0.49
W_2	0.49	0.49
W_3	0.49	0.49
W_4	0.49	0.49
W_5	N/A	0.49
Current sink W_s	3.66	3.66

Delay Measurement

After performing a DC simulation to insure the gates are operating at the anticipated DC points, a transient simulation is set up to measure the gate propagation delays by applying a low frequency pulse train.

Table 4.4 shows the propagation delays for one gate. The difference between the Standard universal-gate's complementary signals delays is four times that for the modified topology. The difference in arrival times between the complementary signals for the standard topology is 4 times the time for the proposed circuit. The MUX-based universal gate has the best symmetry but also has the highest delay.

In the next setup, the propagation delay of a chain of MCML universal gates is measured and compared for the three topologies. Table 4.5 shows the delay for a chain of five gates.

The results in Tables 4.4 and 4.5 are consistent. If the average delays are compared,

Table 4.4: Worst case delays for different MCML universal gate topologies

Gate Topology	+ve Out delay (ps)	-ve Out Delay (ps)
Standard	22.9	18.8
MUX-based	29.3	29.7
Modified	23.4	24.5

Table 4.5: Worst case delays for a chain of five gates for different MCML universal gate topologies

Gate Topology	+ve Out delay (ps)	-ve Out Delay (ps)
Standard	110	108.5
MUX-based	148.4	146
Modified	128.1	129.8

it can be seen that the average delay per gate is 20.8ps and 23.9 for the standard and modified topologies respectively. The conclusion from this timing test is that for typical low-power high-speed design and a moderate fan-out of 2, the modified topology has a 15% higher delay. The MUX-based topology does not show any visible superiority over the modified circuit in any of the timing categories.

DC-Level Shift and Operation Frequency

The aim of this test is to estimate the maximum throughput of the MCML gates. This is done by applying increasingly higher frequency inputs while observing the gate output waveforms. Table 4.6 shows the unity gain bandwidths for the standard and the modified topologies. Even though the modified gate has higher capacitance, its unit gain bandwidth is only slightly narrower than the standard gate's bandwidth. This is expected since the cascading of M5 on top of M2 offsets the Miller's capacitance effect at the output [26].

In the next test, the gates from the previous bench are driven by high frequency inputs. The reduction in the output swing of the fifth gate is monitored while the frequency is increased until the output swing is reduced to 60% of the nominal value. This value is

Table 4.6: The differential signal unity gain bandwidth for the standard and the modified MCML topologies

Topology	f_{unity} (GHz)
Standard	13.7
Modified	12.6

used as a cutoff mark because it is customary to design MCML gates with a noise margin of $0.4\Delta V$ or higher [3]. Figures 4.6 and 4.7 show the outputs of the standard and the modified universal gates when operated at 8.8Gbps and 13.6Gbps respectively. Even though the standard gate has lower propagation delay, it fails at lower frequencies compared to the modified gate. This is due to the inherent timing and electrical imbalance of the gate. It is also noted that as the input frequency increases, the output offset voltage starts to increase accordingly until at a certain frequency, the reduction in the differential output magnitude becomes larger than the noise margin and the signal is no longer recoverable.

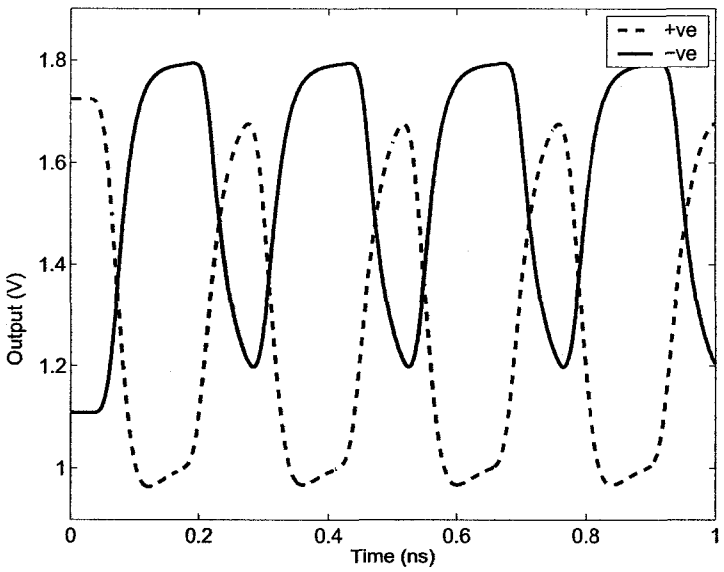

Figure 4.6: Output of the standard universal gate showing that the swing is reduced to 60% of the nominal value at 8.8GHz

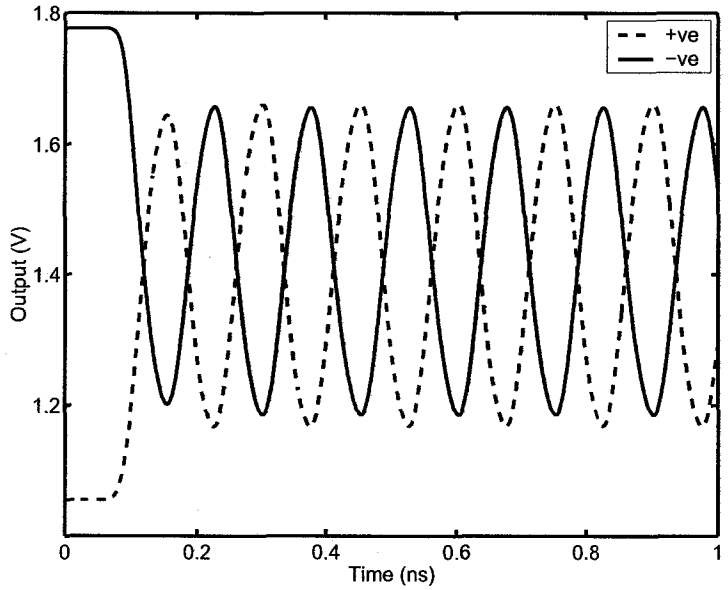

Figure 4.7: Output of the modified universal gate showing that the swing is reduced to 60% of the nominal value at 13.6GHz

The reason for such behavior is the imbalance between the two current branches. When M1 and M3 are on, they tend to push the voltage at node x down to accommodate the tail-current I_{ss}. The same is true for the case when transistor M2 is on. The difference between the two scenarios is that M1 and M3 are stacked on top of each other. That requires V_x to be smaller than in the case when M2 is on. Figure 4.9 shows the voltage activity at node x during switching.

Table 4.7 lists the universal gates maximum operation frequencies for different currents and voltage swings. It is assumed here that the fan-out gate and the driver are equally sized. Maximum operation frequency is directly related to the fan-out capacitance load.

In addition, M2 of the standard universal gate has larger drain to source voltage which reduces the threshold voltage V_{T2} according to the Drain Induced Barrier Lowering phenomenon. This makes M2 harder to switch off. To insure against this, the voltage swing must be increased so that V_{GS2} is lower than V_{T2}. The increased voltage swing is higher than the theoretical value given by [2]

Table 4.7: MCML UG operation frequencies for different currents and voltage swings.

Topology	ΔV (V)	20 μA	50 μA	100 μA
MUX	0.45	3.5 GHz	3.7 GHz	3.9 GHz
Standard	0.45	6.2 GHz	6.6 GHz	6.9 GHz
Modified	0.45	9.5 GHz	10.3 GHz	11.5 GHz
MUX	0.75	3.8 GHz	4.9 GHz	4.8 GHz
Standard	0.75	7.2 GHz	9.3 GHz	9.6 GHz
Modified	0.75	10.5 GHz	13.3 GHz	13.7 GHz

$$\Delta V_{min} \geq V_{DD} - V_x - V_T = \sqrt{\frac{I_{SS}L}{kW}} \tag{4.4.1}$$

Another solution is to increase W_3 significantly to increase V_x and reduce the leakage current through M2. Both solutions lead to an increase in the gate delay.

Switching Noise

Balancing the branches is still important for other reasons. Figure 4.8 shows the power supply current for the standard and the modified topologies.

The standard universal gate's asymmetry causes the voltage V_x to vary significantly during switching. This in turn causes current spiking and induces power supply switching noise.

Ring Oscillator Test

In this test, the gates under investigation are connected in a ring oscillator configuration to examine the outputs eye diagram and measure the oscillation frequencies. Figures 4.10 and 4.11 show the eye diagrams of the differential outputs for the standard and the modified MCML universal gate topologies respectively. Figure 4.10 illustrates the imbalance between the complementary signals for the standard universal gate. The modified design exhibits a fairly symmetric eye shape as shown in Figure 4.11. This is critical for a robust operation

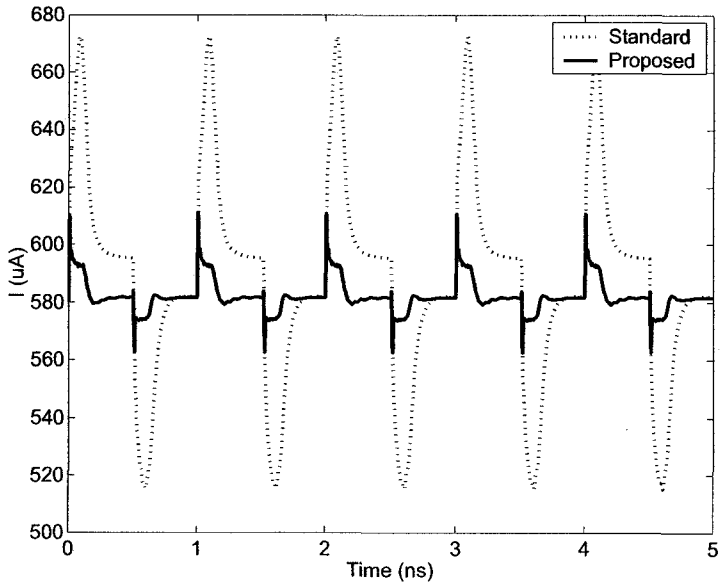

Figure 4.8: Power supply current activity for the standard and the modified universal gates

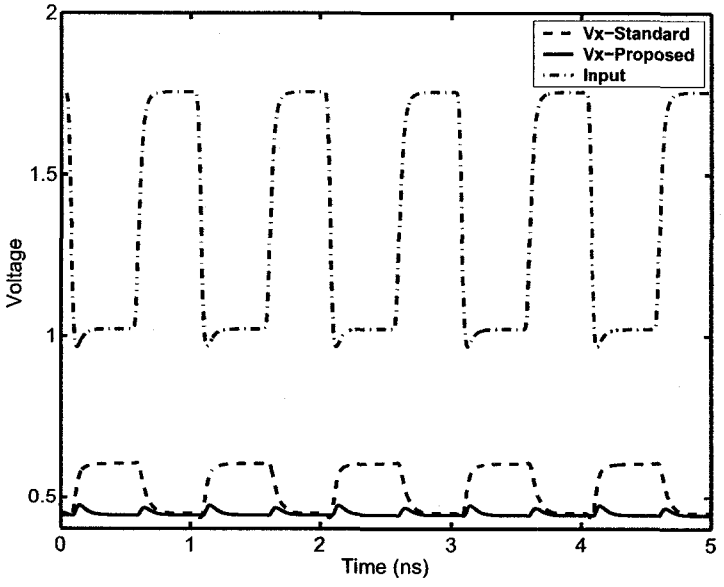

Figure 4.9: The voltage at node x for the standard and the modified universal gates

when dealing with large designs. The standard universal gate oscillates at 5.56 GHz, while the modified gate has an oscillation frequency of 4.26 GHz.

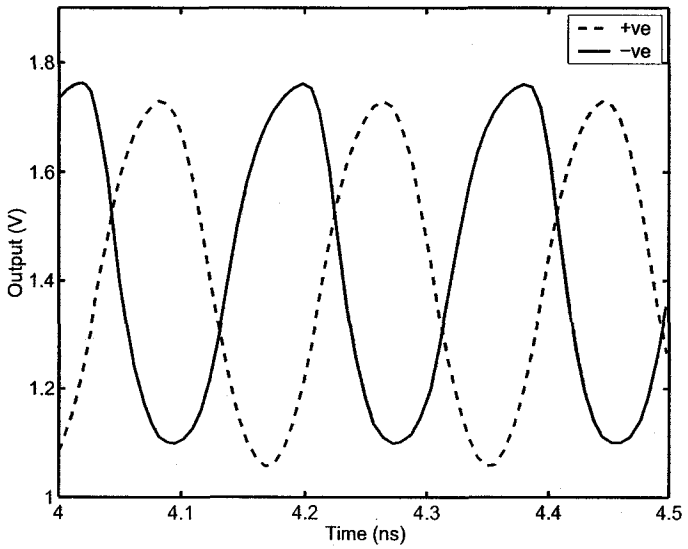

Figure 4.10: Standard MCML UG eye diagram showing timing and DC mismatch

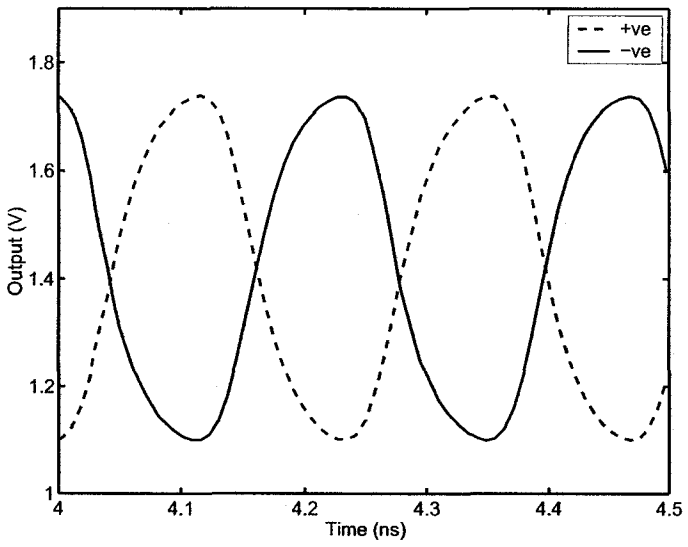

Figure 4.11: Modified MCML UG eye diagram

4.4.2 After Resizing

In Section 4.1.2, we concluded that the standard universal gate's transistors must be resized to eliminate the output voltage offset. Table 4.8 shows the calculated transistors sizes.

Table 4.8: MCML standard universal gate transistor sizes to eliminate the DC offset

Transistor	Width (μm)
M1	0.44
M2	0.39
M3	1.43
M4	1.43
Current Sink	3.66

Tables 4.9 and 4.10 show the delays for one gate and five gates chain respectively. The standard universal gate delay has increased significantly as a penalty for attempting to eliminate the DC voltage offset. This makes our modified topology a better choice when a well balanced output signal is critical, especially in large designs.

Table 4.9: Gate propagation delays for the modified UG and the resized standard UG

	+ve Out delay (ps)	-ve Out delay (ps)
Standard UG	26.5	24
Modified UG	23.4	24.5

Table 4.10: 5-gate chain propagation delays for the modified and the resized standard universal gates

	+ve Out delay (ps)	-ve Out delay (ps)
Standard	155	141
Modified	128	130

4.5 Summary

In this chapter, a modified topology for the MCML universal gate has been analyzed. The goal of the modification is to improve the timing correlation and reduce the voltage offset between the MCML gate's differential outputs. The feasibility of the modified circuit has been investigated. When compared to standard universal gate, the modified topology has a better DC balance, lower switching noise and higher operation frequencies for the same power dissipation. The MUX-based universal gate, on the other hand, experiences higher rising times due to the leakage current through $M5_{MUX}$. It has also been shown that sizing the standard universal gate transistors to balance the dc characteristics increases the timing imbalance and degrades the gate delay.

Chapter 5

MCML Modeling and Design

In this chapter, the relationships between the MCML gate's tail-current, voltage swing and delay are exposed. The aim is to build an understanding of the gate operation and help develop a good mathematical program that is suitable for numerical optimization, which is the prime objective of this thesis. The study begins with the single-level MCML inverter and then evolves to discuss the more complex two-level universal gate. The modified universal gate of Chapter four has been chosen because of its versatility and its good timing, area and power consumption.

The procedure is as follows. First, the conditions for robust operation are defined and expressed analytically. The study then attempts to derive expressions for the static and dynamic behavior of the MCML universal gate, under robustness constraints, that are mathematical-solver friendly. The expressions are then manipulated to expose the relationships between the design metrics in various regions of operation. This step gives the designer some clues into where local minima might lie and help produce better starting guesses. This is critical, since if the problem is not convex, then it may have many local minima. In this case, advanced techniques may be necessary to estimate appropriate initial points to guide the solver towards the global solution.

5.1 MCML Design

5.1.1 Operation Conditions

Let us define ΔV_{min} as the minimum voltage swing required to completely switch the tail-current I_{SS} from one branch to the other. For the inverter in Figure 5.1, this means that if M1's gate is connected to V_{DD} and M2's gate is connected to $V_{DD} - \Delta V$, and if ΔV is larger than or equal to ΔV_{min}, then the current $I_{M1} = I_{SS}$ and $I_{M2} = 0$.

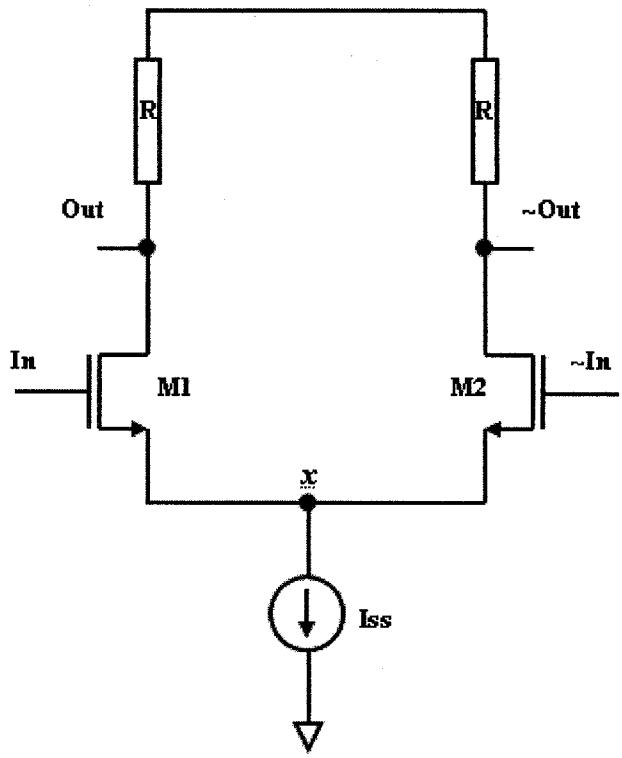

Figure 5.1: MCML inverter

It is important at this point to express ΔV_{min} in terms of the circuit voltages and currents. An NMOS transistor is on when the voltage drop between its gate and source terminals is greater than the threshold voltage. That is

$V_{GS} - V_T > 0$

The condition to switch off M2 is

$V_{GS2} - V_{T2} \leq 0$

Also note that

$V_{G2} = V_{DD} - \Delta V$

$V_{S2} = V_x$ where V_{G2} is the voltage at the gate terminal of M2, V_{S2} is the voltage at the source terminal of M2 and V_x is the voltage at node x. Substituting for V_{GS2} in the switching-off condition yields [27]

$$\Delta V \geq V_{DD} - V_x - V_T \tag{5.1.1}$$

In other words, the minimum voltage swing required to turn off M2 and switch all of the tail-current to the other branch is

$$\Delta V_{min} = V_{DD} - V_x - V_T \tag{5.1.2}$$

The equation above may be expressed in terms of the tail-current and the transistor dimensions. When M2 is off, M1 is necessarily on and its current equals to I_{SS} . By using the alpha-power law model, the saturation current of M1 may be expressed as

$$I_{M1} = I_{SS} = k\frac{W}{L}(V_{DD} - V_x - V_T)^{\alpha} \tag{5.1.3}$$

If M1 and M2 have the same dimensions and ignoring any process variations, then it is safe to assume that both transistors have the same threshold voltage V_T . By using 5.1.2 and making the necessary substitutions into 5.1.3, rearranging and solving for ΔV_{min}, we get [26]

$$\Delta V_{min} = \sqrt[\alpha]{\frac{I_{SS}L}{kW}} \tag{5.1.4}$$

Assume it is required to plot the inverter propagation delay versus the tail-current I_{SS} for a given nominal voltage swing value ΔV_{nom} . To do this, the current is swept between the given bounds and the corresponding delays are measured. The given voltage swing

ΔV_{nom} must satisfy the complete-switching conditions at all times. This can be achieved by changing the transistor width according to equation 5.1.4 so that $\Delta V_{nom} \geq \Delta V_{min}$. Assume that the experiment starts by measuring the higher currents first and then moving down in a descending order. Also assume that at the highest tail-current, the circuit conditions are

$$\begin{aligned} \Delta V_{nom} &= \Delta V_{min} \\ W &> W_{min} \end{aligned} \tag{5.1.5}$$

The first condition suggests that the transistor's width is adjusted to ensure the nominal voltage swing V_{nom} is just enough to guarantee complete-switching. This may be rationalized by noting that a voltage swing higher than the minimum required increases the delay unnecessarily. The second inequality says that the transistor's width is larger than the minimum allowable by the technology.

After recording the first measurement, the tail-current is reduced to the next point. According to equation 5.1.4, reducing I_{SS} will reduce ΔV_{min} and, hence, the nominal voltage swing will become larger than what is required for complete switching. To balance this, the transistor's width W must be reduced according to equation 5.1.4 so that the equality in 5.1.5 is maintained. Reducing the width serves a double purpose. It keeps ΔV_{min} at a constant level and also reduces the inverter intrinsic capacitance. The intrinsic capacitance is the portion of the output capacitance that is due to the logic gate transistors.

The procedure is repeated by descending down to smaller currents while reducing the transistor widths accordingly until a current value is reached, call it I_L, where the transistor's width is equal to the minimum allowable by the technology. At this point, the circuit conditions are

$$\begin{aligned} \Delta V_{nom} &= \Delta V_{min} \\ W &= W_{min} \end{aligned} \tag{5.1.6}$$

If the current is reduced further, the minimum voltage swing will be reduced accordingly and the nominal swing becomes larger than the minimum required for complete-switching.

In this region, the circuit conditions are

$$\begin{aligned} \Delta V_{nom} &\geq \Delta V_{min} \\ W &= W_{min} \end{aligned} \tag{5.1.7}$$

Based on this discussion a number of definitions are in order. We define I_L as the amount of current at which the following conditions are satisfied

$\Delta V = \Delta V_{min}$

$W = W_{min}$

The first condition implies that the voltage swing is just enough to completely turn off the lower-level transistors. The second condition to be satisfied is that the transistor sizes are the minimum allowable by the technology.

We define the low-current region ζ as the region where $I_{SS} \leq I_L$ and

$\Delta V \geq \Delta V_{min}$

$W = W_{min}$

We also define the high-current region φ as the region where $I_{SS} > I_L$ and

$\Delta V = \Delta V_{min}$

$W > W_{min}$

The definitions above may also be reached by observing the voltage at node x. Complete switching occurs when

$$\Delta V = \Delta V_{min} + s \tag{5.1.8}$$

where $s \geq 0$ is a slack variable that follows the change in ΔV to hold the equality in 5.1.8 . When s is greater than zero, the swing is larger than the minimum swing required to completely turn off the transistors in the 'off' branch. Hence, the voltage swing may be reduced without the need to increase V_x . At any I_{ss} value, V_x may be changed by varying the logic transistor's width. Equation 5.1.8 suggests that the voltage swing may be reduced without the need to increase the transistors width. The increase in the voltage swing is compensated by a reduction in the slack term s so that the equality is held true. From this

simple analysis we can predict that when the slack s is positive, a reduction in the swing results in a reduction in the delay.

The more practical case is when the slack variable $s = 0$, that is when the voltage swing is just enough to turn off the transistor in the 'off' branch. A voltage swing larger than the minimum required increases the delay without any meaningful improvement to the gate robustness. Reducing the voltage swing below the minimum value required for complete switching causes an undesired DC current to flow in the 'off' branch . To counter this, V_x must be increased to satisfy the relationship in 5.1.8 . The voltage V_x may be increased by increasing the transistors width which, in turn, leads to an increase in the logic gate's intrinsic capacitance. Thus, reducing the voltage swing when $s = 0$ causes an increase in the gate's intrinsic capacitance. There is little information here to make early predictions about the relationship between the delay and the voltage swing. In the next section, we develop analytical expressions to make reasonable predictions about the relationship between the voltage swing and the delay. The analytical results are then verified and compared to spectre simulations.

5.1.2 MCML Complete Switching

Robustness is one of the most important performance specifications of any library cell. The MCML universal gate is the backbone of a proposed MCML library. Contrary to MCML inverters and custom-made MCML gates for specific applications, MCML gates that are used for high-density digital applications must completely switch the DC current from one branch to the other. This section studies the effects of incomplete switching. Complete switching occurs when the MCML gate's tail-current is completely switched from one branch to the other. Thus, the current in one branch is equal to I_{SS}, while the current in the other branch is equal to zero.

There are two ways to look at complete switching; extrinsically and intrinsically. According to the discussion in Section 5.1.1, a gate switches completely when the voltage swing of the input signal is larger than or equal to ΔV_{min} given by

$$\Delta V_{min} = \sqrt[\alpha]{\frac{I_{SS}L}{kW}} \tag{5.1.9}$$

However, it is also safe to say that as long as the small-signal and large-signal gains are larger than unity, the gates should be able to gradually amplify the signal until the swing is restored to the value $I_{SS} \times R$. This is an extrinsic view, since we are more concerned about the amplitude of the input signal rather than the intrinsic characteristics of the gate(Gain, Noise Margin), which are presumed to be sufficiently high.

When the DC gain of a logic gate is higher than unity, we expect that an input signal with a voltage swing of ΔV_{min} or higher to produce an output with a voltage swing $\Delta V_{output} = I_{SS} \times R$. If the MCML gate is designed properly, then

$$\Delta V_{output} = \Delta V_{min} = I_{SS} \times R \tag{5.1.10}$$

A voltage swing that is higher than ΔV_{min} increases the gate delay and reduces the power-delay-product unnecessarily. Equation 5.1.10 suggests that when the MCML gate DC gain is larger than unity, then

$$\Delta V_{min} = I_{SS} \times R = \sqrt[\alpha]{\frac{I_{SS}L}{kW}} \tag{5.1.11}$$

This is an intrinsic view, since the focus is on the logic gate's ability to propagate the input signal. This last view is the one we choose to use when we talk about complete switching. Based on this, we may label complete switching as a logic gate characteristic rather than an input signal characteristic. A complete-switching gate is a gate that satisfies

$$I_{SS} \times R \geq \sqrt[\alpha]{\frac{I_{SS}L}{kW}} \tag{5.1.12}$$

When the data at the output of an MCML gate is valid, it is expected that one half of the logic transistors to be on, while the other half to be off. The undesired current in the 'off' branch can be referred to as I_{off}.

We define V_{offset} as the difference between the ideal logic levels $V_{DD}, (V_{DD} - \Delta V)$ and

the corresponding actual gate voltage levels V_H, V_L. Then

$$\begin{aligned} V_{offset} &= V_{DD} - V_H \\ V_{offset} &= V_L - \|V_{DD} - \Delta V\| \end{aligned} \tag{5.1.13}$$

In terms of the 'off' branch current

$$V_{offset} = R \times I_{off} \tag{5.1.14}$$

To understand the effect of the 'off' branch current I_{off} on the universal gate performance, the circuit is simulated under numerous input combinations. It is noted that gates with a low Voltage Swing Ratio (VSR) are not capable of preserving the signal swing magnitude. The VSR is the ratio between the current in the 'on' branch to the tail-current. The signal swing degrades as it passes through the stages, and a few gates later, the data fades away completely. To illustrate the effect of the VSR on the signal propagation, a chain of MCML gates with different VRSs has been simulated in spectre. Figures 5.2 and 5.3 show the outputs of gates with a VSR of 98% and 93% respectively. The MCML gates have a tail-current of 50 μA and a voltage swing of $I_{SS} \times R = 0.55$ V. The logic transistors sizes are shown in Table 5.1.

Table 5.1: MCML logic transistor sizes for different VSRs, with a tail-current of 50 μA and a voltage swing $I_{SS} \times R = 0.55$ V

VSR	W (μm)
89%	0.30
93%	0.35
95%	0.40
98%	0.45

A low VSR has another drawback that effects multi-level MCML logic gates. To illustrate this, an input waveform with an amplitude of ΔV_{min} has been applied to the upper-level transistors, M3 and M4, of the standard universal gate in Figure 5.4. The

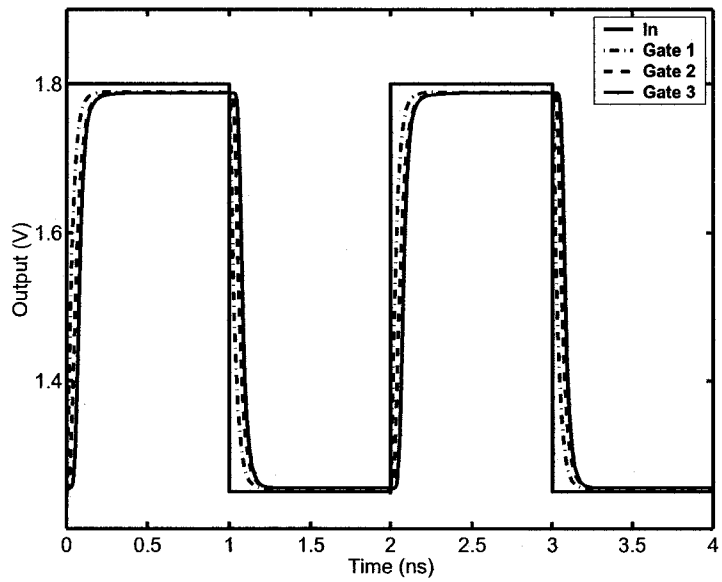

Figure 5.2: Signal propagation through a chain of MCML gates with a VSR of 98%

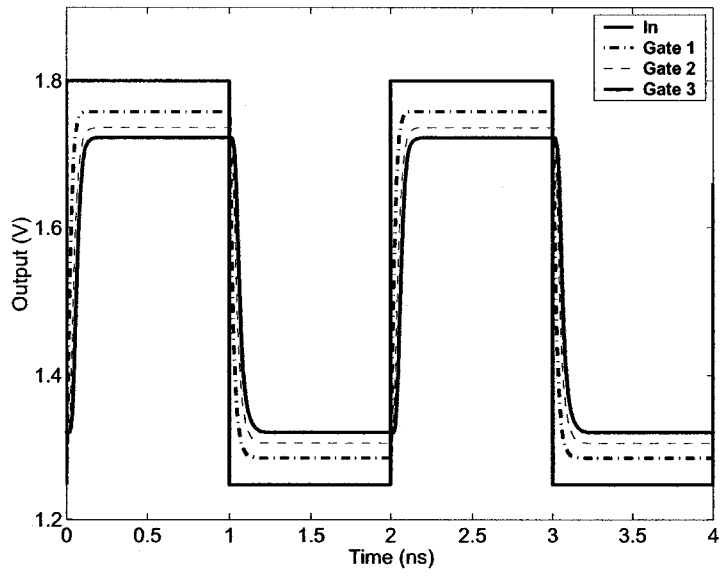

Figure 5.3: Signal propagation through a chain of MCML gates with a VSR of 93%

lower-level transistors, M1 and M2, are connected to Logic '1' and '0' respectively.

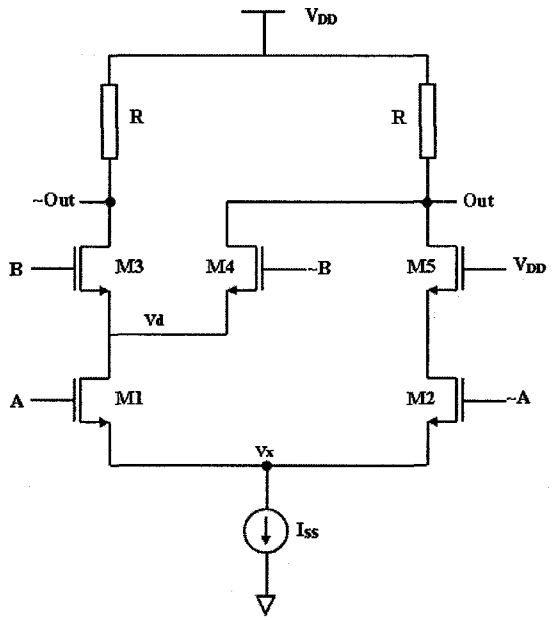

Figure 5.4: Standard MCML universal gate with balancing transistor

Figures 5.5, 5.6 and 5.7 show the gate's output waveform for VSRs of 98%, 95% and 89%. The Figures reveal that as the gate's VSR becomes smaller, the voltage offset between the complementary outputs increases accordingly. The voltage offset can be explained by noting that when the gate's VSR is low, there is a current I_{off} flowing through M2. If M3 is on and M4 is off, the voltage levels at the gate outputs are

$$\begin{aligned} V_{Out} &= V_{DD} - [I_{off} \times R] \\ V_{\overline{Out}} &= V_{DD} - [(I_{SS} - I_{off}) \times R] \end{aligned} \tag{5.1.15}$$

On the other hand, if M3 is off and M4 is on, then the voltages at the gate's outputs are

$$\begin{aligned} V_{Out=} &= V_{DD} - [I_{SS} \times R] \\ V_{\overline{Out}} &= V_{DD} \end{aligned} \tag{5.1.16}$$

This means that the gate's right-hand output can pull down to the ideal logic *Low* level $V_{DD} - \Delta V$, but cannot reach the ideal logic *High* level V_{DD}. The opposite is true for the left-hand output. That is because the upper-level transistors require a smaller voltage swing to completely switch off.

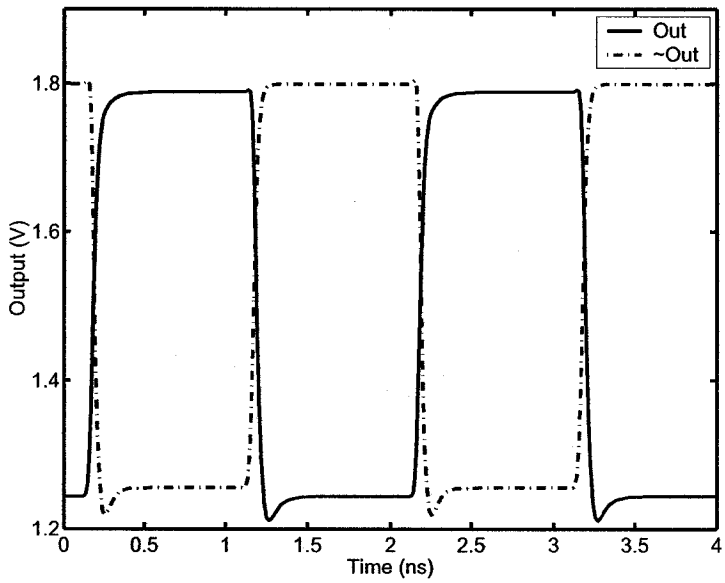

Figure 5.5: Output of MCML gates with a VSR of 98% when the input is applied to upper-level transistors

5.2 The Delay Model

5.2.1 MCML Inverter Delay Model

The small-signal model of the MCML inverter of Figure 5.1half-circuit is shown in Figure 5.8.

Assuming the input is an impulse, the delay of the MCML inverter can be expressed as [28]

$$D = 0.69R \times C \tag{5.2.1}$$

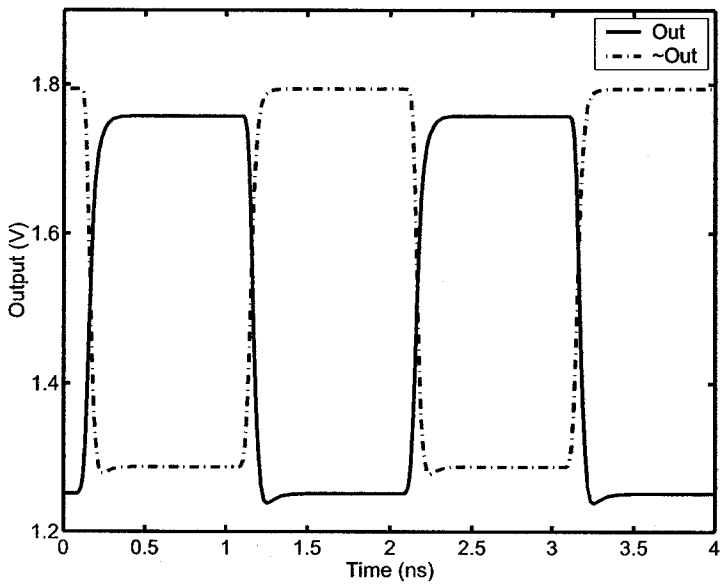

Figure 5.6: Output of MCML gates with a VSR of 95% when the input is applied to upper-level transistors

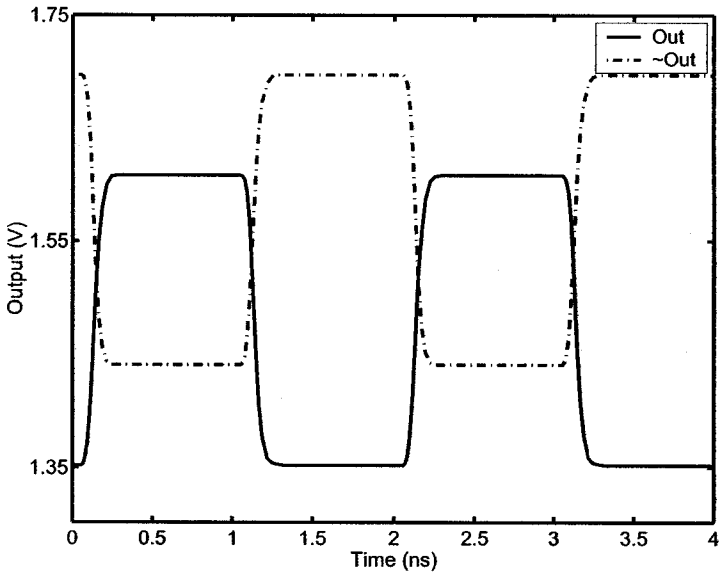

Figure 5.7: Output of MCML gates with a VSR of 89% when the input is applied to upper-level transistors

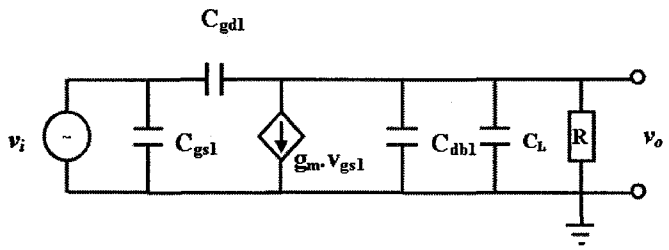

Figure 5.8: MCML inverter small-signal model

where $C = C_{intr} + C_L$. The capacitance C_{intr} is the logic gate's intrinsic capacitance. Assuming that the transistors lengths are set to the minimum feature length, the intrinsic capacitance may be written as $C_{intr} = a_{intr}W + b_{intr}$. Physically, a_{intr} represents all the parasitic capacitances that depend on the transistor's width W. The coefficient b_{intr} represents the other portion of the logic-gate parasitics that are independent of the transistors width.

The assumption here is that the input is an impulse. It was reported in [29] that a factor of 1.5 rather than 0.69 gives a better estimate of the delay when the gate's input has a finite slope, which is a more practical case when approximating the delay across a series of gates. As will be seen later, the delay models that will be derived in the next sections include tunable coefficients. Curve fitting techniques are used to minimize the models errors, and hence, the exact factor value is not critical to the accuracy of the model.

Low-Current Region

When the tail-current $I_{ss} \leq I_L$, we have

$\Delta V = \Delta V_{min} + s$

$W = W_{min}$

The delay may be estimated as

$$D_L = 0.69\frac{(C_{intr} + C_L)\Delta V}{I_{SS}} \tag{5.2.2}$$

In the low-current region, the transistor width is set to W_{min} and hence the intrinsic capacitance is constant. It is then safe to conclude that for any tail-current $I_{SS} \leq I_L$, the delay is directly proportional to the voltage swing.

High-Current Region

When the tail-current $I_{ss} > I_L$ and the voltage swing is equal to the minimum swing ΔV_{min}, we have

$\Delta V = \Delta V_{min}$

$W > W_{min}$

The total capacitance can be approximated as a linear function of the transistor width W. The tail-current can be expressed as

$$I_{SS} = k\frac{W}{L}(\Delta V)^{\alpha} \tag{5.2.3}$$

Substituting for I_{SS} and C_{intr} in equation 5.2.2 and rearranging, we get

$$D_H = 0.69\frac{(a_{intr}W + b_{intr} + C_L)\Delta V}{k\frac{W}{L}(\Delta V)^{\alpha}} \tag{5.2.4}$$

Using equation 5.2.3, the transistor width may be written as

$$W = \frac{I_{SS}L}{k(\Delta V)^{\alpha}} \tag{5.2.5}$$

By substituting for W in 5.2.4, we can expose the relationship between the delay and the voltage swing in the high-current region.

$$D_H = 0.69\left[\frac{a_{intr}L}{k}\frac{1}{\Delta V^{\alpha-1}} + \frac{(b_{intr} + C_L)\Delta V}{I_{SS}}\right] \tag{5.2.6}$$

The derivative of the delay expression can convey a lot of information about the delay sensitivity to the voltage swing.

$$\frac{\partial D}{\partial(\Delta V)} = 0.69\left[\frac{a_{intr}L}{k}\frac{(1-\alpha)}{\Delta V^{\alpha}} + \frac{b_{intr} + C_L}{I_{SS}}\right] \tag{5.2.7}$$

The optimum voltage swing for the lowest delay is

$$\Delta V_{opt} = \sqrt[\alpha]{\frac{(\alpha-1)I_{SS}L}{k}\frac{a_{intr}}{b_{intr}+C_L}} \tag{5.2.8}$$

It is concluded that in general, a small swing is desirable for low tail-current values, while for currents higher than I_L, increasing the voltage swing improves the delay up until a certain value ΔV_{opt}.

5.2.2 MCML Universal Gate Delay Model

In the previous section, we derived a delay model for the basic MCML inverter. In this section, the same procedure is repeated to derive the delay expression for the MCML universal gate. Figure 5.9 shows the small-signal model of the MCML universal gate half-circuit.

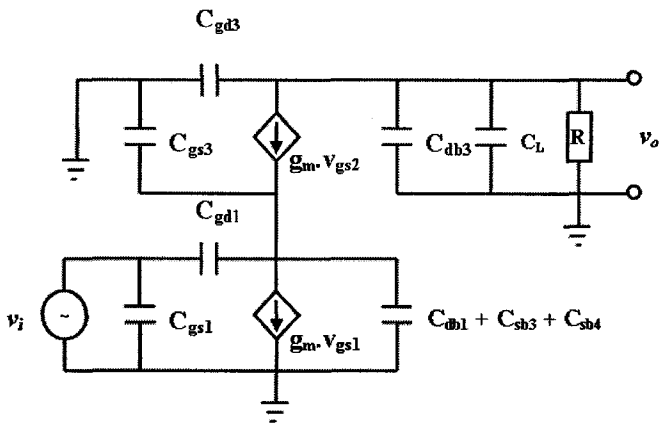

Figure 5.9: Standard universal gate small-signal model

The worst case delay for the universal gate occurs when the input is applied to one of the lower-level transistors and the output is picked up at the drain of M5. Because of the addition of the balancing transistor, the propagation delay from any of the lower-level transistors to either of the output branches is almost identical. In this analysis, it is assumed that the input is applied at the gate of M1 and the output is at the drain of

M5. M4 and M5 gates are connected to V_{DD}, while M3's gate is connected to $V_{DD} - \Delta V$. This arrangement is equivalent to an NAND(NOR) gate with one of its inputs always Low(High).

In the universal gate small-signal model, capacitance can be lumped into three loads: C_1, C_2 and C_L . Capacitor C_1 represents the portion of the intrinsic capacitance seen by the load resistance R . The capacitance C_2 is the other portion that is charged through the upper-level transistor's equivalent resistance $1/g_m$. The capacitance C_L represents the fan-out and wiring capacitance at the output node and is assumed, for now, to be independent of the logic gate's size.

$$\begin{aligned} C_1 &= 2C_{gdo} + 2\left[C_{jsw}(2z + W) + C_j zW\right] \\ C_2 &= 3C_{gdo}W + (2/3)C_{ox}WL + 3\left[C_{jsw}(2z + W) + C_j zW\right] \end{aligned} \tag{5.2.9}$$

where C_{gdo} is the gate to drain overlap capacitance per unit length, C_{jsw} is the drain-to-bulk side wall junction capacitance per unit length, C_j is the drain/source floor to bulk junction capacitance per unit area and z is the drain region extension. The total delay is [14]

$$D = 0.69\left[\frac{(C_1 + C_L)\Delta V}{I_{SS}} + \frac{C_2}{g_m}\right] \tag{5.2.10}$$

The cascading of the NMOS transistors in the universal gate causes the lower-level transistors, M1 and M2, to operate in triode when their respective branches are on. This arrangement makes the upper-level transistors, M3, M4 and M5, in control of the voltage V_x. In this case the current equation will be slightly different than the one used earlier. The current now is expressed as

$$I_{SS} = k\frac{W}{L}(\Delta V - V_{DS1})^{\alpha} \tag{5.2.11}$$

where V_{DS1} is the voltage drop across M1. Using this result and following the same procedure as done for the inverter in Section 5.2.1, the delay for the two operation regions is expressed as

$$D_L = 0.69\left[\frac{(a_1 W_{min} + b_1 + C_L)\Delta V}{I} + \frac{(a_2 W_{min} + b_2)(\Delta V - V_{DS1} - s)}{\alpha I}\right] \quad (5.2.12)$$

$$D_H = 0.69\left[(a_1\Delta V + a_2\frac{\Delta V - V_{DS1}}{\alpha})\frac{L}{k(\Delta V - V_{DS1})} + \frac{(b_1 + C_L)\Delta V}{I} + \frac{b_2(\Delta V - V_{DS1})}{\alpha I}\right] \quad (5.2.13)$$

The complete derivation of equations 5.2.12 and 5.2.13 is shown in Appendix C. This expression suggests that the delay due to the wiring and the fan-out capacitance is directly proportional to the voltage swing. The other portion of the delay has a weak inverse relationship with the swing. Hence, if the wiring and fan-out capacitances dominate the gate intrinsic capacitance, then a smaller swing is desirable. On the other side, if the transistors-width dependent capacitance is dominant, then a larger swing is optimal. This can be rationalized by noting that when the swing is small a large transistor is needed to sufficiently switch the tail-current. Figures 5.10 and 5.11 show the model delay versus current and voltage swing respectively for a fan-out of 2.

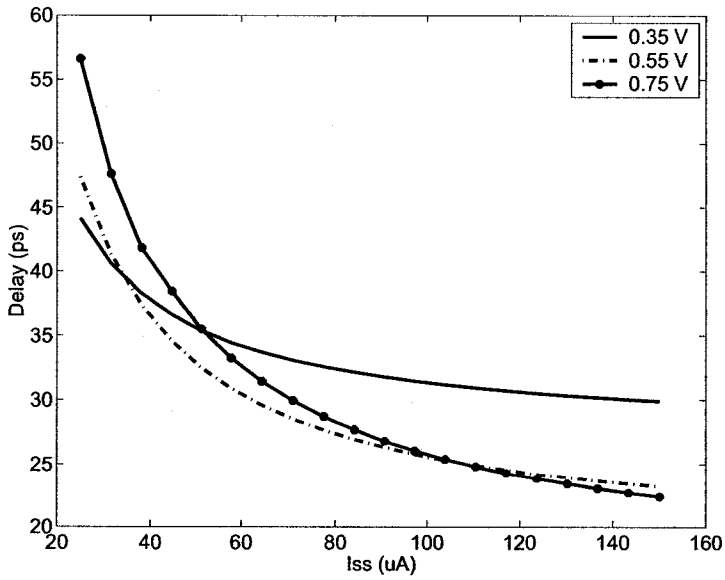

Figure 5.10: Model delay versus tail-current for a fan-out of 2

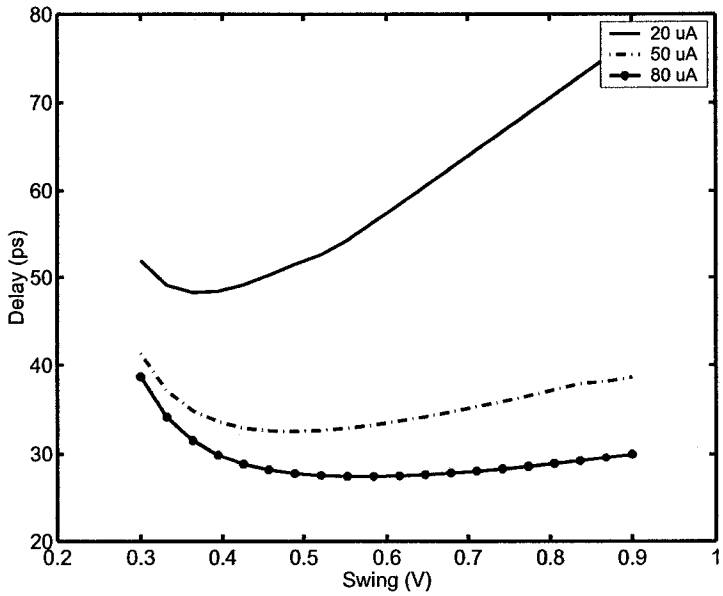

Figure 5.11: Model delay versus voltage swing for a fan-out of 2

Figures 5.12 - 5.14 show plots of the model delay versus fan-out for tail-currents of 20 μA, 100 μA and 140 μA respectively. In Figure 5.13, the delay is lowest for all fan-outs when the swing is 0.35 V. In Figure 5.14, a voltage swing of 0.75 V is optimum for a fan-out of 1. If the fan-out number is between 2 and 7 then a swing of 0.55 V is optimum. For a fan-out higher than 7, the best voltage swing is 0.35 V. The conclusion is that smaller swings are desirable for large fan-outs and low power designs. For high speed applications, larger voltage swings are usually needed to achieve the lowest delay.

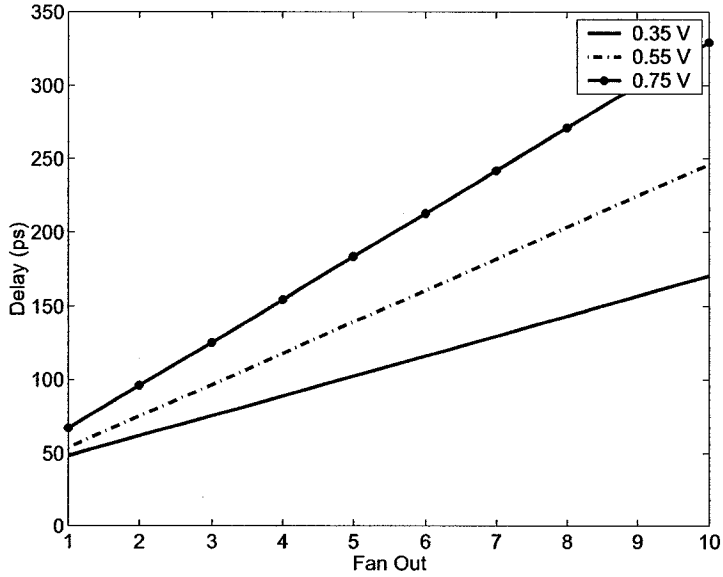

Figure 5.12: Model delay versus fan-out for a tail-current of 20 μA

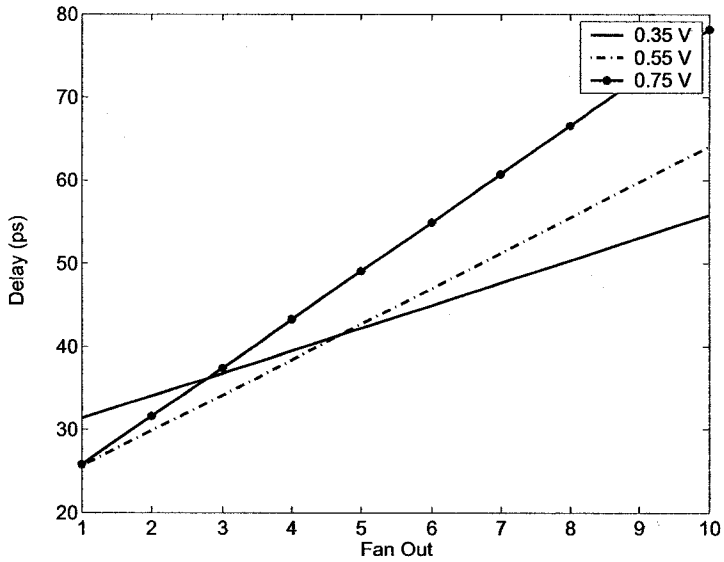

Figure 5.13: Model delay versus fan-out for a tail-current of 100 μA

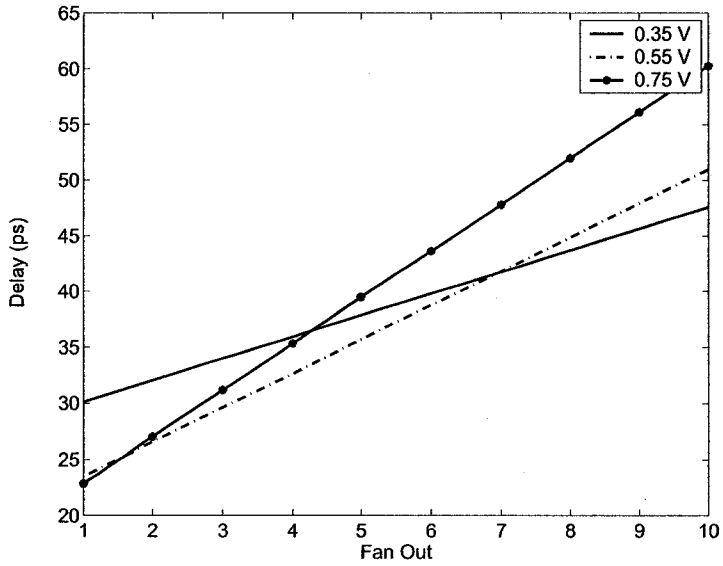

Figure 5.14: Model delay versus fan-out for a tail-current of 140 μA

5.2.3 Model Approximation - Bridging the Gap

So far, we have developed a delay model that is an explicit function of only the tail-current, voltage swing and the technology dependent parameters k , W_{min} , L_{min} , $a_{1,2}$ and $b_{1,2}$. The delay model, however, has some discontinuity in the barrier between the high and low-current regions at the point I_L. It is very important to insure that the model does not have regions where the slope changes rapidly. Figure 5.16 shows the delay versus the tail-current for the delay model and an improved version of the model. The dashed curve is an approximation of the original model. The approximation is obtained using weight functions. A region with a radius r is defined with the current I_L at its center. If the tail-current is higher than $I_L + r$ then the delay is calculated using the high-current delay equation. If the current is lower than $I_L - r$ then the low-current delay equation is used. Within the transitional region, the delay is approximated as a weighted sum of the two equations. The delay model becomes

$$D = \begin{cases} D_H : I_{SS} \geq I_L + r \\ D_L : I_{SS} \leq I_L - r \\ w_H D_H + w_L D_L : I_L - r < I_{SS} < I_L + r \end{cases} \tag{5.2.14}$$

The weight functions w_H and w_L are given by

$$\begin{aligned} w_H &= \tfrac{1}{2r}(I_{SS} - I_L - r) + 1 \\ w_L &= \tfrac{1}{2r}(-I_{SS} + I_L + r) \end{aligned} \tag{5.2.15}$$

The weights are plotted in Figure 5.15 versus the tail-current when $I_L = 30$ μA. There are many elegant approximation techniques that can be used to yield a smooth curve in the transitional region [23], but the developed linear technique is sufficient for the purposes of this thesis. Figure 5.16 shows the model delay before and after the linear approximation.

When applied to a mathematical solver to find the optimum operating points for lowest delay, the discontinuous model caused the solver to struggle around the point I_L. The modified model, on the other hand, has better convergence rate and the solver always reaches the optimum solution regardless of the location of the starting point.

Figures 5.17 and 5.18 show 3-dimensional views of the proposed mode. The delay model is plotted against the tail-current and the voltage swing.

5.3 Model Validation

In this section, we assess the accuracy of the delay model. This is done by comparing the model to simulation results for different tail-currents, voltage swings and output loads.

Tables 5.2 and 5.3 show the model accuracy for a standard fan-out load $C_L = 1.125$ fF. The model average error is 3.6% .

Table 5.2: Delay Model Accuracy

Mean Error	Max Error	Min Error	σ
3.6%	12.9%	0.01%	2%

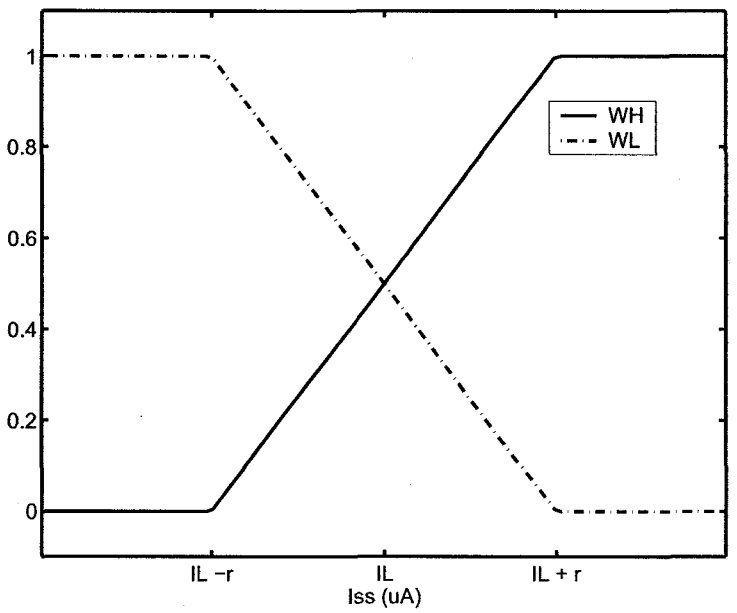

Figure 5.15: Illustration of the weights versus tail-current. $I_L = 30\ \mu$A

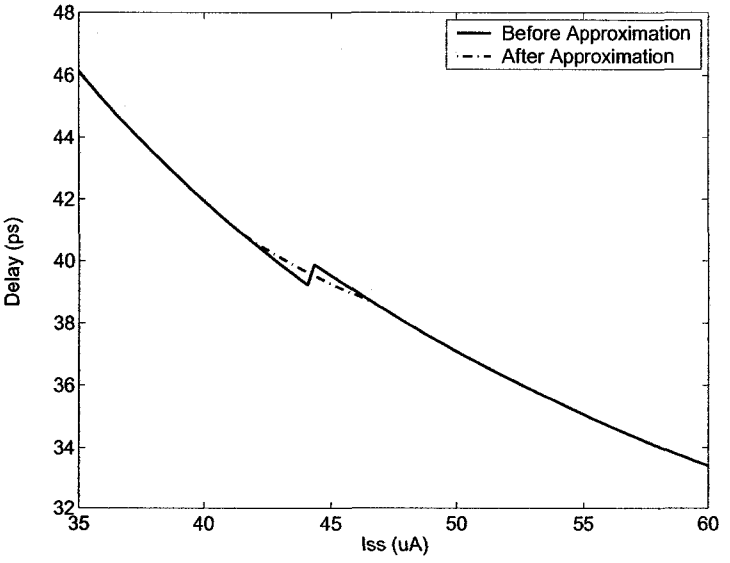

Figure 5.16: Comparison between the original model and approximated model

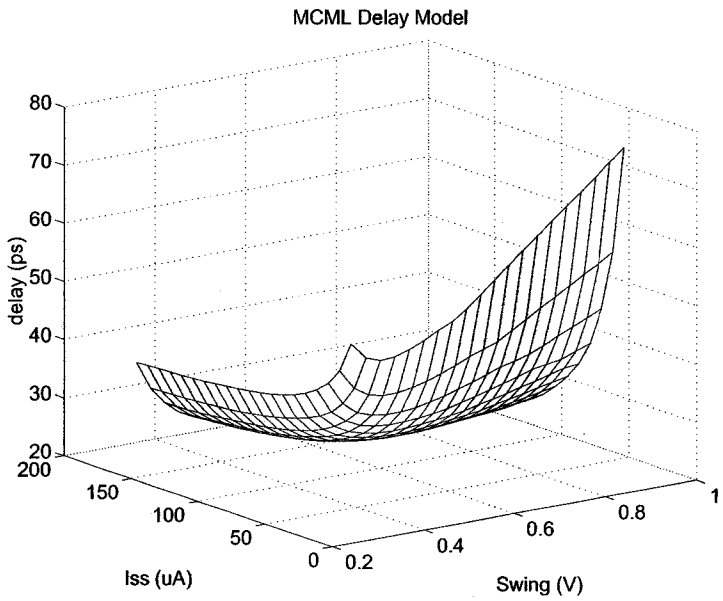

Figure 5.17: 3-dimensional view - Delay model plotted against tail-current and voltage swing

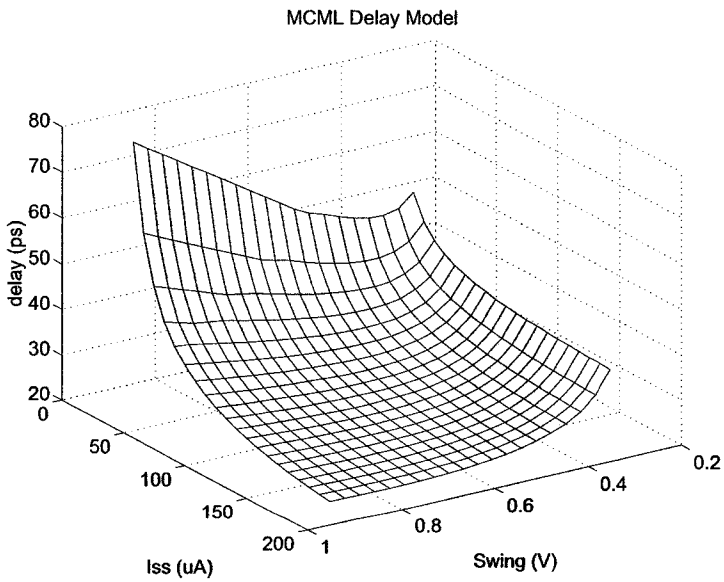

Figure 5.18: 3-dimensional view - Delay model plotted against tail-current and voltage swing

Table 5.3: Model error for various currents

Current	Mean Error
20 μA	3.3%
100 μA	3.2%
140 μA	3.9%
200 μA	5%

Figures 5.19 - 5.22 show a comparison between the proposed model and spectre simulation results. In Figure 5.19, the delay is plotted against the tail-current for voltage swings of 0.55 V and 0.65 V. Figures 5.20, 5.21 and 5.22 show the delay as a function of the output load for voltage swings of 0.35 V, 0.55 V and 0.75 V.

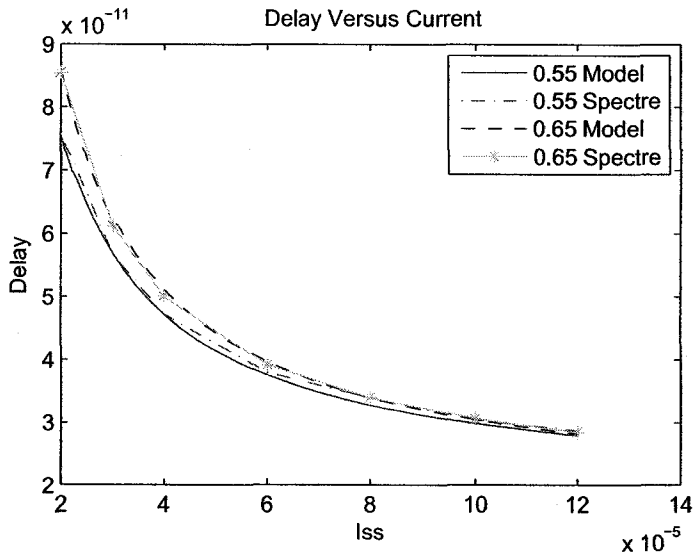

Figure 5.19: Comparison between the model and spectre - Delay versus current

Table 5.4 lists the expected and fitted values of the model coefficients. There are many reasons for the difference between the expected and the fitted numbers. The delay model is based on Elmore's approximation which assumes an impulse input. The model is fitted to measurements obtained from a logic gate that is driven by a similar gate and hence the

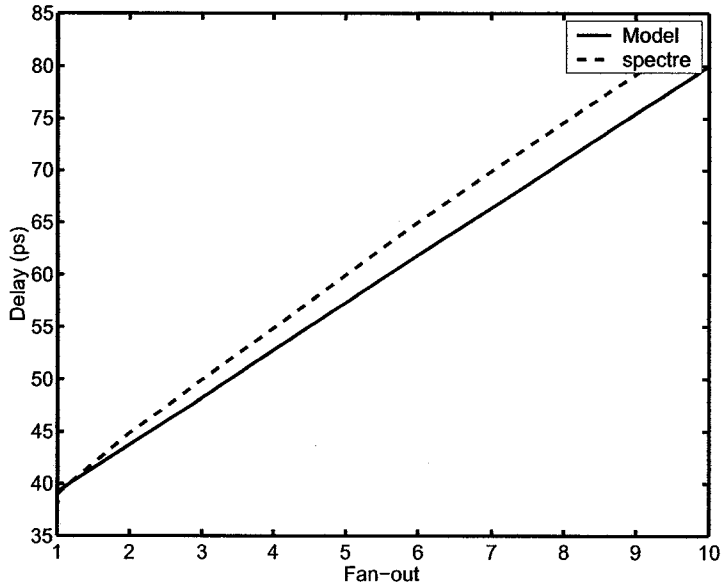

Figure 5.20: Comparison between the model and spectre - Delay versus fan-out for a tail-current of 60 μA and voltage swing of 0.35 V

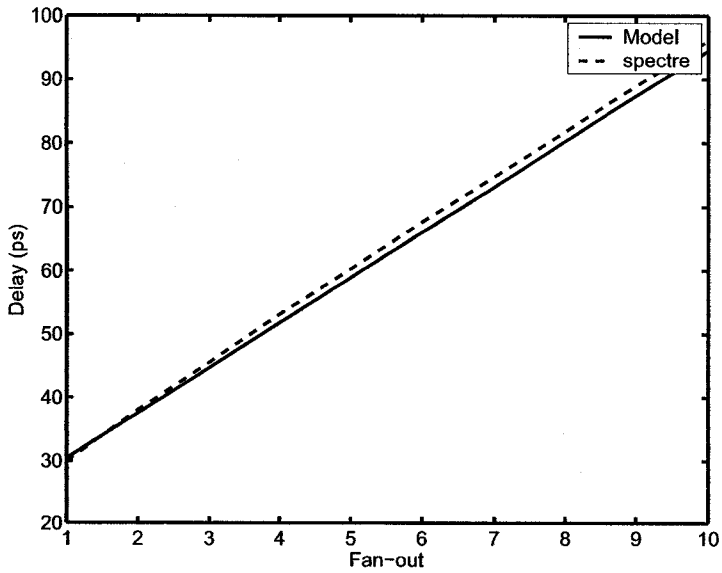

Figure 5.21: Comparison between the model and spectre - Delay versus fan-out for a tail-current of 60 μA and voltage swing of 0.55 V

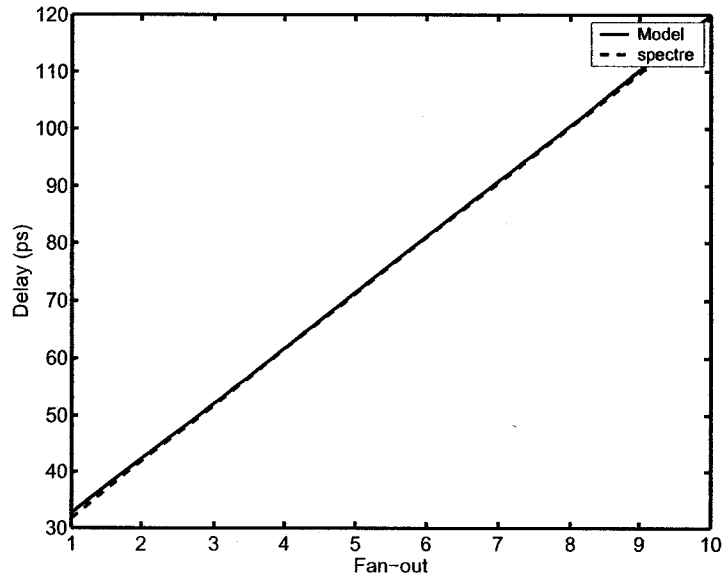

Figure 5.22: Comparison between the model and spectre - Delay versus fan-out for a tail-current of 60 μA and voltage swing of 0.75 V

input signals have finite slopes. Ignoring the biasing effect on the junction capacitance is another reason for the discrepancy between the expected and fitted coefficients values.

Table 5.4: Delay model technology dependent parameters

Coefficient	Expected	Fitted
a_1	2.05E-9	1.96E-9
a_2	7.00E-9	5.59E-9
b_1	6.10E-16	3.49E-16
b_2	1.80E-15	8.00E-16

5.4 MCML with Active Load

Until this point, we have considered only MCML gates with resistive loads. Resistive loads are not suitable for large scale integration for a number of reasons. First, integrated resistors consume larger area when compared to active devices. Moreover, integrated resistors

values often experience large process variation, which reduces the accuracy of the delay models. To combat the unpredictability, designers resort to increasing resistance values to insure that the gates will have enough gain to operate properly under all circumstances. The penalty is a reduction in the circuit power-delay-product.

One drawback of using active loads, on the other hand, is that they impose a limit on the maximum allowable swing. This is because the PMOS loads must operate in the linear region. As the voltage swing increases the devices start to show nonlinear behavior. For this reason, the analysis in this chapter will assume that the PMOS's gates are connected to ground in order to allow the maximum linear range. If a larger swing is desired, then a negative voltage supply is required to extend the PMOS devices linear region. The purpose of this section is to modify the delay models in (5.2.13) to facilitate the inclusion of PMOS active loads. The work in this part has been delayed until later to allow us to focus on the main objective and to reduce the complexity of the procedure by dividing the work into subtasks.

The delay model is based on Elmore's approximation. Based on this, we can add an extra term to account for the PMOS contribution to the gate delay. This extra term has the form

$$D_{PMOS} = 0.69\frac{C_{PMOS}\Delta V}{I_{SS}} \tag{5.4.1}$$

where $C_{PMOS} = C_{gdo}W_p + C_{ox}W_pL_p/2 + C_jzW_p + C_{jsw}(2z + W_p)$

Ideally, we would like to keep the PMOS transistor dimensions as small as possible to improve the power-delay-product. Using the linear-region current equation, the nominal current value at which the PMOS parasitic capacitance is lowest may be expresses as

$$I_{nom} = 2k_p\frac{W_{min}}{L_{min}}\left[(V_{DD} - V_{TP})\Delta V - \frac{\Delta V^2}{2}\right] \tag{5.4.2}$$

where V_{TP} is the PFET threshold voltage. For any voltage swing value ΔV, if the current is to be increased, then the PMOS width must be increased accordingly to preserve the equality in 5.4.2. On the other hand if the current is to be lower than I_{nom}, then the length

must be increased while the width is kept to a minimum. Based on this observation, we can write separate delay expressions for the two cases to eliminate the transistor widths and lengths from the list of variables.

When $W_p/L_p > W_{min}/L_{min}$, the PMOS capacitance may be expressed as $C_{PMOS} = a_pW_p + b_p$, where

$$\begin{aligned} W_p &= \frac{I_{SS}L_{min}}{2k_p\left[(V_{DD}-V_{TP})\Delta V-\frac{\Delta V^2}{2}\right]} \\ a_p &= C_{gdo} + \frac{C_{ox}L_{min}}{2} + C_jz + C_{jsw} \\ b_p &= 2zC_{jsw} \end{aligned} \tag{5.4.3}$$

where $k_p = \mu_pC_{ox}/2$. By using the equations in 5.4.3 and substituting into 5.4.1, we get

$$D_{PMOS} = 0.69\left[\frac{a_pL_{min}}{2k_p(V_{DD} - V_{TP} - \frac{\Delta V}{2})} + \frac{b_p\Delta V}{I_{SS}}\right] \tag{5.4.4}$$

When $W_p/L_p < W_{min}/L_{min}$, then the PMOS capacitance may be expressed as $C_{PMOS} = c_pL_p + d_p$, where

$$\begin{aligned} L_p &= \frac{2k_pW_{min}\left[(V_{DD}-V_{TP})\Delta V-\frac{\Delta V^2}{2}\right]}{I_{SS}} \\ c_p &= \frac{C_{ox}W_{min}}{2} \\ d_p &= C_{gdo}W_{min} + C_jzW_{min} + C_{jsw}(2z + W_{min}) \end{aligned} \tag{5.4.5}$$

By using the equations in 5.4.5 and substituting into 5.4.1, we get

$$D_{PMOS} = 0.69\frac{\Delta V}{I_{SS}}\left[c_p\frac{2k_pW_{min}\left[(V_{DD} - V_{TP})\Delta V - \frac{\Delta V^2}{2}\right]}{I_{SS}} + d_p\right] \tag{5.4.6}$$

The PMOS delay contribution is a function of the voltage swing, tail-current, supply voltage, PMOS threshold voltage V_{TP} and technology dependent parameters. The expected and fitted values of the model coefficients are listed in Table 5.5 .

Table 5.5: PMOS delay model technology dependent parameters

Coefficient	Expected	Fitted
a_p	3.04E-9	4.14E-9
b_p	7.04E-16	5.839E-16
c_p	9.49E-10	2.9E-9
d_p	1.20E-15	1.10E-16

5.5 Summary

In this chapter, the importance of complete switching for Multi-level MCML gates has been emphasized through simulations and then justified by means of small-signal and large-signal analysis. It was found that incomplete switching causes a leakage current in the 'off' branch. If the leakage current is large enough in proportion to the tail-current I_{ss}, the logic gate may fail to propagate the signal.

To prepare a good mathematical model for solvers, a good understanding of the tradeoffs under the robustness conditions is required. A delay model was developed and compared to simulation measurements. The model hides the delay dependence on the internal voltages and transistor sizes and illustrates the impact of the current I_{ss}, the voltage swing and the fan-out load only. The delay was measured against the current I_{ss}, the voltage swing and load capacitance. Results show that the model average error is 3.6%. The technology parameters were extracted using curve fitting techniques and then compared to predicted numbers. The discrepancies are due to the finite slope of the test input and the disregard to the biasing effect on the junction capacitances.

Chapter 6

MCML Mathematical Program

A few attempts have been made to develop an automatic procedure to optimize the performance of MCML circuits. The most relevant attempts are due to [3] and [5], in which an optimization procedure based on mathematical programming has been developed. In these efforts, the MCML gate performance metrics, namely delay, power and area, are described in terms of circuit voltages, currents, transistor sizes and resistive loads. The set of constraints included the minimum mid-swing gain, maximum leakage current allowable and upper and lower bounds on the DC currents and technology related constrains like the transistors sizes. These programs have nonlinear and tight constraints. If the design to be optimized involves one or a handful of gates, then an educated initial guess may be sufficient to produce a global minimum. As the design complexity increases, however, producing an educated guess becomes harder to accomplish. As it stands, state-of-the-art global optimization techniques are not yet capable of solving large scale nonlinear programs with tight constraints. The solution lies in deriving a model that is simple enough for the solver to handle and is yet accurate enough to give dependable results.

6.1 MCML Modeling

It was established earlier that direct translation of the circuit schematic to a mathematical program produces poor results and does not qualify the model to be used in large-scale problems. In Chapter 5, a delay model for MCML universal gates has been derived and

verified. In the next section, the model is configured for MCML optimization.

6.1.1 Delay Model Conditioning

The previous analysis and model derivation in Chapter 5 has been done by assuming a fixed load capacitance C_L, that is independent of the driver gate size. This assumption is valid when the gate is required to drive a fixed external load. For our purposes, it is of most use to define the load C_L in terms of the design variables. The explicit design variables are the MCML tail-currents and voltage swing. The implicit variables are the load resistance, transistors sizes and internal nodes voltages. To accommodate this, the parameters a_L, b_L and j are introduced to express the load capacitance as a linear function of the gate transistor width and the technology dependent constants. The load C_L can be written as

$$C_L = j(a_L W + b_L) \tag{6.1.1}$$

where W is the width of the driver transistors and

$$\begin{aligned} a_L &= C_{gdo} + (2/3)C_{ox}L \\ b_L &= C_{wire} \end{aligned} \tag{6.1.2}$$

The capacitance C_{wire} represents the wiring capacitance at the output. The parameter j can either be an integer to represent the number of fan-outs when all gates have the same size or a real number that represents the ratio between the sum of the fan-out sizes and the size of the driver.

$$j = \frac{\sum W_{Load}}{W_{Driver}} \tag{6.1.3}$$

By making the necessary substitutions and getting rid of the transistor size W as before, the MCML inverter delay becomes

$$D_{Low} = 0.69\left[\frac{(C_{intr}+ja_LW_{min}+jb_L)\Delta V}{I_{SS}}\right]$$
$$D_{High} = 0.69\left[\frac{(a_{intr}+ja_L)L}{k}\frac{1}{\Delta V^{\alpha-1}} + \frac{(b_{intr}+jb_L)\Delta V}{I_{SS}}\right] \quad (6.1.4)$$

Likewise, the delay model for the MCML universal gate may be expressed as

$$D_{Low} = 0.69\left[\frac{(a_1W_{min}+ja_LW_{min}+b_1+jb_L)\Delta V}{I} + \frac{(a_2W_{min}+b_2)(\Delta V-V_{DS1}-s)}{\alpha I}\right]$$
$$D_{High} = 0.69\left[((a_1+ja_L)\Delta V + a_2\frac{\Delta V-V_{DS1}}{\alpha})\frac{L}{k(\Delta V-V_{DS1})} + \frac{(b_1+jb_L)\Delta V}{I} + \frac{b_2(\Delta V-V_{DS1})}{\alpha I}\right] \quad (6.1.5)$$

6.1.2 Model Accuracy

The model accuracy is measured again for the revised model. The model error for different tail-currents and voltage swings is shown in Table 6.1 . The model average error is 3.84%. The error is highest at high tail-currents and voltage swings.

Table 6.1: Model error for various currents and voltage swings

I_{SS} (μA) / ΔV (V)	0.35	0.55	0.75	0.95
60	4.5%	0.9%	1.5%	6.2%
100	3.6%	3.2%	3.9%	2.1%
140	3.7%	3.5%	3.8%	4.9%
200	3.6%	2.8%	4.2%	9.4%

Table 6.2 shows the fitted values of the technology dependent fan-out coefficients.

Table 6.2: Fan-out technology dependent coefficients

Parameter	Expected	Fitted
a_L	1.45E-9	1.98E-9
b_L	0	6.13E-18

6.2 Defining the Constraints

In this section,we demonstrate the procedure to construct a mathematical program for an MCML logic circuit by means of an example. Suppose it is required to minimize the power dissipation of the circuit shown in Figure 6.1 while meeting a specific timing requirement.

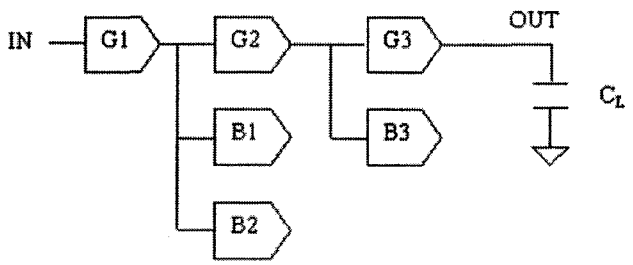

Figure 6.1: A logic circuit example

Also assume that the critical path is from the node labeled IN to the node OUT. The circuit is required to drive the load C_L which could be the input of a storage element. The propagation delay must be less than or equal to a specified time, T_{clk} for example. It is also assumed that the power supply V_{DD} is fixed and hence is not a design variable. The general problem becomes

$$Minimize \quad I_{G1} + I_{G2} + I_{G3} + I_{B1} + I_{B2} + I_{B3}$$

subject to

$$D_{G1} + D_{G2} + D_{G3} \leq T_{clk}$$

$$Gain_m \geq 1 \quad m = 1, ..., 6$$

$$VSR_m \geq VSR_{min}$$

$$\Delta V_{min} \leq \Delta V \leq \Delta V_{max}$$

$$W_{min} \leq W_m \leq W_{max}$$

$$I_{min} \leq I_m \leq I_{max}$$

The term ΔV_{min} in the program is not the same as the value defined earlier as a requirement for complete switching. It is meant to provide a lower bound for the voltage swing.

The delay models that were developed in Chapter 5 do not require the transistor sizes values. Hence, we can immediately discard the lower bound constraints on the transistor size width.

6.2.1 AC Gain

Using the previously made assumptions on complete-switching and robustness, the small-signal gain is estimated as

$$\begin{aligned} A_V &= g_m R \\ g_m &= 2k\frac{W}{L}(0.5\Delta V) \end{aligned} \tag{6.2.1}$$

where $k = u_n C_{ox}$ and g_m is the small-signal mid-swing transconductance. Note that $\Delta V = R \times I_{SS}$. Substituting for g_m and R in the AC gain equation, we get

$$A_V = \frac{2k\frac{W}{L}0.5\Delta V^2}{I_{SS}} = 1 \tag{6.2.2}$$

6.2.2 DC Gain

The DC gain can be expressed as [12]

$$Gain = \frac{\Delta V}{I_{SS}} \times \sqrt{\frac{2I_{SS}kW}{L}} \tag{6.2.3}$$

Also note that

$$k\frac{W}{L} = \frac{I_{SS}}{\Delta V^2} \tag{6.2.4}$$

Substituting again for the term kW/L from equation 6.2.4 into the DC gain expression yields

$$Gain = \sqrt{2} \tag{6.2.5}$$

Thus, when the input voltage swing is larger than the minimum swing required to completely switch the tail-current, the DC gain is always higher than $\sqrt{2}$.

6.2.3 Noise Margin

The noise margin for an MCML inverter is

$$NM = \Delta V \left(1 - \frac{\sqrt{2}}{A_V}\sqrt{1 - \frac{1}{\sqrt{2}A_V}}\right) \tag{6.2.6}$$

A small-signal gain of 1 yields a noise margin of $0.24\Delta V$. This noise margin is relatively low. The differential signalling mode, however, makes it possible to operate the MCML gates safety, even at low noise margins. The low gain and noise-margin have a major advantage in the sense that they reduce the propagation delay.

6.3 The Mathematical Program

The discussions in Sections 6.2.1, 6.2.2 and 6.2.3 reveal that when an MCML gate is completely switched, that is, the gate satisfies

$$I_{SS} \times R \geq \sqrt{\frac{I_{SS}L}{kW}} \tag{6.3.1}$$

Then, the following is also true

$$\begin{aligned} &A_V \geq 1 \\ &Gain_{DC} \geq \sqrt{2} \\ &NM \geq 0.24\Delta V \\ &VSR \approx 100\% \end{aligned} \tag{6.3.2}$$

Based on these results, the MCML mathematical program may now be reduced to

$Minimize \;\; I_{G1} + I_{G2} + I_{G3} + I_{B1} + I_{B2} + I_{B3}$

subject to

$$D_{G1} + D_{G2} + D_{G3} \leq T_{clk}$$

$$\Delta V_{min} \leq \Delta V \leq \Delta V_{max}$$

$$I_{min} \leq I_m \leq I_{max}$$

The upper bound on the voltage swing is determined by observing the requirement that the current sink transistor must remain in saturation. That is

$\Delta V_{max} = V_{DD} + V_T - V_{x,min}$.

Back to the optimization example, having developed the mathematical program, the question becomes how to express the sizes of the gates relative to each other if the transistors widths were eliminated as variables. Referring to the MCML delay model in Section 5.2.2, it was mentioned that j may be the number of fan-outs if all the gates in the design have the same sizes, or the ratio between the sum of the fan-outs sizes to the size of the driver. So for the first gate G1 in the example, the number j is

$$j_{G1} = \frac{W_{G2} + W_{B1} + W_{B2}}{W_{G1}} \tag{6.3.3}$$

When I_{SS} is larger than I_L, the transistor width W for gate m may be expressed

$$W_m = \frac{I_m L}{k \Delta V^{\alpha}} \tag{6.3.4}$$

where I_m is the tail-current of gate m and L is the transistor length. The length L is set to the minimum feature size. By substituting for W_m in 6.3.3, the number j_{G1} becomes

$$j_{G1} = \frac{I_{G2} + I_{B1} + I_{B2}}{I_{G1}} \tag{6.3.5}$$

When a gate is operating in the low-current region, then $I_{SS} \leq I_L$ and the transistor width $W = W_{min}$. The number j for such a gate is

$$j = \frac{\sum W_{Load}}{W_{min}} \tag{6.3.6}$$

Assuming that the voltage swing equals to ΔV_{min}, then the tail-current I_L may be

expressed as

$$I_L = k\frac{W_{min}}{L}\Delta V^2 \tag{6.3.7}$$

Thus, when the gate is operating in the low-current region we can replace the gate's tail-current with I_L . This result can be extended to the case when one or more of the fan-out gates is operating in the low-current region. If the second branch B2 in the example operates in the low-current region, then

$$j_{G1} = \frac{I_{G2} + I_{B1} + I_L}{I_{G1}} \tag{6.3.8}$$

Table 6.3 shows a head-to-head comparison between the different mathematical models for MCML gates in terms of complexity. The symbol N denotes the number of gates in the design.

Table 6.3: Proposed Model complexity compared to previous work

Attribute	[3]	[5]	This Work
Variables	$10N+1$	$7N+1$	$N+1$
Equality Constraints	$2N+1$	$3N$	0
Inequality Constraints	$11N$	$2N$	1

6.4 Model Convexity

The proposed delay models in Section 5.3 have substantially reduced the complexity of the MCML optimization problem. Another potential advantage of the new model is convexity. In most cases, labeling a multi-variable function as convex is a strong claim since it is usually hard to prove convexity. The model at hand is simple and its feasible domain is small making it easier to assess the model convexity.

Convexity can be proven theoretically by showing that the function satisfies the conditions discussed in Chapter 3. Mprobe, a mathematical programming assessment tool,

draws information about the convexity of a function by picking random pairs of points in the feasible domain and testing whether the line segment connecting the two points is completely above the function graph [30].

In this work, the focus is on the ability of the solver to find the global optimum solution. The proposed model will be assessed theoretically and practically. The theoretical approach follows the procedure used in Mprobe. The results should provide valuable information into the shape of the model and expose the regions where the function might be nonconvex. The practical approach involves two tests. The first test is carried out by running the solver many times with different initial solutions. Before each run the solver is provided with a random initial solution and the results are then collected and analyzed. The aim of this test is to sense whether the mathematical program produces different results for different initial guesses. The second practical test is to solve the model using a global optimization method and compare the results to the output of the local optimizer.

6.4.1 Analytical Test

In chapter 3, we stated that a function f is convex on a convex set S if

$f(\theta p_1 + (1-\theta)p_2) \leq \theta f(p_1) + (1-\theta)f(p_2)$

In other words, f is convex if the line segment connecting the points $(p_1, f(p_1))$ and $(p_2, f(p_2))$ lies on or above the function graph [20]. Figure 6.2 illustrates the convexity condition.

To assess the convexity of the proposed model, an algorithm has been developed and implemented in MATLAB. The code picks a large number of random-pairs of points in the feasible domain and checks whether the line segment between any of the pairs is below the model graph. It was found that the convexity condition was often violated around the value I_L . This is expected since the model is not continuous around I_L . The severeness of this violation and its effect on the convergence towards the right solution will be investigated further in the next sections. Figure 6.3 shows a segment of the model where convexity is violated.

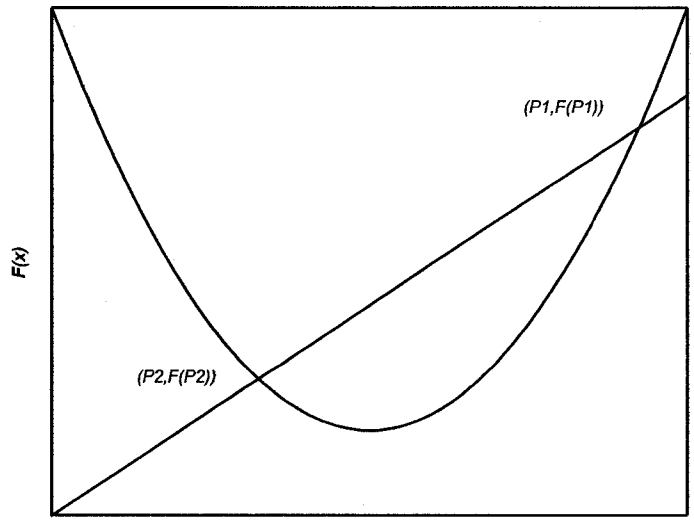

Figure 6.2: Illustration of the convexity condition

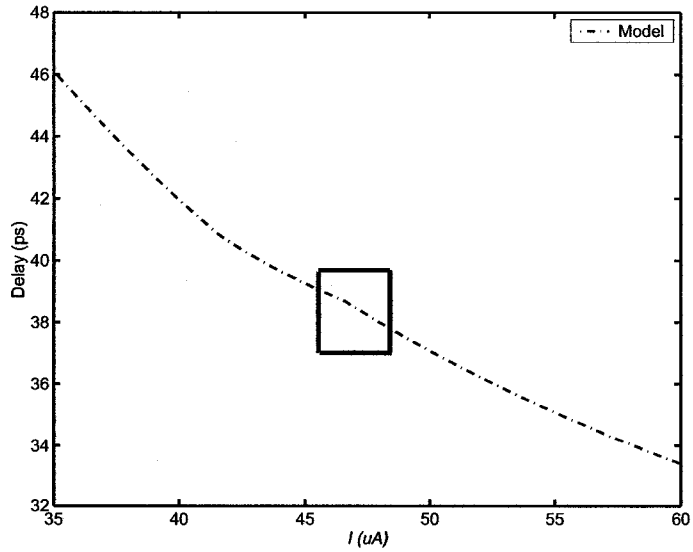

Figure 6.3: A segment of the model curve where convexity is violated

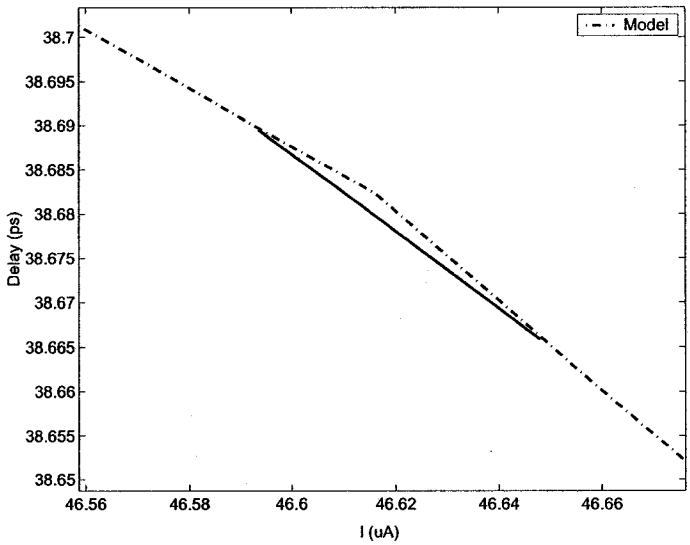

Figure 6.4: The model non-convex segment magnified for illustration

The graph shows that even though a part of the line is below the function, there are no critical points that might trap the mathematical solver.

6.4.2 Practical Tests

Varying the Starting Points

One property of a convex program is that if it has a local minimum x_{opt}, then x_{opt} is also a global minimum [21] [31]. The proposed MCML optimization program has been solved using a local optimization technique known as Sequential Quadratic Programming (SQP) [21]. To test whether the solution is global, the initial point is varied randomly in the feasible design space. It has been found that regardless of the starting point position, the solver has always reached the same solution but with different execution times. The assumption here is that the initial guess is a reasonable one.

Table 6.4 shows the results of an experiment to probe the efficiency of the proposed mathematical program. In this setup, the program is run 100 times with randomly generated initial points. It is required to find the tail-currents and voltage swing that achieve the minimum delay through a 3-gate chain for a maximum power dissipation of 216 μW.

To compare the performance of this program to previous work, a mathematical program similar to the one proposed by [3] and [5] has been constructed and applied to the same problem using the same numerical solver. The results are shown in Table 6.4. In the case of the proposed model, the resultant delay varied from 103 ps to 110 ps with an average of 105 ps. The other program however had a minimum delay of 132 ps and an average delay value of 1806 ps. This shows that the vast majority of the results are actually very far from the global minimum and in order to find the global minimum, the program must be solved a multiple number of times with many initial points to find an acceptable solution. As the design becomes larger, more iterations are required to find the global solution.

Table 6.4: Optimization results and execution times of the proposed model compared against previous work

Statistic	This work	[3], [5]
Number of Iterations	100	10,000
Average Objective Value	105 ps	1,806 ps
Maximum Objective Value	110 ps	4,197 ps
Minimum Objective Value	103 ps	132 ps
Average Power Consumption	216 μW	213 μW
Maximum Power Consumption	216 μW	216 μW
Minimum Power Consumption	216 μW	111 μW
Average CPU time	0.59 s	0.19 s
Maximum CPU Time	1.83 s	2.8 s
Minimum CPU Time	0.23 s	0.11 s

Note that the absolute value of the minimum objective function does not necessarily tip the scales towards one method or the other. The most important figure is the number of attempts carried to find a reasonable solution.

In Figure 6.5, the objective function values that resulted from solving the program in [3] for 10,000 times are normalized to the global minimum value and plotted against

their respective number of occurrences. The plot shows that only 10 iterations out of 10,000 have resulted in an objective function value that is within 10% of the global minimum.

The results also show that the proposed program requires more CPU time than the program in [3]. It will be shown in section 6.4.2 that the proposed program actually converges to the global solution regardless of the location of the starting guess. This means that the solver requires more iterations to converge, if the initial guess is far from the global solution. On the other hand, the programs in [3] and [5] have numerous valleys in the feasible domain. Hence, the solver will quickly proceed downhill to the nearest local minimum.

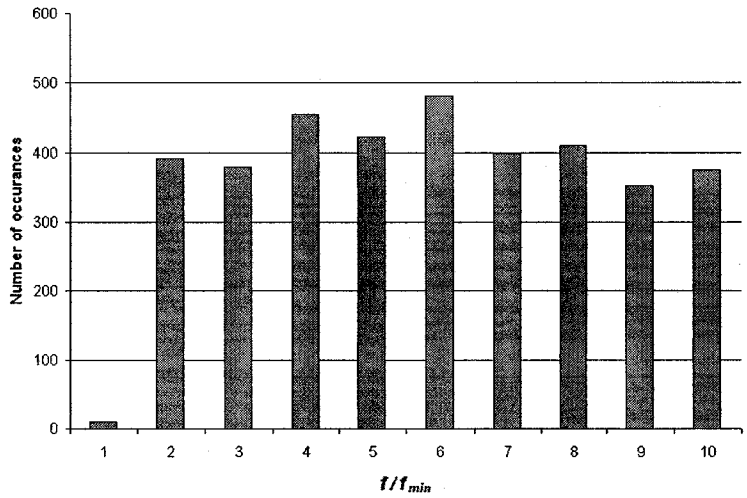

Figure 6.5: Number of occurrences versus the solution value normalized to the global minimum value after 10,000 iterations with the program in [5]

Global Optimization

The second practical experiment involves finding the solution to the problem using a global optimization method. In this test, Simulated Annealing method is used to find the global

solution [32]. A description of the simulated annealing algorithm is provided in Appendix-A. The results are then compared to the output from the SQP algorithm, a local optimization method. The results for different circuits are shown in Table 6.5. Results show that outputs from both the local optimizer and the global optimizer are identical. This means that the proposed model has only one valley in the feasible region. Also note that the global optimizer requires much longer time than the gradient based SQP algorithm.

Table 6.5: A comparison between the results of the simulated annealing technique and the SQP algorithm

	I_1 (μA)	I_2 (μA)	I_3 (μA)	$\sum I$ μA	$\Delta V(V)$	D (ps)	CPU (s)
SA- $1N$	179	N/A	N/A	179	0.82	21.2	863
SQP- $1N$	180	N/A	N/A	180	0.85	21.2	0.3
SA- $2N$	110	89	N/A	199	0.76	51.5	894
SQP- $2N$	104	96	N/A	200	0.77	51.6	0.5
SA- $3N$	79	59	62	200	0.70	87.2	891
SQP- $3N$	69	63	67	199	0.72	87.3	0.6
SA- $10N$	N/A	N/A	N/A	319	0.74	324	1001
SQP- $10N$	N/A	N/A	N/A	320	0.73	329	1.5

Where N is the number of gates in the path. The basic simulated annealing algorithm can not handle constraints. To use simulated annealing to solve the proposed MCML program, some modification to the algorithm or the model is needed. Luckily, in our case, the model is simple and can easily be modified by using penalty methods. In penalty methods, the constrained problem is converted into an unconstrained one by amalgamating the constraints into the objective function. This is done by introducing a penalty term to the objective function. A penalty function imposes a penalty for infeasibility. Appendix A contains a detailed description of penalty methods. The procedure is to assign the objective function a high value when one or more of the constraints is violated. The penalty function is typically required to be continuous and once or twice differentiable depending on the

method used. When the method used is heuristic and uses function evaluations only, as in the case of simulated annealing, it is sufficient to have walls around the feasible region even if this produces discontinuities in the merit function. An algorithm that emulates the barrier effect on the model is outlined in Figure 6.6.

```
Start

if ΔV_min ≤ ΔV≤ ΔV_max
     penalty = objfun;
else
     penalty = +infinity
     return
end

if max(I_1,I_2,...) ≤ I_max && min(I_1,I_2,...) ≥ I_min
     penalty = objfun;
else
     penalty = +∞
     return
end

if V_DD ∑(I) ≤ P_max
     penalty = objfun
else
     penalty = +∞
     return
end/
```

Figure 6.6: Algorithm to evaluate the simulated-annealing cost function

6.5 The Algorithm

To verify the applicability of the mathematical model, an optimization algorithm has been developed in MATLAB and used as a test bench. The algorithm proceeds as follows:

1. Read the circuit netlist. The netlist describes the gate level circuit schematic. The netlist may be represented by a table that consists of 4 columns. Column 1 shows the gate numbers, column 2 lists the first input node numbers, column shows is the second input node numbers and column 4 lists the gate output node numbers.

2. Extract all the possible paths from all the inputs to any of the outputs.

3. Find the critical path. Determining the critical pathes of large logic circuits is a complex procedure [18] [33]. In this example, a simple function has been developed. The function takes all the possible paths, calculates the delays along each path and returns the longest path from the set.

4. Prepare the delay model for the critical path to be used as an objective function.

5. Solve the mathematical program. The mathematical program is passed to a general purpose solver which returns the optimum tail-current and voltage swing values.

6. Collect the results and calculate the circuit component values (W, R, W_s, W_p, L_p).

6.6 Design Example I: 4-bit Carry Ripple Adder

In this example, the procedure to optimize a combinational MCML circuit is demonstrated. It is required to minimize the worst case propagation delay of a 4-bit carry ripple adder for a given maximum power dissipation. To simplify the example, only a 1-bit Full Adder design will be discussed in detail. The results for the 4-bit adder are tabulated at the end of the section. Figure 6.7 shows the full adder circuit schematic.

Table 6.6 shows the node number assignments. These call numbers will be used later to construct the circuit netlist and identify the critical paths.

6.6.1 Mathematical Program

The mathematical program for this example may be written as

Minimize $Delay$

subject to $\sum I_{ss,m} \leq I_{max} \qquad m = 1, .., 6$

For simplicity, it is assumed that the XOR gates have identical delay models to the universal gates. In practice, the delay models would still have the same form but with slightly different coefficients values. We also assume that both gate inputs have the same delay, that is the worst case delay.

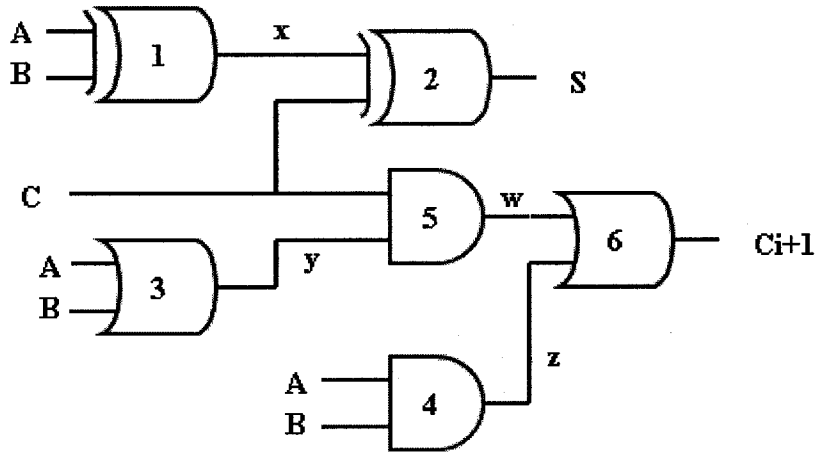

Figure 6.7: A Full Adder schematic

Table 6.6: Full Adder node assignments

Node Name	Node Number
A	1
B	2
C	3
S	5
C_{i+1}	9
X	4
Y	6
Z	7

6.6.2 Netlist

The full adder netlist is shown in Table 6.7 .

Table 6.7: Full adder Neltist

Gate	Input 1	Input 2	Output
1	1	2	4
2	4	3	5
3	1	2	6
4	1	2	7
5	6	3	8
6	8	7	9

6.6.3 Branching Table

The algorithm then calls the function *branch*. This function takes the netlist as an input and produces a 'branch table' as in Table 6.8.

Table 6.8: Branch table

Gate	Branch
1	2
2	0
3	5
4	6
5	6
6	0

The entries in the second column are the call numbers of the fan-out gates. If one logic gate's fan-out number is assigned zero in the table, like the case in rows 2 and 6, then that logic gate does not have any designable fan-outs and its output is a design output. In the

example, the outputs of gates 2 and 6 are the sum and the carry signals. The table may also be expanded horizontally if a gates is driving more than one gate.

6.6.4 Critical Path

There are many ways to identify the critical path. This type of problem is known as the shortest path problem in network programming. Many algorithms to solve such a problem are available [34, 35]. For this example we will exploit the branching table format to write a simple algorithm to identify the critical path. The fan-out entries in Table 6.8 serve as pointers to the rows corresponding to the fan-out gate numbers. For example, the fan-out entry in the first row is 2. This means that gate 2 is the fan-out of gate 1. But gate 2 fan-out information is available in row number 2. This gives a set of all the possible paths. After all the possible paths are determined, a function called *crt-path* reads all the possible paths, calculates the delay for each path and returns the longest path and its corresponding delay. Table 6.9 lists all the possible paths. Path 1, for example, involves gates 1 and 2.

Table 6.9: Possible paths table

Path 1	Path 2	Path 3	Path 4	Path 5
1	2	3	4	5
2	0	5	6	6
0	0	6	0	0

6.6.5 Objective Function

The critical path is identified by the function *crt-path.* This information is passed on to the objective function which is responsible for constructing the delay model for the critical path. For this example, if the critical path was determined to be Path 3 = [3, 5, 6], then the delay model is

$$D = D_3 + D_5 + D_6 \tag{6.6.1}$$

The fan-out number j for gates 3 and 5 is calculated according to equation 6.3.3 . Gate 6 may be assigned a fixed output load C_L.

6.6.6 Optimization

The program is then passed on to a nonlinear optimizer. Note that in this specific example, all the constraints are linear. The last step is to collect the optimization results and calculate the transistors sizes and resistor values that yields the optimum currents and voltage swing. The algorithm is as follows

```
Start

Initialize: x = x0

while (new_delay < old_delay)

     Identify the critical path;

     minimize the objective function along
     the critical path;

     x0 = x;
end

calculate Ri, Wi, Wsi

return Ri, Wi, Wsi

end
```

Figure 6.8: MCML universal gate design and optimization algorithm

6.6.7 Results

The algorithm is applied to the Full Adder under the constraint $I_{TOTAL} \leq 500\mu$A . The Full Adder worst case delay is 83 ps. Table 6.10 shows the gates sizes, currents and voltage swings.

Table 6.11 shows the 4-bit Ripple Carry Adder critical delay and power dissipation. The RCA results are also compared to a similar work reported in [15]. A schematic of the

Table 6.10: Full Adder optimization results

Gate	I_{SS} (μA)	W (μm)	W_s (μm)	R (KΩ	ΔV (V)	I_L (μA)
1	130	0.69	5.15	5.63	0.74	39
2	113	0.59	4.42	6.55	0.74	39
3	94	0.49	3.66	7.92	0.74	39
4	26	0.22	1	28.7	0.74	39
5	62	0.33	2.42	11.9	0.74	39
6	73	0.38	2.84	10.2	0.74	39
$\sum$	500	2.7	19.5	N/A	0.74	39

4-bit adder is shown in Figure 6.9

Table 6.11: 4-bit RCA optimization results

4-bit MCML RCA	Power (mW)	Model Delay (ps)	Simulated Delay (ps)	Error
[15]	5.2	240	261	7.6%
This Work	3.6	210	217	3.2%

Notes and observations about the proposed program are in order. Only MCML universal gates were used to construct the 4-bit RCA. The Length of the logic transistors is kept to the minimum allowable L_{min}. In [3], transistor lengths were set to $2L_{min}$ to improve the fabrication yield. On the other hand, this would kill the delay and is not suitable for high speed and low power applications. The Length of the current source transistor L_s was made to 500 nm to suppress the channel length modulation effect. This is critical for mixed-signal applications with low tolerance to noise. The tail-current control voltage V_n has been set to 0.75 V. Increasing this voltage will reduce the current source transistor size significantly but will cause the transistor to fall off saturation if the voltage swing was high. The full adder circuit has a worst case delay of 83 ps. In practice, a designer would arrange the circuit such that the data would pass through the gates inputs with the lower delay. In such a setting, the FA's delay could be cut to less than 50 ps.

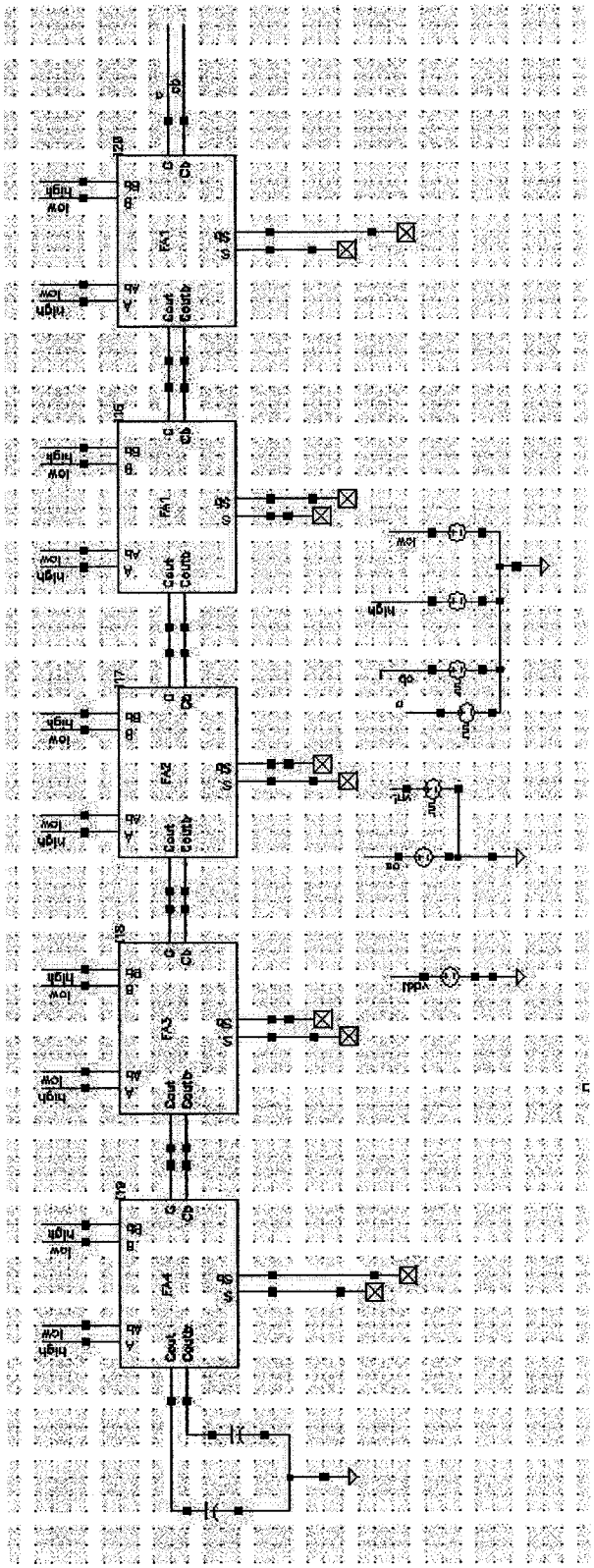

Figure 6.9: 4-bit RCA schematic in Cadence

6.7 Design Example II: 8-bit Decoder/DeMultiplexer

Decoders are commonly used in memory interfacing circuits. Next, an 8 to 256 decoder is optimized using the proposed algorithms. We will assume that the decoder is intended to drive memory word-lines with a capacitance of 500 fF each. The first word line WL_0 may be expressed in terms of the inputs as

$$WL_0 = \bar{A}_0.\bar{A}_1.\bar{A}_2.\bar{A}_3.\bar{A}_4.\bar{A}_5.\bar{A}_6.\bar{A}_7 \tag{6.7.1}$$

Figure 6.10 shows a schematic of the topology that will be used in this example. The topology involves only NAND gates and inverters. Even though the design may be realized by using AND gates only, adding the extra stages reduces the effort of each stage.

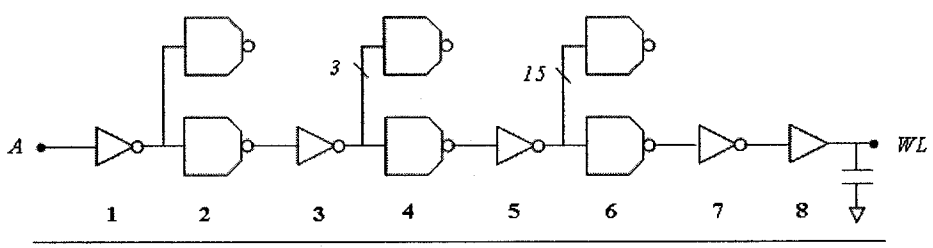

Figure 6.10: 4-bit RCA schematic

The objective is to minimize the power dissipation for a maximum critical delay of 1 ns. The mathematical program becomes

Minimize $8I_1 + 16I_2 + 16I_3 + 32I_4 + 32I_5 + 256I_6 + 256I_7 + 256I_8$

Subject to

$D_1 + D_2 + D_3 + D_4 + D_5 + D_6 + D_7 + D_8 \leq 1n$

$\Delta V_{min} \leq \Delta V \leq \Delta V_{max}$

$I_{min} \leq I_m \leq I_{max}$

The coefficients in the objective function represent the number of the gates of each stage in the whole decoder. The optimization results are shown in Tables 6.12 and 6.13.

The mathematical program yields a minimum power dissipation of 180 mW. The model error is within 4.6% when compared to Spectre simulation measurements.

Table 6.12: 8-bit Decoder optimization results

Stage	I_{SS} (μA)	W_n (μm)	W_p (μm)	L_p (μm)	ΔV (V)
1	58	1.27	0.46	0.18	0.37
2	30	0.66	0.22	0.18	0.37
3	46	1.01	0.34	0.18	0.37
4	24	0.52	0.22	0.25	0.37
5	62	1.37	0.49	0.18	0.37
6	16	0.34	0.22	0.34	0.37
7	59	1.3	0.47	0.18	0.37
8	300	6.6	0.24	0.18	0.37

Table 6.13: 8-bit decoder theoretical and measured delays

Model delay (ps)	Spectre delay (ps)	Error
1000	1046	4.6%

6.8 Model Flexibility

Schematic level transistor models use a default value for the drain region extension from the gate. This length is set to 0.48um in the kit used for this work, which is also the smallest extension allowable by the technology when the drain has a metal interconnect. Figure 6.11 shows a transistor with the minimum dimensions allowable for a drain region in 0.18μm technology. In a multi-transistor layout, drain regions areas are varied to achieve certain floor planning criteria. This coupled with wiring capacitance reduce the delay model accuracy.

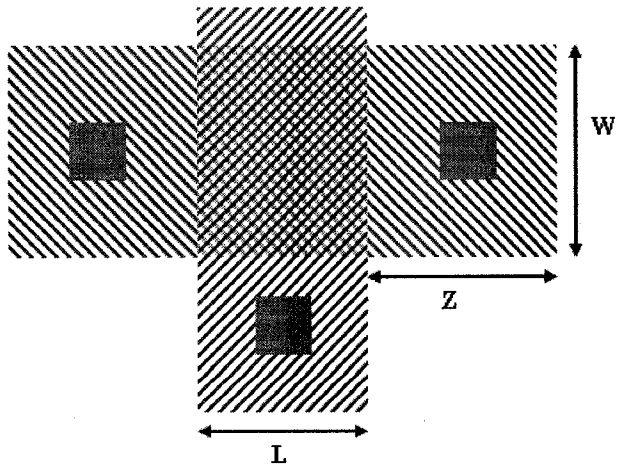

Figure 6.11: MOSFET Layout

Fortunately, the proposed delay model is immune to such degradation. First, the model includes three terms that represents the delay contributions of capacitors that are independent of the gate size. That includes the wiring capacitance and the capacitance of the drain side walls parallel to the gate length. Secondly, the model parameters can be easily adjusted to minimize the model error for any particular design level (Schematic, Layout, Extracted, Chip). This is done by fitting the model to a set of collected measurements from the intended level. Table 6.14 shows the fitted parameters for the Extracted view. A Layout of an MCML universal gate with a swing of 0.55 V and tail-current of 20 μA is shown in Figure 6.12.

6.9 Mathematical Program Efficiency

In this section, the MCML gates that are designed by the proposed procedure are compared to CMOS in terms of power efficiency. Comparing the power dissipation of CMOS versus MCML is not an obvious task, since CMOS's power dissipation is due to many factors other than the input frequency. The switching activity of the gates is an important factor that

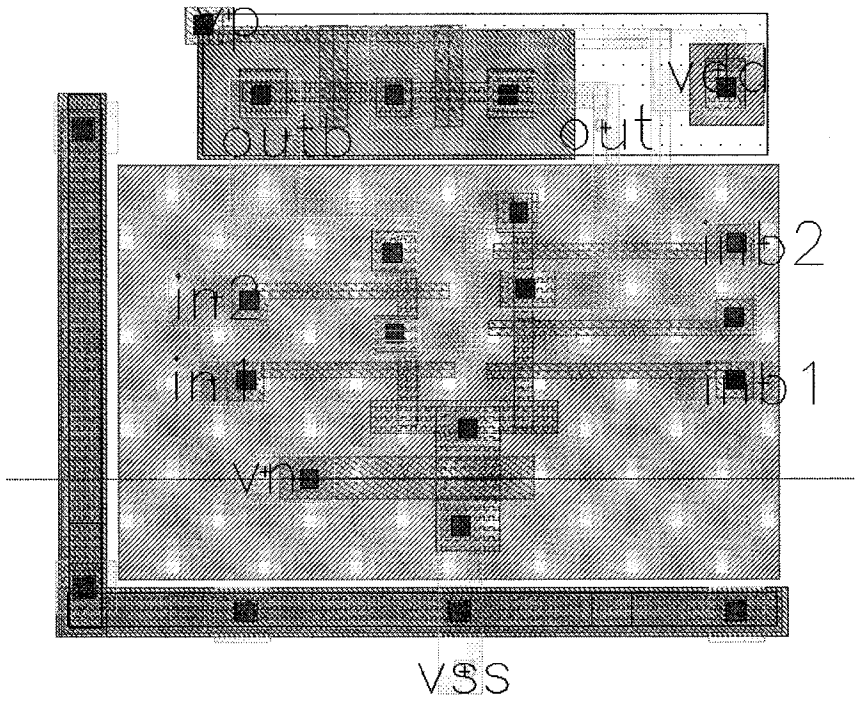

Figure 6.12: MCML NAND gate layout with a tail-current of 20 μA and a voltage swing of 0.55 V

Table 6.14: Extracted view model coefficients

Coefficient	Value
a_1	3.05E-9
b_1	5.15E-15
a_2	3.04E-9
b_1	5.00E-16
a_L	2.05E-9
b_L	1.16E-16
a_p	2.10E-9
b_p	1.99E-15
c_p	4.30E-9
d_p	1.68E-15

cannot be easily estimated, since it depends on the input's switching probability, the gate function (AND, OR, NAND, NOR) and the design architecture. Nonetheless, estimating the crossing point at which MCML becomes more efficient than CMOS is a worthy cause.

To make a fair comparison, the following assumptions are made. The gates have a propagation delay of 47 ps while driving an output load of 12 fF. The input combinations applied are 00, 01, 10 and 11, thus covering all possible combinations. Figures 6.13 and 6.14 show power dissipation comparisons for a NAND and a NOR gate respectively. Simulation results show that the MCML NAND gate is more efficient than its CMOS counterpart at 1.8 GHz. The intersection point for the NOR gate is 1.4 GHz. That is much higher than the reported value of 300 MHz in [7] for the case of the CORDIC DSP unit. Note that the comparison in this thesis is made between individual gates while the conclusion in [7] is based on the performance of the whole DSP circuit.

One may justify the discrepancy between the findings here and the reported numbers in [7] by looking at the fact that CMOS designs tend to have a large number of inverters to avoid using the inefficient and bulky AND and OR gates. This is not required in MCML,

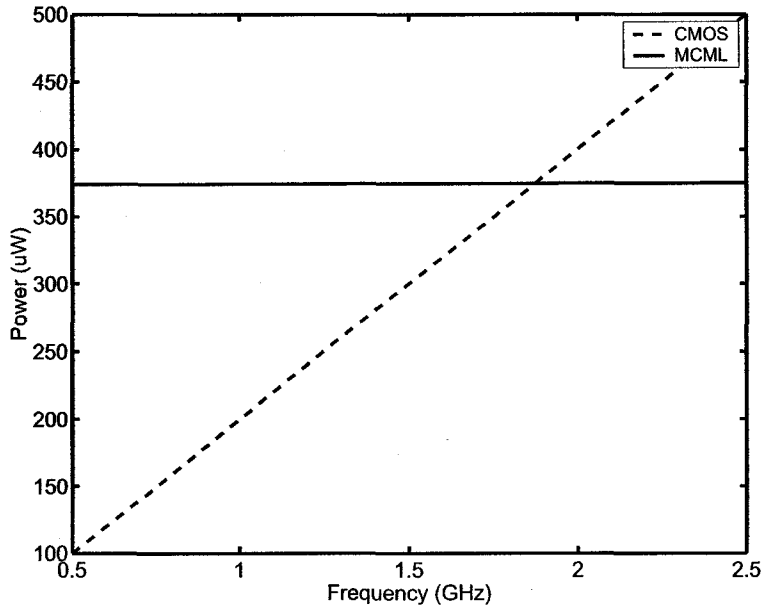

Figure 6.13: Power comparison between CMOS NAND gate and its equivalent MCML universal gate

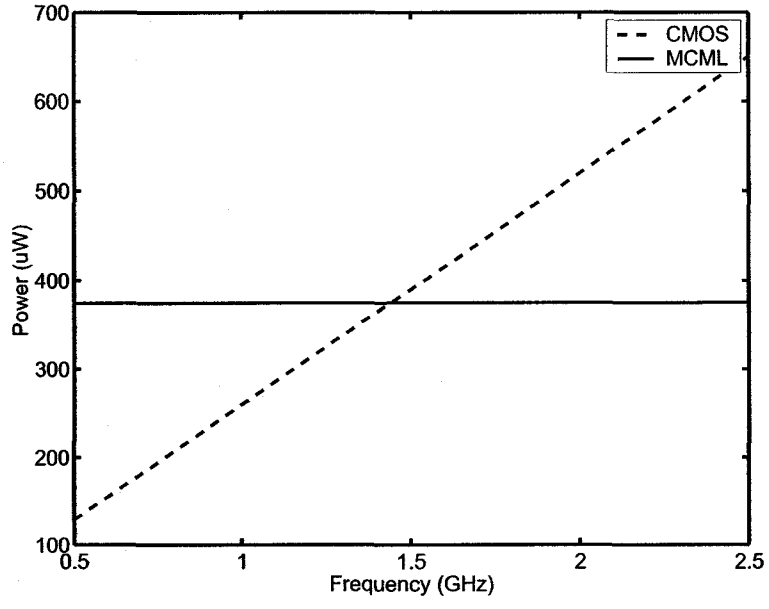

Figure 6.14: Power comparison between CMOS NOR gate and its equivalent MCML universal gate

since universal gates may realize any of the basic function by simply interchanging the inputs and outputs. The inverted signals are also readily available. It was also found that the input capacitance of CMOS gates is usually higher than their MCML counterparts. This means that using the same output load in the previous comparison puts MCML at a disadvantage. That is because MCML gates drive other MCML gates with lower capacitance than that CMOS would drive. The CMOS NOR gate transistors sizes, for example, are 2 μm and 4 μm for the NMOSs and the PMOSs respectively. While the MCML gate transistors size is 2.1 μm. Hence, the CMOS input capacitance in this case is three times larger than that of MCML.

It is safe to say, therefore, that the most accurate way to identify the crossing point between CMOS and MCML is to implement the full design using both logic styles. Only then, a definite decision into which implementation is the most feasible can be made.

6.10 MCML Design Automation Procedure

The following procedure outlines a proposed MCML design flow.

1.Choose the appropriate delay model: Use equations (5.15) and (5.20) for the single level gates, e.g. MCML inverte, or equations (5.27) and (5.28) for 2 level gates, e.g. universal gate, MUX, DLatch.

2. Extract the BSIMv33 DC model parameters for the targeted technology, namely α, k_n , k_p , λ and V_T .

3. Calculate the approximate values for the delay model coefficients. These values will be used as initial points for the curve fitting step.

4. Measure the delay for MCML gates with different tail-currents, voltage swings and fan-outs. Too a few measurements may yield poor accuracy. Too many points require more computing resources. In this work, 30 points fairly distributed over the feasible space were sufficient to yield good accuracy. Size the logic transistors such that the gates have a VSR

of about 98%. A Lower VSR may cause the gates to fail. A higher VSR on the other hand greatly degrades the gate delay.

5. Use a curve fitting algorithm to extract the coefficients values that minimize the average model error, namely a_1 , b_1 , a_2 , b_2 , a_L , b_L , a_p , b_p , c_p , d_p . A good practice is to break this step into two stages. First, the delay model with a fixed load is fitted with a known capacitive load C_L value to extract all the parameters except a_L and b_L . In the second stage, the general delay model is used to extract the load parameters a_L and b_L by fitting the model using delay measurements for gates with different fan-out.

Chapter 7

Concluding Remarks

7.1 Research Contribution

In this thesis, a new method for the automatic design and optimization of MCML has been proposed. The method is based on a modified version of the standard differential-pair MCML universal gate. The motivation for the modifications to the standard universal gate topology is due to two factors. The asymmetry of the standard universal gate creates a major obstacle for the implementation of an equation-based automatic optimization of complex MCML digital designs. The asymmetric topology necessitates the introduction of tight nonlinear constraints.

The imbalance between the two MCML universal gate branches causes serious performance problems in high speed applications. For the same power dissipation, the modified topology has a 54% higher operation frequency over the standard universal gate. This improvement can be traded for power dissipation and silicon area. On the mathematical programming front, applying the symmetric topology has helped reduce the optimization problem size by getting rid of some redundant constraints and variables.

The proposed mathematical program is based on delay and power models that express their respective metrics in terms of the voltage swing, tail-current and process dependent coefficients only. When compared to spectre simulation results, the delay model shows an average error of 3.7% and a maximum error of 9.7%. The delay model has also been modified for use in optimization problems that involve multiple logic gates. To accomplish

this, the relative gates sizes and their respective input capacitance loads are also expressed in terms of the tail-currents. Thus, we are able to eliminate the transistor widths from the variables set. The proposed mathematical program represents a circuit of N gates with $N+1$ variables, compared to $7N+1$ variables for the most recent works in the same topic [3], [5] . The model has one inequality constraint only, while [3] and [5] have $11N$ and $2N$ constraints respectively.

To apply the mathematical program to large designs, an algorithm has been implemented in MATLAB. The algorithm reads in the circuit netlist, the power and delay requirements, converts the data into a mathematical program, solves the optimization problem using a general purpose gradient based nonlinear solver and finally calculates the optimal transistors sizes, bias voltages and resistance values if applicable.

The proposed optimization algorithm has been used to optimize a 4-bit ripple carry adder and a 8-bit decoder. The design involves 24 logic gates or 144 transistors. A number of theoretical and practical tests were carried out to verify the convexity of the program. Results have shown that the model converges rapidly to the global minimum regardless of the location of the initial guess. This is in large contrast with the results from a mathematical program similar to the ones in [3] and [5] . In this case, only 10 out of 10,000 iterations with randomly picked initial points have resulted in a solution with an objective function value within 10% from the global minimum. The thesis ends with a proposed procedure that outlines the steps to building an MCML design and optimization tool.

7.2 Future Work

The work that has been done so far is limited to MCML buffers and the standard universal gate with a balancing transistor. The procedure, however, is applicable to all symmetric MCML gates. That includes MCML XOR gates, multiplexers and latches. The goal is then to extract the technology dependent coefficients for each logic gate. At that point, the automation tool will have the capability to optimize MCML designs that include all basic logic functions (NOT/AND/OR/XOR), datapath elements (Multiplexer/D-Multiplexers)

and memory elements (Latches/Flip Flops).

The MCML universal gate delay model requires the drain-to-source voltage V_{DS} of the lower-level transistors to be known. The voltage V_{DS} may be calculated by using the linear-region DC current equation. This, in turn, requires calculating the transistors widths, and thus, increases the complexity of the mathematical program. To solve this problem, a number of universal gates with different tail-currents and voltage swings have been simulated, and the voltages V_{DS} were recorded for every case. It was found that V_{DS} varies only slightly around the value 0.2 V. Hence, and for simplicity, the voltage V_{DS} is estimated to be 0.2 V in all the delay models in this work. To improve the accuracy of the model without adding extra expressions, a look-up table may be used to assign the V_{DS} values depending the tail-current and the voltage swing values.

The mathematical program will not be complete without taking into consideration the effects of the logic-gates layout. In this work, we assume that the drain and source areas are equal to the transistors widths times a default drain extension length z. In practice, this is not always true. Depending on some design rules and area requirements, the layout may have shorter or longer drain extensions. Also, some parts of the drain might not have the same width as the transistor gate width. This may be dealt with in the mathematical and the physical layout levels. Mathematically, the model may be altered to include compensation coefficients. In the physical level, layouts may be adapted by using consistent drain shapes for all the different gates, and especially, the drain and source regions that have the most significant effect on the worst-case delay.

Appendix A

Optimization Algorithms

A.1 Penalization Methods

Penalization methods solve a sequence of unconstrained problems that will converge to the constrained problem solution. Each unconstrained problem contains a penalization term. This penalization term value is proportional to the amount of infeasibility. Penalization methods are sub categorized into two groups, Barrier methods and penalty methods. Barrier methods impose a penalty for approaching the boundary of the constraint. These methods work well for inequality constraints. The other group, called penalty methods, impose a penalty for violating the constraints. These are better suited for equality constraints. An ideal penalty auxiliary function has the form

$$\sigma(x) = \begin{cases} 0 : x \in S \\ +\infty : x \notin S \end{cases} \tag{A.1.1}$$

where S denotes the feasible set.

A.1.1 Barrier Methods

Barrier methods use an auxiliary term to impose a penalty for approaching the feasible region barrier. In other words, it forms a wall at the inequality constraint barrier and a downward slope terrain into the interior of the feasible region. The auxiliary function we will use here is continuous in the interior of the feasible region and becomes unbounded at

the boundary of the region. Some of the most frequently used functions as barrier terms are the logarithmic function and the inverse function

$$\begin{aligned} \phi(x) &= -\sum_{i=1}^{m} \log(g_i(x)) \\ \phi(x) &= \sum_{i=1}^{m} \frac{1}{g_i(x)} \end{aligned} \tag{A.1.2}$$

The Barrier function has the form

$$\beta(x, \mu) = f(x) + \mu\phi(x) \tag{A.1.3}$$

When the scalar μ approaches zero, the barrier term $\mu\phi(x)$ will approach the ideal function $\sigma(x)$. Barrier methods solve a sequence of unconstrained optimization problems of the form

$minimize \;\; \beta(x, \mu_k)$

for a gradually decreasing μ. The reason why we solve a sequence of problems with gradually decreasing μ instead of solving one problem with very small μ, is that it is much harder to solve a function that increases sharply close to the boundary. We start with a large μ that will give an easier problem to solve, and then μ is reduced gradually with the solution from the previous iteration used as a starting point for the next.

A.1.2 Penalty Methods

When the problem is an equality constrained problem, the penalty method imposes a penalty for violating the equality constraints. In contrast to the Barrier methods, penalty methods start outside the feasible set and hence the name exterior point methods. As the method proceeds, the penalty term forces the iterates gradually towards the feasible set. The penalty function for constraint violation has the form

$\psi(x) = 0 \;\; if x \in S$

$\psi(x) > 0 \;\; if x \notin S$

where S denotes the feasible set. A widely used penalty term is the quadratic-loss function

$\psi = 1/2 \sum g_i(x)^2$

This is the sum of the squares of the constraints values at point x. The problem becomes an unconstrained minimization problem of the form

minimize $\pi(x, \rho_k) = f(x) + \rho\psi(x)$

where ρ a positive scalar used to control the penalty magnitude. As in the case of Barrier methods, ρ is increased gradually after every iteration to force the solution to the feasible set, where $g_i(x) = 0$.

A.2 Sequential Quadratic Programming

This method is a generalization of Newton's method. As the name implies, at every iteration, the method transforms the constrained problem into a quadratic problem -a quadratic objective function with linear constraints- and solves for the search direction p_k.

Methods for solving the problem

minimize $f(x)$

subjectto $g_i(x) = 0$

can be obtained by applying the optimality conditions to the Newton formula. The lagrangian for the problem above is

$$\nabla L(x, \lambda) = f(x) - \lambda^T g(x) \tag{A.2.1}$$

and the first order necessary optimality condition is

$\nabla L(x, \lambda) = 0$

The search direction is the solution to the Newton equations

$$\nabla^2 L(x_k, \lambda_k) \begin{pmatrix} p_k \\ v_k \end{pmatrix} = -\nabla L(x_k, \lambda_k) \tag{A.2.2}$$

where p_k and v_k are the steps

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} + \begin{pmatrix} p_k \\ v_k \end{pmatrix} \tag{A.2.3}$$

Two modification to this classic technique will be discussed. The first modification is the Quasi-Newton update techniques for the approximation of the Hessian of the Lagrangian to reduce the cost of calculating the exact Hessian. The second modification is related to the method's convergence. At each iteration, we usually insist that the new estimate is a better estimate of the solution. In the unconstrained case this is done using function evaluations to test if the new estimate has significantly improved the objective function. In constrained optimization, progress is commonly measured in terms of a merit function. The merit function is usually the sum of the objective function and amount of in feasibility of the constraints. One example of a merit function is the quadratic penalty function.

$$M(x) = f(x) + \rho \sum g_i(x)^2 \tag{A.2.4}$$

where ρ is a positive number. The greater the value of the scale number the greater the penalty for in feasibility.

A.3 Simulated Annealing

Annealing is the process of heating up a material and then cooling it down at a controlled rate. At high temperatures atoms have high energies and more freedom of movement. As the material cools down, the atoms energy is reduced. A crystal structure is obtained when the system energy level is minimum. If the material is cooled very quickly, undesired defects will occur in the crystal structure. In this case the material structure is as polycrystalline and the system energy level is higher than minimum [32].

The probability distribution of system energies at any given temperature is

$$P(E) \propto e^{|-E/KT|} \tag{A.3.1}$$

where K is Boltzmann's constant, T is the system temperature and E is the system

energy.

In the simulated annealing algorithm the system energy corresponds to the objective function value and the minimum energy level is analogous to the optimum or minimum function value. The algorithm starts by taking an initial point from the user. At this point the system temperature is high. The cost function is evaluated and the result is accepted. In the next iteration, a random point is selected in the neighborhood of the previous point. If the new value of the cost function is less than or equal to the old cost, then the algorithm immediately accepts the new point. If not, the algorithm accepts the new point according to Metropolis's criterion. According to Metropolis's criterion, if $cst_{old} \leq cst_{new}$, the new point is accepted if for a generated random number $0 \leq rand \leq 1$

$$rand \leq e^{-\frac{\Delta cst}{KT}} \tag{A.3.2}$$

where $\Delta cst = cst_{new} - cst_{old}$. This condition allows the algorithm to escape local minima at high temperatures. This process is repeated for a given number of times before the temperature is lowered again. The temperature cooling rate is determined by a cooling schedule. The cooling schedule parameters are an initial temperature, cooling rate, number of iterations before each temperature lowering and a stopping temperature. A standard simulated annealing algorithm flow chart is shown in Figure A.1.

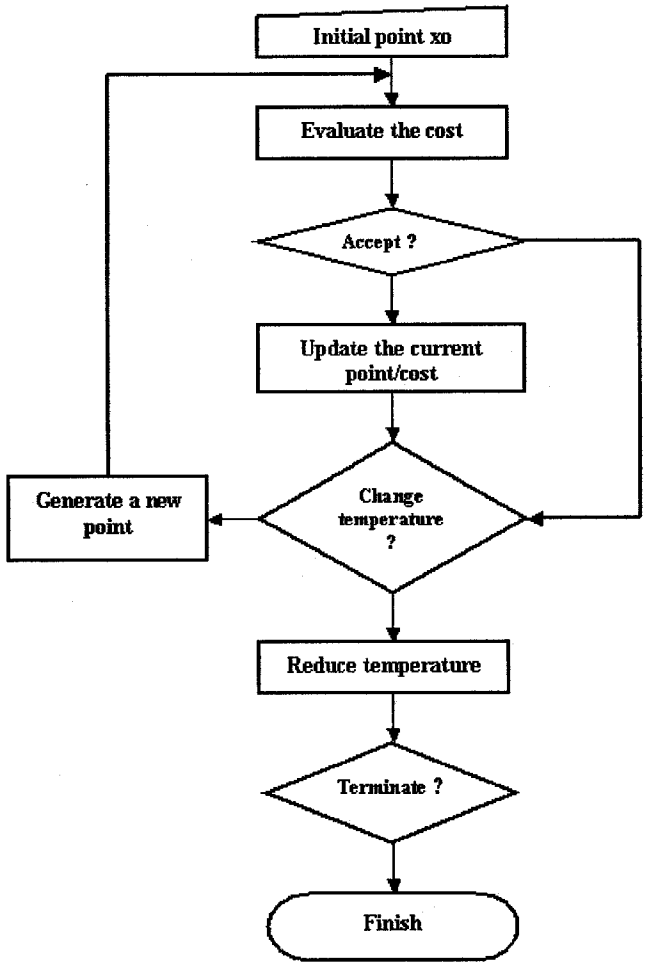

Figure A.1: Simulated Annealing flow diagram

Appendix B

Multi-Level MCML DC Gain

For the standard MCML universal gate, we can express M1 and M2 saturation current in the common mode as

$$i_{D1} = i_{D2} = k\frac{W}{L}(v_{gs} - v_T)^{\alpha} \tag{B.0.1}$$

If a differential signal is applied, the saturation currents become

$$\begin{aligned} i_{D1} &= k\tfrac{W}{L}(v_{in1} - v_T)^{\alpha} \\ i_{D2} &= k\tfrac{W}{L}(v_{in2} - v_T)^{\alpha} \end{aligned} \tag{B.0.2}$$

where

$$\begin{aligned} v_{in1} &= v_{gs} \pm \Delta v \\ v_{in2} &= v_{gs} \mp \Delta v \end{aligned} \tag{B.0.3}$$

The output voltage is given by

$$v_o = R(i_{D1} - i_{D2}) \tag{B.0.4}$$

From B.0.1, the gate to source voltage is

$$v_{gs} = \sqrt[\alpha]{\frac{I_{SS}L}{2kW}} \tag{B.0.5}$$

By substituting for the currents in the output voltage equation and rearranging

$$v_o = Rk\frac{W}{L}\left[\left(\sqrt[\alpha]{\frac{I_{SS}L}{2kW}} + \Delta V\right)^{\alpha} - \left(\sqrt[\alpha]{\frac{I_{SS}L}{2kW}} - \Delta V\right)^{\alpha}\right] \tag{B.0.6}$$

If we use $\alpha = 1$, then

$$\frac{v_o}{v_i} = R\sqrt{\frac{2I_{SS}kW}{L}} \tag{B.0.7}$$

Appendix C

MCML Universal Gate Delay Model

Assuming a resistive load, the delay equals to

$$D = 0.69R(C_1 + C_L) + 0.69\frac{C_2}{g_m} \tag{C.0.1}$$

In resistive load and the capacitances may be expressed in terms of the designable variables as

$$\begin{aligned} R &= \tfrac{\Delta V}{I_{SS}} \\ C_1 &= a_1 W + b_1 \\ C_2 &= a_2 W + b_2 \end{aligned} \tag{C.0.2}$$

In the low-current region, we have

$$\begin{aligned} W &= W_{Min} \\ s &\geq 0 \\ I_{SS} &= k\tfrac{W_{Min}}{L}(\Delta V - V_{DS1} - s)^{\alpha} \\ g_m &= \tfrac{\alpha I_{SS}}{\Delta V - V_{DS1} - s} \end{aligned} \tag{C.0.3}$$

Substituting these expression into the delay equation yields

$$D_L = 0.69\left(\frac{(a_1 W_{Min} + b_1 + C_L)\Delta V}{I_{SS}} + \frac{(a_2 W_{Min} + b_2)(\Delta V - V_{DS1} - s)}{\alpha I_{SS}}\right) \tag{C.0.4}$$

In the high-current region, we have

$$\begin{aligned} W &> W_{Min} \\ s &= 0 \\ I_{SS} &= k\frac{W}{L}(\Delta V - V_{DS1})^{\alpha} \\ g_m &= \frac{\alpha I_{SS}}{\Delta V - V_{DS1}} \end{aligned} \tag{C.0.5}$$

By substituting for these values and eliminating the transistor width W, the delay expression becomes

$$D_H = 0.69\left[\left(a_1\Delta V + a_2\frac{\Delta V - V_{DS1}}{\alpha}\right)\frac{L}{k(\Delta V - V_{DS1})^{\alpha}} + \frac{(b_1 + C_L)\Delta V}{I_{SS}} + b_2\frac{\Delta V - V_{DS1}}{\alpha I_{Ss}}\right] \tag{C.0.6}$$

Bibliography

[1] International Technology RoadMap for Semiconductors, "Radio frequency and analog/mixed-signal technologies for wireless communications," ITRS, Tech. Rep., 2005.

[2] M. Houlgate, "Adaptable MOS current mode logic for multi-band frequency synthesizers," Master's thesis, Carleton University, Ottawa, Canada, 2005.

[3] H. Hassan, M. Anis, M. Elmasry, "MOS current mode circuits: analysis, design, and variability," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 8, pp. 885–898, August 2005.

[4] M. Allam and M. Elmasry, "Dynamic current mode logic (DyCML): a new low-power high-performance logic style," *IEEE Journal of Solid-State Circuits*, vol. 3, pp. 550–558, Mar 2001.

[5] S. Khabiri and M. Shams, "A mathematical programming approach to designing MOS current-mode logic circuits," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, vol. 51, May 2005, pp. 2425–2428.

[6] T.W. Kwan and M. Shams, ""Multi-GHz energy-efficient asynchronous pipelined circuits in MOS Current Mode Logic," in *Proceedings of the 2004 International Symposium on Circuits and Systems*, vol. 2, 2004, pp. 645–648.

[7] J.M. Musicer and J. Rabaey, "MOS current mode logic for low power, low noise CORDIC computation in mixed-signal environments," in *Proc. International Symposium on Low Power Electronics and Design*, 2000, pp. 102–107.

[8] M. Alioto and G. Palumbo, "Design strategies for source coupled logic gates," *Proc. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 5, pp. 640–654, May 2003.

[9] C. Visweswariah, "Optimization techniques for high-performance digital circuits," in *IEEE/ACM International Conference on Computer-Aided Design*, Nov 1997, pp. 198–207.

[10] Jan M. Rabaey, A. Chandrakasan, B. Nikolic, *Digital integrated circuits: A design perspective*, 2nd ed. Pearson Education, Singapore, 2003.

[11] T. Sakurai and A. R. Newton, "A simple MOSFET model for circuit analysis," *IEEE Transaction on ED*, vol. 38, no. 4, pp. 887–894, April 1991.

[12] A. Sedra and K. Smith, *Microelectronic circuits.* Oxford University Press, 1998.

[13] Behzad Razavi, *Design of analog CMOS integrated circuits.* Boston, MA: McGraw-Hill, 2001.

[14] M. Alioto, G. Palumbo, S. Pennisi, "Delay estimation of SCL gates with output buffer," in *Proc. IEEE International Conference on Electronics, Circuits and Systems*, vol. 2, 2001, pp. 719–722.

[15] S. Khabiri, "Design and optimization of Mos current mode logic circuits using mathematical programming," Master's thesis, Carleton University, Ottawa, Canada, 2004.

[16] M. Alioto, L. Pancioni, S. Rocchi, V. Vignoli, "Modeling and evaluation of positive-feedback source-coupled logic," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 12, pp. 2345–2355, Dec 2004.

[17] A.R. Conn, P.K. Coulman, R.A. Haring, G.L. Morrill, C. Visweswariah, Chai Wah Wu, "JiffyTune: circuit optimization using time-domain sensitivities," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, pp. 1292–1309, Dec 1998.

[18] R.K. Brayton, G.D. Hachtel, A.L. Sangiovanni-Vincentelli, "A survey of optimization techniques for integrated-circuit design," *Proceedings of the IEEE*, vol. 69, pp. 1334–1362, Oct 1981.

[19] S. Badel,I. Hatirnaz, Y. Leblebici, "Semi-automated design of a MOS current mode logic standard cell library from generic components," *Research in Microelectronics and Electronics, 2005 PhD*, vol. 2, pp. 155–158,25–28, July 2004.

[20] S. Nash and A. Sofer, *Linear and Nonlinear Programming.* New York:McGrow-Hill, 1996.

[21] Garth P. McCormick, *Nonlinear programming : theory, algorithms, and applications.* New York : Wiley, c1983.

[22] Stephen A. Vavasis, *Nonlinear optimization : complexity issues.* New York: Oxford University Press, 1991.

[23] Gerald W. Recktenwald, *Numerical Methods with MATLAB.* Prentice-Hall Inc., Upper saddle River, New Jersey, 2000.

[24] W. Karush, "Minima of functions of several variables with inequalities as side constraints," Master's thesis, Department of Mathematics, University of Chicago, Chicago, Illinois, 1939.

[25] H.W. Kuhn and A.W. Tucker, "Nonlinear programming," in *Proc. 2nd Barkeley Symposium*, March 1951, pp. 481–492.

[26] Paul R. Gray, *Analysis and design of analog integrated circuits*, 4th ed. New York: Wiley, 2001.

[27] A. Ismail and M. Elmasry, "A low power design approach for MOS current mode logic," in *Proc. IEEE International SOC Conference*, Sept 2003, pp. 134–146.

[28] S. Khabiri and M. Shams, "An MCML four-bit ripple-carry adder design in 1 GHz range," in *Proc. IEEE International Symposium on Circuits and Systems*, vol. 2, May 2005, pp. 23–26.

[29] John Rogers, Calvin Plett, Foster Dai, *Integrated Circuit Design for High-speed Frequency Synthesis.* Boston, Mass. : Artech House, 2006.

[30] J. Chinnek, "Analyzing Mathematical Programs using MProbe," *Annals of Operations Research*, vol. 104, pp. 33–48, 2001.

[31] Reiner Horst and Hoang Tuy, *Global optimization : deterministic approaches.* Berlin; New York : Springer-Verlag, c1993.

[32] D.T. Pham and D. Karaboga, *Intelligent optimisation techniques : genetic algorithms, tabu search, simulated annealing and neural networks.* London ; New York : Springer, c2000.

[33] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proc. IEEE International Conference on Computer-Aided Design*, Nov 1985, pp. 326–328.

[34] William H. Press, *Numerical recipes in C++ : the art of scientific computing.* Cambridge, UK ; New York : Cambridge University Press, 2002.

[35] Dimitri P. Bertsekas, *Network optimization : continuous and discrete methods.* Belmont, Mass. : Athena Scientific, c1998.