BiCMOS for Speed
BiCMOS for Speed

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus
Prof. ir. K.F. Wakker,
in het openbaar te verdedigen
ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op maandag 2 december 1996 te 13:30 uur

door

Viorica SIMION

inginer
Facultatea de Electronică și Telecomunicații – București
geboren te Boekarest, Roemenië
Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. Ralph H.J.M. Otten

Leden van de promotiecommissie:

Rector Magnificus, voorzitter
Prof. dr. ir. R.H.J.M. Otten, promotor
Prof. dr. C.J.M. Beenakker
Prof. dr. ir. J.H. Huijsing
Prof. dr. dipl-ing. J.A.G. Jess
Prof. dr. L.M.F. Kaufmann
Dr. ir. R. Nouta
Dr. ir. C.J.M. Verhoeven

Technische Universiteit Delft
Technische Universiteit Delft
Technische Universiteit Delft
Technische Universiteit Eindhoven
Technische Universiteit Eindhoven
Technische Universiteit Delft
Technische Universiteit Delft

Cip-Data Koninklijke Bibliotheek, Den Haag

Simion, Viorica
BiCMOS for Speed / by Viorica Simion.

Thesis Delft University of Technology.- With ref.- With summary in Dutch.
ISBN 90-5326-025-0
1. Digital integrated circuits - Design and construction. 2. Metal Oxide
Semiconductors, Complementary - Bipolar integrated circuits. 3. Computer
Aided Design.

The text of this book was edited using XEMACS-19.13 running under the LINUX operating
system on a HP Omnibook 5000CTS laptop PC. It is typeset in a 12 points Times-Roman
font by the \LaTeX{} (2e) program. The figures were created using \texttt{xfig}, \texttt{idraw}, \texttt{xmgr}, \texttt{pspice}
and \texttt{matlab}. The bibliography was compiled by \LaTeX{}. Postscript previewing and hacking
was performed using \texttt{ghostview}. Most of the programs used to prepare this document
were made freely available to the public by their programmers. Microsoft programs were
avoided whenever possible.

Copyright © 1996 by Viorica Simion. All rights reserved. No part of this book
may be reproduced in any form or by any electronic or mechanical means (including
photocopying, recording, or information storage and retrieval) without prior
permission in writing from the author.
ISBN: 90-5326-025-0
Printed in The Netherlands
To my mother
Contents

1 Introduction ................................................................. 1
  1.1 Design Style ..................................................... 2
  1.2 Why BiCMOS ? .................................................. 3
  1.3 Synthesis and Optimization ................................... 3
  1.4 Thesis Overview ................................................. 5

2 BiCMOS Drivers ........................................................... 11
  2.1 Basic Configurations ........................................... 12
  2.2 Base Charge Removal .......................................... 14
  2.3 Full-swing Techniques ......................................... 16
  2.4 Driver Analysis ............................................... 18
  2.5 Conclusion ..................................................... 25
  2.a A Device Model for MOS Transistors ...................... 27
  2.b A Device Model for Bipolar Junction Transistors ....... 33

3 Delay Analysis of the Totem-Pole Inverter ......................... 37
  3.1 Some Definitions ............................................... 37
  3.2 Static Characteristic of a BiCMOS Inverter ............... 39
  3.3 Stages in the Pull-up Transient ............................ 40
  3.4 Load Delay Models ............................................ 45
  3.5 Variable Current Gain Model ............................... 47
  3.6 Average Current Gain Model ................................. 52
  3.7 Evaluations ................................................... 55

4 Gate Characterization for CAD ......................................... 57
  4.1 CAD Delay Model ............................................... 57
  4.2 Delay Dependence on Input Slope ............................ 60
  4.3 Gate Power Dissipation versus Input Slope ................ 68
  4.4 Conclusion ..................................................... 73

5 Buffer Optimization ...................................................... 75
  5.1 Device Parameters .............................................. 75
INTRODUCTION

High performance, low cost, and short time to market are the main requirements for application-specific integrated circuits (ASIC's) in today's competitive market. In the early 1970s it was still possible to perform IC design by hand and meet these goals, because the number of components was limited to just few hundred. In the past two decades, advances in processing technology have increased the integration density by no less than 5 orders of magnitude. Nowadays, even ordinary household items such as televisions, telephones and cars are likely to contain integrated circuits with over 1 million transistors. Before the turn of the century, a single chip may contain as much as 100,000,000 transistors. This makes the small fingernail-sized silicon chip by far the 'largest' and most complex man-made structure in the universe. Quite obviously, the design of a chip cannot be performed without the use of advanced CAD tools.

IC CAD tools are extensively used at various levels during the design of an ASIC. CAD tools span the entire spectrum from high-level architectural synthesis, via logic synthesis down to layout placement and routing. Various verification programs exist to check the correctness at each level. Generally, the tools at the lower levels of design are the most established and mature. High-level design tools are a more recent development, and often still in the experimental stage. Unfortunately, despite high expectations in the past decade, a push-button 'silicon compiler' is still a dream. Manual intervention at all design stages is still required to obtain good results. Moreover, processing technology seems to be developing at a faster pace than CAD tools. The tools can hardly keep up with the increased complexity.

In synthesis tools, synthesis and optimization algorithms work together to obtain good results. While the synthesis algorithms ensure a fast design trajectory, the optimization algorithms enhance the design quality.

As indicated before, the task of the designer is not restricted to simple "push button" actions. There is still quite some work to be done manually to overcome the limitations of the software tools. Especially automatic BiCMOS design is a blind spot since most CAD tools are limited to pure CMOS digital circuit and system design. New research and development activities are required to develop dedicated CAD tools, such as for mixed-signal or analog design.
The contribution of this work is to enable high performance CAD tools in these emerging fields. We developed efficient performance optimization techniques which are not confined to the conventional CMOS processes, but which can be extended to other technologies as well, such as BiCMOS. Our optimization techniques cover the widely used CMOS technology without being limited to it.

In the next sections we will present some background information on design styles and target technologies. This is relevant because the choice for one optimization method over another is directly dependent on them.

1.1 Design Style

There are basically two distinct implementation styles for microelectronic circuits: full-custom and semi-custom. In full-custom design the function and the layout of practically every transistor can be optimized. This requires a huge design effort and it is therefore rather expensive. In return, the highest quality circuits can be obtained.

The semi-custom design style uses a limited number of circuit primitives. Each of them is designed by experts and is well characterized. Since the design implementation space is considerably reduced, efficient CAD tools can be developed to design and optimize semi-custom circuits. The resulting circuits may not have the highest performance, but design time and cost are drastically reduced. Nowadays, the semi-custom designs are dominant in the semiconductor market. Full-custom design is only performed for a few small critical components.

The semi-custom style can be further subdivided into a number of subclasses. The foremost implementation styles are, cell-based design, gate-array based design and programmable logic devices such as the highly popular field programmable gate arrays (FPGA's). Each has its own advantages and disadvantages. FPGA's, for example, enable a very fast implementation of a new function without the need to process the chip, but the timing characteristics of these circuits is typically poor.

Cell-based design is a compromise between quick implementation and performance. A fixed set of custom designed gates are used and implemented in a regular layout style. This set is generally referred to as the library. We can make a further distinction between standard-cell design and macro-cell design. The first deals with random and unstructured logic, while the latter includes larger cells which implement regular logic, such as memories, datapath or ALU's. In standard-cell the cells are designed and characterized once, and then stored in a standard library. In macro-cell design the logic functions can by automatically synthesized and generated using a module generator.

The optimization techniques we developed are targeted towards the popular cell-based design style. The techniques are more specifically for high-speed BiCMOS target technologies.
1.2 Why BiCMOS?

BiCMOS circuits aim to combine the advantages of both bipolar and CMOS technologies into a single chip. The first attempt to combine bipolar and MOS transistors in one monolithic structure was made by Hung Chang Lin et al. [30] in 1969. They were claiming low standby power consumption and large driving capability.

Pure CMOS circuits are characterized by low power dissipation, large noise margins, ability to integrate large complex functions with high yields and high packing density. The main advantages of a bipolar technology over CMOS are the higher driving capability, noise performance, analog capability and input/output speed.

A large variety of digital and analog BiCMOS circuit structures have been developed. In a digital BiCMOS IC, most of the logic functions are implemented in CMOS (memory cells, logic gates, registers, multiplexors, etc.), while bipolar devices are typically reserved for the critical paths for word line drivers, sense amplifiers, buffers to drive on-chip global wires, critical logic paths such as carry-propagate, clock drivers on microprocessors, and chip-to-chip drivers and receivers.

BiCMOS is also very suitable for applications that integrate analog and digital functions on the same chip (so called mixed-signal applications). The analog part of the circuit can be implemented using bipolar transistors. They make very good differential pairs for low drift operational amplifiers, reference circuits, and high current output buffers. On the other hand CMOS devices can be used to implement dense logic and memory blocks, and to provide input stages with high input impedance.

Despite the increased popularity of BiCMOS technology in the semiconductor market, surprisingly little commercial CAD design tools exist which address BiCMOS design issues specifically. Most of the tools are targeted towards CMOS technology exclusively. This work attempts to fill a part of this gap by presenting performance optimization techniques which work in a BiCMOS environment.

1.3 Synthesis and Optimization

Synthesis tools assist a human designer in bringing the product on the market as soon as possible. The role of optimization tools is to ensure that the design meets the specifications. Synthesis and optimization tools are specific for each design style. FPGA’s, PLD’s or standard-cells each require different approaches. But for a given design style there is generally no clear distinction between synthesis and optimization even if they are presented as different steps. The optimization is performed along with the synthesis during the entire design flow.

A typical design flow for standard-cell style is shown in Figure 1.1. It starts from a behavioral or functional description of the system to be implemented. This description can be made, for example, in VHDL or Verilog. In case of a behavioral
description a high-level synthesis tool transforms this description into a more concrete functional description. Various problems such as resource sharing under specified constraints are solved. The result is a set of interconnected functional units. At this level still very little is known about the target technology and the design style used to implement the final circuit. Therefore, the optimization tools use technology independent models and the decisions made are focused on architectures.

![Diagram](image)

**Figure 1.1.** ASIC design flow.

Once the high-level synthesis step is completed, logic synthesis tools transform the functional description into a set of interconnected gates in the target technology. Synthesis and optimization algorithms work together to deliver a circuit design which is not only logically correct but also fulfills the constraints such as timing, area, power etc. At this level technology dependent models become very important. The optimization results are directly dependent on the accuracy of these models.
Finally, a layout synthesis tool transforms the netlist description into a physical design. The cells are first placed and then routed. The optimization techniques at this level are related to efficient placement and routing programs. At the end of this step, the circuit layout is ready for the fabrication process.

The design flow we just presented is simplified. There are many smaller steps in between to ensure good quality designs. Simulation and verification tools, for instance, are also necessary to check the correctness of the circuit before fabrication. Along with them accurate models for estimating the circuit performances need to be developed.

Analyzing this simplified design flow one can easily notice that a successful design depends not only on the efficiency of the synthesis and optimization tools, but to an equal extent on the target technology performance, the design quality of the library cells and its characterization accuracy. Therefore, along with developing efficient optimization techniques, our work is also focused on optimum library design and characterization. In Figure 1.1 these areas of interest are emphasized and put into the perspective of the entire design flow.

1.4 Thesis Overview

As mentioned, the choice of optimization tools must account for the target technology and the design style used to implement the circuits. The focus of this thesis is on performance optimization techniques for high-speed cell-based design style in a BiCMOS target technology.

Cell-based design, including both standard-cell and macro-cell design, can be fully automated. A good quality design is generated in a short time by following the full trajectory from VHDL or Verilog description down to layout design. Generally, next to cell-based design blocks, dedicated blocks such as RAM's or ROM's are added on chip. A schematic of an integrated circuit made in cell-based design is shown in Figure 1.2.

In standard-cell design, the elementary cells are pre-designed and stored in a library. The design of all circuits is based on this standard library. Algorithms have been created to automatically synthesize, optimize and verify the designs. The circuit performance is related to synthesis and optimization algorithms. However, the design and characterization of the fundamental cells play a very important role in the final design quality.

In macro-cell design, building blocks are combined to implement the specified logic functions. These macro-cells can be automatically synthesized, placed and wired. In particular, the macro-cell based design is suitable for dedicated blocks such as memory cells, full-adders, etc. The overall circuit performance depends on the circuit techniques used for the macro-cells and the logic minimization algorithms used in functional description implementation.
Even though standard-cell and macro-cell design belong to the same design style, they are conceptually different, requiring different optimization techniques in each case. For example, to enhance the speed of a full-adder we can choose between different architectures, such as carry-ripple or carry-look ahead and implement them in standard-cells, or combine these architectures with new circuit techniques and implement them in macro-cells.

In this work we use two strategies for enhancing the speed of a digital system implemented in a cell-based design and a BiCMOS target technology:

1. For better performance standard-cell designs the critical paths are sped up by using bipolar drivers and efficient timing optimization algorithms. Special attention is paid to optimum driver design.

2. Macro-cell based design is used for logic blocks which have a regular structure. We present a novel circuit design methodology which increases the speed and the integration density of the design.

The speed optimization strategies are organized in seven chapters which cover a brief description of the BiCMOS drivers in chapter 2, followed by the delay analysis of the totem-pole inverter in chapter 3. Chapter 4 addresses the issue of gate characterization for CAD purposes. The buffer sizing methods are addressed in chap-
ter 5. Chapter 6 introduces the timing optimization methods for combinational networks. An efficient fanout optimization algorithm is presented in chapter 7. Chapter 8 covers the macro-cell design and implementation. Finally, some conclusions are derived. An overview of the chapters is provided next.

Many ASIC designs are implemented in a cell-based design environment. The more complex systems are built in both macro-cells and standard-cells, while for less complex ones the standard-cell design alone seems to be adequate.

Our first speed optimization strategy is targeted at standard-cell design style. This design methodology has become very popular especially after logic synthesis tools were made available on the market. Along with them, optimization algorithms were developed to assure high performance circuits. However, these tools are CMOS oriented and little has been done to accommodate BiCMOS technologies.

Our strategy is to use BiCMOS cells in the time critical path to enhance the circuit speed. For this purpose in chapter 2, we briefly introduce various types of BiCMOS gates which can be integrated in a standard-cell library and used to drive high load capacitances. These BiCMOS cells can have different configurations, for full-swing or reduced swing drivers, or suitable for reduced supply voltage, or they use bipolar devices only in the pull-up stages, etc.

The BiCMOS gates are very attractive because they combine the advantages of both bipolar and CMOS technologies. The high driving capability of the bipolar devices and the low power dissipation of the MOS devices are added together in such a BiCMOS driver.

The drawbacks of this design are the more expensive BiCMOS technologies and the additional area used by bipolar devices. In some mixed-signal applications, however, where bipolar transistors are used to implement the analog part, using BiCMOS drivers in the digital part to enhance the speed is an obvious choice. It is also not always the case that the BiCMOS technologies are more expensive than the CMOS technologies. The better performance deep sub-micron CMOS technologies are more expensive than the regular sub-micron BiCMOS technologies because of the more accurate and elaborate process technology. Therefore, the BiCMOS technology is a viable solution for high performance circuits while postponing the device scaling.

In standard-cell based design the optimization results as well as the simulation results rely on the delay models used to implement the tools. If the delay models are not accurate, the timing problems of the design cannot be detected during the design phase but only later on, after the design or even the fabrication process are completed, when it is very expensive. That is why we pay a special attention for modeling and characterization of the BiCMOS cells.

In chapter 3 we develop two analytical delay models. Both models take into account the short and narrow channel effects on MOS device equations and the high level injection effects on bipolar transistor collector current. The first model gives
very accurate results but it is computationally expensive. This model is used in circuit cell design at library development stage. The second model is easy to compute but less accurate. It is used for fast delay estimations.

Chapter 4 addresses the issue of cell characterization for CAD applications. Since the two analytical delay models are still computationally expensive to be used in optimization and simulation tools for IC designs with tens of thousands of cells, we reduce our analytical delay model to a simpler CAD delay model. It depends on the output load capacitance of the cell and the input signal slew rate. These two factors are both very important in developing high performance optimization and simulation tools. A power dissipation versus input signal slew rate analysis is also made to complete the cell characterization part.

The performance of a design implemented in standard-cell is directly dependent on the performance of its components. Special attention is paid to cell design which is not at all trivial. The library is designed once, but it has to be updated when the target technology changes allowing for smaller geometries. Therefore, it is very important that this process is automated.

For BiCMOS cell design the design process is even more complicated than for CMOS. In addition to determine the sizes of MOS devices, we need to size the bipolar devices as well. However, a BiCMOS driver can accommodate, with good delay performance, a larger range of output load capacitances which justify the extra design effort. The cell design process is presented in chapter 5.

The method we propose for sizing the BiCMOS drivers assures near-optimum designs. We design BiCMOS drivers which can drive a given capacitance range in close to minimum time. Our design methodology can be fully automated. The design cycle is short because we use the accurate analytical delay model that we developed for the BiCMOS driver. The design solution is optimum and quick. In the absence of an accurate analytical delay model, SPICE simulations would be necessary during the design process. To obtain the same design solution would then require a much longer time.

A very good designed and characterized standard-cell library is essential for high performance circuit design but it is not enough. Another important role in the design process is played by the synthesis and optimization algorithms. To complete the standard-cell optimization strategy of enhancing the speed of a digital system, we develop a timing optimization algorithm suitable for combinational circuits. The optimization strategy is presented in chapter 6.

Since sequential optimization techniques rely on those for combinational circuits the optimization method we develop here can be extended to work also for sequential circuits. Our assumption is that sequential circuits can be decomposed into combinational logic blocks with sequential elements at their primary inputs and primary outputs. Consequently, the sequential circuit performance is bounded by its combinational circuit performance.
At this phase the CAD delay model we develop is used in estimating the delay performance of the circuit. The model is easy to use and accomplishes the CMOS delay model as well.

An important optimization step is to enable cells to drive a large number of fanouts while satisfying the timing constraints. Chapter 7 addresses the issue of fanout buffering. A tree of buffers and inverters (the fanout tree) is built at the output of the cell.

Finally, chapter 8 introduces the macro-cell based design. As mentioned, macro-cell based design is very suitable for implementing blocks with a regular layout structure. This kind of implementation can increase the integration density. Consequently, the inter-chip communication is reduced and therefore the overall design speed increases.

The performance of the macro-cell design is subject to the circuit design style and the target technology used in its implementation. The use of one circuit design style above another is determined by the specific application and its requirements. Using pure CMOS implementations, circuits with low power consumption will result. But they may not satisfy the timing constraints. Therefore, the designer should look for help, for example in the BiCMOS area and try to combine the high driving capability of the bipolar devices with low power dissipation of the MOS devices.

The macro-cell implementation we develop in this work is suitable for a particular class of circuits called iterative networks. The cell design combines bipolar and MOS devices in a differential logic configuration to attain high speed and low supply voltage performances.

But speed is not the only important factor during the design process. It has to be possible to automatically generate, place and route the cells to assure a short time to market for the design. Choosing differential logic for our implementation we fulfill this condition. The Boolean description of the logic function to be implemented is automatically synthesized using logic minimization algorithms.
The motivation for using combinations of bipolar and MOS transistors for circuits driving large capacitive loads is the prospect of retaining the driving capability of bipolar devices without giving up on the signal processing efficiencies of CMOS circuits. This immediately leads to configurations in which the bipolar device provides the current for charging or discharging the load capacitance while a MOS transistor controlled by a signal level, supplies the base current. This basic configuration can be used for the pull-up and pull-down part of an elementary port to form an inverting buffer with improved driving capability (Figure 2.1).

\[
\begin{array}{c}
\text{(a)} \\
\text{(b)}
\end{array}
\]

**Figure 2.1. Basic BiCMOS driver configuration**

Several varieties are possible by choosing the transistor types and the channel contact of the MOS transistor. They all exhibit the problem that the base charge cannot be removed when the bipolar transistors have to be turned off. All practical implementations must provide a current path for that purpose. Another problem, not so easily solved, is the reduced logic swing of inverters built from such basic pull-up and pull-down configurations. Some solutions will be presented in section 2.3, but BiCMOS full-rail logic is still an active area of research.

In the next chapter a detailed analysis of a conventional BiCMOS inverter with base-charge removal provisions will be carried out to show how macro models suitable for design and optimization can be derived. To obtain a macro model with all relevant properties adequate device models are necessary. The requirements for these models together with a concise model summary concludes this chapter.
2.1 Basic Configurations

A straightforward application of the above ideas was proposed in 1969, by Lin et al. [30] in the first CMOS-bipolar structure aiming at low standby dissipation and large driving capability. The pull-up stage consists of a PMOS transistor and bipolar NPN-transistor while in the pull-down stage an NMOS and bipolar NPN structure are merged. This brings the two bipolar transistors in the well-known totem-pole configuration. In both stages the MOS transistors are connected between collector and base. This early version of a BiCMOS driver is shown in Figure 2.2.

![Figure 2.2. The first version of a BiCMOS inverter.](image)

The operating principle of a BiCMOS buffer is similar to that of a CMOS. When the input voltage is at low logic level, $M_1$ is on and provides current to $Q_1$ which charges the output load capacitance. Both transistors, $M_2$ and $Q_2$ are assumed to be off. If $R_p$ is the equivalent on-resistance of $M_1$ this current is in first approximation equal to $(\beta_F + 1)(V_{DD} - V_{BE,on} - V_{out})/R_p$ which shows the current multiplying effect. The notations we used are $\beta_F$ for the bipolar transistor current gain, $V_{BE,on}$ for the bipolar transistor base-emitter voltage in conducting state, $V_{out}$ for the output voltage of the inverter, and $V_{DD}$ for the supply voltage. Asymptotically the output voltage will be $V_{DD} - V_{BE,on}$, once $Q_1$ is turned off. When the input voltage is at high logic level, $M_1$ and $Q_1$ are assumed off, and $M_2$ is on, providing current to $Q_2$ which discharges the output load capacitance with a current of about $(\beta_F + 1)(V_{out} - V_{BE,on})/R_n$. Here, $R_n$ is the equivalent on-resistance of $M_2$. As the output voltage reaches $V_{BE,on}$, $Q_2$ gradually turns off. The high and low output logic levels of the inverter would therefore be close to $V_{DD} - V_{BE,on}$ and $V_{BE,on}$ respectively. The actual logic swing at the output may be found to be greater, because the bipolar transistor continues to supply collector current until all its base charge is removed.

In a fully complementary BiCMOS technology, both types of bipolar transistors, NPN and PNP are available. This allows more generic types of BiCMOS drivers as shown in Figure 2.3. They are known as common-emitter, gated-diode, and
emitter-follower configuration respectively [52].

In the common-emitter configuration (Figure 2.3a) the MOS transistor operates in common-source mode and the bipolar transistor in common-emitter mode. Due to a large, near constant base current during switching, the circuit is expected to be very fast. There is however additional delay due to the fact that when the output is fully charged or discharged the bipolar transistor will be in saturation. The process of turning the saturated bipolar transistors off is very slow, because there is no good conducting path to remove the base charge. In this case the output logic swing is closed to the full power-supply voltage, namely $V_{DD} - V_{CE, sat}$. The weak point is a substantial static power dissipation.

In the gated-diode configuration (Figure 2.3b) the MOS transistor acts as a switch between base and collector which causes a substantial decrease in MOS transistor current as the output falls or rises. The bipolar transistor is diode-connected and the logic levels are $V_{BE, on}$ and $V_{DD} - V_{BE, on}$. This circuit lacks symmetry between rise and fall times, because the pull-down stage consists of an n-channel MOS transistor and an NPN bipolar transistor, both fast, and the pull-up stage consists of a p-channel MOS transistor and a PNP bipolar transistor, both comparatively slow. Besides, the collector of the PNP bipolar transistor is connected to the output and it must be isolated what usually increases process complexity.

![Diagrams of BiCMOS configurations](image)

**Figure 2.3.** BiCMOS driver configurations: (a) common-emitter configuration; (b) gated-diode configuration; (c) emitter-follower configuration.

In the emitter-follower driver (Figure 2.3c) the MOS transistor, in common-source configuration, drives the emitter-follower. This circuit is the variant closest to the early version proposed by Lin et al. [30]. The difference is in the pull-down stage design where an n-channel MOS transistor drives a PNP bipolar transistor in this case. Density can be improved by merging the two devices in one p-type region in a single diffusion step [42]. In a similar way PMOS and NPN transistors can be
merged. Thus three terminal devices are obtained with good device characteristics: high input impedance inherited from MOS transistor gate, low output impedance thanks to bipolar transistor.

2.2 Base Charge Removal

An inherent problem of these BiCMOS drivers is the charge present in the bipolar transistor base after a complete switch cycle. This charge can cause the bipolar transistor in pull-up stage to stay on when a pull-down event is taking place and vice-versa. As a result the total effective output current is reduced and the achievable speed is limited. To avoid these undesirable currents, bleeding elements must be added to remove the base charge after the switching cycle is finished. In the circuit of Figure 2.4 this is accomplished by extending the basic configuration with two additional NMOS transistors $M_3$ and $M_4$ [50].

![Figure 2.4. The conventional LS BiCMOS totem-pole inverter.](image)

This solution has two advantages. First, it preserves the zero static power dissipation of the gate, because the path between the base of the bipolar transistor and ground is created only when the bipolar transistor is switched off. Second, the MOS transistors belonging to the same basic configuration are never both conducting. Therefore no effective base drive is wasted. The weak point of this design is the limited swing (LS) of the output voltage which will lower the gate voltage of the next stage by the amount $V_{BE,\text{on}}$. As a result, the speed of the circuit degrades quite rapidly as the power-supply voltage is reduced.

Various types of bleeding devices have been proposed. In a so called R-type BiCMOS driver [40], resistors are used to discharge the bipolar base (Figure 2.5a).
But the resistors are conducting permanently. Therefore, to avoid wasting the base drive, high value resistors have to be used. The weak point of the $R$-type circuit is the high power dissipation.

The $R+N$-type gate adds a discharging NMOS transistor $M_3$ to the $R$-type (Figure 2.5b) which leads to improved power dissipation characteristics over the $R$-type [41]. In addition, it has full swing (FS) output voltage due to the resistance between the base and the emitter. The full-swing drivers allow the use of a lower MOS threshold voltage to obtain faster speed and better noise margins. But when the input falls, the base current supplied to the base of $Q_1$ flows partly into the discharging NMOS transistor which slows down the switching speed.

![Figure 2.5. BiCMOS drivers: (a) FS R-type; (b) FS R+N-type; (c) FB-type.](image)

The feedback-type (FB) driver was proposed [41] to overcome this drawback. An NMOS transistor $M_4$ and a feedback CMOS inverter are added to the circuit to stop the leakage current (Figure 2.5c). When the input falls the FB signal is initially low and the NMOS transistor $M_4$ is off. Due to the feedback inverters the FB signal is delayed relative to the output signal and consequently, the PMOS current supplied to the base of $Q_1$ is not leaking via NMOS transistor $M_3$. When the input rises and $Q_1$ turns off, the FB signal is initially high and $M_4$ is still in the on-state for a while due to the feedback inverters. In this way, the base of $Q_1$ is quickly discharged, the penetration current is reduced and the low power dissipation can be achieved.

The solutions adopted in case of a Complementary BiCMOS technology are shown in Figure 2.6 for a typical CBiCMOS 2-input NAND gate. In case of a CBiCMOS with emitter-follower drivers the base nodes of the emitter-follower are clamped to ensure that the bipolar transistors are not turned on simultaneously, but biased near the edge of turn-off in steady states. The clamping device is a CMOS diode.
which is the parallel pair of NMOS diode ($M_6$) and PMOS diode ($M_5$). In case of a CBiCMOS with gated-diode, resistors are used.

![Diagram](image)

**Figure 2.6.** CBiCMOS 2-inputs NAND: (a) EF configuration; (b) GD configuration.

### 2.3 Full-swing Techniques

As process technology progresses toward smaller feature sizes to improve performance, scaling the power supply voltage is necessary to protect MOS transistors from breakdown and hot electron problems, while it also reduces the power dissipation on chip. A major limiting factor of the BiCMOS configurations presented in sections 2.1 and 2.2 is the limited swing, because they will loose their leverage over CMOS circuit drivers at low supply voltage [61, 17, 38]. This loss in the output swing degrades the speed and the noise margins of the circuit, when the supply voltage is scaled down. If the logic swings are increased to the full power supply voltage the speed of the circuits will improve and, in addition, the devices controlled by the driver can be properly switched off.

One technique to achieve the full swing is to use a resistive shunt network between the emitter and the base of the bipolar transistor. Shunt techniques using a simple resistor were already presented before in Figure 2.5.

A number of shunting solutions are illustrated in Figure 2.7 [52]. The pull-up stage is a simple conventional BiCMOS circuit with emitter-follower configuration, while the pull-down operation is presented in two variants: gated-diode configuration and complementary emitter-follower configuration.

A collector-emitter shunt network adds current to the output continuing charging or discharging at the end of the switching. However, a crossover current flows
2.3 Full-swing Techniques

Figure 2.7. Full-swing techniques: (a) collector-emitter shunt for pull-up stage in conventional configuration and pull-down stage in gated-diode configuration and complementary emitter-follower configuration; (b) base-emitter shunt for pull-up stage in conventional configuration and pull-down stage in gated-diode configuration and complementary emitter-follower configuration.

through the shunt element when the output begins to fall or rise, thus increasing the power dissipation and slowing down the transient. In case of a base-emitter shunt the shunt element brings the output to the power-supply or ground level through the MOS transistor that is already on. There is no crossover current in this case, but part of the MOS current, used in driving the bipolar transistor, is bypassed via the shunt element degrading the speed.

The advantages of the CBICMOS drivers are considerably offset by the additional process complexity and cost for the PNP bipolar fabrication. To overcome this limitation and still provide full-swing circuits a so called BiNMOS family was proposed [18, 24, 23] (Figure 2.8). For BiNMOS circuits the pull-down stage is realized only with NMOS transistors to ensure grounded low output level. The initial BiNMOS circuit has still a limited swing since high output logic level remains
$V_{DD} - V_{BE, on}$.

Many alternatives have been proposed which differ through pull-up full swing shunt type, such as in case of a BiRNOS driver a resistor is inserted between the base and the emitter of the pull-up bipolar transistor, for a BiPNMOS driver a feedback base-emitter shunt network is added consisting of a small size PMOS transistor and an inverter, and finally, for a PBiN MOS driver a small size PMOS transistor in parallel with the pull-up bipolar transistor is used. However, BiNMOS family has a small performance leverage over conventional CMOS, especially for 0.5-$\mu$m gate-length MOS transistors or smaller.

![BiNMOS circuit family](image)

**Figure 2.8.** BiNMOS circuit family: (a) Conventional BiNMOS driver; (b) BiRNMOS driver; (c) BiPNMOS driver; (d) PBiN MOS driver.

Many other variants of BiCMOS drivers can be found in the literature. The driver configurations presented here are the most significant. A wide and detailed analysis of BiCMOS types is beyond the scope of this work. We presented them as far as necessary to introduce our speed optimization techniques.

## 2.4 Driver Analysis

Despite the large variety in BiCMOS driver circuits, the basic configuration for pull-up and pull-down stages (Figure 2.1) is always the same. A bipolar transistor drives the output load capacitance while the MOS transistor supplies the base current. Any additional circuitry is generally added to improve the performance of the driver. In pull-up and pull-down timing analysis the basic configurations have a dominant contribution to the switching speed of the driver.

In standard-cell based design the optimization and simulation results rely on the cell characterization models used to implement the tools. The cell characterization models, in their turn, rely on the device models used during the characterization process. In particular, the accuracy of BiCMOS cell characterization depends
directly on the MOS and bipolar device models. In this section model requirements are set for both MOS and bipolar devices to assure an accurate performance characterization for the BiCMOS driver.

MOS Device Requirements

Modern MOS integrated circuits are fabricated in sub-micron technological processes. When the channel geometry is small a variety of modifications to the classical MOS transistor operation must be made. For a certain substrate concentration, the depletion regions of source and drain junctions become comparable with the channel length. Both the gate electric field and the drain source electric field will determine the potential distribution in the channel. Therefore, the gradual approximation used in classical theory is no longer suitable. The effective channel length also decreases as the drain voltage is increased beyond pinch-off which invalidates the assumption of constant saturation current. Moreover, for very short channels even a moderate drain voltage can result in a high field along the channel. For example, high-field effects occur when 1.0 V drops across a channel length of 1.0 \( \mu \text{m} \), giving an electric field of 10 kV/cm. A high field in the channel will cause a degradation of the electron mobility due to the electron velocity saturation.

In short, we can classify the relevant short-channel effects in:

1. Reduction in the channel depletion charge due to the source and drain depleted regions or an increase in the channel surface potential due to the drain-source voltage [2] (see Figure 2.10).

2. The holes and electrons reach the saturation velocity due to the high electric fields in the channel; the carrier mobility in the channel depends on both gate and drain voltage.

A good analytical model must account for these, so called, second order effects. These effects therefore get translated into device parameters:

- device geometry (channel length and channel width) affects the transistor threshold voltage, saturation current and saturation voltage
- high electric fields affect the carrier mobility in the channel with consequences on drain-source current, as well as, saturation voltage

These effects result in very cumbersome device equations that are difficult to deal with in a first order circuit analysis and design. For specific cases numerical methods can be used in solving the device equations, and deriving conclusions on device design. In a design trajectory of a complete integrated circuits a computationally manageable model is needed. It should incorporate the effects relevant to the objective, mostly speed, power and density, without exceeding practical limits to
design time. In appendix 2.a a detailed device model is presented for an NMOS transistor. However, the same results apply also to its PMOS counterpart.

The effective carrier surface mobility (expression 2.a.5), the transistor threshold voltage (expression 2.a.7), and the drain-source saturation voltage (expression 2.a.12) are interdependent parameters. These dependences are not linear. Therefore, it is difficult to analytically determine the transistor saturation voltage for a given gate-source voltage since the threshold voltage is not a constant.

To enable an analytical circuit delay analysis we make the following simplifying assumptions at the device level:

- the threshold voltage is constant \( V_{TH} = V_{TH0} \)
- the surface mobility component due to the gate electric field (expression 2.a.4) is constant; we determine an average value \( \mu_{s,av} \).
- the saturation current is considered constant; in other words there is no channel length variation in the saturation region.

Consequently, the complicated SPICE expressions presented in appendix 2.a can be simplified to an easy to use level. The reader is referred to table 2.1 for device parameter definitions and notations.

The value we considered for the threshold voltage is for grounded source, drain and bulk electrodes. If the substrate is not zero biased then the threshold voltage has to be recalculated with the relation (2.a.7).

When the gate-source voltage varies from its threshold value to the supply voltage \( V_{DD} \), the surface mobility variation is less than 10%. This small variation is due to the small mobility modulation factor \( \theta \). An average value for the surface mobility \( \mu_s \) is, therefore, within 5% from the real value, which is an acceptable error for our analysis. The average gate dependent mobility \( \mu_{s,av} \) is derived from equation (2.a.4) by integrating from \( V_{GS} = V_{TH0} \) to \( V_{GS} = V_{DD} \) as follows:

\[
(2.1) \quad \mu_{s,av} = \frac{1}{V_{DD} - V_{TH0}} \int_{V_{TH0}}^{V_{DD}} \frac{\mu_0}{1 + \theta(V_{GS} - V_{TH0})} dV_{GS} = \frac{\mu_0}{\theta(V_{DD} - V_{TH0})} \log \left( 1 + \theta(V_{DD} - V_{TH0}) \right)
\]

A key factor in determining the drain-source current in saturation is the saturation voltage. The saturation voltage is determined with the expression (2.a.12) in which the surface mobility \( \mu_s \) is replaced with its average value \( \mu_{s,av} \) as follows:

\[
(2.2) \quad V_{DS, sat} = V_a + V_b - \sqrt{V_a^2 + V_b^2}
\]
2.4 Driver Analysis

\[
V_a = \frac{V_{GS} - V_{TH0}}{1 + F_B}
\]
\[
V_b = \frac{v_{max} L_{eff}}{\mu_{s,av}}
\]

Finally, the drain current for linear region can be derived by substituting from equation (2.1) into equation (2.a.13) to account for the average surface mobility. The result is given by:

\[
(2.3) \quad I_{DS}(V_{DS}) = C_{ox} \frac{W_{eff}}{L_{eff}} \frac{\mu_{s,av}}{1 + \frac{\mu_{s,av}}{v_{max} L_{eff}} V_{DS}} \left( V_{GS} - V_{TH0} - \frac{1 + F_B}{2} V_{DS} \right)
\]

The saturation current is the result of the above relation when \( V_{DS} = V_{DS, sat} \). This simpler drain-source current expression for a MOS transistor is used later on in the BiCMOS driver delay analysis (chapter 3).

Bipolar Device Requirements

Bipolar junction transistor (BJT) is used as a switch in digital BiCMOS circuits. Its high driving capability improves the driver switching speed. To accurately estimate the BiCMOS driver switching speed we first need to analyze the large signal behavior of the bipolar transistor.

A BJT has four possible operation modes:

- forward active mode
- reverse active mode
- cutoff mode
- saturation mode

Forward active, saturation and cutoff modes play an important role in bipolar transistor switching behavior. They are briefly presented here. A detailed device analysis can be found in references [19, 47, 2, 39].

In the forward active mode the base-emitter junction is forward biased, while the base-collector junction is reverse biased. The carriers injected from the emitter into the base easily diffuse through the base and are swept by the reverse-biased collector junction. As a result a positive current will flow from collector to emitter. If the recombination rate in the base region is very small, then the collector current is slightly less than the emitter current. The difference is a small base current which controls the collector current.

In saturation mode both junctions are forward biased. The carriers flow into the base region across both the emitter and the collector junctions. The collector current remains essentially constant while the base current increases.
In the cutoff mode, both emitter and collector junctions are reversed biased. The currents through the device are negligible.

There are basically two well known models which describe the bipolar transistor behavior: the Ebers-Moll model [14] and the Gummel-Poon model [22]. The former is a simple first order model which can be used for hand calculations in design and static analysis of the transistor. The latter is a more elaborated model which also accounts for the second order effects in the device and, therefore, it is more accurate. These second order effects include the influence of parasitic resistances and capacitances as well as the high current operation. For example, the transistor current gain is affected at low base-emitter voltage by recombination in the base-emitter depletion region and at high base-emitter voltage by high-level injection in the base.

The most important parameters in modeling the switching behavior of the bipolar transistor are the total excess base charge, the minority carriers lifetime in the base, and the transit time for these carriers to traverse the base region. These second order effects can be summarized as follows [2]:

- **Low-current drop in current gain.** This drop is caused by the additional base current due to recombination in the depletion-layer region. It is important for low-current circuits.

- **High-level injection.** At high transistor currents, the injection of minority carriers into the base region is significant with respect to the majority-carrier concentration. To maintain space-charge neutrality in the base the total majority-carrier concentration is increased by the same amount as the total minority-carrier concentration. The effect of the excess majority carriers is a drop in collector current slope above the knee current [19]. There are also additional consequences due to the base-widening. As the current density crossing the base-collector junction increases, the minority-carrier concentration increases and approaches the collector density, neutralizing the collector charged ions. As a result, the base spreads over the collector epitaxial layer. The current gain degrades and the base transit time increases.

- **Parasitic resistances.** They represent the transistor ohmic resistance from its active region to its emitter, base and collector terminals, respectively. By including these parasitic resistances in transistor model the device characterization is improved.

- **Base-width modulation.** This effect is also known as Early effect. It appears especially in narrow-base transistors where the increase of reverse bias on base-collector junction decreases the neutral base-width, thereby increasing the gain. Consequently, the collector current in saturation increases as the collector-emitter voltage increases.
In appendix 2.b the device model currently use by SPICE simulator is presented for an NPN bipolar transistor. However, the same results apply also to its PNP counterpart with small modifications.

Because the presented bipolar transistor model is too complicated and difficult to use in subsequent circuit design and analysis, some simplifications have to be made. The most important parameters in describing the switching behavior of the bipolar transistor are the base transit time, the current gain, and the device capacitances. Therefore we will concentrate on these parameters. The reader is referred to table 2.2 for device parameter definitions and notations.

As shown in appendix 2.b two expressions are very common in modeling the base transit time of a bipolar transistor. Because of the good agreements with the experimental results, we chose expression (2.b.2) for modeling this parameter. However, this expression is still quite cumbersome and, consequently, not easy to use. An average value of the base transit time over collector current and base-collector voltage is a much better choice.

The average base transit time is obtained by integrating the equation (2.b.2) over the variables collector current $I_C$ and base-collector voltage $V_{BC}$ as follows:

$$\tau_{F,av} = \frac{1}{I_{C,\text{max}}} \int_{I_C}^{I_{C,\text{max}}} \int_{V_{BC,1}}^{V_{BC,2}} \tau_F(I_C, V_{BC}) dI_C dV_{BC}$$

$$= \tau_{F0} \left( 1 + X_{\tau_F} f_{av}(I_C) g_{av}(V_{BC}) \right)$$

(2.4)

where

$$f_{av}(I_C) = \frac{1}{I_{C,\text{max}}} \int_{0}^{I_{C,\text{max}}} \left( \frac{I_C}{I_C + I_{\tau_F}} \right)^2 dI_C$$

$$= \frac{I_C}{I_{\tau_F}} + 2 \frac{I_{C,\text{max}}}{I_{\tau_F}} + 1 - \frac{2}{I_{C,\text{max}}/I_{\tau_F}} \log \left( \frac{I_{C,\text{max}}}{I_{\tau_F}} + 1 \right)$$

(2.5)

and

$$g_{av}(V_{BC}) = \frac{1}{\Delta V_{BC}} \int_{V_{BC,1}}^{V_{BC,2}} \exp \left( \frac{V_{BC}}{1.44 V_{\tau_F}} \right) dV_{BC}$$

$$= \frac{1.44 V_{\tau_F}}{\Delta V_{BC}} \exp \left( \frac{V_{BC}}{1.44 V_{\tau_F}} \right) \bigg|_{V_{BC,1}}^{V_{BC,2}}$$

(2.6)

with

$$\Delta V_{BC} = V_{BC,2} - V_{BC,1}$$

The integration limits for collector current are taken between null and a maximum value $I_{C,\text{max}}$. This choice will be justified in section 3.5 (on page 47). The integration limits for base-collector voltage are taken between two values $V_{BC,1}$ and $V_{BC,2}$ which are also determined in section 3.5.
Another important switching parameter is the transistor current gain in the forward active mode. We will be using the transistor current gain model in both variants, as an explicit function of collector current given by the equation (2.6.2) as well as an average value over the collector current derived as follows:

$\beta_{F,av} = \frac{1}{I_{C,max}} \int_0^{I_{C,max}} \frac{\beta_{F0}}{1 + I_C/I_{KF}} dI_C$

$= \frac{\beta_{F0} I_{KF}}{I_{C,max}} \log \left( 1 + \frac{I_{C,max}}{I_{KF}} \right)$

(2.7)

where the collector current has the same maximum value $I_{C,max}$ as it has already been mentioned in the expression of the average base transit time.

The transistor current gain given by the equation (2.6.2) has a better accuracy than its average value (2.7). However, in circuit switching analysis the former relation will lead to more difficult to solve differential equations than the latter. Using one or the other form is a tradeoff between accuracy and computational effort.

**Figure 2.9.** High current effects of bipolar transistor: (a) Forward current gain and its average value; (b) Forward base transit time and its average value.
2.5 Conclusion

In Figure 2.9 the high current effects over the forward current gain and the forward base transit time of an NPN bipolar transistor are illustrated. The bipolar transistor is part of an BiCMOS driver circuit which has a capacitive load of 3.0 pF. The emitter area of the bipolar transistor is 4.0 \( \mu m^2 \) which corresponds to a knee current of 12.0 mA. The average values for the current gain and base transit time are also shown in the figure for comparison.

Finally, the expressions (2.b.5) for base-emitter and base-collector junction capacitances are also simplified. Their average values are derived over the base-emitter \( V_{BE} \) and base-collector \( V_{BC} \) voltages respectively using the expressions (2.b.5), as follows:

\[
C_{BE,av} = \frac{1}{\Delta V_{BE}} \int_{V_{BE,1}}^{V_{BE,2}} \frac{C_{BE0}}{(1 - V_{BE}/\phi_E)^{m_E}} dV_{BE}
\]

(2.8)

\[
= \frac{C_{BE0} \phi_E}{(1 - m_E)\Delta V_{BE}} \left(1 - \frac{V_{BE}}{\phi_E}\right)^{1-m_E} \bigg|_{V_{BE,1}}^{V_{BE,2}}
\]

with

\[
\Delta V_{BE} = V_{BE,2} - V_{BE,1}
\]

and

\[
C_{BC,av} = \frac{1}{\Delta V_{BC}} \int_{V_{BC,1}}^{V_{BC,2}} \frac{C_{BC0}}{(1 - V_{BC}/\phi_C)^{m_C}} dV_{BC}
\]

(2.9)

\[
= \frac{C_{BC0} \phi_C}{(1 - m_C)\Delta V_{BC}} \left(1 - \frac{V_{BC}}{\phi_C}\right)^{1-m_C} \bigg|_{V_{BC,1}}^{V_{BC,2}}
\]

with

\[
\Delta V_{BC} = V_{BC,2} - V_{BC,1}
\]

The variation limits for base-emitter and base-collector voltages are dependent upon the BiCMOS driver circuit parameters. They will be determined in section 3.5.

With these last junction capacitances all the important switching parameters of the bipolar transistor have been considered. Their expressions have been simplified up to an easy to use level which enables the transient delay analysis of various BiCMOS drivers. This analysis is made in the following chapter.

2.5 Conclusion

A few typical BiCMOS driver configurations have been presented. They have common pull-up and pull-down stages consisting of a MOS transistor (P or N channel) driving a bipolar transistor (NPN or PNP). The difference between the drivers
consists in the circuitry used in discharging the base of the bipolar transistors and if they are designed for full or limited swing. The design of full swing BiCMOS drivers is more complicated and there is no guarantee that they have better speed than their CMOS counterpart, especially when the technology scales down to 0.35-μm or less. However, the limited swing drivers have been proven to be faster than CMOS for a supply voltage down to 3.3 V, and a 0.5-μm process [44]. Therefore, we pay special attention to this type without being limited to them. The BiCMOS driver we are going to used throughout this work is the conventional LS totem-pole inverter shown in Figure 2.4.

Existing analytical device models were also reviewed for MOS and bipolar transistors. Because the modeling of MOS and bipolar devices is beyond the scope of this work, only a brief presentation of these models was made here for convenience. A detailed device model analysis can be found in [58, 19]. To enable the first order circuit analysis and design these analytical models were simplified up to an easy to use form.
Appendices

2.a  A Device Model for MOS Transistors

In this appendix we make a brief presentation of the MOS device model used by well known SPICE simulator [60].

In our approach we start off from the MOS transistor analytical model implemented in SPICE level 3 [2], which gives a very accurate device behavior description. The disadvantage of using SPICE model is the complexity of calculating and measuring some of its parameters. The SPICE parameters, however, are industrial standard and generally they are available. Once we have the SPICE parameters by using simplified, yet accurate, device equations good approximation results can be obtained.

The MOS device model we present next is an N-channel device on a silicon substrate. The same results apply also to its PMOS counterpart.

The level 3 SPICE model for drain-source current $I_{DS}$ has the following simple equation in both linear and saturation regions:

\[(2.a.1) \quad I_{DS} = \beta \left( V_{GS} - V_{TH} - \frac{1 + F_B}{2} V_{DS} \right) V_{DS} \]

where $\beta$ is the transistor transconductance, $V_{GS}$, $V_{DS}$ and $V_{TH}$ are the gate-source, drain-source and threshold voltages respectively. The term $F_B$ expresses the dependence of the channel depletion layer charge ($Q_B$) on the three dimensional geometry of the transistor. It is given by

\[(2.a.2) \quad F_B = \frac{\gamma F_s}{4\sqrt{2}\phi_p - V_{BS}} + F_n \]

where the term $F_s$ captures the short channel effects, while the term $F_n$ accounts for narrow channel effect; $V_{BS}$ is the bulk-source voltage, $\phi_p$ is the Fermi potential in a $p$-type material, and $\gamma$ is the body-effect parameter.

The parameters $\phi_p$, $\gamma$, and $F_n$ are calculated as follows [2]:

\[(2.a.3) \quad \gamma = \frac{\sqrt{2}\varepsilon_S q N_A}{C_{ox}} \]

\[\phi_p = \frac{kT}{q} log \frac{N_A}{n_i} \]

\[F_n = \delta \frac{\pi \varepsilon_S}{2 C_{ox} W_{eff}} \]

with

\[C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}} \]
where \(C_{ox}\) is the capacitance per unit area of the thin oxide layer, \(t_{ox}\) the oxide thickness, \(W_{eff}\) is the effective channel width, \(\delta\) is width channel effect on threshold voltage, \(\varepsilon_{ox}, \varepsilon_S\) are the oxide and silicon permittivity respectively, \(N_A, n_i\) are substrate concentration and intrinsic carrier concentration in silicon, and finally \(k, e, T\) are Boltzmann’s constant, electron charge, and temperature in Kelvin degrees respectively.

In a small channel device the surface electric field is practically always higher than the critical field. Therefore the dependence of the surface mobility on the gate electric field (vertical electric field) can be modeled with a simple equation

(2.a.4) \[ \mu_s = \frac{\mu_0}{1 + \theta(V_{GS} - V_{TH})} \]

where \(\mu_0\) is the surface mobility for small electric fields, and \(\theta\) is the mobility modulation (see Table 2.1). The effect of channel length on the mobility is translated as the mobility variation due to the average electric field between the source and the drain (lateral average electric field). Consequently, the effective carrier mobility can be expressed as

(2.a.5) \[ \mu_{eff} = \frac{\mu_s}{1 + \frac{\mu_s}{v_{max}L_{eff}} V_{DS}} \]

where \(L_{eff} = L - L_d\) is the effective channel length as shown in Figure 2.10; \(L\) is the designed channel length, and \(L_d\) is the total reduction in the channel length due to the lateral diffusion; \(v_{max}\) is the maximum drift velocity of carriers (saturation velocity).

Finally, the expression for transistor transconductance is

(2.a.6) \[ \beta = \mu_{eff} C_{ox} \frac{W_{eff}}{L_{eff}} \]

where, \(W_{eff} = W - W_d\) is the effective channel width, which equals the difference between the designed channel width \(W\) and the total reduction in the channel width due to the lateral diffusion \(W_d\).

The calculation of transistor threshold voltage is made in a complex way. The model accounts for both short and narrow-channel effects, as well as for the dependence on the drain-source voltage. Dang [10] proposed the following equation for the variable threshold voltage:

(2.a.7) \[ V_{TH} = V_{FB} + 2\phi_p - \sigma V_{DS} + \gamma F_s \sqrt{2\phi_p - V_{BS}} + F_n(2\phi_p - V_{BS}) \]

where \(V_{FB}\) is the flat-band voltage and \(\sigma\) is the threshold voltage dependence upon \(V_{DS}\). Its expression is

\[ \sigma = \frac{8.15 \times 10^{-22}}{C_{ox} L_{eff}^3} \]
The relationship between the parameter $\sigma$ and $L_{eff}$ is empirical. It is based on a SPICE fitting parameter $\eta$, called static feedback on threshold voltage.

The flat-band voltage $V_{FB}$ is a process parameter. It depends primarily upon the gate electrode type (aluminum, molibden, polysilicon etc), the doping concentration of the semiconductor and the oxide-silicon interface charges. It is given by:

$$V_{FB} = \phi_{ms} - \frac{Q_o}{C_{ox}}$$

where $\phi_{ms}$ is the work function difference between the gate electrode and the semiconductor and $Q_o$ is the total charges in the oxide and near the oxide-silicon interface. It has a positive value.

![Diagram of MOS transistor](image)

**Figure 2.10.** Parameters used in geometrical calculation of the MOS device current under trapezoidal approximation; the source and drain electrodes are grounded.

We now determine the correction factor $F_s$ for short-channel effect. It is based on a modified trapezoidal model for estimating the depletion layer charge (see Figure 2.10). The term $F_s$ is calculated as [10]:

$$F_s = 1 - \frac{X_j}{L_{eff}} \left( \frac{L_d + W_c}{X_j} \right) \sqrt{1 - \left( \frac{W_p}{X_j + W_p} \right)^2} - \frac{L_d}{X_j}$$
$W_p$ is the thickness of the depleted region on the flat source junction; i.e.,

\begin{equation}
W_p = X_D \sqrt{\phi_j - V_{BS}}
\end{equation}

\begin{equation}
X_D = \sqrt{\frac{2\varepsilon_S}{qN_A}}
\end{equation}

\begin{equation}
\phi_j = \frac{E_g}{2} + kT \log \frac{N_A}{n_i}
\end{equation}

with $E_g$ the silicon bandgap energy and $\phi_j$ the bulk-junction potential. $W_c$ is the thickness of the depleted cylindrical region of the source-substrate junction. It is given by the relation:

\begin{equation}
\frac{W_c}{X_j} = 0.0631353 + 0.8013292 \frac{W_p}{X_j} - 0.0111077 \left( \frac{W_p}{X_j} \right)^2
\end{equation}

In the first-order MOS theory, the border between the linear and saturation regions occurs at $V_{DS, sat} = V_{GS} - V_{TH}$. In short-channel transistors, this value is actually lower due to the velocity saturation effects. In this case the saturation voltage is reached when the carriers in the channel become velocity saturated. This new value is expressed as:

\begin{equation}
V_{DS, sat} = V_a + V_b - \sqrt{V_a^2 + V_b^2}
\end{equation}

\begin{equation}
V_a = \frac{V_{GS} - V_{TH}}{1 + F_B}
\end{equation}

\begin{equation}
V_b = \frac{v_{max} L_{eff}}{\mu_s}
\end{equation}

where $V_a$ is the saturation voltage if $v_{max}$ is not taken into account and $V_b$ is the velocity saturation effect on transistor saturation voltage.

At this point we have determined all the parameters encountered in the drain-source current equation (2.a.1). Consequently, substituting (2.a.5) and (2.a.6) in (2.a.1), the drain current can be further expressed as:

\begin{equation}
I_{DS}(V_{DS}) = C_o \frac{W_{eff}}{L_{eff}} \frac{\mu_s V_{DS}}{1 + \frac{\mu_s}{v_{max} L_{eff}} V_{DS}} \left( V_{GS} - V_{TH} - \frac{1 + F_B V_{DS}}{2} \right)
\end{equation}

This expression is still simplified but we will keep it like this for convenience. The correction factor for short and narrow channel effects $F_B$ is determined step by step with the relations (2.a.2), (2.a.3), (2.a.9) and (2.a.10). Further, the transistor threshold voltage $V_{TH}$ is determined with the relations (2.a.7). The surface mobility $\mu_s$ varies upon the gate-source and threshold voltages. It can be calculated
with the relation (2.a.4). Finally, the drain saturation current can be obtained by replacing the drain-source voltage in the above equation with its saturation value given by relations (2.a.12).

For convenience, the SPICE level 3 model parameters for a NMOS and PMOS transistors are shown in Table 2.1 together with its typical values for 0.8-μm BiCMOS process [11]. Both the symbols we used throughout this presentation and the corresponding SPICE keywords for the device parameters are given in the table together with a brief parameter description. For more informations concerning the MOS SPICE parameters and the way they are obtained, the reader is referred to [2].

<table>
<thead>
<tr>
<th>Symbol</th>
<th>SPICE keyword</th>
<th>Parameter description</th>
<th>NMOS</th>
<th>PMOS</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{TH0}$</td>
<td>VTO</td>
<td>Zero-bias threshold voltage</td>
<td>0.64</td>
<td>-0.75</td>
<td>V</td>
</tr>
<tr>
<td>$t_{ox}$</td>
<td>TOX</td>
<td>Thin oxide thickness</td>
<td>20e-9</td>
<td>20e-9</td>
<td>m</td>
</tr>
<tr>
<td>$N_A$</td>
<td>NSUB</td>
<td>Substrate doping</td>
<td>5.512e16</td>
<td>5.504e16</td>
<td>cm$^{-3}$</td>
</tr>
<tr>
<td>$N_{FS}$</td>
<td>NFS</td>
<td>Surface-fast state density</td>
<td>1.09e12</td>
<td>1e11</td>
<td>cm$^{-2}$</td>
</tr>
<tr>
<td>$X_j$</td>
<td>XJ</td>
<td>Metallurgical junction depth</td>
<td>0.20e-6</td>
<td>0.225e-6</td>
<td>m</td>
</tr>
<tr>
<td>$L_d$</td>
<td>LD</td>
<td>Lateral diffusion</td>
<td>0.0</td>
<td>0.166e-6</td>
<td>m</td>
</tr>
<tr>
<td>$\mu_o$</td>
<td>UO</td>
<td>Surface mobility</td>
<td>436.3</td>
<td>144.6</td>
<td>cm$^2$/Vs</td>
</tr>
<tr>
<td>$v_{max}$</td>
<td>VMAX</td>
<td>Maximum drift velocity of carriers</td>
<td>150100</td>
<td>152100</td>
<td>m/sec</td>
</tr>
<tr>
<td>$\delta$</td>
<td>DELTA</td>
<td>Width effect on threshold voltage</td>
<td>0.0</td>
<td>0.155</td>
<td></td>
</tr>
<tr>
<td>$\eta$</td>
<td>ETA</td>
<td>Static feedback on threshold voltage</td>
<td>0.125</td>
<td>0.0125</td>
<td></td>
</tr>
<tr>
<td>$\theta$</td>
<td>THETA</td>
<td>Mobility modulation</td>
<td>2.45e-2</td>
<td>6.24e-2</td>
<td>V$^{-1}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter description</th>
<th>NMOS</th>
<th>PMOS</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_j$</td>
<td>PB</td>
<td>Bulk-junction potential</td>
<td>0.802</td>
<td>0.470</td>
</tr>
<tr>
<td>$C_j$</td>
<td>CJ</td>
<td>Zero-bias bulk capacitance</td>
<td>5.99e-4</td>
<td>3.68e-4</td>
</tr>
<tr>
<td>$M_j$</td>
<td>MJ</td>
<td>Bulk-junction grading coefficient</td>
<td>0.31</td>
<td>0.213</td>
</tr>
<tr>
<td>$C_{JSW}$</td>
<td>CJSW</td>
<td>Zero-bias side wall capacitance</td>
<td>1.03e-10</td>
<td>1.76e-10</td>
</tr>
<tr>
<td>$M_{JSW}$</td>
<td>MJSW</td>
<td>Sidewall capacitance grading coeff.</td>
<td>0.2</td>
<td>0.33</td>
</tr>
<tr>
<td>$C_{GDO}$</td>
<td>CGDO</td>
<td>Gate-drain overlap capacitance</td>
<td>3.19e-10</td>
<td>3.40e-10</td>
</tr>
<tr>
<td>$C_{GSO}$</td>
<td>CGSO</td>
<td>Gate-source overlap capacitance</td>
<td>3.19e-10</td>
<td>3.40e-10</td>
</tr>
<tr>
<td>$R_D$</td>
<td>RD</td>
<td>Drain ohmic resistance</td>
<td>24.0</td>
<td>100.7</td>
</tr>
<tr>
<td>$R_S$</td>
<td>RS</td>
<td>Source ohmic resistance</td>
<td>24.0</td>
<td>100.7</td>
</tr>
</tbody>
</table>

Table 2.1. SPICE MOS transistor model parameters for a 0.8-μm BiCMOS process.

In this appendix we’ve shown how the MOS transistor drain current can be determined. This drain current is a key parameter in calculating the BiCMOS driver delay. Consequently, accurate drain current estimation is required. To obtain a good accuracy, physical, process, measured and fitted parameters are involved to account for short and narrow channel effects. The resulted equations are cumbersome but
most of them are calculate only once. For the most used equations some simplifications are made.
2.b A Device Model for Bipolar Junction Transistors

In this appendix we make a brief presentation of the bipolar device model used by well known SPICE simulator [60].

In the Gummel-Poon [22] model the bipolar transistor is described as a charge-controlled device, meaning that both emitter and collector currents are proportional to the total excess minority carrier charge stored in the transistor base. This model is valid for all operating modes of the bipolar transistor. In case of a large-signal model, which is the case for switching propose, the charge storage is modeled by three types of capacitances: nonlinear junction capacitances or depletion capacitances, diffusion capacitances, and substrate capacitance.

The diffusion capacitances are associated with the mobile carriers in the transistor. For the carriers injected from the emitter into the base, when the base-emitter junction is forward biased, the associated diffusion capacitance is expressed as [2]:

\[ C_{DE} = \frac{d(\tau_F I_C)}{d V_{BE}} \]

where, \( \tau_F \) is the mean forward transit time, by means of the time required for electrons to travel across the base region, \( V_{BE} \) is the base-emitter voltage, and \( I_C \) is the current in forward active mode, since it is due to the carrier injection from emitter to collector.

The mean forward transit time depends on the collector current and is determined by the following empirical equation [2]:

\[ \tau_F = \tau_{F0} \left( 1 + X_{\tau_F} \left( \frac{I_C}{I_C + I_{RF}} \right)^2 \exp(V_{BC}/1.44V_{TF}) \right) \]

where \( \tau_{F0} \) is the ideal forward transit time, \( V_{BC} \) is the base-collector voltage, and \( X_{\tau_F}, I_{RF} \) and \( V_{TF} \) are fitting parameters described in Table 2.2. The transit time dependence on the collector current is due to the base-widening effect, while its dependence on base-collector voltage is due to the base-width modulation effect.

A simpler expression of the mean forward transit time dependence upon the collector current was derived by Getreu [19]. It is given as:

\[ \tau_F = \tau_{F0} \left( 1 + \frac{I_C}{I_{KF}} \right) \]

where \( I_{KF} \) is the knee current as described in high-level injection behavior of the bipolar transistor [2]. This second expression for the transit time gives, unfortunately, not sufficiently accurate results in circuit analysis.

The base-collector diffusion capacitance is given by [2]:

\[ C_{DC} = \frac{d(\tau_R I_E)}{d V_{BC}} \]
with $\tau_R$ the reverse transit time, and $I_E$ the current in the reverse active mode, since it is due to the injection from the collector to the emitter.

The nonlinear junction capacitances are due to the charge stored in the depletion regions of the transistor, at the emitter and collector junctions. Therefore, whenever there is a voltage change across either of these junctions, the capacitances will also change. These capacitances, denoted by $C_{BE}$ for the base-emitter junction and $C_{BC}$ for the base-collector junction, can be written as follows [2]:

\[
C_{BE} = \frac{C_{BE0}}{(1 - V_{BE}/\phi_E)^{m_E}} \\
C_{BC} = \frac{C_{BC0}}{(1 - V_{BC}/\phi_C)^{m_C}}
\]

where $C_{BE0}$ and $C_{BC0}$ are the values of the emitter-base and collector-base junction capacitances at $V_{BE} = 0$ and $V_{BC} = 0$ respectively, $\phi_E$ and $\phi_C$ are the emitter-base and collector-base junction built-in potentials, and $m_E$ and $m_C$ are the emitter-base and collector-base capacitance grading factors.

![Figure 2.11. Large-signal equivalent circuit of bipolar transistor.](image)

The BJT large-signal equivalent circuit is shown in Figure 2.11. The two diodes represent the intrinsic base-emitter and base-collector junctions. The currents associated with these two diodes are dependent on both the direct (reverse) current through the device and the transistor direct (reverse) current gain $\beta_F$ ($\beta_R$) corresponding to forward (reverse) active operating mode.

A very important model parameter for the device behavior is the transistor direct current gain. It is strongly dependent on collector current at low currents as well as at high currents. However, the prime concern is its variation at high level injection which can be modeled as [19]:

\[
\beta_F = \frac{\beta_{F0}}{1 + I_C/I_{KF}}
\]

where $\beta_{F0}$ is the ideal maximum current gain. In case of low level injection the current gain equals the ideal one.
The large-signal model also includes the parasitic resistance of emitter, collector and base regions. The collector resistance $R_C$ decreases the slope of the curves in saturation region for low collector-emitter voltages. It is a function of collector current and base-collector voltage but it is taken constant in the model.

The emitter resistance $R_E$ is mainly the emitter contact resistance since the emitter region is very heavily doped and, consequently, has a small equivalent resistance. Usually, it has a small value and can be assumed constant.

The base resistance $R_B$ has two distinct components: the resistance of the base contact together with the sheet resistance of the external base resistance, and the resistance of the active area of the base region (lying directly under the emitter). The former has a constant value but the latter is a direct function of the base current. In the SPICE model the total base resistance is expressed as follows:

\[(2.6.7) \quad R_{total} = R_{BM} + 3(R_B - R_{BM})\left(\frac{\tan z - z}{z \tan^2 z}\right)\]

with

\[(2.6.8) \quad z = \frac{-1 + \sqrt{1 + 144I_B/\pi^2 I_{R_B}}}{(24/\pi^2)\sqrt{I_B/I_{R_B}}}\]

where $R_{BM}$ is the minimum high current base resistance; $R_B$ is the base resistance at zero bias; $I_{R_B}$ is the current where the base resistance falls halfway to its minimum value and $I_B$ is the base current.

Table 2.2 resumes the bipolar transistor SPICE parameters for a 0.8-$\mu$m BiCMOS process. The shown parameters characterize an NPN bipolar transistor with an emitter area of 4 $\mu$m$^2$. Both the symbols we used throughout this presentation and the corresponding SPICE keywords for the device parameters are given in the table together with a brief parameter description. We choose the SPICE parameters for an NPN bipolar transistor because this transistor type is most used in modern BiCMOS circuits due to its better performances over its PNP counterpart.

In this appendix we briefly introduced the large-signal model for an NPN bipolar junction transistor. This model is very useful in switching characterization of the device. The model is used in chapter 3 for a transient delay analysis of the BiCMOS driver.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>SPICE keyword</th>
<th>Parameter description</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_S$</td>
<td>IS</td>
<td>Saturation current</td>
<td>4.68e-18</td>
<td>A</td>
</tr>
<tr>
<td>$\beta_F$</td>
<td>BF</td>
<td>Ideal maximum forward current gain</td>
<td>130.25</td>
<td></td>
</tr>
<tr>
<td>$\beta_R$</td>
<td>BR</td>
<td>Ideal maximum reverse current gain</td>
<td>2.35</td>
<td></td>
</tr>
<tr>
<td>$n_F$</td>
<td>NF</td>
<td>Forward current emission coefficient</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$n_R$</td>
<td>NR</td>
<td>Reverse current emission coefficient</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$I_{se}$</td>
<td>ISE</td>
<td>Base-emitter leakage saturation current</td>
<td>174e-18</td>
<td>A</td>
</tr>
<tr>
<td>$I_{sc}$</td>
<td>ISC</td>
<td>Base-collector leakage saturation current</td>
<td>648e-18</td>
<td>A</td>
</tr>
<tr>
<td>$I_{KF}$</td>
<td>IKF</td>
<td>Forward-knee current</td>
<td>12</td>
<td>mA</td>
</tr>
<tr>
<td>$I_{KR}$</td>
<td>IKR</td>
<td>Reverse-knee current</td>
<td>0.2542</td>
<td>mA</td>
</tr>
<tr>
<td>$n_E$</td>
<td>NE</td>
<td>Base-emitter leakage emission coefficient</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>$n_C$</td>
<td>NC</td>
<td>Base-collector leakage emission coefficient</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>$V_A$</td>
<td>VAF</td>
<td>Forward Early voltage</td>
<td>35</td>
<td>V</td>
</tr>
<tr>
<td>$V_B$</td>
<td>VAR</td>
<td>Reverse Early voltage</td>
<td>3.5</td>
<td>V</td>
</tr>
<tr>
<td>$R_C$</td>
<td>RC</td>
<td>Collector resistance</td>
<td>80</td>
<td>Ω</td>
</tr>
<tr>
<td>$R_E$</td>
<td>RE</td>
<td>Emitter resistance</td>
<td>20.3</td>
<td>Ω</td>
</tr>
<tr>
<td>$R_B$</td>
<td>RB</td>
<td>Zero-biased base resistance</td>
<td>312</td>
<td>Ω</td>
</tr>
<tr>
<td>$R_{BM}$</td>
<td>RBM</td>
<td>Minimum high current base resistance</td>
<td>6.25</td>
<td>Ω</td>
</tr>
<tr>
<td>$I_{RB}$</td>
<td>IRB</td>
<td>Base current where $R_B = (R_B(0) + R_{BM})/2$</td>
<td>32</td>
<td>μA</td>
</tr>
<tr>
<td>$\tau_{F0}$</td>
<td>TF</td>
<td>Ideal forward transit time</td>
<td>11.5</td>
<td>ps</td>
</tr>
<tr>
<td>$\tau_{R}$</td>
<td>TR</td>
<td>Ideal reverse transit time</td>
<td>500</td>
<td>ps</td>
</tr>
<tr>
<td>$X_{rF}$</td>
<td>XTF</td>
<td>$\tau_{F0}$ bias dependent coefficient</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>$V_{rF}$</td>
<td>VTF</td>
<td>$\tau_{F0}$ base-collector voltage</td>
<td>2.5</td>
<td>V</td>
</tr>
<tr>
<td>$I_{TF}$</td>
<td>ITF</td>
<td>$\tau_{F0}$ high current parameter</td>
<td>10</td>
<td>mA</td>
</tr>
<tr>
<td>$C_{BE}$</td>
<td>CJE</td>
<td>Zero-bias base-emitter depletion capacitance</td>
<td>9.4</td>
<td>fF</td>
</tr>
<tr>
<td>$\phi_E$</td>
<td>VJE</td>
<td>Base-emitter built-in potential</td>
<td>0.9</td>
<td>V</td>
</tr>
<tr>
<td>$m_E$</td>
<td>MJE</td>
<td>Base-emitter junction grading coefficient</td>
<td>0.41</td>
<td></td>
</tr>
<tr>
<td>$C_{BC}$</td>
<td>CJC</td>
<td>Zero-bias collector substrate capacitance</td>
<td>11.9</td>
<td>fF</td>
</tr>
<tr>
<td>$\phi_C$</td>
<td>VJC</td>
<td>Base-collector built-in potential</td>
<td>0.64</td>
<td>V</td>
</tr>
<tr>
<td>$m_C$</td>
<td>MJC</td>
<td>Base-collector junction grading coefficient</td>
<td>0.333</td>
<td></td>
</tr>
<tr>
<td>$C_{CS}$</td>
<td>CJIS</td>
<td>Zero-bias collector substrate capacitance connected to internal base node</td>
<td>38.1</td>
<td>fF</td>
</tr>
<tr>
<td>$\phi_S$</td>
<td>VJS</td>
<td>Substrate-junction built-in potential</td>
<td>0.65</td>
<td>V</td>
</tr>
<tr>
<td>$m_S$</td>
<td>MJS</td>
<td>Substrate-junction exponential factor</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>$X_{CJC}$</td>
<td>XCJC</td>
<td>Fraction of base-collector depletion</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>$FC$</td>
<td>FC</td>
<td>Coefficient for forward-bias depletion cap.</td>
<td>0.875</td>
<td></td>
</tr>
<tr>
<td>$X_{TB}$</td>
<td>XTB</td>
<td>Forward and reverse $\beta$ temp. coeff.</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>$XTI$</td>
<td>XTI</td>
<td>Saturation current temperature exponent</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2. SPICE BJT model parameters for a 0.8-μm BiCMOS process and an NPN BJT with an emitter area of 4.0 μm².
Chapter 3

DELAY ANALYSIS OF THE TOTEM-POLE INVERTER

Both analysis and modeling of circuit delay are very important steps in the ASIC design process. Especially, for BiCMOS standard gate libraries these steps are essential. BiCMOS gates are very attractive because they have a much better driving capability than their CMOS counterparts. The nonlinear nature of bipolar transistor, however, makes its physical design much more complicated. The current device equation are very cumbersome. Consequently, to get circuit behavior versus physical device parameters for a BiCMOS driver, simplifying assumptions have to be made. However, the estimation of the delay in an ASIC logic system requires accurate, yet fast, tools. Therefore, the delay model incorporated in such tools must be both accurate and easy to use. An accurate analytical gate delay model, which can be used to estimate the pull-up and pull-down times, is a key to various optimization tasks. Such a delay is very useful for understanding the effect of different physical device parameters upon the delay time.

Next the BiCMOS driver analytical model for transient delay analysis is deduced as a function of the output load capacitance. The short and narrow channel effects on MOS device equations and the high level injection effects on bipolar transistor device equations are considered when deriving the circuit equations. Two analytical delay models are developed. One which is very accurate but computational expensive, and another one less accurate but easier to use.

3.1 Some Definitions

Before analyzing the delay characteristics of the gate some parameters need to be defined to avoid later ambiguities. These ambiguities originate in delay time definition and its dependence on the input signal slew rate.

As shown in the previous section the BiCMOS gate will switch state when the input voltage equals the inversion voltage. This level is also referred in the literature as switch voltage of the gate or, shorter, the gate threshold. Since the inversion voltage is a function of gate geometry, its value is specific to the gate. This assertion is also valid for a CMOS gate. However, the gates are not used independent.
Therefore, the definition of the gate delay requires the same logic switch voltage for every gate in the circuit. In this work the logic switch voltage is defined as 50% power supply voltage $V_{DD}$. This value differs from the actual inversion voltage of an individual gate, but it is very convenient and practically useful in circuit design and optimization. The main advantage consists in an easy to compute delay for a long chain of gates by adding the individual gate delays, since the same gate delay is assigned for all the gates.

DEFINITION 1 (GATE DELAY OR PROPAGATION DELAY) The time between the moment when the input signal crosses the logic switch voltage (50%$V_{DD}$) to the moment when the output signal crosses the same logic switch voltage is referred to as the gate delay $t_d$; in particular the gate delay for a rising input signal and falling output signal is referred to as the fall gate delay ($t_{d,fall}$), while the gate delay for a falling input signal and rising output signal is referred to as the rise gate delay ($t_{d,rise}$).

Because of the logic switch voltage chosen to be 50%$V_{DD}$ the gate delay is often referred to as 50% delay of the gate. The fall and rise gate delays are illustrated in Figure 3.1. These delays are not necessarily equal. They are dependent on gate design. The gate parameters must be very well tuned to assure equal delays.

![Figure 3.1. Definition of gate delay, fall and rise delay.](image)

Furthermore, another two typical delays are defined for the gate output transitions.

DEFINITION 2 (RISE DELAY) The time interval between the moment when the output signal crosses its 10%$V_{DD}$ value to the moment when the output signal crosses its 90%$V_{DD}$ value is referred to as the rise delay $t_{rise}$.

DEFINITION 3 (FALL DELAY) The time interval between the moment when the output signal crosses its 90%$V_{DD}$ value to the moment when the output signal crosses its 10%$V_{DD}$ value is referred to as the fall delay $t_{rise}$. 
However, in circuit timing analysis the signal waveforms are linearized to alleviate the computational complexity. When the linearization takes place we calculate the rise and fall times between the ground and the power supply voltage values.

3.2 Static Characteristic of a BiCMOS Inverter

The delay of the BiCMOS gate depends, among the others, upon the input signal slew rate. For a better understanding of this problem a static switching analysis of the BiCMOS inverter is helpful. The static characteristics of the totem-pole BiCMOS inverter shown in Figure 3.3 (on page 41) are discussed. The input voltage of the inverter is changed step by step so that the instantaneous output voltage of the gate is determined by the instantaneous input voltage. The relationship between the input and the output voltages can be found using the MOS and bipolar equations presented in the previous chapter.

The SPICE simulations of the static transfer characteristic are shown in Figure 3.2, together with the corresponding current flows. The characteristic has seven distinct regions. Each region has been explained in detail in [15]. Here, we make only a brief review as far as necessary for understanding input slope influence on BiCMOS inverter behavior.

In the first region both bipolar transistors and the NMOS transistor $M_2$ are off, while the PMOS transistor $M_1$ operates in the linear region. The output voltage is at its high level. As the input voltage increases, the NMOS transistor $M_2$ turns on charging the base of the bipolar transistor $Q_2$ (region 2). The bleeding transistors $M_3$ and $M_4$ are also on. When the base emitter voltages of both bipolar transistors $Q_1$ and $Q_2$ reach $V_{BE, on}$, they both start conducting (region 3). The output voltage slowly starts dropping. The input voltage value at the beginning of region 3 is denoted $V_h$.

In region 4, both MOS transistors $M_1$ and $M_2$ are saturated and the collector currents of both bipolar transistors reach the same maximum value. This region 4 corresponds to a point. It is the moment when the gate will switch from high to low level. In this point $V_{in} = V_{inv}$ is denoted as the inversion voltage.

After the top point in region 4, all the currents in the inverter devices start decreasing (region 5). In region 6, the bipolar transistors are off again. The bleeding devices are discharging the bipolar transistor bases. The output voltage slowly reaches its low level value. The input voltage value when the output voltage is low is denoted $V_l$.

In the last region (7), the MOS transistors $M_2$ and $M_3$ are in the linear region with their drain-source voltages equal zero. All other devices are off.
3.3 Stages in the Pull-up Transient

As shown in chapter 2 a large variety of BiCMOS drivers exists. However, they all have a common basic structure for pull-up and pull-down stages. The pull-up stage consists of a PMOS transistor driving a bipolar transistor (NPN or PNP), while the pull-down stage consists of an NMOS transistor driving a bipolar transistor (NPN or PNP). Consequently, the switching behavior of all the conventional BiCMOS gates is qualitatively the same. We can limit our analysis to the basic switching structure and extend this one with minor modifications to any other gate. Since the pull-up and pull-down analysis are very similar, only the pull-up analysis is presented here. For this particular stage we have chosen the PMOS -NPN structure because it is widely used in BiCMOS circuit design.

The delay model of the BiCMOS gate must account sufficiently for the important device parameters and circuit phenomena. They can be summarized as:

- the short channel effects on drain-source current of the MOS transistor
3.3 Stages in the Pull-up Transient

- the parasitic capacitances for both MOS and bipolar transistors
- the high-level injection influence on the current gain and base transit time of bipolar transistor
- the parasitic resistances of the bipolar transistor

The analytical model presented in this chapter accounts for most of these so-called "secondary effects" to obtain a better physical insight into the gate delay. Consequently, near optimum physical BiCMOS gate design is made possible. An optimized design of BiCMOS gate at the circuit level is also presented in this work.

We choose a BiCMOS inverter for our delay analysis. However, the results can be easily extended to a variety of BiCMOS logic circuits, e.g. for each PMOS or NMOS logic block, in the standard cell configuration, an equivalent PMOS or NMOS transistor can be found to approximate the current driving characteristics of their counterparts for any given input combinations [28]. Thus, the delay analysis of any BiCMOS standard cell can always be reduced to the problem of estimating the delay of a BiCMOS inverter circuit.

For the totem-pole BiCMOS inverter in Figure 3.3 the pull-up and pull-down stages are sketched with dashed lines. During the pull-up switch the output capacitance $C_{\text{load}}$ is charged via $M_1$ and $Q_1$, while during the pull-down phase the same capacitance is discharged via $M_2$ and $Q_2$.

![Pull-up and pull-down stages for a conventional BiCMOS inverter.](image)

The equivalent circuit of the pull-up stage driving a load capacitance is shown in Figure 3.4. The PMOS transistor is modeled by a constant current source whose
current equals the average device current $I_{MOS}$. The average PMOS current depends upon the gate and drain biases.

The equivalent large-signal circuit shown in Figure 2.11 (on page 34) has been used to model the bipolar transistor. In BiCMOS gate configuration the bipolar transistor is operated in the forward active mode. The reverse currents in the device are small. Therefore, they can be neglected. The voltage drop across the base resistance is also typically small. Its value is significant at zero bias, but it drops substantially when the base current increases. Consequently, we choose not to account for the base resistance in our analysis. The reader is referred to [4] for a transient delay analysis which incorporates the base resistance as well.

In Figure 3.4, $C_{BE}$ and $C_{BC}$ are the base-emitter and the base-collector junction capacitances respectively. Their average values are calculated using the expressions (2.8) and (2.9) which include the capacitance dependence on the bias.

$C_P$ and $C_{CS}$ are parasitic capacitances. The capacitance $C_P$ accounts for all the parasitic capacitances at the drains of $M_1$ and $M_3$. It is the sum of the drain-substrate depletion capacitances of $M_1$ and $M_3$ and the gate-oxide capacitance of $M_4$. The capacitance $C_{CS}$ at the bipolar transistor emitter includes the drain-substrate depletion capacitance of $M_2$, the base-collector and the collector-substrate capacitances of $Q_2$. The total load capacitance equals the sum of external load capacitance $C_{LOAD}$ and $C_{CS}$.

$C_D$ is the diffusion capacitance of $Q_1$ calculated with the expression (2.b.1). The diode represents the intrinsic base-emitter junction as has been shown in the BJT large-signal equivalent circuit. Finally, $R_C$ and $R_E$ are parasitic resistances of collector and emitter respectively.

Figure 3.4. The transient equivalent circuit for the pull-up delay.

As we already mentioned the input signal slew rate is very important for the transient delay analysis of the BiCMOS as well as CMOS gates. However, attempts of
incorporating the input waveform slope in BiCMOS transient delay analysis has led to a non-analytical delay expression in [16]. Therefore, we first perform the transient delay analysis by applying a step voltage to the input of the gate. Subsequent, the gate delay dependence on the slew rate is derived from the gate characterization.

![Waveform diagrams](image)

**Figure 3.5.** Transient delay analysis for the totem-pole BiCMOS inverter; the current and voltage waveforms are derived using SPICE simulation.

The significant time intervals in the pull-up switching transient of the gate are shown in Figure 3.5. Looking at the physical circuit behavior we can distinguish four distinct time regions in the switching transient:

- **Junction delay** $t_i$:
  
  When a step voltage ($V_{DD}$ to zero volts) is applied at the BiCMOS inverter input, $M_1$ turns on, entering the saturation region. It supplies the saturation current $I_{DS, sat}$. The BJT $Q_1$ is off and its emitter and collector currents are practically zero. The base current $I_{BQ_1}$ equals the PMOS saturation current. The time required to turn $Q_1$ on, $t_i$, is the first delay component of the pull-up transient delay. It is also referred to as the junction delay [45]. During
this time the capacitive network is charged by \( I_{DS,sat} \) until the base-emitter voltage reaches \( V_{BE,on} \) value. The junction delay expression is:

\[
(3.1) \quad t_i = (C_P + C_{BC} + C_{BE}) \frac{V_{low} + V_{BE,on}}{I_{DS,sat}}
\]

where \( I_{DS,sat} \) is calculated using formula (2.3) in which drain-source voltage equals \( V_{DS,sat} \); \( V_{low} \) is the low output logic level.

**diffusion delay** \( t_d \)

The second delay component, referred to as the diffusion delay, is the time required to charge the diffusion capacitance of the bipolar transistor. The collector current is just beginning to rise and the output voltage changes slowly. As the base current continues to increase, so does the base voltage \( V_{BQ1} \). Initially, the transient is dominated by the charging of the diffusion capacitance but it rapidly turns in charging the output load capacitance. It is very difficult to make a demarcation between these two regions. As was found in [45] the diffusion delay is very short and therefore we neglect it.

**load delay** \( t_l \)

The third delay component \( t_l \), referred to as the load delay, is the time required to charge the output load capacitance to the switch voltage value \( V_{switch} = V_{DD}/2 \). The base and the output voltages continue to rise. Their time variations can be considered identical. The collector current is increasing until a maximum peak value is reached. The time when the collector current of the bipolar transistor peaks is about the time when the output voltage reaches its switch value. During the load delay the base-emitter voltage is assumed to be a constant. The load delay is derived in the following sections.

**saturation delay** \( t_{sat} \)

When the bipolar transistor operates at very high current density, which is the case for very high output load capacitances, it may saturate if the collector resistance is large. The result is a degradation in switching performance. Saturation occurs when

\[
(3.2) \quad V_{CE,sat} = V_{DD} - V_O(t_s) - I_C(t_s) R_C - (I_C(t_s) + I_{MOS}) R_E
\]

where \( t_s \) is the time at which \( Q_1 \) enters the saturation region. This time can be obtained by solving numerical the above equation where \( V_O(t_s) \) and \( I_C(t_s) \) will be derived later on in this chapter. The time required to charge the output node to \( V_{DD}/2 \) while the bipolar transistor is saturated is given by:

\[
(3.3) \quad t_{sat} = C_O(R_C + R_E) \log \frac{V_{DD} + I_{MOS} R_E - V_{CE,sat} - V_O(t_s)}{V_{DD}/2 + I_{MOS} R_E - V_{CE,sat}}
\]

A detailed saturation analysis is presented in [15].
Finally, the delay expression for the pull-up transient delay is given by:

\[ t_{\text{rise}} = t_i + t_d + t_i + t_{\text{sat}} \approx t_i + t_i \]  

The analytical model for the load delay is derived next.

### 3.4 Load Delay Models

During the load delay time the PMOS transistor \( M_1 \) operates in the saturation as well as the linear region. To calculate an accurate average current for the \( M_1 \) both these regions need to be considered. There are two distinct possibilities differentiated by the drain-source voltage \( V_{DS,1} \) applied to the transistor \( M_1 \) at the beginning of the load delay time. The \( V_{DS,1} \) value is given by:

\[ V_{DS,1} = V_{DD} - V_{BE,\text{on}} - V_{\text{low}} \]  

Therefore,

- if \( V_{DS,1} \geq V_{DS,\text{sat}} \)
  
  the PMOS transistor \( M_1 \) operates first in the saturation region and then enters the linear region.

- if \( V_{DS,1} < V_{DS,\text{sat}} \)
  
  the PMOS transistor \( M_1 \) is already operating in the linear region; during the load delay time the transistor never operates in the saturation region.

The transistor saturation voltage is calculated using the expression (2.a.12), page 30. Consequently, we derive two expressions for the average PMOS current corresponding to these distinct regions. The voltage drop across the transistor drain source terminals varies between \( V_{DS,1} \) and \( V_{DS,2} \), where \( V_{DS,2} \) is the drain-source voltage drop at the moment when the output reaches its switch value. It is given by:

\[ V_{DS,2} = V_{DD} - V_{BE,\text{on}} - V_{\text{switch}} \]  

When the PMOS transistor \( M_1 \) operates in both saturation and linear regions, its average current is calculated as follows:

\[ I_{\text{MOS}} = \frac{1}{\Delta V_{DS}} \left( I_{DS,\text{sat}} V_{DS}_{V_{DS,1}}^{V_{DS,\text{sat}}} + \int_{V_{DS,1}}^{V_{DS,2}} I_{DS}(V_{DS}) \, dV_{DS} \right) \]

if \( V_{DS,1} > V_{DS,\text{sat}} \)
The first term on the right hand side of the above equation is due to the saturation component of the current, while the second term is the linear region component. The gate-source and drain-source voltages as well as the transistor threshold voltage are taken in absolute value to avoid any ambiguities.

When the PMOS transistor $M_1$ operates only in the linear region then

\[
I_{\text{MOS}} = I_{\text{MOS,lin}} = \frac{1}{\Delta V_{DS}} \int_{V_{DS,1}}^{V_{DS,2}} I_{DS}(V_{DS}) \, dV_{DS} \quad \text{if } V_{DS,1} < V_{DS,\text{sat}}
\]

where \[
\Delta V_{DS} = V_{DS,2} - V_{DS,1}
\]

The saturation current $I_{DS,\text{sat}}$ is calculated using the formula (2.3) for $V_{DS,\text{sat}}$. The average PMOS transistor current in the linear region is derived by integrating the drain current (relation 2.3) over the integration limits. This results in:

\[
I_{\text{MOS,lin}} = \left\{ \frac{C_{ox}W_{eff}v_{\text{max}}}{\Delta V_{DS}}(V_{GS} - V_{TH0}) \left( V_{DS} - \frac{v_{\text{max}}L_{eff}}{\mu_{s,av}} \log(1 + \frac{\mu_{s,av}}{v_{\text{max}}L_{eff}}V_{DS}) \right) \right. \\
- \frac{C_{ox}W_{eff}v_{\text{max}}}{2\Delta V_{DS}}(1 + F_B) \left( \frac{V_{DS}^2}{2} - \frac{v_{\text{max}}L_{eff}}{\mu_{s,av}} V_{DS} \right) \\
+ \left( \frac{v_{\text{max}}L_{eff}}{\mu_{s,av}} \right)^2 \log(1 + \frac{\mu_{s,av}}{v_{\text{max}}L_{eff}}V_{DS}) \left\} \right|_{V_{DS,1}}^{V_{DS,2}}
\]

To determine the linear current component in equation (3.7) the integration limit $V_{DS,1}$ in the above equation should be replaced with $V_{DS,\text{sat}}$.

By applying Kirchhoff’s current and voltage laws in the equivalent circuit set up in Figure 3.4, the circuit switching transient can be described mathematically. Some simplifying assumptions need to be made to solve the resulted equations analytically. These simplifications are related to the high level injection effects which we already have presented in section 2.4. The relations between the bipolar transistor base, emitter and collector currents are also simplified.

When we use complex functions in modeling the high level injection, we need to simplify the circuit in order to derive a closed form analytical delay model. In contrary, when we use the average values to model the high level injection, we enable more accurate circuit analysis.

These two distinct simplifying assumptions lead to two analytical delay models. They are both derived and compared. They both use the average forward transit time given by the expression (2.4). The differences are summarized as follows:

- variable current gain model
3.5 Variable Current Gain Model

the current gain depends directly on the collector current (expression 2.b.6)
the emitter current is considered approximatively equal to the collector current

- **average current gain model**

- an average value for the forward current gain is taken (expression 2.7)
- the relation between the emitter and collector currents is derived by applying Kirchhoff's current law at the emitter node in the equivalent pull-up circuit (Figure 3.4)

A detailed presentation of these two models is made in the following sections. The delay models are derived with the assumption that the BJT operates in the forward active region.

### 3.5 Variable Current Gain Model

For the first transient delay model (variable current gain) we use the relation (2.b.6) to model the BJT forward current gain. Consequently, the relation between bipolar base and collector currents is expressed as:

\[
I_B = \frac{I_C}{\beta_F} = \frac{I_C}{\beta P_0} \left(1 + \frac{I_C}{I_{KF}}\right)
\]

Applying Kirchhoff's law at the base node (Figure 3.4) and replacing the base current with the above expression the following current equation is derived:

\[
I_{MOS} = C_P \frac{dV_B}{dt} + \frac{I_C}{\beta P_0} \left(1 + \frac{I_C}{I_{KF}}\right) + C_D \frac{dV_{BE}}{dt} + C_B \frac{dV_{BC}}{dt}
\]

where

\[
\frac{dV_B}{dt} \approx \frac{dV_O}{dt}
\]

\[
C_D \frac{dV_{BE}}{dt} \approx \tau_{F,av} \frac{dI_C}{dt}
\]

\[
I_E \approx I_C
\]

The average base transit time \(\tau_{F,av}\) is calculated using the relation (2.4), in which the integration limits for the base-collector voltage are

\[
V_{BC,1} = -V_{DD} + V_{BE, on} + V_{low}
\]

\[
V_{BC,2} = -V_{DD}/2 + V_{BE, on} + (R_E + R_C)I_C
\]
corresponding to the beginning and the end of the load delay time interval. The same integration limits are used in deriving the average base-collector junction capacitance using the expression (2.9).

Furthermore, the base-collector voltage $V_{BC}$ is derived from Kirchhoff's voltage law as:

$$\begin{align*}
V_{BC} &= V_{BE} + R_E I_C + V_O - (V_{DD} - R_C I_C) \\
&= \frac{I_C^2}{\beta F_0 I_{KF}} - \frac{1}{\beta F_0 I_{KF}} I_C - \beta F_0 I_{KF} I_{MOS} = -\beta F_0 I_{KF} \tau_0^* \frac{dI_C}{dt}
\end{align*}$$

where the base-emitter voltage is assumed to be a constant.

Substituting the relations (3.12), (3.13) in (3.11), the current equation becomes

$$\begin{align*}
I_C^2 &+ \frac{\beta F_0 I_{KF}}{\beta F_0} I_C - \beta F_0 I_{KF} I_{MOS} = -\beta F_0 I_{KF} \tau_0^* \frac{dI_C}{dt}
\end{align*}$$

where

$$\begin{align*}
\frac{1}{\beta F_0} &= \beta F_0 + \frac{C_{BC} + C_P}{C_O} \quad \text{with} \quad C_O = C_{CS} + C_{LOAD}
\end{align*}$$

and

$$\begin{align*}
\tau_0^* &= \tau_{F,av} + C_{BC}(R_C + R_E)
\end{align*}$$

The integral form of the above differential equation is obtained by separating its variables as follows:

$$\begin{align*}
\int dt &= -\beta F_0 I_{KF} \tau_0^* \int \frac{dI_C}{I_C^2 + \frac{\beta F_0 I_{KF}}{\beta F_0} I_C - \beta F_0 I_{KF} I_{MOS}}
\end{align*}$$

with the initial condition = \begin{align*}
t &= 0 \\
I_C(0) &= 0
\end{align*}

The solution of the integral equation is

$$\begin{align*}
I_C(t) &= \frac{\lambda_1 + \lambda_2}{2} + \frac{\lambda_1 - \lambda_2}{2} \tanh \left( \frac{t + \xi}{2d} \right)
\end{align*}$$

where

$$\begin{align*}
\lambda_{1,2} &= \frac{\beta F_0 I_{KF}}{2 \beta F_0} \pm \frac{1}{2} \sqrt{\left( \frac{\beta F_0 I_{KF}}{\beta F_0} \right)^2 + 4 \beta F_0 I_{KF} I_{MOS}}
\end{align*}$$

with $\lambda_1 > 0, \lambda_2 < 0, \lambda_1 - \lambda_2 > 0$ and $\lambda_1/\lambda_2 < 0$

$$\begin{align*}
d &= \frac{\beta F_0 I_{KF} \tau_0^*}{\lambda_1 - \lambda_2} \\
\xi &= -d \cdot \log \left( \frac{-\lambda_1}{\lambda_2} \right)
\end{align*}$$
In the collector current differential equation (3.14) the time is a positive quantity. Therefore to satisfy the physical conditions the collector current must vary between its initial value and a maximum limit, e.g. \(0 \leq I_C \leq \lambda_1\). Consequently, \(I_{C,\text{max}} = \lambda_1\).

Applying Kirchhoff's current law at the output node (Figure 3.4), and keeping in mind that we approximate the emitter current with the collector current, the output voltage \(V_O\) can be expressed as

\[
(3.19) \quad V_O(t) = \frac{1}{C_O} \int I_C(t) \, dt
\]

with the initial condition \[
\begin{aligned}
  t &= 0 \\
  V_O(0) &= V_{\text{low}}
\end{aligned}
\]

Substituting from (3.18) into (3.19) and integrating we obtain the expression for the output voltage

\[
(3.20) \quad V_O(t) = \frac{1}{C_O} \left( \frac{\lambda_1 + \lambda_2}{2} t + d(\lambda_1 - \lambda_2) \log \frac{\cosh \frac{t + d}{2}}{\cosh \frac{d}{2}} \right) + V_{\text{low}}
\]

in which the cosines hyperbolic function can be replaced with

\[
cosh(x) = \frac{e^x + e^{-x}}{2} = \frac{e^x}{2} (1 + e^{-2x})
\]

to obtain the final relation for the output voltage

\[
(3.21) \quad V_O(t) = \frac{\lambda_1 d}{C_O} \left( \frac{t}{d} + \left( 1 - \frac{\lambda_2}{\lambda_1} \right) \log \frac{1 - \frac{\lambda_1}{\lambda_2} \exp(-t/d)}{1 - \frac{\lambda_1}{\lambda_2}} \right) + V_{\text{low}}
\]

At this point we should be able to calculate the load delay \(t_l\) at which the output voltage reaches \(V_{DD}/2\).

the switch point is \[
\begin{aligned}
  V_O(t) &= V_{\text{switch}} \\
  \Delta V_O &= V_{\text{switch}} - V_{\text{low}}
\end{aligned}
\]

At the switch point the expression (3.21) becomes

\[
\frac{\Delta V_O C_O}{\lambda_1 d} = \frac{t_l}{d} + \left( 1 - \frac{\lambda_2}{\lambda_1} \right) \log \frac{1 - \frac{\lambda_1}{\lambda_2} \exp(-t_l/d)}{1 - \frac{\lambda_1}{\lambda_2}}
\]

which is a transcendental equation in \(t_l\). In other words, this equation does not have an analytical solution. It needs to be simplified to get a solvable equation in \(t_l\). It can be rewritten as

\[
-\frac{\Delta V_O C_O}{\lambda_1 d} = \log \frac{\exp(-t_l/d)}{1 - \frac{\lambda_1}{\lambda_2} \exp(-t_l/d)} + \left( 1 - \frac{\lambda_2}{\lambda_1} \right) \log \left( 1 - \frac{\lambda_1}{\lambda_2} \right) + \frac{\lambda_2}{\lambda_1} \log \left( 1 - \frac{\lambda_1}{\lambda_2} \exp(-t_l/d) \right)
\]
with $|\lambda_2| > |\lambda_1|$. Interchanging two terms yields:

$$
(3.22) \quad -\log \frac{\exp(-t_1/d)}{1 - \frac{\lambda_1}{\lambda_2} \exp(-t_1/d)} = \frac{\Delta V_0 C_0}{\lambda_1 d} + \left(1 - \frac{\lambda_2}{\lambda_1}\right) \log \left(1 - \frac{\lambda_1}{\lambda_2}\right) + \frac{\lambda_2}{\lambda_1} \log \left(1 - \frac{\lambda_1}{\lambda_2} \exp(-t_1/d)\right)
$$

In this form it should be noticed that if we manage to make the last term in the above equation independent on $t_i$, an analytical solution for $t_i$ can be obtained. At this point the exact value of $t_i$ is unknown, but we can easily derive its upper and lower values (denoted $t_{\text{max}}$ and $t_{\text{min}}$). Therefore we approximate the last term in the equation with its average over the interval $(t_{\text{min}}, t_{\text{max}})$. Consequently, the expression can be simplified to

$$
(3.23) \quad -\log \frac{\exp(-t_1/d)}{1 - \frac{\lambda_1}{\lambda_2} \exp(-t_1/d)} \approx \frac{\Delta V_0 C_0}{\lambda_1 d} + \left(1 - \frac{\lambda_2}{\lambda_1}\right) \log \left(1 - \frac{\lambda_1}{\lambda_2}\right) + \frac{\lambda_2}{\lambda_1} \Delta t_i \int_{t_{\text{min}}}^{t_{\text{max}}} \log \left(1 - \frac{\lambda_1}{\lambda_2} \exp(-t_1/d)\right) dt_i
$$

with $\Delta t_i = t_{\text{max}} - t_{\text{min}}$. We use the notation $F_{av}$ for the average value

$$
(3.24) \quad F_{av} = \frac{1}{\Delta t_i} \int_{t_{\text{min}}}^{t_{\text{max}}} \log \left(1 - \frac{\lambda_1}{\lambda_2} \exp(-t_1/d)\right) dt_i
$$

and the notation $\Theta$ for the right hand side term of the equation (3.23)

$$
(3.25) \quad \Theta = \frac{\Delta V_0 C_0}{\lambda_1 d} + \left(1 - \frac{\lambda_2}{\lambda_1}\right) \log \left(1 - \frac{\lambda_1}{\lambda_2}\right) + \frac{\lambda_1}{\lambda_2} F_{av}
$$

The solution of the equation (3.23) is now

$$
(3.26) \quad t_i = d \left(\Theta + \log \left(1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta)\right)\right)
$$

The delay time $t_i$ defined by the equation (3.26) is a function of variable $\Theta$ (written $t_i(\Theta)$), which has the following property:

**PROPERTY 1** The function $t_i(\Theta)$ is monotonically increasing for $\Theta > 0$, e.g., if $0 < \Theta_1 \leq \Theta_2$ implies $t_i(\Theta_1) \leq t_i(\Theta_2)$.

**PROOF** To prove that the function is monotonically increasing, we have to prove that if $\Theta_1 \leq \Theta_2$ then

$$
d \left(\Theta_1 + \log \left(1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_1)\right)\right) \leq d \left(\Theta_2 + \log \left(1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_2)\right)\right)
$$
The above expression can be rewritten as

\[ \Theta_1 - \Theta_2 + \log \frac{1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_1)}{1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_2)} \leq 0 \]

since \( 0 < \Theta_1 \leq \Theta_2 \) then \( \Theta_1 - \Theta_2 \leq 0 \) and \( \exp(-\Theta_1) \geq \exp(-\Theta_2) \). We know also that \( \frac{\lambda_1}{\lambda_2} < 0 \). That all implies

\[ 1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_1) \leq 1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_2) \]

\[ \Rightarrow \frac{1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_1)}{1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_2)} \leq 1 \]

\[ \Rightarrow \log \frac{1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_1)}{1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_2)} \leq 0 \]

q.e.d. \( \square \)

Since \( \frac{\lambda_1}{\lambda_2} < 0 \) and \( 0 < \exp(-t_i/d) < 1 \) it follows that the term we need to approximate is bounded by:

\[ 0 < \log \left( 1 - \frac{\lambda_1}{\lambda_2} \exp(-t_i/d) \right) < \log \left( 1 - \frac{\lambda_1}{\lambda_2} \right) \]

Consequently, the lower and upper values for \( \Theta \) in relation (3.25) are derived as:

\[ \Theta_{\text{min}} = \frac{\Delta V_0 C_0}{\lambda_1 d} + \left( 1 - \frac{\lambda_2}{\lambda_1} \right) \log \left( 1 - \frac{\lambda_1}{\lambda_2} \right) + \frac{\lambda_1}{\lambda_2} \log \left( 1 - \frac{\lambda_1}{\lambda_2} \right) \]

\[ = \frac{\Delta V_0 C_0}{\lambda_1 d} + \log \left( 1 - \frac{\lambda_1}{\lambda_2} \right) \]

\[ \Theta_{\text{max}} = \frac{\Delta V_0 C_0}{\lambda_1 d} + \left( 1 - \frac{\lambda_2}{\lambda_1} \right) \log \left( 1 - \frac{\lambda_1}{\lambda_2} \right) \]

As has been shown the function \( t_i(\Theta) \) is monotonically increasing in \( \Theta \). Therefore, we can write

\[ t_{\text{min}} = d \left( \Theta_{\text{min}} + \log \left( 1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_{\text{min}}) \right) \right) \]

\[ t_{\text{max}} = d \left( \Theta_{\text{max}} + \log \left( 1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta_{\text{max}}) \right) \right) \]

Now, we can find the average value \( F_{av} \) solving the defined integral (3.24) between the previous calculated limits \( t_{\text{min}} \) and \( t_{\text{max}} \). To solve the integral we need to make the following variable change:

\[ u = \exp(-t_i/d) \Rightarrow du = -\frac{1}{d} \exp(-t_i/d) dt_i \]
and the new integration limits become

\[ u_{\text{min}} = e^{p(-t_{\text{min}}/d)} \]
\[ u_{\text{max}} = e^{p(-t_{\text{max}}/d)} \]

Finally, the average function \( F_{av} \) can be approximated as

\[
F_{av} = \frac{1}{\Delta t_i} \int_{t_{\text{min}}}^{t_{\text{max}}} \log \left ( 1 - \frac{\lambda_1}{\lambda_2} e^{p(-t_i/d)} \right ) dt_i
\]

\[
= -\frac{d}{\Delta t_i} \int_{u_{\text{min}}}^{u_{\text{max}}} \frac{\log \left ( 1 - \frac{\lambda_1}{\lambda_2} u \right )}{u} du
\]

\[
\approx \frac{d}{\Delta t_i} \frac{\lambda_1}{\lambda_2} \left ( e^{p(-t_i/d)} + \left ( \frac{\lambda_1}{\lambda_2} \right )^2 \frac{e^{p(-2t_i/d)}}{4} + \left ( \frac{\lambda_1}{\lambda_2} \right )^3 \frac{e^{p(-3t_i/d)}}{9} + \cdots \right ) \bigg |_{t_{\text{min}}}^{t_{\text{max}}}
\]

Substituting this final value of \( F_{av} \) in (3.25), and then the resulting \( \Theta \) in (3.26) the load delay is obtained. Hence, the total rise delay can be written as:

(3.27) \[ t_{d,\text{rise}} = t_i + t_i \]

\[ = t_i + d \left ( \Theta + \log \left ( 1 + \frac{\lambda_1}{\lambda_2} e^{p(-\Theta)} \right ) \right ) \]

where \( t_i \) is replaced by the expression (3.1).

This analytical expression for the total pull-up rise delay depends upon the device physical parameters as well as the electrical parameters of the BiCMOS gate. Therefore, the expression is very useful for the physical design of the BiCMOS cell. We will use it in the following chapter to determine the optimum device geometry for minimum delay.

### 3.6 Average Current Gain Model

The analytical delay expression derived in the previous section gives very accurate results but it is far too complicated to be used for hand calculation circuit analysis. Therefore, we derive here an easier to use, yet accurate delay expression. This model is the second analytical delay model (average current gain) we implement in this work.
In this case we use both average values for the bipolar transistor current gain (expression 2.7) and the base transit time (relation 2.4) to account for the high level injection effects. Consequently, the relation between the base and collector currents of the bipolar transistor can be simple written as:

$$I_B = \frac{I_C}{\beta_{F,av}}$$

Similar to the previous model, Kirchhoff's current equation at the base node (Figure 3.4) is

$$I_{MOS} = C_F \frac{dV_B}{dt} + \frac{I_C}{\beta_{F,av}} + C_D \frac{dV_{BE}}{dt} + C_{BC} \frac{dV_{BC}}{dt}$$  \hspace{1cm} (3.28)

with

$$I_E = I_C + C_D \frac{dV_{BE}}{dt} + \frac{I_C}{\beta_{F,av}}$$

$$\frac{dV_B}{dt} \approx \frac{dV_O}{dt} = \frac{I_E}{C_O}$$ \hspace{0.5cm} \text{with} \hspace{0.5cm} C_O = C_{CS} + C_{LOAD}

$$C_D \frac{dV_{BE}}{dt} \approx \tau_{F,av} \frac{dI_C}{dt}$$

$$V_{BC} = V_{BE} + R_E I_C + V_O - (V_{DD} - R_C I_C)$$

Notice that in this case the emitter current no longer equals the collector current. Substituting the above relations in (3.28), the current equation becomes

$$\frac{1}{\beta_{F}} I_C - I_{MOS} = -\tau_{F}^{*} \frac{dI_C}{dt}$$  \hspace{1cm} (3.30)

where

$$\frac{1}{\beta_{F}} = \frac{1}{\beta_{F,av}} + \frac{C_{BC} + C_P}{C_O} + \frac{C_{BC} + C_P}{\beta_{F,av} C_O}$$

$$\tau_{F}^{*} = \tau_{F,av} + C_{BC}(R_C + R_E) + \tau_{F,av} \frac{C_{BC} + C_P}{C_O}$$  \hspace{1cm} (3.31)

Furthermore, it can be rewritten as

$$\beta_{F}^{*} \tau_{F}^{*} \int \frac{1}{I_C - \beta_{F}^{*} I_{MOS}} dI_C = - \int dt$$  \hspace{1cm} (3.32)

with the initial condition

$$\begin{cases} 
 i = 0 \\
 I_C(0) = 0 
\end{cases}$$

and for physical reasons since the time variable is a positive quantity:

$$0 \leq I_C \leq \beta_{F}^{*} I_{MOS} \hspace{1cm} \text{with} \hspace{1cm} I_{C,\text{max}} = \beta_{F}^{*} I_{MOS}$$
The solution of the differential equation (3.32) is

\[(3.33) \quad I_o(t) = \beta_F^* I_{MOS} \left( 1 - e^{\frac{-t}{\beta_F^* \tau_F^*}} \right) \]

The expression for the output voltage \( V_o \) differs also from the previous model. It is given by

\[(3.34) \quad V_o(t) = \frac{1}{C_o} \int I_e(t) \, dt \]

\[= \frac{1}{C_o} \int \left( (1 + \frac{1}{\beta_{F,av}}) I_C + \tau_{F,av} \frac{dI_c}{dt} \right) \, dt \]

with the initial condition = \( \begin{cases} t = 0 \\ V_O(0) = V_{low} \end{cases} \)

The solution of the output voltage differential equation is

\[(3.35) \quad V_o(t) - V_{low} = (g - d) \lambda + d \frac{\lambda^2}{2} \]

where

\[(3.36) \quad \lambda = \frac{t}{\beta_F^* \tau_F^*} \]

\[g = \frac{\beta_F^* \tau_F^* I_{MOS}}{C_o} \left( 1 + \frac{1}{\beta_{F,av}} \right) \]

\[d = \frac{\beta_F^* I_{MOS}}{C_o} \left( \beta_F^* \tau_F^* + \frac{\beta_F^* - \beta_{F,av}}{\beta_{F,av}} \right) \]

The only term dependent on the time is \( \lambda \). This one can be determined easily by solving the second order equation (3.35) and retaining its positive value as follows:

\[(3.37) \quad \lambda = (1 - g/d) + \sqrt{(1 - g/d)^2 + \frac{2(V_o(t) - V_{low})}{d}} \]

Finally, the load delay \( t_l \) is calculated at

the switch point = \( \begin{cases} V_O(t) = V_{switch} \\ \Delta V_O = V_{switch} - V_{low} \end{cases} \)

The result for the load delay obtained in our second delay analysis is

\[(3.38) \quad t_l = \beta_F^* \tau_F^* \left( (1 - g/d) + \sqrt{(1 - g/d)^2 + \frac{2\Delta V_O}{d}} \right) \]
and the total rise delay is the sum of the junction delay and the load delay given by:

\[ t_{d,\text{rise}} = t_i + t_l \]

\[ t_i + \beta^* \tau_E \left( (1 - g/d) + \sqrt{(1 - g/d)^2 + \frac{2\Delta V_G}{d}} \right) \]

where \( t_i \) is replaced by the expression (3.1).

This last expression is very easy to use and helps us understand the device physical parameters influence on the circuit transient behavior. However, it is not suitable for optimum buffer design purpose since this solution is less accurate than the \textbf{variable current gain} delay model derived here.

### 3.7 Evaluations

To check the validity of our results, a comparison is made between SPICE simulation results and our new BiCMOS delay expressions (3.26) and (3.38) using the electrical parameters discussed in appendix 2.a and 2.b. We implemented our delay models as a set of expressions in the MATLAB environment [35]. These results are shown in Figure 3.6. Other two models [4, 15] reported in the literature are also implemented to compare their results with ours.

A quick analysis of the curves in Figure 3.6 shows that our new expressions for the BiCMOS delay are more accurate and in excellent agreement with SPICE simulations. Note that the discrepancies between the complex model (3.26) predictions and SPICE simulation results are within 5%, while the simplified delay model (3.38) still gives a good accuracy within 10% from SPICE.

The results presented in Figure 3.6 demonstrate the validities of the simplifying assumptions we made to obtain a BiCMOS analytical delay model in which short channel and high level injection effects are incorporated.

In the delay model proposed by Fang \textit{et al.} [4] (Figure 3.6) the PMOS transistor is modeled by an equivalent channel resistance while for bipolar transistor equivalent circuit, base and collector resistances and some equivalent capacitances are retained. Their work shows that the base series resistance of the bipolar transistor is not important when it is much smaller than the channel resistance of the PMOS transistor. This conclusion justifies our decision of neglecting the base resistance in the circuit delay analysis. However, they did not consider the high level injection effects on base transit time and transistor current gain.

The delay model proposed by Embabi \textit{et al.} [15] is very similar to our first approach. Their simplified assumptions, however, are valid only for very high load
Figure 3.6. Pull-up BiCMOS delay as a function of output load capacitance in case of SPICE simulation and different analytical models.

capacitive area. Therefore, their results have good accuracy in this area but they are less in agreement with SPICE simulations for the rest.
Chapter 4

GATE CHARACTERIZATION FOR CAD

In standard-cell based design the performance of the design systems rely on the cell characterization models used to implement the tools. The accuracy of logic and timing simulators, as well as the accuracy of timing optimization tools depends on the accuracy of their delay models for cell delay. If the delay models are not accurate, the timing problems of the design cannot be detected during the design phase but only later on, after the design or even the fabrication process are completed, when it is very expensive.

Using SPICE simulator for this purpose can be very expensive in terms of CPU time. Also the delay models presented in previous chapter are too cumbersome to be used for this propose. A simple yet accurate delay model is required for CAD applications. The input signal slew rate should also be incorporated in the delay model to improve the design characterization performance.

This chapter presents a CAD delay model which also accounts for input signal slew rate. A power dissipation analysis versus input slope is made to complete the cell characterization step.

4.1 CAD Delay Model

To incorporate both CMOS and BiCMOS gate delays in our model, we derive an efficient parameterized nonlinear delay model. The model is based on the following parameters: input capacitance per input pin, intrinsic delay, and extrinsic delay (delay per unit load). It is easy to use and is a good compromise between accuracy and productivity.

In the expressions (3.26) and (3.39) derived in chapter 3 the gate propagation delays are nonlinear functions in output load capacitance $C_{LOAD}$. Therefore, it is very difficult, if not impossible, to make any of these two delay expressions an explicit function of the load capacitance. For fast CAD tools, however, the delay expression must depend explicitly on the output load capacitance.

It has been shown in [29] that for an optimum designed driver, the gate propagation delay is proportional with the square root of the load capacitance, while in the first
order delay analysis presented in [50] the minimum propagation delay is proportional with \( C_{LOAD}^{2/3} \). These earlier works give two possible lower bounds for the gate propagation delay. We choose the smallest limit for our CAD analysis. For the upper limit we use the experimental results presented in Figure 3.6. The SPICE simulation results show that the relation between the propagation delay and the load capacitance is at most linear. Consequently, this is the upper limit we choose for the gate delay.

The CAD delay model we propose here is iteratively approximated from the analytical delay model (3.26). The delay variation with the load capacitance is considered to be between the lower and upper limits previously discussed. A delay model nonlinear in its output load capacitance is obtained from the following approximation:

\[
(4.1) \quad t_{d,\text{rise}} = t_i + d \left( \Theta + \log \left( 1 + \frac{\lambda_1}{\lambda_2} \exp(-\Theta) \right) \right) \approx A + BC_{LOAD}^\alpha
\]

where the gate dependence delay on load capacitance \( \alpha \in [\alpha_{\text{min}}, \alpha_{\text{max}}] \) is in \([0.5, 1]\) corresponding to the lower limit suggested by [29] for an optimum designed driver and the upper limit suggested by the experimental results; \( A \) is the intrinsic delay of the gate, and \( B \) is the drive capability of the gate.

Furthermore, the rise delay is a function of the output load capacitance \( C_{LOAD} \) via the variables \( \Theta, d, \lambda_1, \lambda_2 \). Consequently, we can write

\[
(4.2) \quad t_{d,\text{rise}}(C_i) = t_{di} = t_i + d_i \left( \Theta_i + \log \left( 1 + \frac{\lambda_{1i}}{\lambda_{2i}} \exp(-\Theta_i) \right) \right)
\]

where \( C_i \) is an instantiation value of output load capacitance \( C_{LOAD} \). If we consider the variation of \( C_{LOAD} \) within a limited range \([C_{\text{min}}, C_{\text{max}}]\) and sample this interval in \( n \) distinct values, then \( \Theta_i, d_i, \lambda_{1i}, \lambda_{2i} \) are functions of instantaneously values of the output load capacitance \( C_i \), where \( i \) takes integer values from 1 to \( n \) (\( i = 1, n \)).

Similar, for \( \alpha \in [\alpha_{\text{min}}, \alpha_{\text{max}}] \), we can sample this interval in \( k \) distinct values and for each value associate an \( \alpha_j, j = 1, k \). For each \( \alpha_j \) and \( C_i \) set (\( i = 1, n \)), \( C_i \) to the power of \( \alpha_j \) can be treated as an entity. Consequently, the expression (4.1) becomes a linear equation in variable \( C_i^{\alpha_j} \). Note that for each \( \alpha_j \) corresponds a set
of parameters \((A_j, B_j)\). They can be calculated as follows:

\[
A_j = \frac{\left(\sum_{i=1}^{n} C_i^{\alpha_j}\right) \left(\sum_{i=1}^{n} C_i^{\alpha_j} t_{di}\right) - \left(\sum_{i=1}^{n} C_i^{2\alpha_j}\right) \left(\sum_{i=1}^{n} t_{di}\right)}{\left(\sum_{i=1}^{n} C_i^{\alpha_j}\right)^2 - n \left(\sum_{i=1}^{n} C_i^{2\alpha_j}\right)}
\]

\[
B_j = \frac{\left(\sum_{i=1}^{n} C_i^{\alpha_j}\right) \left(\sum_{i=1}^{n} t_{di}\right) - n \left(\sum_{i=1}^{n} C_i^{\alpha_j} t_{di}\right)}{\left(\sum_{i=1}^{n} C_i^{\alpha_j}\right)^2 - n \left(\sum_{i=1}^{n} C_i^{2\alpha_j}\right)}
\]

The average square error between the results obtained with the analytical model and the proposed nonlinear model is given by:

\[
\text{error}_j = \frac{\sum_{i=1}^{n} \left(t_{di}^2 - \left(A_j + B_j C_i^{\alpha_j}\right)^2\right)}{n}
\]

Finally, the parameters which give the minimum error are chosen for the BiCMOS CAD delay model:

\[
(A, B, \alpha) = (A_j, B_j, \alpha_j) = \arg \min_{1 \leq j \leq k} \frac{\sum_{i=1}^{n} \left(t_{di}^2 - \left(A_j + B_j C_i^{\alpha_j}\right)^2\right)}{n}
\]

The iterative algorithm used to determine the nonlinear delay parameters is presented in Figure 4.1. The number of samples \(n\) and \(k\), taken for the load capacitance and \(\alpha\) respectively, will determine the accuracy of the CAD delay model.

The gate delay versus load capacitance, calculated analytically, simulated with SPICE, and derived with the CAD model, is plotted in Figure 4.2. The agreement is quite good, showing that (4.1) accurately represents the gate delay. The deviation of the analytical delay model from SPICE is within 5%.

The main advantage of deriving the CAD delay model from the analytical delay model is the possibility of building a fully automated ASIC system. By using the analytical delay model, smart cells can be generated to accommodate a certain output load capacitance. Besides, with minimum effort its simple CAD delay model is derived for later optimization and simulation steps. How this model is used in a design timing optimization process it is shown in the following chapters.
\begin{algorithm}
\textbf{CAD model}

\textbf{input}: output capacitance samples $C_i$, $i = 1, n$
analytical delay model $t_{d,\text{rise}}(C_i)$

\textbf{for} $i = 1$ to $n$
\{\textbf{calculate} $\Theta_i$, $d_i$, $\lambda_{1i,2i}$ and $t_{di}$\}

\textbf{foreach} $\alpha_i \in [\alpha_{\text{min}}, \alpha_{\text{max}}]$ with $\alpha_{\text{step}} = \frac{\alpha_{\text{max}} - \alpha_{\text{min}}}{k}$ \{\textbf{do} \}

\begin{equation}
A_j = \left( \sum_{i=1}^{n} C_i^\alpha \right) \left( \sum_{i=1}^{n} C_i^{2\alpha} t_{di} \right) - \left( \sum_{i=1}^{n} C_i^{2\alpha} \right) \left( \sum_{i=1}^{n} t_{di} \right)
\end{equation}

\begin{equation}
B_j = \left( \sum_{i=1}^{n} C_i^{\alpha} \right) \left( \sum_{i=1}^{n} t_{di} \right) - \left( \sum_{i=1}^{n} C_i^{2\alpha} \right) \left( \sum_{i=1}^{n} C_i^{\alpha} t_{di} \right)
\end{equation}

\begin{equation}
error_j = \frac{\sum_{i=1}^{n} \left( t_{di}^2 - (A_j + B_j C_i^{2\alpha}) \right)}{n}
\end{equation}

\}

$p = \arg\min_{1 \leq j \leq k} error_j$
$(A, B, \alpha) = (\bar{A}_p, \bar{B}_p, \alpha_p)$

\textbf{end CAD model}
\end{algorithm}

Figure 4.1. CAD delay model algorithm.

4.2 Delay Dependence on Input Slope

As we mentioned before an accurate gate delay model must account for the slope of the input waveform. A couple of models [48, 49, 36] for CMOS gates have been proposed, which have the capability of acceptable accurate model the switching behavior of the gate. However, we do not know about any similar analytical delay model for the BiCMOS gates. A two-step iterative approach has been proposed in [16] but it needs more than 30 parameters to model the gate delay and considerable computational effort. All the other analytical delay models we found [21, 46, 13, 4, 15, 37, 45], are not dependent on input signal slew rate.

An exact analysis of the BiCMOS gate delay with time-dependent input, yields nonlinear differential equations that are very difficult to solve [16]. Therefore, we decide to limit the exact circuit analysis to a gate delay model independent on input slope, and to extract this dependency from the actual behavior of the gate.

In this section, we present a simple BiCMOS gate delay model which depends on
4.2 Delay Dependence on Input Slope

![Graph showing delay dependence on input slope](image)

**Figure 4.2.** Pull-up buffer delay as a function of output load capacitance determined with the CAD delay model.

the input signal slew rate. In addition to the propagation delay, the rise and fall delays of the gate will be derived. This is important because the output ramp of a cell serves as the input ramp to the subsequent cells in the circuit. Our model is similar to the CMOS gate delay model proposed in [36], which is reported to be very accurate even when the technology goes down to 0.35-μm.

Two cases can be distinguished for gate propagation delay versus input signal slew rate: the pull-up delay complemented by output waveform rise time and the pull-down delay complemented by output waveform fall time. We choose to illustrate our delay model for a pull-up transient and output rise time. Yet, the results apply for the pull-down transient and output fall time as well. The test gate is a BiCMOS totem-pole inverter but similar results are obtained for a variety of other BiCMOS cells. In this section, $t_d$ will stand for gate delay, either rise or fall.

The leakage current through the bleeding devices is ignored in the following analysis. As pointed out, our model is based on the actual behavior of the gate. That means, for a fixed output load capacitance and $a$, so called, fast input signal slew rate the current provided by the BJT in the pull-up stage of the inverter is entirely
used for charging the load capacitance. The NMOS and BJT transistors in the pull-down stage of the gate are totally off.

For a fixed load capacitance and a slow input signal slew rate, both PMOS and NMOS transistors are conducting supplying currents to both bipolar transistors. If the input slope exceeds a certain limit the capacitive network of the BJT in the pull-down stage will be charged forcing the transistor into conduction. Consequently, the effective output current will be limited.

The input signal slew rate separating these two regions is called the critical input slope (CIS). In practice, this is translated as an output ramp region independent on the input ramp, denoted as “fast ramp” region, and a linear dependency of the output ramp on the input ramp, denoted as “slow ramp” region.

![Graph](image)

**Figure 4.3.** Output waveforms and their linear approximation for different input slopes, a low to high output transition and 2.0 pF load capacitance.

We linearize the output waveform by a line passing through $V_{DD}/2$ at $t_d$ with a slope equal with that of the line between the two points 10%$V_{DD}$ and 90%$V_{DD}$, or $V_{low}$ and $V_{high}$ if these last ones are within the previous limits (Figure 4.3).

In the fast ramp region, the linearized output waveforms are almost parallel (see the output response for input slope $IS=0.1$ ns, 0.3 ns, and 0.5 ns in Figure 4.3). This is translated as a constant output ramp function of input ramp. This simplified assumption is based on our SPICE simulation results. In the slow ramp re-
4.2 Delay Dependence on Input Slope

region, however, the output slope increases linear with the input slope (see the output waveforms for $IS = 0.7$ ns and $0.9$ ns in the figure).

Figure 4.4 shows a simplified low to high transition of a gate. The input signal waveform changes from the supply voltage $V_{DD}$ to the ground $V_{SS}$ with an input slope $IS$, while the output rises from $V_{low}$ to $V_{high}$. With the assumption of a linear output waveform, two quantities can be defined to characterize the gate response. These are denoted by $\Delta t_i$ for the time interval between the moment when the input starts falling and the output starts rising, and $\Delta t_h$ for the time interval between the moment when the input starts falling and the output reaches $V_{high}$ level. In short they are denoted as output slope parameters.

![Figure 4.4. Input/Output waveforms of a BiCMOS driver and their relations.](image)

All the delays, including the propagation delay as well as the rise and fall delays, can be expressed as functions of the output slope parameters. Thus, the propagation delay is given by

\[ t_d = \frac{\Delta t_i + \Delta t_h - IS}{2} \]

(4.7)

while, the output ramp is

\[ t_{rise} = \Delta t_h - \Delta t_i \]

(4.8)

for a low to high transition.

To accurately determine $\Delta t_i$ and $\Delta t_h$, we consider that in the fast ramp region the output slope for a step input and the output slope for a ramp input are practically the same. They are only shifted in time as a linear function of the input signal slope $IS$. Consequently, the step input response is used to determine the output slope in the fast ramp region.
In the previous sections it has been shown that the gate delay for a step input signal can be expressed as

\[ t_d = t_i + t_l = \tau + \beta \cdot C_{load}^\alpha \]

where \( t_i \) is the junction delay and \( t_l \) the load delay defined in section 3.3; \( \tau, \beta, \) and \( \alpha \) are the CAD delay parameters denoted intrinsic delay, drive capability, and gate dependence delay on load. With these parameters we can determine the output slope for a step input as follows:

\[ \Delta t_{i, IS=0} = t_i \]
\[ \Delta t_{h, IS=0} = t_i + 2t_l = 2(\tau + \beta \cdot C_{load}^\alpha) - t_i \]

which results in

\[ t_{rise, IS=0} = 2t_l = 2(\tau + \beta \cdot C_{load}^\alpha - t_i) \]

Between the step input and the critical input ramp \( CIS \) the variation of \( \Delta t_i \) and \( \Delta t_h \) with \( IS \) is the same. After the input signal critical slope is reached, the output slope will increase linearly with the input slope. To model accurately this behavior the output slope parameters can be defined as follows:

\[ \Delta t_i = \tau_i + \delta \cdot \min(IS, CIS) + \psi_i \cdot \max(0, IS - CIS) \]
\[ \Delta t_h = \tau_h + 2\beta \cdot C_{load}^\alpha + \delta \cdot \min(IS, CIS) + \psi_h \cdot \max(0, IS - CIS) \]

with

\[ \tau_i = t_i \]
\[ \tau_h = 2\tau - t_i \]

The linear coefficient in the fast ramp region \( \delta \), as well as the slow ramp region parameters \( \psi_i \) and \( \psi_h \) can be determined by characterization.

**Definition 4 (Critical Input Slope)** The critical input slope \( CIS \) is the value of the input slew rate at which the output transition reaches \( V_{inv} \) precisely at the end of the input transition (see Figure 4.5).

In other words, at the critical input slope, the output is significantly changed. For ramps between the step input and \( CIS \) the gate operates in the fast ramp region. After the input waveform reaches the \( CIS \) value the gate enters the slow ramp region.

From Figure 4.5, by taking

\[ x = \Delta t_{h, CIS} - CIS \]
Figure 4.5. Critical input slope $CIS$ definition for a BiCMOS gate.

we can determine

\[ x = \frac{V_{high} - V_{inv}}{V_{high} - V_{low}} (\Delta t_{h,CIS} - \Delta t_{l,CIS}) \]  

with

\[ \Delta t_{l,CIS} = \tau_l + \delta \cdot CIS \]  
\[ \Delta t_{h,CIS} = \tau_h + 2 \beta \cdot C_{load}^\alpha + \delta \cdot CIS \]

Substituting (4.16) and (4.15) in (4.14) the expression for the critical input slope results in

\[ CIS = \frac{(V_{inv} - V_{low})(\tau_h + 2 \beta \cdot C_{load}^\alpha) + (V_{high} - V_{inv}) \cdot \tau_l}{(V_{high} - V_{low})(1 - \delta)} \]

From the static analysis of the gate in section 3.2 we saw that when the input falls very slowly, the output gradually will start rising when the input reaches $V_i$ value, will switch at $V_{inv}$, and gradually will reach $V_{high}$ when the input equals $V_h$. We determine the parameters $\psi_l$ and $\psi_h$ from the limit condition when the output follows the static transfer curve. This limit condition is illustrated in Figure 4.6 with continuous lines. With dashed lines the gate response at the critical input slope is also sketched.

From the figure, using geometrical equivalences we can write

\[ \Delta IS_{limit} = IS_{limit} - CIS \]  
\[ \psi_l = \frac{x}{\Delta IS_{limit}} \]  
\[ \psi_h = \frac{y}{\Delta IS_{limit}} \]
using the approximations

\[
\frac{\Delta t_i, CIS + x}{CIS + \Delta IS_{\text{limit}}} \approx \frac{x}{\Delta IS_{\text{limit}}} = \frac{V_{DD} - V_i}{V_{DD} - V_{SS}}
\]

\[
\frac{\Delta t_h, CIS + y}{CIS + \Delta IS_{\text{limit}}} \approx \frac{y}{\Delta IS_{\text{limit}}} = \frac{V_{DD} - V_h}{V_{DD} - V_{SS}}
\]

the desired parameters can be written

\[
\psi_i = \frac{V_{DD} - V_i}{V_{DD} - V_{SS}}
\]

\[
\psi_h = \frac{V_{DD} - V_h}{V_{DD} - V_{SS}}
\]

The final expressions for the output slope parameters are obtained by substituting the relations (4.20) in (4.12). The result is as follows:

\[
\Delta t_i = \tau_i + \delta \cdot \min(IS, CIS) + \frac{V_{DD} - V_i}{V_{DD} - V_{SS}} \max(0, IS - CIS)
\]

\[
\Delta t_h = \tau_h + 2 \beta \cdot C_{\text{load}}^0 + \delta \cdot \min(IS, CIS) + \frac{V_{DD} - V_h}{V_{DD} - V_{SS}} \max(0, IS - CIS)
\]

Finally, the gate delay expression as a function of both, load capacitance and input signal slew rate, is derived using the relation (4.7) in which the output slope
4.2 Delay Dependence on Input Slope

parameters are substituted with their final version (4.21). It is given by

\[
(4.22) \quad t_d(C_{load}, IS) = \frac{1}{2} \left( 2 \tau + 2 \beta \cdot C_{load}^\alpha \right. \\
+ 2 \delta \cdot \min(IS, CIS) + \frac{2V_{DD} - (V_i + V_h)}{V_{DD} - V_{SS}} \max(0, IS - CIS) - IS \bigg) 
\]

The gate delay depends upon the load capacitance and the input signal slope but these two components are independent one of the other. Consequently, the delay can be written as

\[
(4.23) \quad t_d(C_{load}, IS) = t_d(C_{load}) + t_d(IS)
\]

where

\[
(4.24) \quad t_d(C_{load}) = \tau + \beta \cdot C_{load}^\alpha \\
t_d(IS) = \delta \cdot \min(IS, CIS) + \frac{2V_{DD} - (V_i + V_h)}{2(V_{DD} - V_{SS})} \max(0, IS - CIS) - \frac{IS}{2}
\]

The expression for the output signal slope is now derived by substituting the relations (4.21) into (4.8). The result is

\[
(4.25) \quad t_{rise} = 2(\tau + \beta \cdot C_{load}^\alpha - t_i) + \frac{V_i - V_h}{V_{DD} - V_{SS}} \max(0, IS - CIS)
\]

which with the relation (4.11) becomes equivalent to

\[
(4.26) \quad t_{rise} = t_{rise, IS=0} + \frac{V_i - V_h}{V_{DD} - V_{SS}} \max(0, IS - CIS)
\]

By taking into account both, fast and slow, input ramp regions our model gives a very realistic characterization of the gate transition. Besides, the simple linear model "derived" here makes it very practical and easy to use. The model proposed in [16] only consider the fast ramp region for delay analysis. This is a major limitation, especially when the model is used for simulation or optimization purposes.

We implemented our model in a MATLAB environment [35]. The accuracy of the delay expression (4.24) is verified by comparing it with SPICE simulation results. In Figure 4.7 the pull-up BiCMOS gate delay versus input signal slope for 6 different output loads is plotted. Both the proposed model and the SPICE results are shown together with calculated critical input slope value CIS for each output load capacitance. The results presented in the figure are taken between the moment when the input waveform starts falling and the output waveform is half way to V_{DD}. This choice is justified by the fact that when the propagation delay time definition 1 is used, the delay of the gate may become negative if the input signal slew
Figure 4.7. Rise time delay as a function of input slope for different output load capacitance and its linear approximation; the propagation delay is taken between the moment when the input starts falling and the output is 50% supply voltage.

rate is too slow. We thought this choice will give a better picture of the gate delay versus input slope.

The delay model presented has many practical applications. For example, it can be used for timing simulation or optimization purposes. The delay and the output waveforms of each stage in the circuit can be accurately determined. The use of this delay model in timing optimization of the combinational circuits will be presented in chapter 6.

4.3 Gate Power Dissipation versus Input Slope

Power consumption is rapidly becoming a very important issue in ASIC design. A consistent effort is made to minimize the power dissipation at all the design levels starting with the behavioral level, going through architectural level, logic level and ending with the physical level and packaging. Existing ASIC design tools map high level requirements to library cells early in the design cycle. To do any kind of
4.3 Gate Power Dissipation versus Input Slope

design or optimization for low power, a power model for the cells must be available. In this section we will analyze the library cell power dissipation.

There are four major sources of power dissipation in digital BiCMOS circuits which are summarized in the following equation:

\[
P_{\text{avg}} = P_{\text{switching}} + P_{\text{glitch}} + P_{\text{short-circuit}} + P_{\text{leakage}} = \alpha_{0\rightarrow1}C_{\text{load}} \cdot V_{\text{swing}}^2 \cdot f_{\text{clk}} + P_{\text{glitch}} + I_{\text{sc}} \cdot V_{\text{DD}} + I_{\text{leakage}} \cdot V_{\text{DD}}
\]

where the first three terms represent the dynamic power dissipation of the gate, and the last term is the static power dissipation due to the circuit leakage current \(I_{\text{leakage}}\). The first term of the dynamic power dissipation is the switching power, effectively used in charging and discharging the output load capacitance \(C_{\text{load}}\); \(f_{\text{clk}}\) is the clock frequency, \(V_{\text{swing}}\) the voltage swing of the gate and \(\alpha_{0\rightarrow1}\) is the node transition activity. The second term of the dynamic power dissipation \(P_{\text{glitch}}\) is the power consumed in false transitions, either partial or complete swing. Finally, the third term is due to the short-circuit current \(I_{\text{sc}}\) in the direct path from \(V_{\text{DD}}\) to the ground.

To minimize the power dissipation of a circuit, each of the power components stated above have to be minimized. The switching power is dependent on the logic design, while the leakage component of power is primarily determined by fabrication technology. The only component of power which can be truly characterized at the gate level is the short-circuit power. Finite rise and fall times at the input of logic gates result in short currents which will exist for a certain period of time during switching.

For a closer analysis of the short-circuit power dissipation in BiCMOS cells we refer to the SPICE simulation results presented in Figure 4.8. The transient delay analysis is performed for a BiCMOS gate with an output load capacitance of 3.0 pF. The input signal slope is varied between 0.0 ns and 3.5 ns. In Figure 4.8a the response of the gate is given. The output waveform linear approximation obtained with our model (relation 4.25) is also shown with dashed lines. We also show the corresponding average charging and short-circuit currents in Figure 4.8b. These two currents have a very important role in power characterization of the gate.

For a step input there is no short-circuit current and therefore the only power dissipated by the gate is in charging the output load capacitance. As the input slope increases the short-circuit current continues to stay at a constant low level until the input reaches its critical slope value \(C_{\text{IS}}\) (1.5 ns in this particular case). After this point, the short-circuit current will gradually increase with the input signal slope and so does the short-circuit power consumption. First, the short-current power dissipation is less than the switching power but after a while it will become larger. We call the input signal slew rate separating these two regions unsafe input slope \(U_{\text{IS}}\), and the region starting at this point unsafe region. In the unsafe region the
Figure 4.8. BiCMOS driver response for a variable input slope ($IS$); (a) output voltage waveforms produced by SPICE (solid lines) and by linear approximation (dashed lines); (b) charge current waveforms (solid lines) and short current waveforms (dashed lines) during the output transition from low to high.

A better insight in the physical behavior of the circuit is given in Figure 4.9. The internal currents of the BiCMOS inverter are simulated with SPICE for 0.1 ns and 6.0 ns input slopes, and an output load capacitance of 2.0 pF. When the input slope equals 0.1 ns (below $CIS$) the current supplied by the bipolar transistor $Q_1$, $IE(Q_1)$, is almost entirely used in charging the load capacitance ($I(C_{load})$). For a very short time a short-circuit current $ID(M_2)$ will flow via the NMOS transistor in the pull-down stage. The bipolar transistor in the pull-down stage $Q_2$ is off.
Figure 4.9. BiCMOS inverter currents for a low to high output voltage transition: (a) \( IS = 0.1 \) ns; (b) \( IS = 6.0 \) ns.

When the input slope becomes 6.0 ns the gate is in deep unsafe region. The current supplied by the NMOS transistor in the pull-down stage will charge the capacitive network of the bipolar transistor \( Q_2 \) which will start conducting (Figure 4.9b). Consequently, the effective charging current \( I(C_{\text{load}}) \) will be dramatically decreased. To avoid these undesired effects, the input slope has to be kept between certain limits. This is why we defined and "derived" the safe and unsafe operating regions for a BiCMOS gate. Notice that this problem is more serious for BiCMOS than for CMOS circuits because of the large bipolar current gain. Once, the bipolar transistor in the pull-down stage starts conducting a large short-circuit current will be generated.

**Definition 5 (Unsafe Input Slope)** The unsafe input slope \( UIS \) is the value of the input slew rate at which the output transition is completed precisely at the same time with the completion of the input transition (see Figure 4.10).
The unsafe input slope is the border line between the power safe operating region of the gate and power unsafe operating region.

![Figure 4.10. Unsafe region definition for a BiCMOS driver.](image)

From Figure 4.10 it is easy to see that

\[ UIS = \Delta t_{h,UIS} \]

Furthermore, \( \Delta t_{h,UIS} \) is given by the relation 4.21 for \( I_{S}=UIS \) as follows:

\[ (4.28) \quad \Delta t_{h,UIS} = \tau_h + 2\beta C_{load}^\alpha + \delta \cdot CIS + \frac{V_{DD} - V_h}{V_{DD} - V_{SS}} \cdot (UIS - CIS) \]

By substituting it in (4.28), the unsafe input slope can be calculated with the following formula:

\[ (4.29) \quad UIS = \frac{\tau_h + 2\beta \cdot C_{load}^\alpha + \left( \delta - \frac{V_{DD} - V_h}{V_{DD} - V_{SS}} \right) \cdot CIS}{\frac{V_h - V_{SS}}{V_{DD} - V_{SS}}} \]

In Figure 4.11 the average charge and short-circuit currents versus input signal slew rates are shown for different output load capacitances spaced between 0.1 pF and 10.0 pF. These simulation results are for a low to high output voltage transition of a BiCMOS inverter. The limits of the unsafe operating region calculated with the relation (4.29) are also shown. The \( UIS \) limits increase monotonically with the load capacitances, which was expected since a larger output capacitance will enforce a larger charge current.

The conclusion derived from this gate power analysis is that the gate power dissipation is small and constant when the input slew rate of the gate is between zero and a critical value \( CIS \). This is a very safe operating region for the gate. After the \( CIS \) is reached the average short-circuit current increases almost linear with the input slope until it exceeds the average charge current at \( UIS \). From now on the gate enters the unsafe operating region and the short-circuit power consumption becomes very expensive.
Figure 4.11. Average BiCMOS driver currents for a low to high output transition as a function of input slope: (a) average charge current; (b) average short current.

4.4 Conclusion

For standard-cell design, the design and characterization of library primitives is essential in order to design good quality circuits. The quality of optimization and verification results rely on the quality of the delay models used in implementing this tools. The objective of this chapter was to accurate characterize the essential library cells parameters.

A CAD timing model was derived which gives accurate delay evolutions and is not computationally expensive. The short and narrow channel effects on the MOS tran-
sistor current equations are incorporated into the model, as well as the high injection effects upon the current gain and the base transport factor of the bipolar transistor.

A simple approach was proposed to characterize the input signal slew rate effects upon the gate delay and power dissipation.
BUFFER OPTIMIZATION

To optimize the switching speed of BiCMOS gates, designers scale the size of MOS and bipolar transistors. In the previous chapter we showed that the propagation time of a BiCMOS gate depends on the size of PMOS and NMOS transistors in the pull-up and pull-down stages, and the emitter size of the bipolar transistors. The latter controls the high injection characteristics. Moreover, the dimensions of both MOS and bipolar transistors determine the parasitic capacitances which affect the propagation delay time. Thus a proper sizing of MOS transistors as well as bipolar transistors can increase the circuit speed significantly.

Finding this optimal scaling using conventional circuit simulation techniques (that is running SPICE) requires a large number of runs and therefore a timing consuming process. An analytical delay model, however, may provide a design optimization strategy which can be both time efficient and flexible.

Research efforts have been reported on BiCMOS circuit optimization, e.g. [50, 13, 44, 4, 29]. None of these earlier works are easy to implement in a fast and fully automated cell generator. Some of the existent analytical delay expressions do not provide the required accuracy. The numerical models generally require considerable computational effort. Other models are based on look up tables which must be filled with pre-simulated results.

In this chapter a comprehensive strategy for BiCMOS buffer optimum design is presented. The method takes advantage of the accurate analytical delay model we derived in chapter 3. We extend the optimization process from a single specific load capacitance to a continuous range of capacitances. Since the delay model does not require any SPICE simulations, the gate optimization process is quick and efficient.

5.1 Device Parameters

To build a fully automatic buffer design system the device parameters required by the optimization process have to be available. This means that the SPICE device parameters presented in chapter 2 (tables 2.1 and 2.2) are required for both CMOS
and bipolar transistors. As we pointed out, the key circuit parameters which define the BiCMOS gate are the width of the MOS transistors and the emitter length of the bipolar transistors. The length of the MOS transistor is usually kept at the minimum value in integrated circuit design. Its value is determined by the fabrication technology. Meanwhile, the base and collector length of the bipolar transistor will scale with the emitter length.

For a MOS transistor the SPICE device parameters do not change with the transistor size. They only depend on the technology. The effect of the transistor width or length is taken into account later on in the device equations.

This is not the case for the bipolar transistor parameters. Changing the emitter size will directly influence the device parameters. Besides, chip manufacturers generally provide a number of different layout designs for the same bipolar transistor size. These transistors can vary in base and collector resistances, number of base contacts, number of emitters etc. Obviously, the device parameters for each will differ. We do not expect the designer to provide all of these parameters each time the size of the bipolar transistor changes or a new bipolar transistor layout type is chosen. In our design system we only require the "basic" device parameters. All the other necessary parameters will be automatically generated by the system.

We classify the device parameters of a bipolar transistor as follows:

- **size parameters** - the device parameters which change with the emitter length of the bipolar transistor.

- **class parameters** - the device parameters which are layout dependent. These parameters are invariant with the transistor size, they only depend on the transistor layout.

- **technology parameters** - the device parameters which only depend on the fabrication technology.

An overview of the size parameters is given in Table 5.1 together with their dependency on the emitter length $L_E$. In this table the variable $K_{xy,i}$ stands for the initial value of the device parameter $K_{xy}$, while the variable $K'_{xy}$ stands for the normalized value of the device parameter $K_{xy}$. Note that this last value can be either multiplied with or divided by the emitter length. The device parameter $K_{xy}$ can be any of the size parameters of the bipolar transistor.

The SPICE parameters which were not enumerated in this table, are either class or technology parameters. They are independent on the emitter size. Consequently, they do not change during the gate optimization process.

A module which automatically generates the size parameters was implemented in C++ programming language. The error between the generated parameters and SPICE parameters was found to be less than 2% over an emitter size range between 1.0 $\mu$m$^2$ and 6.0 $\mu$m$^2$. 
5.2 Transistor Sizing

We use the totem-pole BiCMOS inverter (Figure 3.3 on page 41) to present our buffer optimization methodology. A large variety of other BiCMOS drivers could just as well be optimized in the same way. The BiCMOS gate delay implemented in our gate design system is the delay model developed in section 3.5 (variable current gain model).

For the totem-pole BiCMOS inverter the width of the MOS transistors $M_1$, $M_2$, $M_3$, and $M_4$ need to be determined, as well as the emitter length of the bipolar transistors $Q_1$ and $Q_2$. The bleeding devices $M_3$ and $M_4$ should be small to achieve the overall minimum, but large enough to discharge the bipolar transistor base in a short time. Our experiments indicated that a good size for these devices would be 10% of the size of $M_1$ or $M_2$. To get equal pull-up and pull-down delays, the bipolar transistors $Q_1$ and $Q_2$ should be identical. Consequently, the device unknowns are the width of the MOS transistors in pull-up and pull-down stages and the emitter length of the bipolar transistor.

Increasing the emitter area will increase the knee current. It will also decrease the collector, emitter and base resistances. Therefore, the high injection related degradation of the current gain and the base transit time will be reduced, as well as the saturation probability. As a result the switching speed will improve. The increase in the emitter area will also contribute to large parasitic capacitances, and will decrease the ideal maximum current gain. This increases the delay. We need to find the emitter length at which the former effects are dominant over the latter.

The propagation delay is inversely proportional to MOS transistor currents, thus
any increase in MOS transistor sizes will lead to a decrease of the propagation time. However, choosing a large MOS transistor size will increase the gate input capacitance and, consequently, the overall delay. To take into account this contradictory effects of MOS transistor size on the gate delay, a CMOS gate is chosen to drive the BiCMOS inverter throughout all the optimization steps. Figure 5.1 shows this practical schematic of a CMOS gate driving a BiCMOS one. During the optimization process the overall delay of CMOS and BiCMOS gates will be minimized.

![Figure 5.1. Optimization schematic.](image)

The following simple model has been chosen for CMOS gate propagation delay:

\[
t_d,C_{\text{CMOS}} = \tau + \beta \cdot C_l
\]

where \( \tau \) is the intrinsic delay of the CMOS gate, \( \beta \) is the gate drive capability, and \( C_l \) is the load capacitance at the output of the CMOS gate. This model is commonly used in network optimization.

However, to improve the design accuracy our optimization algorithm can accommodate any other CMOS delay model.

The strategy in designing the optimum BiCMOS gate is to vary the width of the MOS transistors (\( M_1 \) and \( M_2 \)) and the emitter length of the bipolar transistors (\( Q_1 \) and \( Q_2 \)) over a predefined range. The overall delay is calculated. If \( t_d,C_{\text{CMOS}} \) is the propagation delay of the CMOS gate calculated with expression (5.1), and \( t_d,B_{\text{BiC}} \) is the propagation delay of the BiCMOS gate calculated with the expression (3.27), then the overall delay is \( t_d,C_{\text{CMOS}} + t_d,B_{\text{BiC}} \). Under the assumption that the signal \( V_{\text{input}} \) at the input of the CMOS gate makes a step transition from zero to \( V_{\text{DD}} \) or vice versa, \( t_d,C_{\text{CMOS}} \) and \( t_d,B_{\text{BiC}} \) are calculated and than the overall delay is minimized.

The delay calculation indicates that for a fixed output load capacitance there is a pair \( (W, L_E) \) of the MOS transistor width and bipolar transistor emitter length, which minimize the delay. This conclusion has also been confirmed by SPICE simulations [4, 29]. Despite this result, we do not perform the optimization process for both MOS transistor width and bipolar emitter length in the same time. The reason is that in practical BiCMOS processes the bipolar transistor size is standardized. Unlike the MOS transistor, the bipolar transistor is only available in a limited number of sizes, such as unit size, double size etc. Therefore, it does not make sense to calculate an optimum value for the emitter length only to find out that this value is not available.
5.2 Transistor Sizing

To find the minimum driving delay we iteratively set the bipolar transistor emitter to a standard size, then calculate the width of the MOS transistor which assure the overall minimum delay. Finally, the pair \((W, L_E)\) which gives the minimum delay is chosen.

![Graph showing overall delay as a function of MOS transistor size and two standard bipolar transistor: \(S_E = 4.0 \mu m^2\) for upper sheet and \(S_E = 10.0 \mu m^2\) for lower sheet.]

**Figure 5.2.** Overall propagation delay as a function of MOS transistor size and two standard bipolar transistor: \(S_E = 4.0 \mu m^2\) for upper sheet and \(S_E = 10.0 \mu m^2\) for lower sheet.

In Figure 5.2 the overall rise delay of CMOS and BiCMOS gates is shown as a function of output load capacitance, MOS transistor width, and for two standard bipolar transistor sizes. The two emitter sizes are \(4.0 \mu m^2\) for the upper sheet and \(10.0 \mu m^2\) for the lower sheet.

Increasing the width of the MOS transistor will decrease the gate delay down to a minimum value. After this minimum point is reached, the delay starts increasing.

Increasing the emitter area will also decrease the gate delay. For lower load capacitances, however, this might not be true anymore. Below a certain value of the load capacitance the delay of the smaller size bipolar will even be better than the delay performance of the larger size bipolar. In the latter case, the time necessary to charge the capacitive network of the bipolar transistor is dominant over the time necessary to charge the output load capacitance. Consequently, a bigger emitter size causes an increased delay.

We made the following observation: even for high capacitive loads, there is not much difference between the performance of the two bipolar transistors. For ex-
ample, at 10.0 pF load capacitance, the delay improvement is less than 300 ps for an emitter area increase of $2\frac{1}{4}$ times. In other words, for an area increase of 150%, a delay decrease of only 20% is obtained. In practice, the design of the BiCMOS gate is clearly a compromise between speed and silicon area. It is also well known that most of the silicon area is occupied by the bipolar transistor. Therefore, sometimes it is better to choose a smaller size bipolar transistor to realize a good area/delay tradeoff.

![Graph showing overall delay versus MOS transistor width](image)

**Figure 5.3.** Overall propagation delay versus MOS transistor width in case of our delay model and SPICE simulations; $S_E = 4.0 \, \mu m^2$.

The results presented in Figure 5.2 are derived using our analytical model developed in chapter 3 (variable current gain model). In Figure 5.3 both the analytical results and the SPICE simulation results are plotted to verify the accuracy of our model. The overall propagation delay versus MOS transistor width is shown for different output load capacitances. The bipolar transistor used in this experiment has an emitter area of $4.0 \, \mu m^2$. The maximum deviation compared to SPICE simulations is less than 10%.

The results in Figure 5.3 show that given an output load capacitance $C_{load}$, the BiCMOS propagation delay can be minimized by adjusting the size of the MOS transistor for a fixed bipolar transistor size. SPICE simulations also confirm these results.
5.2 Transistor Sizing

As already mentioned, the minimum delay depends on the size of both CMOS and BICMOS gate. By increasing the size of the CMOS gate, the width of the MOS transistors in BICMOS pull-up and pull-down stages can be increased without affecting the overall performance. Larger bipolar currents will be supplied and the minimum delay will decrease. This adjustment cannot be continued endlessly. The larger the width of the MOS transistor, the faster the bipolar will be driven into the high-current region where the forward current gain decreases and the forward transit time increases. Therefore, bipolar transistor performance will become worst, and the propagation delay will increase.

Besides, a larger size for the CMOS gate will require a gate with more driving capability at its input. The result is a chain reaction which may become difficult to control. This case is similar to the CMOS tapering buffer design. One of the reasons to use BICMOS drivers is to avoid this kind of designs which are area consuming. If the output load capacitance can be split, than a tree of buffers will be a better solution to this problem. However, for driving large off-chip load capacitances, or output pads, or on-chip clock drivers is still necessary to scale the buffers with the output load [25, 4, 29].

![Figure 5.4](image)

**Figure 5.4.** Overall propagation delay versus MOS transistor width and output load capacitance, and its minimum value; $S_E = 4.0 \mu m^2$.

In Figure 5.4 the overall propagation delay versus MOS transistor width and load capacitance is shown in a three-dimensional plot. The BICMOS gate is driven by a fixed size CMOS gate and the bipolar transistor used in the buffer output stage has
an emitter area of 4.0 \( \mu \text{m}^2 \). The minimum propagation delay for each output load capacitance is also shown with small circles.

From this figure one can easily see that the width of the MOS transistor is gradually increasing with the minimum delay when the load capacitance increases. This leads to the conclusion that for each output load capacitance we need to design a new gate to attend the minimum propagation delay. This is a very expensive solution to buffering problem, which is not always possible especially in ASIC design where a pre-designed standard library is used. In previous optimization works [4, 29, 44] this is the only solution proposed.

We propose an efficient methodology of designing BiCMOS drivers. The designed gate can drive a larger range of output load capacitances without much penalty in delay performance. To introduce this methodology we use the results shown in Figure 5.5. The overall propagation delay is calculated for different load capacitances and MOS transistor widths.

![Graph showing overall propagation delay versus MOS transistor width for different output load capacitances; minimum delay and 0.25% increase in minimum delay.](image)

**Figure 5.5.** Overall propagation delay versus MOS transistor width for different output load capacitances; minimum delay and 0.25% increase in minimum delay.

Increasing the MOS transistor width not only generates a delay plot which has a minimum, but its shape looks like it saturates before this minimum is reached. This saturation effect is visible especially for higher load capacitances when larger MOS width is necessary to reach the minimum. Consequently, the delay differences in this region become very small. So small that if we increase the minimum delay
with only 0.25%, a decrease in MOS transistor width of up to 48% can be achieved. The result is a very good area/delay tradeoff. In this particular case the delay penalty is very small.

The points corresponding to the 0.25% increase in minimum delay, have the MOS transistor widths very close to each other. For the example given in Figure 5.5, these points are now spaced between 23.0 μm and 40.0 μm, while for the minimum delay points the range is (30.0 μm, 80.0 μm). For the optimum gate design, the width average of the former points is taken. In case that these points have a larger width range, subregions of the load capacitance need to be made.

5.3 The Optimization Procedure

The BiCMOS gate optimization process can be summarized as follows:

1. Set the CMOS gate to drive the BiCMOS gate; give its delay parameters.
2. Set a standard bipolar transistor.
3. Calculate the overall propagation delay with the expressions developed in section 3.5 (variable current gain model); a desired range of load capacitances should be available.
4. Find the minimum delay points (MOS width, delay value) for each load capacitance.
5. Find the points (MOS width, 0.25% increase in minimum delay value); the increase in minimum delay can also be specified by the user, otherwise the increase which assures the best area/delay trade-off will be taken.
6. Calculate the average MOS transistor width.
7. Repeat from step 2.
8. Choose the best performance gate; performance can cover delay, area or both.

The previous optimization steps are implemented in a C++ program.

The BiCMOS pull-down stage has a configuration slightly different from the pull-up stage. Consequently, optimizing both pull-up and pull-down stages for delay may not always assure equal propagation times for both rise and fall delay. More likely, the delays will be slightly different. The gate size can only be tuned for a given value of the load capacitance.

For example, in Figure 5.6 the rise and fall delays are calculated versus MOS transistor width for a 3.0 pF output load capacitance. For the delay calculation we used
our analytical delay models. The standard bipolar transistor chosen for the gate has an emitter area of 4.0 $\mu$m$^2$. The overall propagation time has a minimum at the point denoted 2 in the figure, for both rise and fall delay. However, the rise and fall delays are slightly different. An equal propagation time is achieved at the point denoted 1 in the figure. Here, there is a small increase in minimum delay but the rise and the fall times are equal.

![Graph showing delay vs. transistor width](image)

**Figure 5.6.** Rise and fall time delays as a function of MOS transistor width; $C_{load} = 3.0$ pF, $S_E = 4.0$ $\mu$m$^2$.

### 5.4 Conclusion

The optimization methodology presented here assures a good area/delay tradeoff for the BiCMOS driver. The gate optimization algorithm can be used at the very early stage of the ASIC design when the library design takes place. Alternatively, the driver optimization can be made along with the combinational network optimization process to improve the design performance. This step is known as resizing. Next chapters deal with the optimization techniques for technology mapped circuits.
Chapter 6

NETWORK OPTIMIZATION

Optimizing a circuit is a very complex task. We can optimize the circuit performance, area, or testability. They are not independent, optimizing with respect to one may cause degradations with respect to the others. The testability can be ensured without degrading circuit performance, but area and performance optimization at the same time are usually not compatible. The focus of this research is to develop optimization techniques which improve the circuit speed, while keeping the area increase as small as possible.

After the technology dependent optimization phase is completed, a netlist of gates in the specified target technology is available for the layout synthesis process. This resulted circuit, however, does not always satisfy the timing constraints. Besides, additional wire and interconnect capacitances will be added to the gates output during the layout synthesis process. The circuit will slow down even more. Therefore, additional optimization steps are necessary. At this level the optimization is usually performed on the mapped circuits, but local re-mapping routines can also be considered.

To reduce the complexity of the optimization problem the logic circuits are first partitioned into:

- **combinational circuits** - logic acyclic circuits;
- **sequential circuits** - logic circuits with feedback.

Since sequential synthesis and optimization techniques rely on those for combinational circuits, our work is targeted towards combinational circuits. The optimization methods can be extended to work also for sequential circuits under the assumption that sequential circuits can be decomposed into combinational logic blocks with sequential elements at their primary inputs and primary outputs. Consequently, the sequential circuit performance is bounded by its combinational circuit performance.

The performance of a combinational logic circuit is measured by the input/output propagation delay. Therefore, the sequential circuit performance is related to the
worst case propagation delay of signals between two register boundaries. This delay determines the fastest clock that can be applied to the circuit. The main objective in performance optimization is to minimize the propagation delay from inputs to outputs. The result is a faster combinational block which will lead to a higher clock frequency for the system.

While in technology independent optimization the combinational circuits are represented by Boolean functions which have to be minimized, in technology dependent optimization the methods strongly depend on the technology and the design style used to implement the circuits. Our design style is cell-based design in a BiCMOS technology.

For cell-based design new performance problems arise in the layout synthesis phase. The cells need to be placed and interconnected. Placement and routing problems are generated. A less efficient placement program may place two connected cells far apart which leads to longer wires connecting them. The increased capacitive loading may cause the circuit to slow down, and ultimately affect the clock frequency.

6.1 Assumptions and Abstractions

Before introducing the optimization methods used in cell-based design, some assumptions have to be made to decide which optimization procedures are suitable for this particular design and to correctly evaluate the optimization effect on the design performance.

We assume that the following decisions have been made before the optimization process started:

1. The target technology is BiCMOS technology.
2. The design style is cell-based design.
3. There exists a standard-cell library. The primitives of the library are a set of fixed logic gates designed in BiCMOS technology.
4. A library cell is characterized by:
   - a single output combinational logic function;
   - an area cost;
   - the input/output propagation delays specified as a function of the load capacitance and input signal slew rate, for rising/falling transitions at the output or by worst case values.
5. The optimization techniques are applied to a logic combinational circuit.
6. The combinational circuit is mapped into an interconnection of gates that are instances of the given standard-cell library.

7. The delay of the circuit is determined by the longest path in the circuit. Since the longest path may be a false one, this is a pessimistic assumption which can lead to an overestimation of the delay.

8. The layout effect on the delay is taken into account by including an additional delay based on wire capacitance information.

9. The objective is to minimize the maximum propagation delay from inputs to the outputs.

The well known technology dependent optimization techniques which can be applied to a cell-based design are:

- gate resizing
- applying local transformation to mapped circuits
- buffering/repowering

Since we made the assumption that the circuit is mapped into an interconnection of cells that are instances of the given standard-cell library and these cells are fixed, the gate resizing optimization method is not applicable. However, local driver resizing may be allowed for high capacitive loads which cannot be split. In this particular case the design methodology presented in the previous chapter is applied.

We also choose not to re-map the circuit. For optimization algorithms using local transformations to mapped circuits the reader is referred to [59]. This method implies logic minimization techniques which are beyond the scope of this work.

The optimization methods we use in this work are buffering and repowering. They are briefly introduced in the following sections. A detailed presentation of these techniques is made in the following chapter.

Optimizing the circuit delay will affect the circuit area and power dissipation. We try to keep area increase and power consumption as small as possible while reducing the circuit delay.

The timing optimization program we develop provides an easy to use interactive environment which is an extension of the familiar environment provided by SIS package [51]. Its input is a technology mapped netlist of a combinational logic network, the user defined I/O specifications, and the library used in mapping the circuit. The logic network can be a result of a logic synthesis system, e.g. SIS [51]. Alternatively, it can be a user defined circuit which includes the wire capacitances. The user defined I/O specifications consist of arrival time of the primary inputs (the time when the signals actually arrives at the primary inputs of the circuit), and the
required time of the primary outputs (the time when the signal is required to arrive at the primary outputs).

The combinational logic network is represented as a DAG (directed acyclic graph). Each vertex in the graph corresponds to a cell in the circuit, and each cell is assumed to be single output cell. The edges of the graph represent the connections between the cells in the circuit (see Figure 6.1).

Two lists of edges are associated with each vertex $v$. The edges ending in a vertex $v$ represent the fanin set of the vertex $v$ (the cell inputs), while the edges starting in a vertex $v$ represent the fanout set of the vertex $v$ (the driven cells). We denote these two groups as $fanin(v) = \{e \in E \mid target(e) = v\}$ for fanin set, and $fanout(v) = \{e \in E \mid source(e) = v\}$ for fanout set with $E$ the set of all edges associated with the vertex $v$.

![Figure 6.1. Schematic of a network represented by a parameterized DAG.](image)

The library consists of cells represented by several templates for each type of gate. These templates implement the same logic function, but differ in area, delay parameters, and the maximum capacitive load they can drive.

To determine the circuit speed performance, static timing analysis is performed on the network. To implement the static timing analysis algorithm, we need a delay model to predict the delay of each individual cell in the network.

### 6.2 Delay Models

Static timing analysis is based on finding the longest paths in the network. These paths are usually critical paths which do not meet the timing constraints. Thus, their speed needs to be improved. The propagation delay defined in section 3.1, definition 1 is very convenient in evaluating the path delay. This type of delay
computes the delay of a long chain by adding the individual cell delays. This is possible, because the propagation delay of a cell is taken at 50% supply voltage for each individual cell.

The BiCMOS delay model we choose is the one derived in section 4.1, which we will repeat here for convenience. It depends on the output load capacitance of the cell. The routing capacitance is also added to the load to account for wire and interconnect effects. They are treated as one entity.

The cell delay $d_g$ is calculated between the output and the $k$-th input of the cell $g$ as follows:

$$(6.1) \quad d_g(k) = \tau_g(k) + \beta_g(k) \cdot \gamma_L^{\alpha_g(k)} \text{ with } \gamma_L = \sum_{i \in \text{fanout}(g)} \gamma_i$$

where, $\tau_g(k)$ is the intrinsic delay from the $k$-th input to the output of cell $g$, the pair $(\beta_g(k), \alpha_g(k))$ represent the extrinsic delay parameters with $\beta_g(k)$ as the drive capability and $\alpha_g(k)$ as the gate dependence on load capacitance. In short, $(\tau, \beta, \alpha)$ are the delay parameters of cell $g$. The parameter $\alpha_g(k)$ can take values between $(0.5, 1)$ as shown in section 4.1. In case $\alpha_g(k) = 1$, the gate delay model becomes linear and is identical to the well known CMOS delay model. Finally, $\gamma_L$ is the load capacitance calculated as sum of input pin capacitance of all the cells in the fanout set of cell $g$, $\text{fanout}(g)$, including the corresponding routing capacitance.

To keep the algorithm complexity at a low level we do not make any distinction between rise and fall delay. Both the rise delay $d_r(k)$ and the fall delay $d_f(k)$ of the cell are calculated as shown before and the maximum value is taken so that $d_g(k) = \max(d_r(k), d_f(k))$.

So far, this delay model does not depend on input signal slew rate. Taking into account the input slope will give more accurate delay estimations, but will result in a very complex algorithm. For our particular optimization algorithms we look for cells with big fanout and try to lower it. Besides, improving the speed will lead to faster output slope of the cell as well. The input slope is a necessity for dynamic timing verification where additional information concerning the switching activity of the gates is available.

In static timing analysis we suppose that the gate is switching and the worst case delay is taken. Sometimes, bigger errors can arise if we consider the input slope. For example, when the cell propagation delay becomes negative, because of a too slow input slew rate the path delay decreases which in fact is a false information. To avoid this kind of problems complex algorithms need to be implemented. The drawback of a complex algorithm is that it will slow down the optimization process, especially for large circuits.
6.3 Static Timing Analysis

Before starting the timing analysis, the network cells must be arranged in topological order. As a next step the timing performance of the circuit can be estimated by evaluating the arrival and required times at all the nodes in the circuit taking into account the primary input arrival times, the primary output required times, and the propagation delay through all the cells in the circuit.

The arrival time at the cell \( g \), denoted by \( a_g \), is the time when the signal arrives at the output of the cell, while the required time \( r_g \) is the time when the signal should arrive at the inputs of the cell. A recursive procedure is evaluating the arrival and the required times for every gate \( g \) in the network with the formulas:

\[
(6.2) \quad a_g = \max_{i \in \text{fanin}(g)} (a_i + \tau_i + \beta_i \gamma_j^{\alpha_i})
\]

\[
(6.3) \quad r_g = \min_{i \in \text{fanin}(g)} r_i - \max_{i \in \text{fanin}(g)} (\tau_i + \beta_i \gamma_j^{\alpha_i})
\]

where, \( a_i \) is the arrival time at the \( i \)-th input of the cell \( g \) (\( i \in \text{fanin}(g) \)), and \( r_i \) (\( i \in \text{fanout}(g) \)) are the required times at the output of the cell \( g \) (Figure 6.1).

The slack of the cell \( g \), denoted \( \text{slack}_g \), is defined as the difference between the minimum required time at the output of the cell and the arrival time of the cell:

\[
(6.4) \quad \text{slack}_g = \min_{i \in \text{fanout}(g)} r_i - a_g
\]

A positive slack at an internal node measures how much additional delay the corresponding signal may tolerate without violating the delay constraints.

If the slack at the primary outputs is negative, then the circuit does not meet the timing constraints. In this case the speed-up procedures as described in this work must be applied. Consequently, a negative slack at an internal node means the signal arrives later than required.

6.4 Critical Path Detection

After the initial slack calculation, the paths with negative slack are extracted and grouped in a so called \( \epsilon \)-network [12, 57]. To ease the optimization process, an as small as possible set of paths should be chosen. A critical path is defined as the path from input to the output of the logic network on which all the nodes have the slack less than \( (\text{slack}_{\text{min}} + \epsilon) \leq 0 \), where \( \text{slack}_{\text{min}} \) is the minimum slack value in the network, and \( \epsilon \) is a user defined constant. This set of critical paths is a subnetwork of the initial input network (Figure 6.1). The nodes laying on a critical path are referred as critical nodes.
6.5 Global Optimization

A node cutset of the $\varepsilon$-network is a maximal subset of critical nodes dynamically generated as a function of local optimization results. The node cutset paradigm was proposed in [12], and was successfully implemented in later works [57, 56, 62].

The cutset extraction is based on a criticality coefficient $c_g$ associated with each cell $g$ in the $\varepsilon$-network which measures how suitable the node is to be subject to the optimization process to fulfill the delay constraints of the circuit [31]. This coefficient is derived from the cell extrinsic delay parameters, the position of the cell on the critical path, and the slack of the cell as follows:

$$c_g = \left| \frac{\beta_g \times (\sum_{i \in \text{fanout}(g)} \gamma_i)^{\alpha_g}}{\text{slack}_g / \text{length}_g} \right|$$

The position of the cell on the critical path, denoted $\text{length}_g$, represents the maximum signal path from the primary inputs of the circuit to the cell inputs. It can be calculated with the recursive formula:

$$\text{length}_g = 1 + \max_{i \in \text{fanin}(g)} (\text{length}_{g_i})$$

In the node cutset the cells with the highest criticality are chosen.

The criticality coefficient assures that the gates with a large fanout will be optimized. However, a large criticality coefficient is not the only criterion in deciding whether the node should be included in the cutset or not. We have to make sure that it is really necessary to optimize the node to assure the delay performance with small increase in area and power consumption.

If the required time of the node is a minimum required time for at least one of the gates driving this node, then the node has to be optimized. Else, improving the delay of this node yields no overall gain, since the required times of the cells driving this node will not be improved.

Consequently, the condition which needs to be satisfied by a node $g$ to become a cutset node is:

$$\bigvee_{g \in \text{fanin}(g)} (r_g = \min_{i \in \text{fanout}(g)} r_i)$$

where $g_i$ are the cells in the fanin set of cell $g$ (the cells driving the inputs of $g$) and $r_i$ are the required times at the output of each cell $g_i$.

Optimizing the speed of a cell not satisfying the condition 6.7 does not improve the delay performance of the circuit, but increases its area and power consumption.

6.5 Global Optimization

The algorithm presented in this section reduces the critical path delay while keeping the area and power consumption increase as small as possible. The optimization procedure is applied for each cell in the cutset to generate the appropriate fanout tree of buffers and inverters at a minimum cost.
If the cost to minimize is the delay of the network, the minimum delay fanout tree is constructed at any area expense. If the cost to minimize is the delay under area constraints a new cost function is used, denoted by sensitivity $s_g$ of the cell $g$, and defined as follows:

**Definition 6 (Gate Sensitivity)** The cell sensitivity is the required time improvement per additional area:

$$s_g = \frac{r_{go} - r_g}{\sum_{i \in \text{fanout\_tree}(g)} \text{area}_b}$$

where, $r_g$ and $r_{go}$ are the required times before and after optimization process of the cell $g$, and $\sum_{i \in \text{fanout\_tree}(g)} \text{area}_b$ denotes the area of the fanout tree $\text{fanout\_tree}(g)$ constructed to optimize the delay of the cell $g$. We use $b$ to designate a buffer or an inverter in the fanout tree.

A lower limit $s_{min}$ for sensitivity is imposed and only the fanout trees with a sensitivity greater than this limit are accepted. Obviously, the sensitivity limit is at least zero, meaning no decrease in delay is achieved after buffering process.

After all gates in the cutset are processed, other paths in the network may become critical so that delay recalculation and optimization process is repeated until the timing constraints are met or there is no improvement in the circuit delay. The network optimization algorithm is presented in Figure 6.2.

The algorithm starts with performing static timing analysis on the input netlist. It calculates the arrival times, the required times and the slacks for all the nodes in the network with the expressions given in section 6.3. Using the slack of the nodes, the critical paths are detected and grouped in an $\epsilon$-network. Then the node cutset is extracted. Each node in the cutset is sent to the fanout optimization procedure, denoted here `single\_node\_buffering`.

After buffering the delay reduction and the area consumption are compared by calculating the sensitivity coefficient. If the delay/area tradeoff is better than the imposed limit the optimization results are accepted. If not, no change is made in the fanout of the node.

The `single\_node\_buffering` algorithm, also known as fanout problem, is presented in the following chapter.

### 6.6 Conclusion

In standard-cell based design, after the Boolean description of the circuit has been mapped into a specific target library, optimization techniques are performed to improve the circuit performance. One way to enhance the circuit speed is to resize the network primitives along with buffering process. The cell design as well as the
algorithm network_optimization

input: topologically ordered netlist, primary input arrival times, 
primary output required times, constant $\epsilon$

repeat {
    foreach gate $g \in$ netlist from (primary inputs) to (primary outputs)
        $a_g = \max_{i \in fanin(g)} (a_i + \tau_i + \beta_i * \left( \sum_{j \in fanout(g)} \gamma_j \right)^{\alpha_i})$
    foreach gate $g \in$ netlist from (primary outputs) to (primary inputs)
        $r_g = \min_{i \in fanout(g)} r_i - \max_{i \in fanin(g)} (\tau_i + \beta_i * \left( \sum_{j \in fanout(g)} \gamma_j \right)^{\alpha_i})$
    foreach gate $g \in$ netlist
        $slack_g = \min_{i \in fanout(g)} r_i - a_g$
        $slack_{\min} = \min_{g \in \text{netlist}} slack_g$
        generate an $\epsilon$-network($slack_{\min} + \epsilon$)
    foreach gate $g \in \epsilon$-network
        $c_g = \frac{\beta_g \left( \sum_{i \in fanout(g)} \gamma_i \right)^{\alpha_g}}{slack_g / length_g}$
        generate the cutset: ($c_g \geq c_{\min}$) and \( \bigcup_{g \in \text{fanout}(g)} (\min_{i \in \text{fanout}(g)} r_i = r_g) \)
        /* $c_{\min}$ adjustable during the optimization process */
    foreach $g \in$ cutset
        single_node_buffering($g$
            $s_g = \frac{r_{go} - r_g}{\sum_{i \in \text{fanout-tree}(g)} \text{area}_i}$
        if $s_g \geq s_{\min}$
            insert the fanout tree in the circuit netlist
}

until (timing constraints are met or no more delay improvement)

output: netlist of the optimized network

end network_optimization

Figure 6.2. Network optimization algorithm.

Optimization are not trivial steps. They can be very time consuming. Therefore, computer help is necessary to complete the job.

A global optimization strategy for mapped circuits was presented. The network optimization goal is to speed-up the critical paths in the circuit while keeping the area and power consumption increase as small as possible. The fanout optimization algorithm is presented in the following chapter.
Chapter 7

FANOUT OPTIMIZATION

After the high level description of a design is synthesized and mapped in a target technology, is the turn of the optimization techniques to enhance the design quality. At this level the design performance can be estimated more accurate since the target technology is already known and the layout data can be made available.

In chapter 6 the global strategy for optimizing a mapped circuit was presented. After static timing analysis is performed, the critical paths of the network are extracted and grouped in the ε-network. The most critical nodes of the ε-network are then defined using the criticality coefficient. These nodes form the node cut-set. The cutset nodes have a large number of fanouts or they have to drive large capacitive nodes. Therefore, their driving capability needs to be improved.

This chapter presents the fanout optimization techniques we developed for improving the speed of a mapped circuit. These techniques are applied to each node in the cutset.

First, the fanout problem is introduced followed by the algorithms we proposed for solving this problem. The fanout buffering, the node repowering and sink assignment methods are described. Finally, an example and the experimental results are presented.

7.1 Problem Definition

The fanout problem was defined by Berman, Carter and Day [3] in 1989 as follows: the process of choosing a circuit that does not compute any function but simply distributes a signal to its destinations or sinks, so that the timing constraints on these sinks are met.

The fanout tree is constructed by building a tree of buffers and inverters between the source and its destinations. Since the fanout tree does not compute any function, but only enhances the propagation speed of a signal at the cost of area increase and power consumption. The interesting optimization problems are to minimize the circuit delay or to minimize the area or power consumption under delay constraints.
Determining whether there is a fanout circuit that meets given timing and area constraints is an NP-complete problem. Specifically, given a list of required times, a target technology, and an area A, the decision problem, "Is there a solution to the fanout problem with area at most A and nonnegative required time?" is NP-complete. This theorem was formulated and proved by Berman, Carter and Day [3]. Its significance is that one cannot expect to find the best fanout circuit in polynomial time, even for moderate size problems. In other words, the fanout problem is NP-hard.

The important factors in solving the fanout problem are the delay model, the required times and the sink loads. The delay model can be taken constant, unity or more complex, the required times can be equal or varying, the same is valid for sink loads. However, simple delay models are not realistic and more complex ones preclude an efficient solution method. An overview of possible fanout solutions for different delay models, required times and sink loads together with the complexity issues involved is given in [59].

In this work, because of the NP-hardness of the fanout problem, we resort to the heuristic methods. Some simplifying assumptions are made during the optimization steps to limit the complexity of the problem.

The fanout tree optimization process is carried out for all cells in the node-cutset of the combinational logic network. Two goals are of interest:

- to maximize the required time at the source
- to minimize the number of internal nodes (that is minimizing the area of the circuit) under delay constraints

We restrict our attention to the simpler problem of designing fast fanout circuits. For multiple variants of equal delay performance we select the minimum area configuration.

We solve the fanout problem by finding a tree of buffers and inverters at the output of a cell g, denoted by source s, that distributes the signal to various destinations and maximizes the required time of g.

We do not make any distinction between complemented and un-complemented source signals. If we consider the un-complemented signal as the signal provided directly by the source then the complemented signal is obtained from the source via an inverter. If the performance of the gate does not deteriorate and the polarity does not change, this inverter will collapse later on after the local optimization is completed.

Our approach is based on combinatorial merging algorithm [20], but we merge the destinations in more than one node at a time.
7.1 Problem Definition

The fanout tree is a rooted tree of variable degree \( p \), \( 1 \leq p \leq n \) where \( n \) is the size of the fanout problem. \( S = (s_n, ..., s_1) \) is the set of destinations (sinks) having the same polarity and ordered with decreasing required time, i.e. \( (r_n, ..., r_1) \) (Figure 7.1), and \( S = \{(s_k, s_{k-1}, ..., s_{k-p}), \forall 1 \leq p < k \leq n\} \) are the subsets of \( S \).

![Figure 7.1. A single node fanout problem.](image)

**Definition 7 (Consecutive Subsets)** Given the set of subsets of sinks \( S = \{(s_k, s_{k-1}, ..., s_{k-p}), \forall 1 \leq p < k \leq n\} \). Two subsets in \( S \), \( s(i, j) = (s_i, ..., s_j) \) and \( s(q, r) = (s_q, ..., s_r) \), are called consecutive if and only if \( q = j - 1 \).

The minimum required time of the fanout problem is denoted \( r_{\text{min}} \) and equals the required time \( r_1 \) in the beginning. The loads of the sinks are denoted \( (\gamma_n, ..., \gamma_1) \). All the quantities \( \gamma_{n,j} = \sum_{n \geq j \geq 2} \gamma_j \) are precomputed, and their derivatives \( \gamma_{i,j} = \sum_{n \geq k \geq j} \gamma_k = \gamma_{n,j} - \gamma_{n,i+1} \) are available.

Similarly, for all buffer templates \( b \), with \( (\tau_b, \beta_b, \alpha_b) \), in the library \( L \) we evaluate the expected required times at the input of the buffers for all subsets in \( S \) which have the total load \( \gamma_{i,j} \) greater than the buffer input load \( \gamma \).

Our single node buffering algorithm is shown in Figure 7.2.

The optimum buffer to drive the merged sinks \( (s_i, ..., s_j) \) is chosen taking into account the delay due to the input load of the buffer \( \gamma \), by means of the time in which the source gate will drive the input capacitance of the buffer.

The final fanout tree will result in a structure with a small output capacitive load of the source. Therefore, we make the assumption that the source is of CMOS type. Our assumption is supported by the fact that, in general, BiCMOS gates are only used for driving high loads to assure a good area/delay tradeoff. Consequently, selecting the optimum buffer to drive the merged sinks \( (s_i, ..., s_j) \) is equivalent with maximizing the value \( r_j - (\tau_b + \beta_b \gamma_{n,j} \alpha_b) - \beta_s \gamma \).

The number of subsets in \( S \) is \( n(n - 1)/2 \) and there are \( d \) buffer templates in the library. Thus, the complexity of the algorithm which finds the optimum buffers to drive the sinks is \( O(dn^2) \).
algorithm single_node_buffering(g)
    input: source $s = g$ with required time $r_s = r_g$
    sinks $S = (s_{n_1}, ..., s_{1})$ with $(r_{n_1}, ..., r_{1})$ and $(\gamma_{n_1}, ..., \gamma_{1})$
    repeat { precompute $\gamma_{n_i} = \sum_{n \geq j \geq i} \gamma_j \forall 1 \leq i < n$
          foreach $1 < i \leq n$
                  foreach $1 \leq j < i$
                          $b_{i,j} = \arg \max_{x \in C}[r_j - (r_b + \beta_b \gamma_{i,j}^{a_k} - \beta_s \gamma_b)] \text{ if and only if } \gamma_{i,j} > \gamma_b$
                          $r_{i,j} = r_j - (\gamma_{b_{i,j}} + \beta_{b_{i,j}} \gamma_{i,j}^{a_k_3})$
                          $r_{g_{i,j}} = \beta_s (\gamma_{i,j} - \gamma_{b_{i,j}})$
                  }
          if $\max_{1 \leq i \leq n}(r_{i,j}; \forall 1 \leq j < i) < r_1$
                 build p.h.tree
                 output: fanout tree
                 break
          else {
                 build trace.tree
                 output: partial solution of the fanout problem
                 and a new fanout problem $S' = (s'_{n_1}, ..., s'_{1})$
          }
    }
until (only one sink is left or no more delay improvement)
output: fanout tree of an optimized node $g$
end single_node_buffering

Figure 7.2. Single node buffering algorithm.

Inserting buffers between the source and its sinks will alleviate the driving load of the source. Consequently, merging the sinks $(s_i, ..., s_j)$ in a common node driven by a buffer $b_{i,j}$ will result in a lower source delay. For a strict quantitative estimation of this source delay improvement we introduce the following definition:

**Definition 8 (Rough Gain)** The rough gain $rg_{i,j}$ is the time improvement in source delay due to the lowering of its output load capacitance by $(\gamma_{i,j} - \gamma_{b_{i,j}})$.

In the previous definition, $\gamma_{b_{i,j}}$ is the input load of the optimum buffer chosen to drive the merged sinks $(s_i, ..., s_j)$. The rough gain does not take into account the buffer delay itself.

Once the optimum buffer selection for each pair $(i, j)$ is done, the buffer required
time \( r_{i,j} \) and the rough gain \( rg_{i,j} \) are also evaluated with the formulas:

\[
\begin{align*}
    r_{i,j} &= r_j - (\tau_{b_{i,j}} + \beta_{b_{i,j}} \gamma_{i,j} \alpha_{b_{i,j}}) \\
    rg_{i,j} &= \beta_s (\gamma_{i,j} - \gamma_{b_{i,j}})
\end{align*}
\]

with \((\tau_{b_{i,j}}, \beta_{b_{i,j}}, \alpha_{b_{i,j}})\) the delay parameters of the buffer \( b_{i,j} \), and \( \beta_s \) the drive capability of the source.

We develop two optimization techniques to improve the delay of the source node, by means of increasing the source required time. The choice between these two techniques is made based on the required time values after buffering. They are presented in the following sections.

### 7.2 Node Optimization Based on \texttt{p.h.tree}

In case no value of the required times after buffering \( r_{i,j} \) is greater than or equal to \( r_1 \), we conclude that the required times of the sinks are spread on a narrow range. This situation is similar to a fanout problem with equal required times for the sinks. We solve the problem by constructing a complete \( p-ary \) [9] fanout tree in which all leaves have the same depth and all internal nodes have degree \( p \). Since the number of children grows exponential with the depth of the tree, the number of leaves for a tree of height \( h \) is \( p^h \). Consequently, the height of a complete \( p-ary \) tree which solves a fanout problem of size \( n \) must be less than or equal to \( \log_p n \).

**Definition 9** (\texttt{p.h.tree}) \texttt{A p.h.tree is a balanced tree of degree p, which solves a fanout problem of size n and has a height h \leq \log_p n.}

The number of buffers required by such a tree is

\[
    p + p^2 + \cdots + p^h = \sum_{i=1}^{h} p^i = \frac{p(p^h - 1)}{p - 1}
\]

which grows very rapidly with \( p \) and \( h \).

It has already been shown [59] that the size of a fanout problem is typical in the range of 10 to 20, only rarely above 50. In conclusion, for a usual fanout problem a \texttt{p.h.tree} of height 1 is the best solution, which in addition assures a good tradeoff area/delay. In this particular case both, CMOS and BiCMOS, types are allowed for source template.

The optimum degree \( p \) of the \texttt{p.h.tree} of height 1, calculated for minimum delay, is obtained with the formula

\[
    p_{opt}(b) = \left( \frac{\alpha_b \beta_b \gamma_{l_{1,n}}}{\alpha_s \beta_s \gamma_{l_{s,n}}} \right)^{\frac{1}{\alpha_s + \alpha_b}}
\]
and the value of \( p_{\text{opt}}(b) \) is then rounded to the nearest integer; \( b \) stands for a buffer in the library, and \( s \) for the source template. We choose the buffer which assures the minimum delay of the tree.

For bigger size fanout problems the minimum delay \((p, h)\) pair is determined. The source and the internal nodes of the fanout tree are chosen to be of CMOS type since they will not be very heavily loaded, while the leaves of the tree have the freedom to be of both CMOS or BiCMOS type. We need this restriction to simplify the complexity of the problem and to be able to determine the optimum solution. Allowing BiCMOS buffers to drive internal nodes will lead to a mathematical equation with one unknown which does not have an analytical solution. Besides, this restrictions seems very reasonable and realistic.

Thus, the optimum degree \( p \) of the \( p.h.\text{tree} \) as a function of the tree height \( h \), which assures the minimum delay from the source to the sinks is

\[
p_{\text{opt}}(h) = \left( \frac{\alpha_{b_2} \beta_{b_2} \gamma_{b_1} \gamma_{b_2} h}{\beta_{b_1} \gamma_{b_1} + \beta_{b_1} \gamma_{b_1} (h - 2) + \beta_{b_1} \gamma_{b_2}} \right)^{\frac{1}{\alpha_{b_2} h + 1}} \quad \text{with} \quad 2 \leq h \leq \log_p n
\]

where, \( b_1 \) denotes the CMOS buffer corresponding to the internal nodes of the tree, and \( b_2 \) the buffer assigned to the leaves of the tree.

The height \( h \) of the tree is incremented each step and the value which gives the minimum delay is taken. In this calculation we suppose that the same load is assigned to each buffer at the leaves. This is not always true since the sinks do not have the same loads.

Finally, the sinks must be assigned to the buffer leaves. The final assignment of the sinks to the buffers is a very important step in buffering process. The delay reduction is strongly dependent upon this step. Therefore, choosing the right set of buffers to drive the sinks is similar with finding the optimum distribution of the sinks among the buffers, by means of the distribution which will lead to a minimum delay \( p.h.\text{tree} \).

### 7.3 Sink Assignment

The sinks do not have equal required times and, moreover, they do not have equal loads. Consequently, a distribution of an equal number of sinks to each buffer will not be a good solution. We propose here a greedy algorithm which assigns \( n \) sinks to \( p \) identical buffers in such a way that the minimum delay from the source to the sinks is achieved.

A simplified version of this algorithm is given in Figure 7.3. The sinks are sorted in order of decreasing required times. The algorithm starts by distributing the
sinks consecutively to the buffers in such a way that for each buffer an equal load is assigned. After assignment process is finished the consecutive sinks \( (s_n, \ldots, s_{j_1}) \) are linked to the buffer \( b_p \), then the consecutive sinks \( (s_{j_1-1}, \ldots, s_{j_p-1}) \) are linked to the buffer \( b_{p-1} \) and so on. In other words, to consecutive buffers are assigned consecutive subsets of sinks. If the sinks have equal loads then it is very probable that the required times at the buffers will satisfy the relation:

\[
\text{required}(b_p) \geq \text{required}(b_{p-1}) \geq \cdots \geq \text{required}(b_1)
\]

This is possible because the sinks are ordered with decreasing required times and assigned in consecutive order to the consecutive buffers. However, this distribution may not be optimal. Our objective is to maximize the minimum required time at the buffers by means of maximizing \( \text{required}(b_1) \). To reach this goal we need to redistribute the sinks among the buffers. The sink ordering does not change, they are only shifted from one buffer to its successor or predecessor taking into account some cost functions.

There are two cases to be distinguished:

1. For each two consecutive buffers \( b_j, b_{j-1}, j = p, \ldots, 1 \), if \( \text{required}(b_j) < \text{required}(b_{j-1}) \) then the load assigned to the buffer \( b_j \) is too high. Such a situation can arise if the loads of the sinks are spread on a large area. In this case we need to move sinks from \( b_j \) to \( b_{j-1} \) until the balance between the two required times of the buffers changes (Figure 7.3-Case1).

2. For each two consecutive buffers \( b_j, b_{j-1}, j = p, \ldots, 1 \), if \( \text{required}(b_j) \gg \text{required}(b_{j-1}) \) then it is still possible to improve the minimum buffer required time by moving the sinks from \( b_{j-1} \) to \( b_j \) as long as \( \text{required}(b_j) \) is still bigger than \( \text{required}(b_{j-1}) \) (Figure 7.3-Case2)

Depending on the relationship between the required times of the two consecutive buffers the sinks can be redistributed over the buffers following the algorithms presented in Case 1 or Case 2. This redistribution can be iteratively repeated until no further improvement is noticed.

The assignment of the sinks to the buffers is a key step in fanout optimization process. An inappropriate sink distribution can lead to poor optimization results. It is not always possible to find the optimum sink distribution. The algorithm is limited by restricting the sinks to be ordered with decreasing required time. Allowing different sink orders, however, will increase too much the complexity of the algorithm and therefore the CPU time. Our algorithm is fast and, usually, does not need more than two iterations to complete the job. The algorithm runs in linear time.
**Algorithm sink.assignment**

**Input:** sinks $S = (s_n, ..., s_1)$ with $(r_n, ..., r_1)$ and $(\gamma_n, ..., \gamma_1)$ to be assigned to the buffers $(b_p, ..., b_1)$

let $\text{load}_b = \frac{n \cdot \alpha}{p} \; i = n$

**Foreach** $b_j \in (b_p, ..., b_1), j = p, ..., 1$

- $\text{load}(b_j) = 0$
  - **While** $\text{load}(b_j) < \text{load}_b$ and $i \geq 1$
    - *assign* $(s_i)$ to $b_j$
    - $\text{load}(b_j) = \text{load}(b_j) + \gamma_i$
    - $i = i - 1$

**Next** buffer $b_j$ with $j = j - 1$

**Assign** all the remaining sinks to $b_1$

/*to get better performance redistribute the sinks to the buffers*/

/*Case1: when \textbf{required}(b_j) < \textbf{required}(b_{j-1})*/

**Foreach** $b_j \in (b_p, ..., b_2), j = p, ..., 2$

- **While** \textbf{required}(b_j) < \textbf{required}(b_{j-1})
  - move the last sink of $b_j$ to $b_{j-1}$
  - recalculate \textbf{required}(b_j) and \textbf{required}(b_{j-1})

**Next** buffer $b_j$ with $j = j - 1$

/*Case2: when \textbf{required}(b_j) \gg \textbf{required}(b_{j-1})*/

**Foreach** $b_j \in (b_p, ..., b_2), j = p, ..., 2$

- let $s_i, i \in (n, ..., 1)$ be the first sink of buffer $b_{j-1}$
- **While** \textbf{required}(b_j) - \textbf{required}(b_{j-1}) \geq
  - $r_{i+1} - r_i + 2\beta_h \gamma_i$
  - move the first sink of $b_{j-1}, s_i$ to $b_j$
  - recalculate \textbf{required}(b_j) and \textbf{required}(b_{j-1})
  - $i = i - 1$

**Next** buffer $b_j$ with $j = j - 1$

**Output:** the sink distribution to the buffers

**End sink.assignment**

Figure 7.3. Sinks assignment to the buffers.
7.4 Node Optimization Based on trace_tree

In case the required times of the sinks are spread on a wider area, i.e. at least one required time after buffering is greater than or equal to \( n_1 \), a variable degree \( p \) fanout tree is constructed.

For a better understanding of our algorithm we need to introduce some new notions.

**Definition 10 (trace_tree)** The space of all possible solutions for buffering the source is covered by a trace_tree system \( (M, m_0, f) \), where \( M \) is the set of nodes, \( m_0 \in M \) is the root of the tree, and \( f \) is a function from the set \( S = \{(s_k, s_{k-1}, \ldots, s_p), \forall 1 \leq p < k \leq n\} \) to the set \( M \); \( M(i, j) \) denotes a subset of \( M \) which covers the subsets \( \{(s_i, \ldots, s_j)\} \).

In fact, \( M(i, j) \) is a subtree of the trace_tree, sometimes denoted trace_tree\((i, j)\).

**Definition 11 (trace_tree node)** A trace_tree node, denoted node\((i,j)\), is a node in the trace_tree obtained by merging the sinks \( \{(s_i, \ldots, s_j)\} \) together; \( (i, j) \) is called the identity number of the node.

**Definition 12 (trace_tree path)** A trace_tree path, denoted path\((n-p,n-p-i,\ldots)\), is a possible partial solution for the fanout problem; it has the property that all the nodes laying on it cover consecutive subsets of sinks.

Each path represents a possible merging variant for the sinks it covers. The path coefficients \( n - p, n - p - i, \ldots \) are the last element indexes of the node identity numbers.

The nodes in the trace_tree are potentially internal nodes of the final fanout tree. For every fanout problem of size \( n \) a trace_tree with maximum \( (n-1) \) subtrees can be constructed. Figure 7.4 shows the schematic of a trace_tree subtree \( M(n, n-p) \). The fields of the internal nodes are also shown in the figure but they will be defined later on in this chapter.

The trace_tree covers the total number of fanout trees \( C_n \) for an optimization problem of size \( n \). It can be calculated with the recursive formula:

\[
C_n = C_{n-1} + C_{n-2} \quad \text{with} \quad n \geq 2 \quad \text{and} \quad C_2 = 1, C_3 = 2
\]

\( C_n \) is in the order of Fibonacci numbers \( F(n) = \phi^n / \sqrt{5} \) rounded to the nearest integer, with \( \phi = (1 + \sqrt{5}) / 2 = 1.61803\ldots \).

A cost function evaluates what fraction of this space is searched. It is almost never the case that all this space is searched. Besides, the size of a fanout problem is
Figure 7.4. A complete subtree \( \text{trace.tree}(n, n - p) \).

typical 10 to 20, rarely above 50. For extreme cases, the search space is first partitioned in smaller subproblems (for example with parallel repowering) and then recursively solved. In any case the space we search is considerably smaller than the total number of rooted trees with \( n \) leaves, which is of the order of the Catalan numbers \( C(n) = \frac{1}{n+1} \binom{2n}{n} \).

The fanout problem to be solved has size \( n \). The source is driving the sinks \( (s_n, \ldots, s_1) \), which have the required times \( (r_{n}, \ldots, r_{1}) \) and the input loads \( (\gamma_{n}, \ldots, \gamma_{1}) \). The minimum required time of the problem is initially \( r_{\text{min}} = r_{1} \). The source delay parameters are \( (\tau_{s}, \beta_{s}, \alpha_{s}) \) and its initial required time is \( \tau_{s} \).

The information over the best buffers \( b_{i,j}, n \geq i, j \geq 1 \), to drive all the consecutive subsets in \( S = \{(s_{k}, s_{k-1}, \ldots, s_{k-p}), \forall 1 \leq p < k \leq n \} \) is available. They have been determined previously (see Figure 7.2). All the required times \( r_{i,j} \) of the buffers together with the associated rough gains \( r_{g_{i,j}} \) have also been calculated.

It is not necessary to build the whole \( \text{trace.tree} \) immediately. The subtrees of the \( \text{trace.tree} \) can be built one by one, then solved for the best solution. Since their results are independent of one another, the global best solution is chosen by comparing the partial solutions among them and picking the best. This strategy will also reduce the complexity of the program.

To build the subtree \( \text{trace.tree}(n, n - p) \) we start with creating the subtree node \( m(n, n - p) \). The sinks \( (s_n, \ldots, s_{n-p}) \) are merged to the buffer \( b_{n,n-p} \) which has the required time \( r_{n,n-p} \). Its identity number is \( (n, n - p) \).

**Definition 13 (Dynamic Index)** The dynamic index \( k_{\text{min}}, (n - 1 \geq k_{\text{min}} \geq 0) \) is the minimum argument in the set of sinks \( S = (s_n, \ldots, s_1) \) such as \( r_{i,j} \geq r_{k_{\text{min}}} \); \( r_{i,j}, n \geq i, j \geq 1 \), is any required time of the buffers.
A set of dynamic indexes is generated during the optimization process corresponding to local fanout problems solved at a given moment. If there are no sink required times to satisfy the inequality, then the dynamic index becomes 0.

The size of the current fanout problem is dynamically adjusted, i.e. if the required time \( r_{n,n-p} \) is greater than or equal to \( r_1 \), then the size is reduced from \( n \) to \( n - k_{\min} + 1 \); if \( r_{n,n-p} \) is less than \( r_1 \) then the size of the problem stays \( n \) but a local minimum required time \( r_{\min} = r_{n,n-p} \) is generated. The dynamic problem adjustment works for every new node inserted into the subtree. The information concerning the dynamic index \( k_{\min} \) and the minimum required time \( r_{\min} \) of the local fanout problem is stored in every node of the tree.

**Definition 14 (Net Gain)** The net gain of a node \((i,j)\) in the tree, denoted \( g_{i,j} \), is the source delay improvement due to merging the sinks \((s_i, ..., s_j)\) is a common node driven by the buffer \( b_{i,j} \).

The net gain takes into account the buffer delay itself. Its value depends on the position of the required time \( r_{i,j} \) relative to the local minimum required time of the parent node \( (r_{\min-parent}) \).

In Figure 7.4 a schematic of a \( \text{trace.tree}(n, n-p) \) is given together with the fields of its nodes. A recursive procedure (Figure 7.5) inserts each node \((i,j)\) and evaluates its states as follows:

\[
\begin{align*}
    r_{\min} &= \min(r_{i,j}, r_{\min-parent}) \\
    k_{i,j} &= \min(k, 1 \leq k \leq n, r_k \geq r_{i,j}) \\
    k_{\min} &= \max(k_{i,j}, k_{\min-parent}) \\
    g_{i,j} &= \begin{cases} 
    r_{g_{i,j}} & \text{if } r_{i,j} \geq r_{\min-parent} \\
    r_{i,j} - r_{\min-parent} + r_{g_{i,j}} & \text{otherwise}
    \end{cases}
\end{align*}
\]

Whether a node is inserted into the subtree depends on its gain. If the gain value is greater than or equal to zero then a new node is inserted into the \( \text{trace.tree}(n, n-p) \). If the gain value is negative then the searching process in the current direction stops and all following subsets \( \{(s_{j_1}, ..., s_{j_q}) \mid \forall 2 \leq q < j\} \) are abandoned. The insertion process continues until all the subsets \( \{(s_{n-p-1}, ..., s_{n-p-q}) \mid \forall 2 \leq q < n-p\} \) are evaluated. In addition, each node contains an identity number \( id \), and the fields \( child \) and \( parent \) corresponding to its children and its parent respectively.

After the insert operation is finished a \( \text{trace.tree}(n, n-p) \) in which every path represents a possible merging variant of the sinks is obtained. By choosing a certain path a new fanout problem for the next step is generated.

**Definition 15 (Expectation)** We define the expectation of a path \((n-p, i, j, \ldots)\), denoted \( path.expectation(n-p, i, j, \ldots) \), the minimum expected improvement in source delay due to the new created fanout problem.
algorithm trace_tree_insert(i, j, parent)
    input: the degree (i, j) of the subtree to be built; the parent node parent;
    sinks $S = (s_n, ..., s_1)$ with $(r_n, ..., r_1)$;
    optimum buffer $b_{i,j}$, its required time $r_{i,j}$, and its rough gain $r g_{i,j}$;

    if $j < parent \to k_{min}$ return
    if $r_{i,j} \geq parent \to r_{min}$ {
        gain$_{i,j} = r g_{i,j}$
        $r_{min} = parent \to r_{min}$
        $k_{i,j} = \min (k, 1 \leq k \leq n, r_k \geq r_{i,j})$
        $k_{min} = \max (k_{i,j}, parent \to k_{min})$
    }
    else {
        gain$_{i,j} = r_{i,j} - parent \to r_{min} + r g_{i,j}$
        $r_{min} = r_{i,j}$
        $k_{min} = parent \to k_{min}$
    }
    if gain$_{i,j} \geq 0$ {
        create child = node(i, j)
        with
        child $\to$ id = (i, j)
        child $\to$ $r_{min} = r_{min}$
        child $\to$ $k_{min} = k_{min}$
        child $\to$ gain = gain$_{i,j}$
        child $\to$ parent = parent
        parent $\to$ addChild(child)
        $k = j - 1$
        $p = j - 1$
        while $(p = p - 1) \geq 2$
            trace_tree_insert(k, p, child)
    }
    return
output: the subtree $M(i, j)$
end trace_tree_insert

Figure 7.5. trace_tree insertion algorithm.
algorithm trace_tree
  input source s; sinks S = (s_2, ..., s_l);
  optimum buffers b_i,j and their required times r_{i,j};
  create root m_0 with m_0 \rightarrow r_{min} = r_1 and m_0 \rightarrow k_{min} = 1
  for 1 \leq p < n {
    build trace_tree(n, n - p) with trace_tree_insert(n, n - p, m_0)
    foreach path \in trace_tree(n, n - p) {
      path_gain = \sum_{e_{i,j} \in path} gain_{i,j}
      generate a new fanout problem of size n' and r'_{min}, S' = (s'_2, ..., s'_l)
      b_{n',n'-1} = \arg\max_{b_{n',n'-1}}[r_{n'-1} - (r_b + \beta_b r_{n',n'-1}) - \beta_s r_s]
      r_{n',n'-1} = r_{n'-1} - (\tau_{n',n'-1} + \beta_{n',n'-1} \gamma_{n',n'-1} \alpha_{n',n'-1})
      \gamma_{n',n'-1} = \beta_s (\gamma_{n',n'-1} - \gamma_{n',n'-1})
      path_expectation = { r_{n',n'-1} \text{ if } r_{n',n'-1} \geq r'_{min}
                          r_{n',n'-1} - r'_{min} + r_{n',n'-1} \text{ otherwise}
      }
    }
    select trace_tree(n, n - p) best path e.i.
    max_{path \in trace_tree(n,n-p)} path_gain or
    max_{path \in trace_tree(n,n-p)} path_expectation or
    max_{path \in trace_tree(n,n-p)} r'_{min}(path) or
    min_{path \in trace_tree(n,n-p)} path_length or
    min_{path \in trace_tree(n,n-p)} n'(path)
  }
  select trace_tree best path
  output fanout problem partial solution and a new fanout problem
end trace_tree

Figure 7.6. trace_tree algorithm.

The trace_tree buffer insertion algorithm is presented in Figure 7.6. The expectation calculation is given in Figure 7.6, line marked 5.

The total gain per path is now evaluated and the best solution, that is the path with the maximum gain, is chosen. In case of equal solutions the priorities are: biggest expectation, biggest minimum required time, shortest path (smallest amount of buffers), minimum number of sinks transferred to the next step. All the other subtrees of the trace_tree are constructed and evaluated in a similar way. Finally, the best solution together with the new fanout problem are returned and the process is repeated from the beginning (Figure 7.2) until there are no sinks left or no additional improvement in source delay is obtained.

The order of our priorities is chosen such that the fastest tree of buffers is generated. The net gain per path and the new minimum required time assure the smallest
delay. In case of equal delay performance, the tree with minimum area is taken to keep the area increase at a minimum. However, if the requirement is to get a minimum area under delay constraints then the priority order will change. We will look for minimum area path which has a delay smaller than the one specified.

Example

For a better understanding of our fanout buffering algorithm we solve the fanout problem in Figure 7.7 step by step.

The size of the problem is \( n = 7 \). The source cell drives seven sinks \((s_7, \ldots, s_1)\) with sorted required times \((7, 7, 7, 6, 6, 5, 3)\) and input loads \((\gamma_i = 1, \forall 1 \leq i \leq 7)\) (see Figure 7.7a). The delay parameters of the source are \((\tau_s, \beta_s, \alpha_s) = (0, 1, 1)\). We have chosen these parameters for simplicity of presentation. Consequently, the initial source required time is \( r_s = -4 \). To simplify the problem, we suppose that there is only one buffer \( b \) in the library with delay parameters \((\tau_b, \beta_b, \alpha_b) = (0, 1, 1)\) and input load \( \gamma_b = 1 \).

First, the buffer required times and the rough gains are calculated using the algorithm presented in Figure 7.2. The results for our example are given in the left hand side tables of Figure 7.7b. Since at least one required time is greater than minimum sink required time \( r_{\text{min}} = r_1 = 3 \), our \textit{trace_tree} procedure is chosen to build the fanout tree.

The root node \( m_0 \) is created, with \( r_{\text{min}} = r_1 = 3 \) and \( k_{\text{min}} = 1 \) (see Figure 7.6 line marked 1). Then, as the next step, the \textit{trace_tree}(7, 6) is recursively constructed with the relations \((7.7 \div 7.10)\) as shown in section 7.4. The sinks \( s_7 \) and \( s_6 \) are merged into a new node \((7, 6)\) (Figure 7.6 line 2). This one becomes the parent of the subtree \textit{trace_tree}(7, 6). The required time of this node is \( r_{7,6} = 5 \) (shown on the edge ending in this node) and its rough gain \( r_{7,6} = 1 \). In conformity with the relations \((7.7 \div 7.10)\) the node parameters are respectively \( r_{\text{min}} = 3, k_{7,6} = 2, k_{\text{min}} = 2 \) and \( \text{gain}_{7,6} = 1 \) (see \textit{trace_tree} subtree \((7, 6)\) in Figure 7.7b, right hand side).

The sinks \( s_5 \) and \( s_4 \) are merged together to create the node \((5, 4)\), and so on. The insertion operation continues until \( k_{\text{min}} = 2 \) in sinks ordering is reached or the node gain becomes negative. For example, on the path \((6, 4, 2)\) the insertion stops before the sink \( s_4 \) because \( k_{\text{min}} = 2 \). But, on the next path \((6, 3)\), because the next node \((2, 1)\) does not bring any gain, the insertion stops before the sink \( s_2 \).

After the insertion operation for the subtree \textit{trace_tree}(7, 6) is finished the path evaluation takes place (Figure 7.6 line 3). By choosing a certain path, a new fanout problem for the next step is generated (Figure 7.6 line 4). The path expectation is calculated as given in the \textit{trace_tree} algorithm in Figure 7.6, line marked 5.

After evaluating the total gain per path (by summing the gains of all the nodes laying on the same path) the best solution, that is the path with the maximum gain, is taken. In case of equal gains the priorities are: biggest expectation, biggest minimum required time, shortest path (smallest amount of buffers), minimum number of sinks transferred to the next step (Figure 7.6 line 6).
Figure 7.7. A complete fanout optimization example with trace_tree: (a) Schematic of the fanout problem; (b) First step solution; (c) Second step solution; (d) Final solution of the fanout problem.
All the other subtrees of the \textit{trace.tree} are constructed and evaluated in a similar way. Finally, the best solution together with the new fanout problem are returned and the process is repeated from the beginning (Figure 7.2).

In our example the \textit{path}(6,3) with a gain of 3 and an expectation of 1 is the best solution of the subtree \textit{trace.tree}(7,6). In the same way all the other subtrees (\textit{trace.tree}(7,5) and \textit{trace.tree}(7,4)) are constructed and evaluated, but the \textit{path}(6,3) remains the best. This means that the sinks \((s_7, s_6)\) and \((s_5, s_4, s_3)\) are merged in two distinct nodes driven by two buffers of type \(b\). A new fanout problem of size 4, \((5,5,3,3)\) is created for the next step (Figure 7.7c).

The fanout process is repeated until no delay improvement can be obtained anymore. The final solution is shown in Figure 7.7d. The new source required time is now equal to 0, which is the optimum solution.

The same example was given in [59], where it could not be solved optimally. The algorithm proposed in [59] results in a fixed topology of the fanout tree. Therefore, the optimum topology could not be obtained. The main advantage of the work presented here lies in the flexibility of the fanout tree topology. Our method covers the fixed fanout tree topology proposed in [59] without being limited to it. We do not always guaranty the optimum solution but the chance to get is bigger.

### 7.5 Node Repowering

If the number of sinks is large, for example larger than 30, the optimization results obtained with the buffering process may not be very good. Besides, the complexity of the \textit{trace.tree} algorithm will grow unnecessarily. In this case is better to repower the node before building the fanout tree.

There are two ways to repower a source node:

**Definition 16 (Repowering)** A node is repowered if it is replaced with a cell from the library which has the same functionality, but better driving capability.

**Definition 17 (Parallel Repowering)** A source node driving a couple of sinks is parallel repowered, if the source is replaced with two or more cells from the library which have the same functionality and the sinks are split between the new cells.
The drawback of repowering is that the total input load of the source will increase causing a higher load for the cells driving it. If simple repowering is used then the load will increase because a cell with a better driving capability is usually a cell with a higher input load. If parallel repowering is used then two or more sources are used. Thus the input load will increase at least two times. The repower is only useful when the increase in the source input load is not dominant over the speed increase.

In case of parallel repowering the number \( p \) of cells to replace the source is calculated by minimizing the overall delay of the source and the cells driving it. The result is similar to the optimum number of buffers used in \( p,h,tree \) algorithm for a tree of height 1.

\[
p_{opt}(g) = \min_{g_h \in fanin(g)} \left( \frac{1}{\alpha g_h} \frac{\gamma g_h^{\alpha g_h}}{\alpha g_h^{\beta g_h} \gamma g_h^{\alpha g_h}} \right) \left( \frac{1}{\gamma g_h^{\alpha g_h}} \right)
\]

where \( g_h \) are the cells driving the node \( g \) to be repowered; \( \gamma_{1,n} \) is the total load of the node \( g \) and \( \gamma g_h \) its input load. For each cell driving the node \( g \) a different optimum \( p \) will be calculated. We decide to take its minimum value to assure no decrease in speed performance.

After determining \( p_{opt}(g) \), the sinks are assigned to the new sources following the algorithm described in section 7.3.

### 7.6 Experimental Results

We implement the algorithm in C++ as a stand-alone fanout optimization system. We use the well-known set of benchmark circuits from MCNC [32] together with the mcnc library to test the efficiency of our method.

To illustrate the advantage of using non-restricted topology over the fixed topology fanout trees, the same fanout problem is solved with our algorithm as the one used in sis [51]. In this particular example the source drives 17 sinks. The initial required time at the source is -31.04 ns. The optimization result is presented in Figure 7.8. The root of the tree represents the source, while the leaves are the sinks of the initial fanout problem. For each node in the tree the gate type is given together with its required time. After the fanout buffering process is performed our \( trace_tree \) algorithm leads to a required time of -28.14 ns (Figure 7.8), which is a better solution than -28.34 ns obtained with sis optimization procedure (Figure 7.9).

We also test our method on some benchmark circuits. General information on these benchmark circuits is provided in Table 7.1. We use the common term logic in case the logic function implemented by the circuit was not known. The circuits are first down-sized for minimum area and the circuit delay and area are measured. Then
we apply our fanout optimization algorithm and perform the same measurements again. The results are stated in Table 7.1. For comparison we also perform a parallel optimization for the same set of benchmark circuits using the SIS synthesis system [51]. The buffer_opt procedure from SIS is used to optimize the mapped circuit. This optimization procedure is based on the works presented in [59, 57]. The results are also reported in Table 7.1.

Unfortunately, we could not interface the optimization procedure used in SIS with our method. A comparison of both results is difficult to make because we could not test our fanout optimization algorithm under the same conditions. It is very likely that the nodes are not called in the same order and, moreover, not on the same data.

However, even though an entirely fair comparison is not made, the experimental results obtained with our fanout optimization algorithm are very promising. An overall average delay reduction of 30% is obtained for the original minimum area version, while an average reduction of 4% is obtained over the SIS delay optimization procedure.

The increase in circuit area is not relevant in analyzing the performance of our fanout buffering algorithm since, in this case, the size of the cutset is very important.
The way in which the parameter $\varepsilon$ is taken can determine a larger of smaller area for the same delay performance.

The *mcnc* library we use in testing our method is CMOS oriented. We could not find a similar BiCMOS library in the available benchmark sets. To solve this problem we built ourself a BiCMOS like library and used it in optimization process. The optimization results show that in case of random logic, in which the load of the nodes can be split among the buffers, BiCMOS gates are very rarely used. Not surprisingly they only become effective for high indivisible capacitive loads.
<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>Circuit Function</th>
<th>In Nr.</th>
<th>Out Nr.</th>
<th>min area gates</th>
<th>delay</th>
<th>fanout opt delay</th>
<th>red %</th>
<th>SIS opt delay</th>
<th>red %</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1355</td>
<td>error correcting</td>
<td>41</td>
<td>32</td>
<td>544</td>
<td>53.20</td>
<td>48.19</td>
<td>9.42</td>
<td>48.70</td>
<td>8.45</td>
</tr>
<tr>
<td>C1908</td>
<td>error correcting</td>
<td>33</td>
<td>25</td>
<td>371</td>
<td>61.90</td>
<td>54.19</td>
<td>12.45</td>
<td>55.00</td>
<td>11.15</td>
</tr>
<tr>
<td>C3540</td>
<td>ALU and control</td>
<td>50</td>
<td>22</td>
<td>798</td>
<td>84.50</td>
<td>74.49</td>
<td>11.85</td>
<td>74.70</td>
<td>11.59</td>
</tr>
<tr>
<td>C499</td>
<td>error correcting</td>
<td>41</td>
<td>32</td>
<td>240</td>
<td>50.10</td>
<td>45.19</td>
<td>9.80</td>
<td>45.70</td>
<td>8.78</td>
</tr>
<tr>
<td>apex6</td>
<td>logic</td>
<td>135</td>
<td>99</td>
<td>485</td>
<td>27.00</td>
<td>26.30</td>
<td>2.60</td>
<td>26.00</td>
<td>3.70</td>
</tr>
<tr>
<td>dalu</td>
<td>dedicated ALU</td>
<td>75</td>
<td>16</td>
<td>1181</td>
<td>92.80</td>
<td>48.78</td>
<td>47.44</td>
<td>49.40</td>
<td>46.76</td>
</tr>
<tr>
<td>des</td>
<td>data encryption</td>
<td>256</td>
<td>245</td>
<td>2612</td>
<td>131.10</td>
<td>34.2</td>
<td>73.91</td>
<td>34.10</td>
<td>73.98</td>
</tr>
<tr>
<td>frg2</td>
<td>logic</td>
<td>143</td>
<td>139</td>
<td>2344</td>
<td>56.00</td>
<td>28.40</td>
<td>49.30</td>
<td>25.40</td>
<td>54.64</td>
</tr>
<tr>
<td>i10</td>
<td>logic</td>
<td>257</td>
<td>224</td>
<td>1733</td>
<td>101.40</td>
<td>76.80</td>
<td>24.26</td>
<td>77.20</td>
<td>23.86</td>
</tr>
<tr>
<td>k2</td>
<td>logic</td>
<td>45</td>
<td>45</td>
<td>1181</td>
<td>68.90</td>
<td>30.55</td>
<td>55.66</td>
<td>32.26</td>
<td>53.17</td>
</tr>
<tr>
<td>pair</td>
<td>logic</td>
<td>173</td>
<td>137</td>
<td>1058</td>
<td>56.00</td>
<td>40.49</td>
<td>27.70</td>
<td>41.60</td>
<td>25.71</td>
</tr>
<tr>
<td>rot</td>
<td>logic</td>
<td>135</td>
<td>107</td>
<td>451</td>
<td>54.20</td>
<td>42.89</td>
<td>20.86</td>
<td>43.50</td>
<td>19.74</td>
</tr>
<tr>
<td>vda</td>
<td>logic</td>
<td>17</td>
<td>39</td>
<td>584</td>
<td>40.00</td>
<td>22.68</td>
<td>43.30</td>
<td>23.00</td>
<td>42.50</td>
</tr>
<tr>
<td>x3</td>
<td>logic</td>
<td>135</td>
<td>99</td>
<td>629</td>
<td>30.60</td>
<td>29.30</td>
<td>4.25</td>
<td>29.10</td>
<td>5.22</td>
</tr>
</tbody>
</table>

**Table 7.1.** Fanout optimization experimental results.

### 7.7 Conclusion

In this chapter we presented an efficient speed optimization algorithm for BiCMOS circuits. Our approach uses a novel and very accurate delay model applicable for both CMOS and BiCMOS gates which improves the performance of the algorithm. The network and the single node optimization algorithms are mixed together to get better results. The algorithm builds hierarchically a fanout buffer tree by partitioning the fanout problem into local subproblems and recursively solving them. No restrictions on the fanout tree topology are imposed. The space of all possible solutions for a given fanout problem is implemented as a rooted tree which is dynamically constructed and searched. Near optimum fanout trees are obtained.
Chapter 8

MACRO-CELL LIBRARY DEVELOPMENT

At the architectural level the VLSI designs are seen as large dedicated blocks, such as ALU's, RAM's, ROM's etc, and random logic used to glue these parts together. The random logic is generally specified by logic functions which will be implemented by logic synthesis tools using a specified standard cell library.

The dedicated blocks are taken from a macro-cell library or are generated with regular array generator from user's input description. These dedicated blocks can incorporate parts which are regular or repetitive in nature. In particular some complex logic functions can be implemented as linear iterative networks. These networks have a regular structure. They are repetitive in space domain, while performing a sequential function in time domain. For example, a carry ripple adder is such an iterative network.

One of the main challenge in the development of ASIC libraries is to provide cells with high speed capability while area and power dissipation is kept at a minimum. This requirement is valid for macro-cell libraries as well.

In this chapter we present the design methodology of a macro-cell library suitable for implementing iterative networks. These kind of networks lend themselves well for implementation in differential logic. We use bipolar transistors in the time critical path of the complex circuits for speeding up signal propagation and MOS transistors in the remaining parts to allow reduced area and supply voltage.

This chapter is organized as follows: first the iterative network concept is introduced followed by differential logic basics. After that our macro-cell design is explained. An idea for an regular array generator is briefly sketched. Finally, two macro-cell design examples are given.

8.1 Iterative Networks

An iterative network is an one-dimensional array of $m$ identical modules. A module consists of one or more internal cells. Each module is connected only to its neighbors. The overall circuit is strictly combinational in nature, and there are no feedback paths (Figure 8.1).
The network has both external/internal inputs/outputs. They are either vectors or scalars. In our notation vectors are underlined. In certain applications the external outputs $z^i (i = 0, \ldots, m - 1)$ are not required and only the last module is specialized in providing outputs.

![Diagram](image)

**Figure 8.1. A linear iterative network.**

The $c^i$ inputs/outputs are internal connections between neighboring modules. The dimension of $c^i$ vector is dependent on the specific application. It is at least a scalar. The rightmost module receives an externally generated, fixed $c^0$ input.

When the $i^{th}$ module implements the functions $G$ and $F$ for both internal and external outputs we can write:

\[(8.1) \quad c^{i+1} = G(z^i, c^i) \quad i = 0, \ldots, m - 1\]

and

\[z^i = F(x^i, z^i) = F(x^i, G(z^{i-1}, G(\ldots, G(z^0, z^0), \ldots)))\]

This shows that the external/internal outputs of module $i$ depend only on the inputs of the modules $0 \leq j \leq i$ via internal connections $c^j$.

Functions with a large number of variables, such as adders, are sometimes implemented as iterative networks. Due to propagation delay through the array the network might become too slow when the variable number is large.

Ercegovac and Lang [6] describe the module delay by two parameters: the output-delay $\Delta_o$, which is the time between input stabilization (external and internal) and external output stabilization; and the internal-delay $\Delta_c$, which is the time between input stabilization and internal output stabilization. For a network with $m$ modules the delay $\Delta$ is:

\[(8.2) \quad \Delta = (m - 1)\Delta_c + \max(\Delta_o, \Delta_c)\]

As is seen from the relation (8.2), the most significant factors in the network delay are the internal-delay and the number of modules. To reduce the network delay these factors have to be minimized.

We can decrease the number of modules $m$ by increasing the number of external inputs $k$ per module and by using skip technique in the path of internal variables. An example of such a solution is the well known carry look-ahead architecture for a full-adder. However, implemented in conventional static logic this design is still
not fast enough. Besides, for a given technology the internal-delay is fixed by the
delay of the basic static gates and it can not be reduced.

Another possibility is to reduce the internal-delay by using circuit techniques other
than the conventional ones. A successful implementation is the differential logic
design presented in the following section.

8.2 Differential Logic

A differential CMOS logic family called Cascode Voltage Switch Logic (CVSL),
capable of processing complex logic functions within a single circuit delay was
proposed for the first time in 1984 by Heller and Griffin [27]. Short circuit delay,
high layout density, low power dissipation and logic flexibility are among the ad-
vantages of this technique over conventional NAND/NOR logic [8].

Figure 8.2 shows the basic CVSL circuit. The logic design implementation is real-
ized in CVSL block by cascoding differential pairs of MOS devices. If the height
of the resulted logic tree is equal \( n \), then Boolean functions with up to \( 2^n - 1 \) input
variables can be processed at a time. A pair of PMOS transistors are used in the
pull-up stage as load in a latch configuration.

![Figure 8.2. The first differential Cascode Voltage Switch Logic.](image)

The operating principle is as follows: depending on the differential inputs, either
the output \( Q \) or \( \overline{Q} \) is pulled down by the NMOS combinational logic tree network.
Automatically, the output which is not pulled down, will be pulled up by the PMOS
latch. Both the non-inverted and inverted input signals are required, but in return
for that both polarities for the output logic function are available.

As we mentioned before there are a couple of major advantages in using differen-
tial logic implementation. Besides these, the input gate capacitance is typically 3
times smaller than in CMOS circuits since only the NMOS network is used for implementing the logic function. The number of circuit delays are reduced in comparison with the conventional logic, and last but not least, the logic tree network can be built automatically using logic minimization algorithms.

A variation of cascoded logic is the differential split-level logic, DSL [43]. The basic building block for DSL is shown in Figure 8.3. The circuit concept is basically the same as in cascoded logic, with exception of the output voltage swing which is lowered through NMOS transistor pair. A reference voltage \( V_{\text{ref}} \) driving the NMOS transistors is controlling the voltage swing. In this way even faster circuits can be obtained. DSL has been used in high-speed CMOS complex gate applications, and has been shown [8] to be the fastest static CMOS logic so far developed.

![Diagram](image)

**Figure 8.3.** Reduce voltage swing differential logic implementation.

Later, Lu and Ercegovac [33, 34] improved the delay of the CVSL logic by introducing the enable/disable CMOS Differential Logic (ECDL). They presented an extension to this logic particularly suited for iterative networks. But when the power supply is low, relative to the threshold voltage, a large NMOS transistor chain is discharged sequentially and therefore the discharging time is significantly increased [53]. Besides, the smaller gate-source voltage and the body effect require large sizes for the top transistors. Consequently, structures with a large NMOS stacking height cannot be optimal.

### 8.3 Macro-cell Circuit Design

In this work we propose a new way to reduce the delay of certain iterative networks. The internal-delay \( \Delta_c \) is minimized by lowering the voltage swing on the internal connections. Therefore we implement the time critical path with bipolar
transistors. We use differential cascode logic and skip technique but with a lower number of stacking levels. Our methodology is practical only under the following restricted condition:

\[
G(x^i, c^j) \quad i = 0, \ldots, m - 1, \quad j = 0, \ldots, q - 1
\]

That means, the \( q \) internal outputs are propagating parallel through the network without interfering one with the others, whereas in the relation (8.1) \( c^{i+1} \) could depend on each internal variable. Although it is a considerable restriction still many iterative networks can be designed in a suitable form for this implementation.

In Figure 8.4 the general structure for an iterative network module implemented with Bipolar/CMOS differential cascode logic is shown together with an internal cell schematic indicating the operating principle.

**Figure 8.4.** A general Bipolar/CMOS differential network tree. (a) Iterative network module. (b) Operating principle.

The cell consists of a resistive load, a differential MOS network which accepts the differential external inputs \( x^i \) and a bipolar core for internal connections. The MOS core has \( k \) stacking levels. It is possible to use both NMOS and PMOS transistors inside the core. The external inputs are driven directly by a CMOS voltage swing. The top level \( k+1 \) of the differential network tree is occupied by bipolar transistors.

The internal cell is designed such that for the "true input vector" \((x^i, c^j)\) of the switching function \(G(x^i, c^j)\), \(c^{j+1}\) is disconnected from \(V_S\) and node \(\overline{c}^{j+1}\) is connected to \(V_S\) by a unique conducting path through the tree. When \((x^i, c^j)\) is the "false vector" of \(G(x^i, c^j)\) the reverse is true.

The current and the resistor values adjust the output voltage swing. The voltage drop across the resistive load is kept small because the deep saturation of the bipolar transistors must be avoided. If an output MOS voltage swing is needed a
CMOS level converter with built-in logic capability can be used [7]. The differential network tree can be operated at both, positive and negative, supply voltage depending on the specific application.

Heimsch et al. [26] used a similar technique for implementing a BiCMOS ripple adder. They introduced a Merged CMOS/Bipolar Current Switch gate in analogy to an ECL gate. Their circuit technique allows a reduced supply down to 3.3 V.

In our approach we remove the current source which supplies the structure with a constant current. The maximum current flowing through the resistive load is now determined by the MOS transistor sizes. At reduced supply voltage the MOS transistor sizes have to be larger to maintain the operating current and, consequently, the capacitive loads on the internal nodes are higher. These are constraints which need to be taken into account when the macro-cell is designed.

### 8.3.1 Switching Circuitry

Figure 8.5a presents the pull-down circuit configuration for an internal cell. The bipolar and MOS transistors represent the pull-down stage. All MOS transistors in the chain are in the conducting state. The bipolar transistor operates in the forward active mode with $V_{\text{base}} = V_C$.

We can deduce the voltage drop over the MOS and bipolar core as follows:

$$V_{\text{MOS,core}} = V_C - V_S - V_{\text{be,on}}$$

and

$$(8.4) \quad V_{ce} = V_{\text{be,on}} - \Delta V_{\text{output}} \geq V_{\text{saturation}}$$

where $\Delta V_{\text{output}}$ is the output voltage swing. The high and low output logic levels are $V_C$ and $V_C - \Delta V_{\text{output}}$ respectively.

The voltage drop over the MOS transistor core must be high enough to assure a large operating current for small transistor sizes but in the same time it must be small for reducing supply voltage.

In some special cases the Bipolar/C莫斯 differential internal cell can present a direct pull-down path only via the MOS transistors (Figure 8.5c). The MOS transistors from the first $k-1$ stacked levels are shared between these two pull-down paths.

The sizes of the top transistors $M_k$ and $N_k$ are adjusted to obtain the same output voltage swing. The same operating current must be preserved. Since the voltage drop over the MOS core in the latter case is higher, the transistor $N_k$ must have a smaller size than the transistor on the same level $M_k$. The selection between these two pull-down paths is a function of the external inputs $x$.

The pull-up stage is represented by a passive resistor $R_C$ in the collector of the bipolar transistor. Using an active pull-up transistor does not improve the switching time since the pull-down time is dominant.
8.3 Macro-cell Circuit Design

![Pull-down circuit configuration](image)

Figure 8.5. Pull-down circuit configuration. (a), (b) Pull-down path via bipolar transistor. (c) Direct pull-down path through MOS transistors.

8.3.2 Temperature Dependency

The boundary of the saturation region and the reverse saturation current of the bipolar transistor rise with increasing temperature. It is reasonable to expect that the temperature variation will result in different output voltage swings. However, for proper operation, the condition imposed in equation (8.4) must be valid even at the highest operating temperature.

Merging the discrete MOS transistors (Figure 8.5a) one by one from the first to the $k^{th}$ level, after the model proposed by Shoji [53], leads into a single long-channel NMOS transistor shown in Figure 8.5b. Applying the simplified quadratic model for a nonuniform NMOS transistor results in:

\[
I(M_1, M_2, \ldots, M_k) = \frac{1}{\mu C_{ox} (V_{gs} - V_{th} - V_{ds}/2) V_{ds}} \sum_{i=1}^{k} \frac{L_{M_i}}{W_{M_i}}
\]

where \(V_{ds} = V_C - V_S - V_{be, on}\)

\(I(M_1, M_2, \ldots, M_k)\) is the current of the composite NMOS transistor constructed by joining \(M_1, M_2, \ldots, M_k\).

Although not very accurate, equation (8.5) is very useful for our proposal and does not affect the generality of the problem. Consequently, we can extract the equivalent non-linear resistor of the composite NMOS transistor as a function of temperature in the form:

\[
R_{equiv}(T) = \frac{1}{\mu(T) C_{ox} (V_{gs} - V_{th}(T) - V_{ds}(T)/2)} \sum_{i=1}^{k} \frac{L_{M_i}}{W_{M_i}}
\]
One of the main temperature-dependent parameters in equation (8.6) is the surface mobility $\mu(T)$, which is known to decrease with temperature following a $T^{-1.5}$ dependency [58]. Thus a higher temperature tends to enlarge the equivalent non-linear resistor through $\mu(T)$. The others temperature-dependent parameters are the threshold voltage $V_{th}(T)$ and the base-emitter voltage $V_{be, on}(T)$.

The threshold voltage was found to exhibit an almost straight-line decrease with the temperature in the range $25^\circ C$ to $300^\circ C$ [54] so that the decreasing rate can be modeled by $-k_1 \Delta T$. The constant $k_1$ is dependent on the technological parameters such as substrate doping concentration and oxide thickness. For the technological process, which we use in our simulations, it has a typical value of $0.3 \ mV/\circ C$ [11].

The base-emitter voltage of the bipolar transistor decreases with a rate $k_2$ of approximately $1.45 \ mV/\circ C$. After some algebra the $R_{equiv}(T)$ results in:

\begin{equation}
R_{equiv}(T) = \frac{1}{\mu(T)C_{ox}\{V - (k_2/2 - k_1)\Delta T\}} \sum_{i=1}^{k} \frac{L_{M_i}}{W_{M_i}}
\end{equation}

with $V = V_{gs} - V_{th,T=300^\circ C} - (V_C - V_S - V_{be, on,T=300^\circ C})/2$

Since $k_2/2 > k_1$ we can conclude that the equivalent non-linear resistor increases with the temperature.

At this moment we can calculate the output voltage swing of the circuit in Figure 8.5b as follows:

\begin{equation}
\Delta V_{output}(T) \simeq \frac{R_C}{R_{equiv}(T)}\{V_C - V_S - V_{be, on}(T)\}
\end{equation}

The voltage swing depends on the temperature through base-emitter voltage $V_{be, on}(T)$ and through equivalent non-linear resistor $R_{equiv}(T)$. With a constant resistor in the emitter of the bipolar transistor the output voltage swing would rise. Therefore the condition (8.4) to avoid deep saturation becomes more critical.

The advantage of our methodology is a temperature-dependent resistor in the emitter of the bipolar transistor which rises with temperature and consequently, the output voltage swing decreases. We estimate that the base-emitter voltage and the equivalent non-linear resistor contribute to the relative voltage swing change in the amount of $+0.32 \ mV/\circ C$ and $-0.73 \ mV/\circ C$ respectively. The overall effect is a shift of the output voltage swing with an average value of $-0.4 \ mV/\circ C$. This final result assures the operation of the bipolar transistor beyond deep saturation even for higher temperatures.

### 8.3.3 Noise Behavior

The noise margin of the iterative networks implemented with Bipolar/CMOS differential logic circuits is distributed to each macro-cell. The small logic swing of the internal connections and the absence of the constant current source result in
a poor noise margin of the individual modules. Consequently, the noise voltage in the power supply line must be minimized. A temperature-dependent regulated voltage supply can be used. Our SPICE simulation results, however, show that for a noise voltage induced to the input of the network, the output voltage does not change erroneously but causes only a little loss in signal. This level deterioration is recovered by the following macro-cells. This behavior will be illustrated later on in this chapter on an example.

8.4 Regular Array Generator

With the demand of a quick-turnaround in nowadays ASIC designs, even the fastest circuit design will lose its leverage over the conventional logic if it can not be automatically generated. Fortunately, one of the differential logic advantages is that it can be synthesized using logic minimization algorithms. For this propose binary decision diagrams have been used.

8.4.1 Binary Decision Diagrams

A binary decision diagram (BDD) is a way to determine the output value of a logic function by examining the values of its inputs [1]. The diagram is represented as a directed acyclic graph. There are a couple of procedures to build a BDD depending on the logic function form. Since in most of the cases the logic function is given as a Boolean expression, we will limit our discussion to this type. In this case a top-down procedure based on Shannon expansion can be used.

The classical Shannon expansion formula for a logic function \( f(a, b, c, \ldots) \) is:

\[
(8.9) \quad f(a, b, c, \ldots) = a \ f(1, b, c, \ldots) + \overline{a} \ f(0, b, c, \ldots)
\]

and this one is applied to the function until all its variables are check out.

Before we start building a BDD a very important step is to choose the order in which the function variables are checked out. It is well known that the BDD size depends on the ordering of the variables. Fail to find the right order may lead to suddenly "blow up" either running out of memory or failing to complete the job. This drawback has been intensively study and some solutions have been found [5].

The easiest way to explain how a BDD is built is on an example. Let's take a function of 4 variables

\[
(8.10) \quad f(a, b, c, d) = a \overline{b} \overline{c} \overline{d} + \overline{a} \ (b + c + d)
\]

The arguments order in the diagram is taken in such a way that the size of the resulted graph is minimum. We begin by setting the variable \( b = 0 \) in \( f \) to obtain the function \( f_0 = a \overline{c} \overline{d} + \overline{a}(c + d) \). Then, we set \( b = 1 \) to obtain \( f_1 = \overline{a}(c + d) \) as
Figure 8.6. How to build a BDD using Shannon expansion.

shown in Figure 8.6a. This process is repeated in a similar way by checking out c, d, and as last a. Each time another variable is expanded. If all the paths end in an 1 or 0 the diagram is completed. The result, step by step, is shown in Figure 8.6.

For common combinational functions such as half adder, full adder, parallel to serial multiplexer, serial to parallel multiplexer, n-bit parity function, the BDDs are already built and they are available.

As we mentioned the size of the BDD may become critical if the variables order is not proper. However, in our particular application for differential logic this aspect is not important. In this case the maximum number of variables is limited by the maximum number of MOS transistors which can be stacked on top of each other and still assure a correct operation of the circuit. This number depends on the process technology as well as the supply voltage. In general it is a small number, around 7 for a 0.8-μm technology and 5 V supply. Consequently, for such a small number all the possible ordering of the variables can be tried out to get the best solution.

8.4.2 Macro-cell Generation Using BDDs

In this section we will show how a differential logic tree can be built using BDDs. Considering the logic function given by the Boolean expression (8.10), and its bin-
ary decision diagram representation in Figure 8.6, for each node in the diagram will correspond a pair of MOS transistors in the differential tree. Each transistor pair shares their sources.

For example, the transistors $m_1$ and $m_2$ are the equivalent pair for the node $b$ in the diagram (Figure 8.7). The transistor $m_1$ is driven by the signal $b$ corresponding to the edge $b = 1$ in the graph, while the transistor $m_2$ is driven by the signal $\bar{b}$ corresponding to the edge $b = 0$ in the graph. These transistors have the sources at the ground since the node $b$ is the root of the BDD. The edge $b = 0$ leads to the node $c$ which means that the drain of $m_2$ should be linked with the transistor pair corresponding to the node $c$. In a similar way the drain of $m_1$ should be linked to its correspondent $a$. There are two nodes $a$ in the diagram. Consequently, there are two pairs of MOS transistors in the logic tree. From $a$ the direct link to inverted and non-inverted outputs is made.

![Figure 8.7. Cascode voltage switch logic implementation of $f = a\overline{b}\overline{c}\overline{d} + \overline{a}(b + c + d)$ in a CMOS technology.](image)

In our example in Figure 8.7 we use plain CMOS technology. However, in the same manner we can implement the BiCMOS macro-cells. In this last case, to the node denoting the variable in the critical path will correspond a pair of bipolar transistors. For the rest, the procedure is the same.
8.5 Examples

8.5.1 An Iterative Magnitude Comparator

We will prove the feasibility of our methodology by applying it to some functions. First, an iterative magnitude comparator is presented. This one is, usually, a basic component in an ALU block.

The iterative magnitude comparator is an iterative network which compares two positive integers \( A \) and \( B \). They are represented by the \( n \)-bit binary vectors \( a \) and \( b \). The network output has three possible values \( \text{A>B}, \text{A=B} \) and \( \text{A<B} \). An arbitrary module \( i \) has two external inputs \( a_i \) and \( b_i \) and no external outputs. The internal variables \( g_i, e_i \) and \( s_i \) correspond to \textit{greater than}, \textit{equal} and \textit{less than} function respectively. The output of the network is obtained from the internal outputs of the last module. The internal outputs \( (g_{i+1}, e_{i+1}, s_{i+1}) \) depend on the internal/external inputs as follows:

\[
\begin{align*}
  g_{i+1} &= a_i \bar{b}_i + g_i (a_i b_i + \bar{a}_i \bar{b}_i) \\
  e_{i+1} &= e_i (a_i b_i + \bar{a}_i \bar{b}_i) \\
  s_{i+1} &= \bar{a}_i \bar{b}_i + s_i (a_i b_i + \bar{a}_i \bar{b}_i)
\end{align*}
\]

This implementation is made in such a way to satisfy the restricted equation (8.3). The network module consists of three internal macro-cells, one for implementing each internal variable. The comparison is performed beginning with the least significant bits of the positive integers. The internal inputs of the first module are \( g_0 = 0, e_0 = 1 \) and \( s_0 = 0 \).

![Binary decision diagrams for internal outputs.](image)

Figure 8.8. Binary decision diagrams for internal outputs.

From the binary decision diagram description of the internal variables (Figure 8.8) the differential network trees are constructed [5]. The diagrams for \textit{greater than} and \textit{less than} functions are similar. Therefore, only the schematic for \textit{greater than} and \textit{equal} functions are depicted in Figure 8.9.

In the first two levels we used only NMOS transistors. In the second level the gate-source voltage is smaller than in the first level. Consequently, the transistor width has to be larger to preserve the same current or both transistor sizes have to be
enlarged. It is possible to use PMOS transistors in the second level, but no area will be saved in that case. Note that the MOS transistors from the first level can be shared between the cells.

![Iterative magnitude comparator module](image.png)

Figure 8.9. Iterative magnitude comparator module. (a) greater than function cell. (b) equal function cell.

**Design Parameters** An 8-bit iterative magnitude comparator has been designed and simulated with SPICE for a 0.8-μm single polysilicon layer self-aligned BiCMOS process [11]. The typical device parameters are listed in Table 8.1. A detailed device parameters analysis is made in chapter 2. The effective gate length of the MOS transistors is 0.8 μm. We designed the transistor widths of 4.0 μm and 8.0 μm in the first and the second level respectively. The bipolar transistors have a polysilicon emitter and an $f_T$ of 13 GHz. For this specific technological process we designed the optimal circuit parameters.

<table>
<thead>
<tr>
<th>Bipolar transistor</th>
<th>MOS transistor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_E$</td>
<td>Gate oxide thickness</td>
</tr>
<tr>
<td>$H_{FE}$</td>
<td>$n$-channel $L_{eff}$</td>
</tr>
<tr>
<td>Base Transit Time</td>
<td>$V_{thn}$</td>
</tr>
<tr>
<td>$C_{EB}$</td>
<td>$p$-channel $L_{eff}$</td>
</tr>
<tr>
<td>$C_{CB}$</td>
<td>$V_{thp}$</td>
</tr>
<tr>
<td>$C_{CS}$</td>
<td></td>
</tr>
<tr>
<td>$R_E$</td>
<td></td>
</tr>
<tr>
<td>$R_C$</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.1. Typical Transistor Parameters at 0.8-μm Design Rule
Supply Voltage For a proper operation of the circuit the supply has to cover one $V_{be, on}$ plus the minimum voltage drop over the MOS transistor core. The MOS transistors in this BiCMOS process function down to 1.0 V drain-source voltage. We simulated that a voltage of 1.15 V is enough for operating two series NMOS transistors. Therefore, with a $V_{be, on}$ of 0.85 V and a voltage drop of 1.15 V over the MOS core a supply of 2.0 V is required. This value also assures the network operation under a 10% power fluctuation and with a 25% lower voltage swing (Figure 8.10).

![Figure 8.10. Circuit behavior for a sudden drop of 10% in supply voltage.](image)

Voltage Swing The minimum voltage swing for which the ideal network is still operating is approximately 80.0 mV. When the swing is larger than 500.0 mV the bipolar transistors cannot be kept out of deep saturation. Figure 8.11 shows the propagation time through the critical path of the network as a function of internal voltage swing. The propagation time does not improve too much under a swing of 250.0 mV. Therefore, the voltage swing value which assure a very good propagation time and, at the same time, a good noise margin (10% power fluctuation, temperature and process variation) is 250.0 mV.

Temperature Simulation To support our theoretical estimations on the temperature behavior of the circuit, presented in section 8.3.2, in Table 8.2 some conclusive results are presented. Simulations have been performed for a temperature range from 27°C (room temperature) up to 150°C. The propagation time through the network is almost constant over the temperature range since both the voltage swing and the operating current are decreasing with rising temperature. The voltage swing variation with the temperature is not linear. It has an average rate of $-0.53$ mV/°C. Since the operating current decreases with the increasing temperature a negative feedback is established between these two variables which avoids the over-heat of the circuit.
Figure 8.11. Internal variable propagation time for a greater than function cell as a function of the output voltage swing in a 0.8 μm nominal BiCMOS process.

<table>
<thead>
<tr>
<th>T [°C]</th>
<th>ΔV [mV] output voltage swing</th>
<th>Δt [ps/bit] internal variable propagation time</th>
<th>P [μW] power dissipation per cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>250</td>
<td>134</td>
<td>376</td>
</tr>
<tr>
<td>50</td>
<td>237</td>
<td>131</td>
<td>354</td>
</tr>
<tr>
<td>100</td>
<td>212</td>
<td>130</td>
<td>316</td>
</tr>
<tr>
<td>150</td>
<td>195</td>
<td>130</td>
<td>291</td>
</tr>
</tbody>
</table>

Table 8.2. Temperature behavior of the Bipolar/CMOS differential logic circuits for an 8-bit greater than function cell chain in a 0.8 μm nominal BiCMOS process.

**Process Dependent Performances** To account for process variations, simulations have been performed for the worst case situations corresponding to two extreme technological processes (fast for high current drive and slow for low current drive). The variations of polysilicon length, deep-oxide width, sheet resistance, epitaxial doping, base pinch resistance and normalized hole recombination at the poly-mono silicon interface have been chosen for NPN devices. They have been translated in variation of the SPICE parameters.

For MOS transistors, fast and slow model parameters are generated from the nominal model parameters corresponding to lateral diffusion, threshold voltage, gate oxide thickness and capacitances. The SPICE model parameter variation for bipolar and NMOS transistors are given in Tables 8.3 and 8.4.
together with the simulation results.

For a fast technological process the voltage swing rises with about 64% comparative with the nominal value, while for a slow process it decreases with about 38%. However, the new voltage swing values are still fulfilling the conditions enumerated in section 8.5.1 and, therefore, a nominal swing of 250.0 mV assures a proper operation of the network even for technological process variations.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1.45e-18</td>
<td>47</td>
<td>116</td>
<td>3.67</td>
<td>1960</td>
<td>42.65</td>
<td>3.6</td>
<td>4.73</td>
</tr>
<tr>
<td>slow</td>
<td>3.57e-18</td>
<td>25</td>
<td>250</td>
<td>9.49</td>
<td>3267</td>
<td>46.79</td>
<td>5.95</td>
<td>7.22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process</th>
<th>LD [μm]</th>
<th>VTO [V]</th>
<th>TOX</th>
<th>capacitances</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>+.075</td>
<td>-.15</td>
<td>-10%</td>
<td>-20%</td>
</tr>
<tr>
<td>slow</td>
<td>-.075</td>
<td>+.15</td>
<td>+10%</td>
<td>+20%</td>
</tr>
</tbody>
</table>

**Table 8.3.** SPICE parameters for fast and slow fabrication process.

<table>
<thead>
<tr>
<th>Process</th>
<th>T=27°C</th>
<th>T=150°C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ΔV [mV]</td>
<td>Δc [μs/bit]</td>
</tr>
<tr>
<td>fast</td>
<td>410</td>
<td>114</td>
</tr>
<tr>
<td>slow</td>
<td>156</td>
<td>186</td>
</tr>
</tbody>
</table>

**Table 8.4.** Effect of the process variation on the circuit performance of an iterative network with a nominal voltage swing of 250.0 mV.

**Noise Behavior Simulation** Figure 8.12 illustrates the noise behavior of the iterative network implemented with Bipolar/CMOS differential logic circuits. A noise voltage is induced at the beginning of the greater than function chain so that the low and the high logic levels are modified with 30% in the worst direction. The result is a loss in the output signal of the first stage. After that the level deterioration is completely recovered. The waveforms in Figure 8.12 represent the non-inverted greater than bit propagation through the network.

**Comparison** For comparison we implemented the same iterative networks with differential split-level CMOS logic, which is claimed to be the fastest logic among the differential CMOS logic techniques [43]. The result, for a
0.8-μm nominal CMOS process, is an internal variable propagation time of 350.0 ps/bit for 5.0 V supply at a reduced voltage swing of 2.4 V. For 3.3 V supply a delay of 1.4 ns/bit was obtained, which is ten times as long as we get using our Bipolar/CMOS differential logic methodology.

8.5.2 Gray-to-binary Code Converter

Another popular example of iterative network is a Gray-to-binary code converter. The integers from 0 to $2^n$-1 can be represented in an $n$-bit Gray code so that two consecutive numbers differ only in one bit. This property is very useful especially in power optimized systems where reducing the switching activity is essential.

If the Gray representation of an integer is $g = (g_{n-1}, \ldots, g_0)$ and the binary representation of the same integer is $b = (b_{n-1}, \ldots, b_0)$ then the bit $b_j$ is the parity function $P$ of the bits $g_{n-1}, \ldots, g_j$ as follows:

$$b_j = P(g_{n-1}, \ldots, g_j) = b_{j+1} \oplus g_j \quad 0 \leq j \leq n - 1 \text{ and } b_n = g_n$$

Implementing the parity functions with an iterative network of classical XOR gates will result in a slow conversion. To speed it up we use Bipolar/CMOS differential logic methodology together with the skip technique. The input vector is divided in $n/k$ external input groups of $k$ elements each. One group is applied to each module. Consequently, the network has $n/k$ identical $M$ modules connected as indicated in Figure 8.13.

The network macro-cell $M$ has $k$ internal cells and produces $k$ external outputs $b_i^j$ and one internal output $c_i^j$. The first $k-1$ cells are differential XOR network trees. They produce the external outputs $b_{k-1}^j \ldots b_1^j$. The $k^{th}$ cell, called the skip cell,
Figure 8.13. Gray-to-binary code converter iterative network.

is specialized in providing the propagation bit \( c^i = b_0^i \) without waiting for the computing of the "most significant" bits of the module. It has \( k+1 \) differential inputs, including the internal input \( c^{i+1} \).

The \( n \)-bit conversion time is

\[
t_{\text{conv}} = ((n/k) - 1)t_{\text{skip}} + kt_{\text{XOR}}
\]

and

\[
c^i = g_0^i \oplus g_1^i \oplus \cdots \oplus g_{k-1}^i \oplus c^{i+1}
\]

Figure 8.14. Gray-to-binary code converter skip cell.

In Figure 8.14 the general structure of the skip cell in the Gray module is presented together with the particular case \( k=3 \). For this case the third level in the MOS core is implemented with PMOS transistors to minimize the circuit area. The transistor sizes are adjusted for optimum delay taking into account the cell operating current and the supply voltage.
For an 190.0 $\mu$A operating current at 2.0 V supply we obtained an internal delay of 180 ps per skip cell in the particular case when $k=2$, 185.0 ps for $k=3$, and 190.0 ps for $k=4$ respectively, for gradually increasing supply voltage. The optimal design which satisfies both high speed and low area consumption requirements is for $k=3$ MOS transistor stacking height.

### 8.6 Conclusion

We presented in this chapter an efficient way of implementing high speed/low voltage iterative networks. Our Bipolar/CMOS differential logic macro-cells improve the overall speed of the networks considerably and allows for a low supply. Besides, the design can be generated automatically by using binary decision diagrams. This is a major advantage in ASIC design. High performance circuits can be generated easily and made available to the market in a short period of time. Two examples have been given to support our methodology.
Chapter 9

CONCLUSIONS

The focus of this research was on speed optimization techniques targeted towards cell-based designs in a BiCMOS environment. Cell-based design has become the dominating technique on todays ASIC market because of its high performance, low cost, and short time to market capability. On the other hand, BiCMOS technology is established as a technology which combines high speed with the high integration density and low power dissipation of CMOS technology. Consequently, joining the quality of cell-based design to design circuits fast with the quality of BiCMOS technology to design fast circuits seems a very attractive merger. The main result of our work is that we not only show how to design fast circuits, but we also made possible to design them in a short time.

We have shown that different optimization methods have to be applied in case of macro-cell design and standard-cell design. We took the speed advantages of bipolar transistor and differential logic design and merged them in a high speed macro-cell design. These macro-cells can be automatically generated by using binary decision diagrams. A regular array generator can be built which accepts the functional description and generates the circuit netlist.

In case of standard-cell design, we optimized the critical paths by using bipolar drivers and efficient timing optimization algorithms. Since in standard-cell design the library primitives play a very important role, these ones need to be very well designed and characterized. To optimal design the BiCMOS drivers we developed an analytical delay model. The model is based on physical device parameters and can be used to estimate both pull-up and pull-down times for a variety of circuit configurations. For the intrinsic delay associated with the bipolar transistor we used a charge control model that incorporates the high injection effects upon the current gain and the base transport factor. For modeling the MOS transistor we took into account the short and narrow channel effects upon the device current equations. A simple approach was proposed for the accurate determination of the delay dependency on input signal slope. The effect of input signal slew rate on the circuit power dissipation was also considered. The comparison with SPICE circuit simulation results showed that the new model predicts the gate delay with less than 5% error in most of the cases.
An automatic design system for BiCMOS drivers was built based on the analytical delay model. The transistor sizing problem was optimal solved. The new model provides significant improvement of the necessary computation time for optimization, where numerous circuit simulations are required to determine the optimum solution. In addition, while other existent systems design drivers in an unrealistically wide range of sizes and delays in order to achieve maximum speed for each load, our design system can accommodate a wider range of loads per driver and still get the maximum speed.

Further more, we extrapolated a CAD model from the analytical delay model. This model is not only useful for timing optimization algorithms but also for high-level timing simulation purposes. We implemented a timing optimization algorithm which speeds up the combinational circuits. An efficient iterative algorithm that generates fanout tree with close to optimum delay has been proposed. No restrictions on fanout tree topology are imposed. The space of all possible solutions for a given fanout problem is implemented as a rooted tree which is dynamically constructed and searched. An average delay improvement of 30% has been obtained on benchmark circuits.

For the future work, we plan to have a closer look to the influence of deep submicron technologies on the BiCMOS delay model and optimization techniques. We expect that our analytical delay model will still give accurate results for a 0.5-μm technology as well. But we are not sure if the model is still valid for 0.35-μm feature size. Since our delay model is based on the device models, as long as these ones are still valid our delay model will still give good results. Besides, some delay improvements need to be made to accommodate the BiCMOS full swing gates. Another field to be investigated is related to the timing optimization algorithms. First, for better timing optimization performance direct layout information is necessary. A link between our timing optimization program and a global routing tool can make a very useful dynamic feedback. Second, better results will be obtained if our timing optimization algorithm for combinational circuits will be mixed with the existent retiming algorithms for sequential circuits.
Bibliography


SUMMARY

This thesis presents speed optimization techniques for cell-based designs in a BiCMOS environment. Cell-based design, including both standard-cell and macro-cell, is the dominant design style on today's ASIC market because of its high performance, low cost, and short time to market capability. BiCMOS technology is established as a very high speed technology while retaining the high integration density and the low power dissipation of CMOS technology. So, joining the ability of cell-based design to design circuits in a short time with the quality of BiCMOS technology to design fast circuits seems a very attractive merger. In this thesis I will show how such high speed circuits can be designed. The main result of this work is to enable efficient techniques for the various tasks related to a BiCMOS design environment.

The first speed optimization strategy is to use BiCMOS drivers in the time critical path of a standard-cell design to enhance the circuit speed. Chapter 2 introduces various types of BiCMOS gates which can be integrated in a standard-cell library and used to drive high capacitive loads. Accurately modeling of the delay of these BiCMOS drivers requires device models with enough detail, yet manageable complexity.

Chapter 3 demonstrates the analysis and characterization of such a driver. I introduce an accurate analytical delay model for a totem-pole BiCMOS inverter. The model takes into account the short and narrow channel effects on the MOS device equations and the high level injection effects on the bipolar transistor collector current. Two analytical models are developed: one model which gives very accurate delay estimations but also rather 'expensive' computationally, and another model which uses more approximations but which is easier to use. A BiCMOS CAD delay model is extrapolated from the former in chapter 4. It depends on the input signal slew rate and the output load capacitance. The CAD delay model is used in implementing the speed optimization algorithms for standard-cell based designs. Finally, a power dissipation analysis versus input signal slew rate is made to complete the cell characterization part.

The performance of speed optimization techniques rely on the performance of the design components. Special attention is paid to cell design, which is not at all
trivial. Chapter 5 presents a comprehensive strategy for BiCMOS buffer optimum design. A fully automated BiCMOS design system is implemented. It can be used at the early stage of ASIC design to build the design library, or to be integrated with the network optimization algorithm, e.g. for gate resizing.

In chapter 6 the optimization methods used in standard-cell based design are introduced. An important speed optimization step is to enable cells to drive a large number of fanouts while satisfying the timing constraints. This is made possible by building a tree of buffers and inverters (the fanout tree) at the output of the cell. The fanout optimization algorithm is presented in chapter 7. The algorithm is tested on the standard set of benchmark circuits. The experimental results are also presented in this chapter.

Chapter 8 addresses my second speed optimization strategy targeted towards macro-cell based designs. A high-speed/low-voltage macro-cell design methodology is proposed for implementing applications which use iterative networks. I show how to automatically generate and optimize the macro-cell circuit design. Examples are given to illustrate my design methodology.
SAMENVATTING

In dit proefschrift presenteer ik schakelsnelheidsoptimalisatietaftechnieken voor cel-gebaseerde BiCMOS geïntegreerde schakelingen. Cel-gebaseerd IC-ontwerp is een veel gebruikte techniek waarbij de schakeling wordt opgebouwd uit een aantal vooraf ontworpen elementen. Elk van deze elementen is door experts ontworpen en uitvoerig getest. Hierdoor kan een schakeling relatief snel en met voorspelbare eigenschappen ontworpen worden, hetgeen de kosten omlaag brengt.

De BiCMOS technologie is een IC-fabricageproces waarbij zowel CMOS als bipolaire transistors op dezelfde wafer geïntegreerd kunnen worden. Fabricage in deze technologie is iets duurder dan in de conventionele CMOS technologie. In BiCMOS kunnen echter de beste eigenschappen van beide transistortypen gecombineerd worden: hoge integratiedichtheid, laag energieverbruik, en zeer hoge schakelsnelheden. Met de in dit proefschrift voorgestelde optimalisatietechnieken kunnen deze snelle schakelingen efficiënter ontworpen worden.

In hoofdstuk 2 wordt een nieuwe methode gepresenteerd waarbij BiCMOS drivers in het kritische pad van de gebruikt worden. Een aantal varianten voor BiCMOS poorten die eenvoudig in een standaard-celbibliotheek opgenomen kunnen worden passeren daar de revue. Deze poorten zijn speciaal geschikt voor het aansturen van grotere capacitatieve belastingen. Voorts worden in dit hoofdstuk de eisen die deze BiCMOS drivers stellen aan de transistormodellen bepaald, waardoor de poorteigenschappen nauwkeurig voorspeld kunnen worden.

In het volgende hoofdstuk (3) presenteer ik een nauwkeurig analytisch vertragingstijdmodel voor de totem-pole BiCMOS inverter. Dit nieuwe model houdt rekening met de z.g. short-channel en de narrow-channel effecten op de transistorschakelingen. Bovendien worden de high-level injection-effecten op de collectorstroom van de bipolaire transistor in de beschouwing meegenomen. Ik heb twee analytische modellen ontwikkeld. Het eerste model geeft een zeer nauwkeurige schatting voor de vertragingstijd, maar is tamelijk rekenintensief. Het tweede model geeft een minder accurate benadering maar is eenvoudiger te gebruiken en sneller. In hoofdstuk 4 heb ik op basis van het nauwkeurige model een CAD vertragingstijdmodel ontwikkeld dat zowel de steilheid van de ingangssflank als de uitgangsbelasting als parameters gebruikt. Dit model is zeer toepasbaar in schakelsnelheids-
optimalisatiealgoritmen in automatische circuitsyntheseprogramma's. Als laatste aspect in dit hoofdstuk heb ik de energieconsumptie van de schakeling geanalyseerd en nauwkeurig gekarakteriseerd. Deze parameter hangt af van de flanksteilheid van het ingangssignaal.

Het effect van de schakelsnelheidsoptimalisatietechnieken hangt vanzelfsprekend ook af van de individuele ontwerpcomponenten. Het ontwerp van zulke componenten is niet trivial. In hoofdstuk 5 ga ik in op de snelheidsoptimalisatie van combinatorische schakelingen die met deze componenten worden opgebouwd. Na een inleiding over de basistechnieken voor standaard-cel optimalisatie presenteer ik een algemene strategie voor BiCMOS bufferontwerp. Ik heb een automatisch BiCMOS ontwerpsysteem geïmplementeerd dat gebruikt kan worden voor bibliothekcelontwikkeling. Het zou ook geïntegreerd kunnen worden in de netwerkoptimalisatiestap van een logisch syntheseprogramma.

Ik behandel weer een andere relevante schakelsnelheidsoptimalisatiestap in hoofdstukken 6 en 7. Sommige uitgangssignalen moeten een groot aantal ingangen aansturen, vaak meer dan het aantal waarvoor de poort ontworpen was. Door een fan-out tree van buffers en invertors in te voegen kan het uitgangssignaal versterkt worden, terwijl de vertraging binnen zekere grenzen blijft. Ik presenteer hiervoor een nieuw fanoutoptimalisatiealgorithmie in hoofdstuk 7. Het werk is eveneens geïmplementeerd. De experimentele resultaten steken gunstig af bij ander werk op hetzelfde probleem.

In Hoofdstuk 8 presenteer ik mijn tweede schakelsnelheidsoptimalisatiestrategie voor macro-cel ontwerp: een laag-voltage/hogesnelheidsontwerpstijl voor iteratieve (d.w.z. repeterende) netwerken. Ik toon aan hoe dergelijke circuits automatisch gegenereerd kunnen worden. Een schakeling is ook op een chip geïmplementeerd.
ACKNOWLEDGMENTS

This thesis is the result of my research work at the Circuits and Systems Group, Delft University of Technology.

I especially would like to thank Prof.dr.ir. Ralph H.J.M. Otten, my promotor, for giving me the opportunity to perform my Ph.D. research under his guidance and for his help along the path towards completion of this thesis.

I would also like to thank all people who contributed in any way towards creating a stimulating and positive work environment on the 17th floor. In particular I would like to express my gratitude to Paul Stravers for his constant encouragements and expert help, to Jack Glas for tolerating me as his office mate, to Marion de Vlieger for (among many other things) her great help with the final stages of the thesis, and to Martin Huisman for teaching me Dutch.

Furthermore, I am indebted to Alex Hegt and Leo de Jong for their warm support and help during the first years I joined Delft University of Technology.

Above all, I would like to thank my husband Patrick for his love and support.
ABOUT THE AUTHOR

Viorica Simion was born in Bucharest, Romania, on March 11, 1960. In June 1979 she received the secondary school diploma from the "Mihai Viteazul Lyceum" in Bucharest. In September the same year she started her study in electrical engineering at the Polytechnic University in Bucharest. In 1984 she received the Electrical Engineer degree (the equivalent of an M.Sc.).

From 1984 to 1986, Mrs. Simion has been with the Integrated Circuits and Semiconductors Company of Romania (IPRS) in Bucharest, where she worked on automatic IC layout design and testing. From 1986 to 1991, she has been with the Research and Development Center for Semiconductors (ICCE), in Bucharest as an R&D engineer. Her research was focused on modeling and characterization of MOS devices. She was involved in implementing a parameter extraction software system for characterization and developing a self-aligned CMOS technology.

In 1991, Mrs. Simion joined the Circuits and Systems group at Delft University of Technology to embark in the chartered engineering program (TWA10). In March 1993, she received the TWA10 diploma. Immediately after that, she started her Ph.D. research in the same group under the supervision of Prof. dr. ir. R.H.J.M Otten. Her research interests include the design of high-speed/low-voltage CMOS and BiCMOS circuits and systems together with speed optimization methods.

Mrs. Simion is currently working with Cadence Design Systems Inc. in San Jose, California, USA.