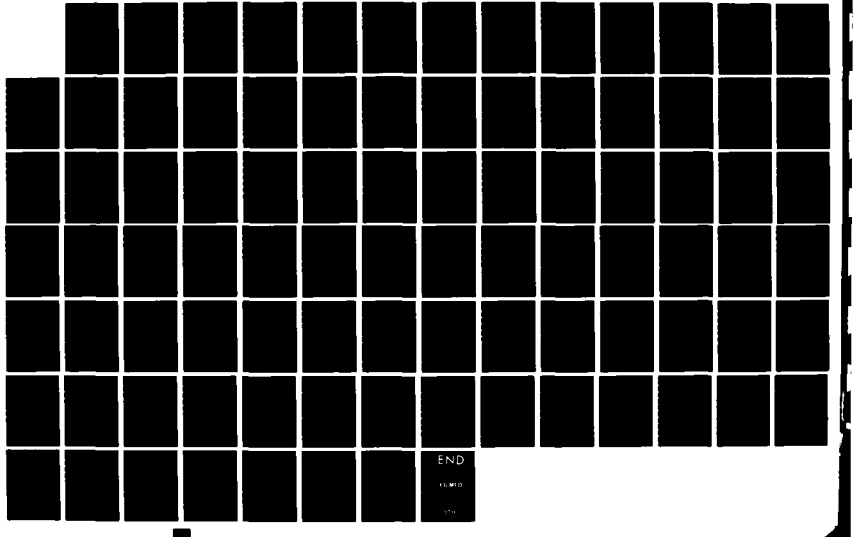
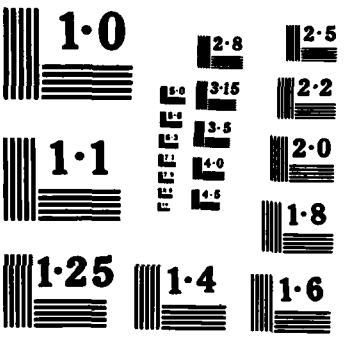
AD-A158 540 A COHERENT VLSI DESIGN ENVIRONMENT(U) MASSACHUSETTS 1/1
INST OF TECH CAMBRIDGE DEPT OF ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE P PENFIELD ET AL. 31 MAR 85
UNCLASSIFIED N00014-80-C-0622 F/G 9/5 NL





NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CAMBRIDGE, MASSACHUSETTS 02139

A COHERENT VLSI DESIGN ENVIRONMENT

Semiannual Technical Report
for the period
October 1, 1984 to March 31, 1985

Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Principal Investigators: Paul Penfield, Jr. (617) 253-2506
Lance A. Glasser (617) 253-4677
Thomas F. Knight, Jr. (617) 253-7807
Charles E. Leiserson (617) 253-5833
Ronald L. Rivest (617) 253-5880
John L. Wyatt, Jr. (617) 253-6718
Richard E. Zippel (617) 253-6028

This research was sponsored by Defense Advanced Research Projects Agency (DoD), through the Office of Naval Research under Contract N00014-80-C-0622.

TABLE OF CONTENTS

---

[1] Abstract and/or Table of Contents only. Complete version available from Microsystems Program Office, Room 36-575, MIT, Cambridge, MA 02139; telephone (617) 253-7308.

RESEARCH OVERVIEW

This report covers the period from October 1, 1984 through March 31, 1985. The research discussed here is described in more detail in several published and unpublished reports cited below.

Several fundamental bounds on the complexity of network architecture, parallel computation, VLSI design, and algorithms have been established and/or improved during this period. The grid-matching problem, of importance to wafer-scale integration, is close to solution. Improved algorithms for two-layer channel routing have been developed.

The "fat-tree" interconnection network has been studied further, and a better algorithm for on-line routing of messages in this network has been developed. There is continued interest in compaction, and a provably fast algorithm for solving constraint systems has been devised.

The CAD frame Schema has been solidified in several ways during this period. It is now possible to use Schema as a schematic capture and data storage system. There is better support being developed for PC-board designs. Some advanced ideas in describing waveforms qualitatively are being incorporated.

A novel PROM device that is UV-enabled for writing has been designed and tested. The tradeoff between speed and fault probability in A/D converters has been viewed from a new angle. The same tradeoffs have been investigated for inverters, in an embrionic study of reliability software. Development of CAD tools for the IBM PC has continued.

The previously reported bounds for interconnect delay in MOS circuits have been improved in several ways. Some bounds now pertain to RC meshes rather than RC trees; some hold with resistors to ground. Tighter bounds have been found by exploiting slew-rate limits on node voltages.

THE DESIGN OF SCHEMA

During the past six months, Schema has been solidified in a number of ways. The database and schematic capture tools have stabilized to the point that people are now being to use Schema to enter and store designs which will be maintained for long periods of time. Our colleagues at Harris are confident enough in the architecture that they are developing the first application tool to be built on top of Schema rather than an integral part of Schema. In addition a number of the internal components that are needed for VLSI design are being put into place.

Anuja Kohli has enhanced the schematic capture portion of the system so that it can handle logic symbols with an arbitrary number of inputs, and is developing a basic spatial management system for schematics. This tool will permit the creation of routing programs for schematics and may also be used for gate arrays. In addition they are used to manage the placement of text on the screen and generally improve the aesthetics of the designs as entered by hand.

Our colleagues at Harris, Inc. are developing a wirewrap/PC board development tool on Schema. This package makes use of Schema's newly enhanced ability to deal with logic diagrams and hierarchical designs. There are three phases to the project. First, the logic schematic is converted to technology specific diagram by binding gates to particular implementations, e.g. a NAND gate is converted to a 74S00. Second, the gates are partitioned into packages and the packages are placed. At this point a wire wrap board can be created. Finally, a more detailed adjustment of the placement is made and the signals are assigned layers and routed. This last phase is being done by Don Becker here at MIT.

Brian Williams has been refining his temporal constraint propogation tools in preparation for their incorporation into Schema. Margaret St. Pierre has begun specifying the waveform representations and simulation interface for Schema. Unlike previous versions of these representations, Margaret's will permit qualitative values to be used both for the time specification and for the value. The basic idea is that a waveform is a mapping between time and some value space. The value space can be a continuous quantitative domain as is used by Spice for voltages and a currents, a discrete quantitative domain as is used in logic simulation, or a qualitative domain as is used by Brian Williams qualitative reasoning system. When a waveform is asked for the value at some time, the time value can be any open or closed interval, including a point. If the time value is not a point, the value returned may be a qualitative value. This mechanism will ease greatly the effort required to incorporate qualitative reasoning mechanisms to Schema.

THE WAVEFORM BOUNDING APPROACH TO TIMING ANALYSIS

Our work since October 1, 1984 has been concentrated on bounds for signal delay in linear RC models for on-chip interconnect. Taking as a starting point the work on Rubinstein, Penfield, and Horowitz (IEEE Trans. CAD, July, 1983), we have been able to include more general networks than RC trees driven by voltage step inputs. We have also succeeded in reducing the region of uncertainty in the original bounds for certain classes of networks of practical interest.

One extension we have completed is a method of bounding the response of RC meshes, which are more general than RC trees in that resistor loops are allowed. These networks are important in practice 1) as models for the gates of large MOS pad driver transistors, 2) whenever linear resistor models are used for transistors in logic gates or CMOS pass gates, as in Chris Terman's program RSIM, and 3) to model interconnect networks with closed loops sometimes created by automatic routing programs. Another successful extension is to networks with resistive paths to ground, which are appropriate models for, e.g., interconnect to bipolar logic gates.

Tighter bounds have been achieved for unbranched lines and certain classes of RC trees by exploiting slew rate limits on the node voltages and exploiting the spatial convexity of interconnect voltage during transients in a novel way.

Two master's level graduate students, Ray Schnitzler and David Standley, are being supported by this contract.

HIGH PERFORMANCE CIRCUIT DESIGN

Progress was made in four areas: PROMs, A/D converters, understanding noise margin/speed/reliability tradeoffs, and PC microtools.

We have recently demonstrated a new type of programmable read-only memory (PROM). The invention allows users of conventional nMOS and CMOS processes, such as those available through MOSIS, to place several hundred bits of electrically alterable read-only memory on any custom VLSI chip. Typical applications might include the storage of cryptographic codes, special addresses, calibration data, or repair locations for fault-tolerant systems.

The size of the new non-volatile PROM cell is about twice the size of a conventional static--but volatile--memory cell. The programming process, while it does not require the use of high voltages or special processing, must be done in the presence of ultraviolet (UV) light. Each cell can be individually written while the UV light floods the entire chip.

The experimental chips were fabricated through MOSIS in 4 micron nMOS. Write times on the order of ten minutes were observed. So far, the cells have retained their state for months, and years of storage is projected. We hope the new PROM, which is not seen as replacing commercial EEPROMs built with special processes, will find wide system application where the use of exotic and expensive processes for just a few bits of field programmable, non-volatile storage is uneconomical or infeasible.

In the area of A/D convertion, we examined the fundamental limits on the speed of A/D converters as a function of the probability of a fault. This fault problem is completely analogous to the synchronizer problem in digital circuits. After all, if one could build a perfect A/D converter, one could build a perfect synchronizer. We have found that, for extremely high levels of reliability, flash and self-timed successive approximation converters are equally slow because they both spend virtually all their time resolving the one hard bit.

We have continued our investigation of tradeoffs between speed and reliability. We have discovered that the lower bound on inverter pair delay increases by 50% as the noise margins are increased from zero to their maximum values of half the power supply rail. We have also investigated the transient step response of inverters and seen tradeoffs between the reliability measure (the noise margin divided by the worst-case noise) and the ultimate speed.

We have continued our back-burner effort on VLSI microtools--a set of programs for helping with the early design stages of a VLSI chip. These tools run on IBM PCs and either use LOTUS 1-2-3 or TK!Solver. They solve such problems as finding the temperature rise in a metal line or its fringing capacitance as a function of the wire geometry. They also do characteristic impedance calculations for PCBs. Community members may have copies of these programs for free, and at their own extreme risk, by sending me a diskette. We promise bugs for all.

ARCHITECTURAL DESIGN

Professor Leighton is continuing research on several problems involving novel network architectures, parallel computation, VLSI design and the development of algorithms for NP-complete problems which provably work well on the average. Advances have been made in several areas during the past six months. Highlights are described in the following paragraphs.

In the algorithms area, Professors Leighton and Sipser, Thang Bui and Soma Chaudhuri (University of Washington at Seattle) have developed graph bisection algorithms which (provably) almost always find the minimum bisection of graphs with small bisections. These algorithms perform dramatically better than known techniques for large classes of relevant graphs. The work will form an important part of Thang Bui's PhD thesis, which should be completed by this summer.

In the area of fault-tolerant construction of VLSI networks, Professors Leighton and Rosenberg (Duke University) and Dr. Chung (Bell Communications Research Labs) have developed efficient algorithms and bounds for representing useful networks as a small number of "stacks" of wires. As the stacks are easily implemented in VLSI, the results make possible the efficient configuration of fault-free networks in environments that contain defective components.

In related work, Professor Leighton and Peter Shor (who is expected to finish his PhD thesis this summer) are close to solving the grid matching problem. Roughly stated, the problem is to determine the expected minimum maximum edge length over all perfect matchings of N random points to N fixed points that are arranged in an $N^{1/2} \times N^{1/2}$ grid with unit spacing between consecutive rows and columns. Professors Leighton and Leiserson proved an upper bound of $O(\log N)$ and a lower bound of $\Omega(\log N)^{1/2}$ for this problem in their work on wafer-scale integration of systolic arrays in 1982. Determination of the precise bound has remained a difficult and important open problem ever since. It now appears that the exact bound for the grid matching problem is $\Theta(\log^{3/4} N)$, improving both the upper and lower bounds. As a direct result of this work, it will be possible to improve the best bounds known for the average case behavior of algorithms for wafer-scale integration as well as for a variety of other packing and assignment problems.

Professor Leighton and Johan Hastad (a first year graduate student) are developing efficient circuits for parallel division. Currently, the only known circuit that can compute the N most significant bits of a quotient in $O(\log N)$ parallel steps requires $\Theta(N^5)$ processors. Preliminary work by Leighton and Hastad indicates that the number of processors can be decreased to $O(N^{1+\epsilon})$ where ϵ is an arbitrarily small positive constant. Although not yet practical, the improvement in hardware requirements is significant.

Professor Leighton and Bonnie Berger (another first year graduate student) are developing improved algorithms for 2-layer channel routing. Initial progress in this area suggests that it may be possible to achieve the performance of the Baker-Bhatt-Leighton Manhattan routing algorithm and the Rivest-

Baratz-Miller knock-knee routing algorithm with a single, simpler algorithm. More importantly, it appears that the new algorithm can be extended to the unit-vertical-overlap model (in which wires can overlap only for unit distance and only in the vertical direction) where a factor of two in channel width can be saved. The factor of two is significant because the new algorithm always routes 2-point net channels with width $d+O(d)$ instead of the best previously known bound of $2d-1$. Here d denotes the density of the channel which, of course, is a lower bound on channel width. The results also hold for multi-point net problems, except that an additional factor of two in channel width is required.

Peter Shor has been investigating the average-case behavior of bin packing algorithms. In the case where the item sizes are uniformly distributed, he has derived much tighter bounds on the wasted space produced by the algorithm First Fit than were previously known, and has the exact answer, up to a constant, for the wasted space produced by the algorithm Best Fit. He has also derived a lower bound for any on-line that shows that on-line algorithms cannot do as well as off-line algorithms, and that Best Fit comes within a small factor of being optimal among on-line algorithms.

Charles Leiserson and Ron Greenberg have further improved their algorithm for on-line routing of messages in the "fat-tree" interconnection network. This probabilistic algorithm is novel in that it does not randomize in the choice of message paths or in the operation of the switches, but rather in the choice of whether or not to send a particular message in a particular delivery cycle. The algorithm ensures that a set of messages, M, can be routed with high probability within $O(\lambda(M)\log|M|)$ delivery cycles, where $\lambda(M)$ is the maximum over all communication links of the ratio of the number of messages in M which must pass through the link to the capacity of the link. This work may also have some applicability to routing networks other than "fat-trees."

Miller Maley has developed a provably fast algorithm for solving constraint systems in VLSI layout compaction. Constraint solving is usually done by the Bellman-Ford algorithm, which has $O(|V||E|)$ running time in the worst case. Heuristics have been developed which allow the system of constraints arising in compaction to run much more quickly than this bound. The new algorithm runs in $O(|E|+|V|\log|V|)$ time in the worst case.

Susmita Sur is currently writing up the work on channel stretching in the PI project.

PUBLICATIONS LIST

C. E. Leiserson and J. B. Saxe, "A Mixed-Integer Linear Programming Problem Which is Efficiently Solvable," 21st Annual Allerton Conference on Communication, Control and Computing, October, 1983; also MIT VLSI Memo No. 84-216, December, 1984.

Andrew L. Ressler, "A Circuit Grammar for Operational Amplifier Design," PhD Thesis, Department of Electrical Engineering and Computer Science, MIT, January, 1984; also MIT VLSI Memo No. 84-211, November, 1984.

F. T. Leighton and A. L. Rosenberg, "Three-Dimensional Circuit Layouts," MIT-LCS TM #262, September, 1984.

T. Bui, S. Chaudhuri, F. T. Leighton and M. Sipser, "Graph Bisection Algorithms with Good Average Case Behavior," Proc. 25th Conf. on Found. of Comp. Sci., October, 1984, pp. 181-192.

H.-N. Tan and J. L. Wyatt, Jr., "Time Optimal Trajectories Associated with Voltage Bounds in RC Tree Networks," MIT VLSI Memo No. 84-205, October, 1984.

P. S. Whitney and C. G. Fonstad, "Manganese as a p-type Dopant for Liquid Phase Epitaxial $In_{0.53}Ga_{0.47}As$," MIT VLSI Memo No. 84-206, October, 1984.

J. L. Wyatt, Jr., and Q. Yu, "Signal Delay in RC Meshes, Trees, and Lines," Proceedings, 1984 IEEE International Conference on Computer-Aided Design, Santa Clara, CA, pp. 15-17, November, 1984.

Mark D. Matson, "Macromodeling of Digital MOS VLSI Circuits," MIT VLSI Memo No. 84-212, November, 1984.

Mark D. Matson, "Optimization of Digital MOS VLSI Circuits," MIT VLSI Memo No. 84-213, November, 1984.

C. A. Zukowski and J. L. Wyatt, Jr., "Sensitivity of Nonlinear 1-port Resistor Networks," IEEE Transactions on Circuits and Systems, vol. CAS-31, no. 12, pp. 1048-1051, December, 1984.

J. Buss and P. Shor, "On the Pagenumber of Planar Graphs," 16th Annual Symposium on Theory of Computing, pp. 98-100, 1984.

P. Shor, "The Average-Case Analysis of some On-Line Algorithms for Bin Packing," 25th Symposisum on Foundations of Computer Science, pp. 193-200, 1984.

J. L. Wyatt, Jr., "Monotone Sensitivity Theorem for Nonlinear, Nonuniform RC Transmission Lines, with Application to Timing Analysis of Digital Integrated Circuits," IEEE Transactions on Circuits and Systems, vol. CAS-32, no. 1, pp. 28-33, January, 1985.

J. L. Wyatt, Jr., Q. Yu, C. Zukowski, H. N. Tan, and P. O'Brien, "Improved Bounds on Signal Delay in MOS Interconnect," MIT VLSI Memo No. 85-221, January, 1985.

A. L. Robinson, L. A. Glasser, and D. A. Antoniadis, "A Simple Cost Model for Multilayer Integrated Circuits," MIT VLSI Memo No. 85-223, January, 1985.

F. Berman, F. T. Leighton, P. Shor and L. Snyder, "Generalized Planar Matching," MIT VLSI Memo No. 85-234, March, 1985.

F. R. K. Chung, F. T. Leighton and A. L. Rosenberg, "Embedding Graphs in Books: A Layout Problem with Applications to VLSI Design," MIT VLSI Memo No. 85-235, March, 1985.

J. L. Wyatt, Jr., Q. Yu, C. Zukowski, H. N. Tan, and P. O'Brien, "Improved Bounds on Signal Delay in MOS Interconnect," accepted for publication in Proceedings, IEEE International Conference on Circuits and Systems, Beijing, China, June, 1985.

F. M. Maley, "An Observation on Constraint Solving for VLSI Layout Compaction," unpublished manuscript.

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CAMBRIDGE, MASSACHUSETTS 02139

VLSI Memo No. 84-205 October 1984

Time Optimal Trajectories Associated with Voltage Bounds in RC Tree Networks*

Han-Ngee Tan and John L. Wyatt, Jr.

ABSTRACT

Linear RC tree networks are appropriate models for branching interconnect lines in MOS integrated circuits. Bounds on the step response of these networks, first derived in Rubinstein, Penfield, and Horowitz (1983), are useful in timing analysis as bounds on the signal delay in MOS interconnect. Those results become more transparent if the bounds are derived in terms of the payoff function for associated minimum-time and maximum-time linear optimal control problems with state constraints. This approach, first introduced in Yu and Wyatt (1984), provides a natural way of incorporating new information, such as bounds of the form $\dot{v}_e(t) \geq - v_e(t)/T_{se}$ on the slew rate of node voltages, and yields tighter bounds on signal delay than were given in the work by Rubinstein et al. above. This report gives a simple and rigorous derivation of the solutions to this class of optimal control problems with slew rate limitations.

---

*This work was supported by the National Science Foundation under Grant No. ECS-8310941 and by the Air Force Office of Sponsored Research under Contract No. F49620-84-C-0004.

**Address inquiries to Wyatt: Department of Electrical Engineering and Computer Science, M.I.T., Room 36-865, Cambridge, MA 02139; (617) 253-6718.

Time Optimal Trajectories Associated with Voltage Bounds in RC Tree Networks

Han-Ngee Tan and John L. Wyatt, Jr.

Abstract

Linear RC tree networks are appropriate models for branching interconnect lines in MOS integrated circuits. Bounds on the step response of these networks, first derived in [1], are useful in timing analysis as bounds on the signal delay in MOS interconnect. The results in [1] become more transparent if the bounds are derived in terms of the payoff function for associated minimum-time and maximum-time linear optimal control problems with state constraints. This approach, first introduced in [2], provides a natural way of incorporating new information, such as bounds of the form $\dot{v}_e(t) \geq -v_e(t)/T_{ee}$ on the slew rate of node voltages, and yields tighter bounds on signal delay than were given in [1]. This report gives a simple and rigorous derivation of the solutions to this class of optimal control problems with slew rate limitations.

This work was supported by the National Science Foundation under Grant No. ECS-8310941 and the Air Force Office of Sponsored Research under Contract No. F49620-84-C-0004.

1. Introduction

The work of Rubinstein et. al. [1] and Horowitz [3] on bounds for the step response of a linear RC tree concerns the system (for the case of a falling transient)

$$\sum_{\ell=1}^{N} R_{k\ell} C_\ell \dot{v}_\ell = -v_k, \qquad v_k(0) = 1, k = 1, \ldots, N. \tag{1.1}$$

It was shown that for an RC tree,

$$R_{ii} R_{jk} - R_{ki} R_{ji} \geq 0 \tag{1.2}$$

and

$$-\dot{v}_k(t) \geq 0, \qquad \forall\, t \geq 0. \tag{1.3}$$

It follows from (1.2) and (1.3) that

$$T_{Re} v_e(t) \leq g_e(t) \leq T_p v_e(t), \tag{1.4}$$

where

$$g_e(t) \triangleq \int_t^\infty v_e(\tau)\, d\tau, \tag{1.5}$$

and

$$T_p \triangleq \sum_{k=1}^{N} R_{kk} C_k, \tag{1.6}$$

$$T_{Re} \triangleq \sum_{k=1}^{N} \frac{R_{ke}^2}{R_{ee}} C_k. \tag{1.7}$$

Using (1.1) - (1.7), upper and lower bounds for $v_e(t), 0 \leq t < \infty$ were derived.

Charles Zukowski of M. I. T. made the remarkable observation that these bounds can be interpreted as solutions of certain optimal control problems. Consider, for example, the lower bound. Suppose $v_e^*(t)$ solves the following minimum-time problem in which an input $u(\cdot)$ is introduced to represent the unknown waveform $\dot{v}_e(\cdot)$:

Minimize t_f

with

$$\dot{g}_e(t) = -v_e(t), \qquad g_e(0) = T_{De}, \tag{1.8}$$

$$\dot{v}_e(t) = u(t), \qquad v_e(0) = 1, \tag{1.9}$$

$$v_e(t_f) = V_e^*, \qquad 0 < V_e^* \leq 1, \tag{1.10}$$

$$u(t) \leq 0, \qquad \forall\, t \geq 0, \tag{1.11}$$

$$T_{Re} v_e(t) \leq g_e(t) \leq T_p v_e(t), \qquad \forall\, t \geq 0. \tag{1.12}$$

We denote this minimum time $t_{min}(V_e^*)$, since it depends on the "target" voltage V_e^*. Since $v_e(0) \geq V_e^*$ and $v_e(\cdot)$ is continuous, $v_e(t) \geq V_e^*$, $\forall\, t \in [0, t_{min}(V_e^*)]$: in particular V_e^* is a lower bound on $v_e(t_{min}(V_e^*))$. Thus the inverse of the function $t_{min}(V_e^*)$, denoted here $v_{e,min}(t)$, is a lower bound on all solutions $v_e(t)$ of (1.8) – (1.12).

This interpretation allows a particularly simple derivation of the bounds in [1]. We can also use the optimal control approach to derive tighter bounds that result from imposing additional constraints on the control $u(\cdot)$. Yu [2] has shown that for any RC tree and any node e there exists a $T_{ee} > 0$ such that

$$-\frac{\dot{v}_e(t)}{v_e(t)} \leq \frac{1}{T_{ee}}, \qquad \forall\, t \geq 0. \tag{1.13}$$

In terms of (1.9) and (1.11), (1.13) translates into a new constraint on the control,

$$u(t) \geq -\frac{1}{T_{ee}} v_e(t). \tag{1.14}$$

The nature of the optimal trajectory for the minimum-time problem with the additional constraint (1.14) is discussed in [2] for some special cases. The purpose of this memo is to provide a complete and rigorous derivation of the solutions of the optimal control problems for both maximum-time (upper bound) and minimum-time (lower bound) and for all values of T_{ee}, without invoking Pontryagin's maximum principle [4]. The problem of deriving a numerical value for T_{ee} for a given tree was studied in detail in [2] and will not be discussed here.

2. Minimum-Time Trajectory

Problem Statement (Minimum-Time Problem) :

For each "target" voltage $V_e^*, 0 < V_e^* \leq 1$, determine

$$\min_{u \in \mathcal{U}} \quad t_f$$

with

System Dynamics :

$$\dot{g}_e(t) = -v_e(t), \qquad g_e(0) = T_{De}; \tag{2.1}$$

$$\dot{v}_e(t) = u(t), \qquad v_e(0) = 1; \tag{2.2}$$

Target :

$$v_e(t_f) = V_e^*, \qquad 0 < V_e^* \leq 1; \tag{2.3}$$

Admissible Control Set :

$$\mathcal{U} \triangleq \left\{ \text{All piecewise continuous} \quad u(\cdot) \mid -\frac{1}{T_{se}} v_e(t) \leq u(t) \leq 0, \quad 0 \leq t \leq t_f \right\} \tag{2.4}$$

State Constraints :

$$T_{Re} v_e(t) \leq g_e(t) \leq T_p v_e(t). \tag{2.5}$$

The solution of the minimum-time problem can be derived from the following two lemmas, without resorting to optimal control theory.

Lemma 2.1

Let $u^*(t), 0 \leq t \leq t_f$ be the optimal control of the minimum-time problem *without* the state constraints (2.5). If the solution $g_e^*(t)$ and $v_e^*(t)$ of (2.1) and (2.2) with control $u(t) = u^*(t)$ does not violate the state constraints (2.5), then $u^*(t)$ is also the optimal control for the minimum-time problem *with* state constraints (2.5).

The proof of Lemma 2.1 is trivial and is omitted. Its value is that the optimal strategy for the problem *without* state constraints is particularly simple : to drive $v_e(t)$ from 1 to $V_e^* < 1$ as fast as possible, simply decrease $v_e(t)$ at the maximum allowable rate at each t, i.e.

$$u^*(t) = -\frac{1}{T_{se}} v_e^*(t). \tag{2.6}$$

(To see this rigorously, assume $u(\cdot)$, piecewise continuous and that $u(t) > u^*(t), \quad \forall\ t \in J$ where $J \subseteq [0, t_f]$ is a nonzero time interval. Therefore $\dot{v}_e(t) = u(t) \geq -v_e(t)/T_{se} \Rightarrow T_{se}\dot{v}_e(t) + v_e(t) = \delta(t) \geq 0 \quad \forall\ t \in [0, t_f]$ and $\delta(t) > 0 \quad \forall\ t \in J$. By integration, we obtain $v_e(t) = v_e(0)\exp\{-t/T_{se}\} + \int_0^t \delta(t-\tau)\exp\{-\tau/T_{se}\}\,d\tau > v_e(0)\exp\{-t/T_{se}\} = v_e^*(t)$, since both $\delta(t)$ and the impulse response $\exp\{-t/T_{se}\}$ are positive over J.)

Morecver the trajectory under the action of $u^*(\cdot)$ is a straight line with slope T_{se} in the $g - v$ plane because

$$\frac{dg_e^*}{dv_e^*} = \frac{\dot{g}_e^*(t)}{\dot{v}_e^*(t)} = \frac{-v_e^*(t)}{u^*(t)} = T_{se}. \tag{2.7}$$

Lemma 2.2

Consider two trajectories I and II with common initial and final states A and B, such that I lies entirely above II as shown in Fig. 2.1. Then the time taken to reach B from A along I is strictly longer than that along II.

Proof :

Since $\frac{dg_e}{dt} = -v_e$, the time taken to reach B from A along any path P in the plane is given by

$$t^P_{A-B} = \int_P -\frac{1}{v}\, dg.$$

Since path *I* lies *above* *II* , path *II* must lie to the *right* of *I*, as shown in Fig. 2.1. Therefore

$$\begin{aligned} t^I_{A-B} &= \int_I -\frac{1}{v}\, dg = \int_{g_B}^{g_A} \frac{1}{v^{(I)}}\, dg \\ &> \int_{g_B}^{g_A} \frac{1}{v^{(II)}}\, dg \qquad \text{(since } v^{(II)} > v^{(I)} \text{ for each fixed } g.) \\ &= \int_{II} -\frac{1}{v}\, dg = t^{II}_{A-B}. \end{aligned}$$

■

2.1. Case A : $0 \le T_{se} \le T_{Re}$

See Figs. 2.2(i – iii).

Proposition 2.1

(i) For $1 \ge V_e^* \ge \frac{T_{De} - T_{se}}{T_p - T_{se}}$,

$$t_f^* = T_{se} \ln\left[\frac{1}{V_e^*}\right]. \tag{2.8}$$

(ii) For $\frac{T_{De} - T_{se}}{T_p - T_{se}} \ge V_e^* \ge \frac{T_{Re} - T_{se}}{T_p - T_{se}}$,

$$t_f^* = T_{De} - T_{se} - (T_p - T_{se})V_e^* + T_{se} \ln\left[\frac{1}{V_e^*}\right]. \tag{2.9}$$

(iii) For $\frac{T_{Re} - T_{se}}{T_p - T_{se}} \ge V_e^* > 0$,

$$t_f^* = T_{De} - T_{se} + T_{Re} \ln\left[\frac{1}{V_e^*}\frac{T_{Re} - T_{se}}{T_p - T_{se}}\right] + T_{se} \ln\left[\frac{T_p - T_{se}}{T_{Re} - T_{se}}\right]. \tag{2.10}$$

Proof :

(i) From Fig. 2.2(i), V_c^* can be reached from A by decreasing $v_c(t)$ at the maximum allowable rate, without violating the state constraints (2.5) This is in fact the optimal strategy for the minimum-time problem without the state constraints (2.5). Hence path AB is the optimal trajectory by Lemma 2.1. Therefore

$$\dot{v}_c^*(t) = u^*(t) = -\frac{1}{T_{oc}} v_c^*, \qquad v_c^*(0) = 1,$$

$$\Rightarrow \qquad v_c^*(t) = \exp\{-\frac{t}{T_{oc}}\}$$

$$\Rightarrow \qquad t_f = T_{oc} \ln\left[\frac{1}{V_c^*}\right].$$

(ii) We claim that trajectory ABC in Fig. 2.2(ii) is optimal. To see this, consider any other trajectory which is has the same initial condition $(1, T_{Dc})$ and terminates at the target set, but different from ABC on some portion of its path. Such a trajectory either lies entirely above ABC, such as *I*, or meets ABC at some point, say B', such as *II*. *I* is nonoptimal by Lemma 2.2. *II* is also nonoptimal because along *II* AB' takes longer than ABB' by Lemma 2.2 and B'C' takes longer than B'C by Lemma 2.1. Adding the times along AB and BC yields (2.9).

(iii) Path ABCD in Fig. 2.2(iii) is optimal by the same reasoning in part (ii). Adding the times along AB, BC and CD yields (2.10).

■

2.2. Case B : $T_{Rc} \leq T_{oc} \leq T_{Dc}$

See Figs. 2.3(i – ii).

Proposition 2.2

(i) For $1 \geq V_c^* \geq \frac{T_{Dc} - T_{oc}}{T_p - T_{oc}}$, the optimal trajectory is AB (Fig. 2.3(i)) and

$$t_f^* = T_{oc} \ln\left[\frac{1}{V_c^*}\right]. \tag{2.11}$$

(ii) For $\frac{T_{Dc} - T_{oc}}{T_p - T_{oc}} \geq V_c^* > 0$, the optimal trajectory is ABC (Fig. 2.3(ii)) and

$$t_f^* = T_{Dc} - T_{oc} - V_c^*(T_p - T_{oc}) + T_{oc} \ln\left[\frac{1}{V_c^*}\right]. \tag{2.12}$$

The proof of Proposition 2.2 is exactly the same as that of Proposition 2.1 and is omitted.

2.3. Case C : $T_{Dc} < T_{oc}$

In this case, the minimum value of V_c^* attainable from $v_c(0) = 1$ using admissible controls $u(\cdot) \in \mathcal{U}$ without violating the state constraints (2.5) is $(T_{Dc} - T_{oc})/(T_{Rc} - T_{oc})$, as is apparent from Fig. 2.4. No trajectory can satisfy all the constraints and tend asymptotically to the origin as $t \to \infty$. In terms of the RC tree problem, case C can only arise through a modelling error, so we will not prusue it further.

3. Maximum-Time Trajectory

Problem Statement (Maximum-Time Problem) :

For each "target" voltage $V_e^*, 0 < V_e^* \leq 1$, determine

$$\max_{u \in \mathcal{U}} \quad t_f$$

with

System Dynamics :

$$\dot{g}_e(t) = -v_e(t), \qquad g_e(0) = T_{De}; \tag{3.1}$$

$$\dot{v}_e(t) = u(t), \qquad v_e(0) = 1; \tag{3.2}$$

Terminal Time Constraint :

$$v_e(t_f) \geq V_e^*, \qquad 0 < V_e^* \leq 1; \tag{3.3}$$

Admissible Control Set :

$$\mathcal{U} \triangleq \left\{ \text{All piecewise continuous } u(\cdot) \mid -\frac{1}{T_{se}} v_e(t) \leq u(t) \leq 0, \quad 0 \leq t \leq t_f \right\} \tag{3.4}$$

State Constraints :

$$T_{Re} v_e(t) \leq g_e(t) \leq T_p v_e(t). \tag{3.5}$$

The maximum-time problem is easier than the minimum-time problem and its solution for different values of T_{se} and V_e^* can be derived using Lemma 2.2 only. We state the solutions below. Proofs are omitted.

3.1. Case A : $0 \leq T_{se} \leq T_{De}$

See Figs. 3.1(i – ii).

Proposition 3.1

(i) For $1 \geq V_e^* \geq \frac{T_{De} - T_{se}}{T_p - T_{se}}$, the optimal trajectory is AB (Fig. 3.1(i)) and

$$t_f^* = T_{se} \ln \left[\frac{1}{V_e^*} \right] + \frac{1}{V_e^*} \{ (T_{se} - T_{Re}) V_e^* + (T_{De} - T_{se}) \}. \tag{3.6}$$

(ii) For $\frac{T_{De} - T_{se}}{T_p - T_{se}} \geq V_e^* > 0$, the optimal trajectory is ABCD (Fig. 3.1(ii)) and

$$t_f^* = T_p - T_{Re} + T_{se} \ln \left[\frac{T_p - T_{se}}{T_{De} - T_{se}} \right] + T_p \ln \left[\frac{1}{V_e^*} \frac{T_{De} - T_{se}}{T_p - T_{se}} \right]. \tag{3.7}$$

3.2. Case C : $T_{De} < T_{se}$

The results here are identical to those in section 2.3.

4. Conclusion

We invert, where possible, the expressions derived in Props. 2.1 – 2.3 and Props. 3.1 – 3.2 to obtain the final expressions below, for the upper and lower waveform bounds. For the case of $T_{De} < T_{se}$, where not all values of $V_e^* < 1$ are attainable, the upper and lower bounds are not meaningful and therefore are omitted here.

4.1. Lower Bounds

4.1.1. Case A : $0 \le T_{se} \le T_{Re}$

(i) For $0 < t \le T_{se} \ln\left[\frac{T_p - T_{se}}{T_{De} - T_{se}}\right]$,

$$V_e^*(t) = \exp\left\{-\frac{t}{T_{se}}\right\}. \tag{4.1}$$

(ii) For $T_{se} \ln\left[\frac{T_p - T_{se}}{T_{De} - T_{se}}\right] < t < T_{De} - T_{se} + T_{se} \ln\left[\frac{T_p - T_{se}}{T_{Re} - T_{se}}\right]$, there is no explicit function for $V_e^*(t)$, because (2.9) is not invertible.

(iii) For $t \ge T_{De} - T_{se} + T_{se} \ln\left[\frac{T_p - T_{se}}{T_{Re} - T_{se}}\right]$,

$$V_e^*(t) = \frac{T_{Re} - T_{se}}{T_p - T_{se}} \exp\left\{-\frac{t - (T_{De} - T_{se}) - T_{se} \ln[(T_p - T_{se})/(T_{Re} - T_{se})]}{T_{Re}}\right\}. \tag{4.2}$$

4.1.2. Case B : $T_{Re} \le T_{se} \le T_{De}$

(i) For $0 < t \le T_{se} \ln\left[\frac{T_p - T_{se}}{T_{De} - T_{se}}\right]$,

$$V_e^*(t) = \exp\left\{-\frac{t}{T_{se}}\right\}. \tag{4.3}$$

(ii) For $t > T_{se} \ln\left[\frac{T_p - T_{se}}{T_{De} - T_{se}}\right]$, there is no explicit expression for $V_e^*(t)$, because (2.12) is not invertible.

4.2. Upper Bounds

4.2.1. Case A : $0 \le T_{se} \le T_{De}$

(i) For $0 \le t < T_p - T_{Re} + T_{se} \ln\left[\frac{T_p - T_{se}}{T_{De} - T_{se}}\right]$, there is no explicit expression for $V_e^*(t)$, because (3.6) is not invertible.

(ii) For $t \ge T_p - T_{Re} + T_{se} \ln\left[\frac{T_p - T_{se}}{T_{De} - T_{se}}\right]$,

$$V_e^*(t) = \frac{T_{De} - T_{se}}{T_p - T_{se}} \exp\left\{-\frac{t - (T_p - T_{Re}) - T_{se} \ln[(T_p - T_{se})/(T_{De} - T_{se})]}{T_p}\right\} \tag{4.4}$$

Acknowledgement

It is a pleasure to acknowledge helpful conversations with Charles Zukowski, Prof. Dimitri Bertsekas and Prof. Paul Penfield of the Department of Electrical Engineering and Computer Science at Massachusetts Institute of Technology and Prof. Violet Haas of the School of Electrical Engineering at Purdue University.

Reference

[1] J. Rubinstein, P. Penfield, Jr., and M. A. Horowitz,"*Signal Delay in RC Tree Networks*", IEEE Transc. on CAD, CAD-2(3):202 - 211, July 1983.

[2] Q. Yu and J. L. Wyatt, Jr., "*Improved Bounds on Signal Delay in RC Trees Using Inequalities on the Derivatives of Node Voltages*", VLSI Memo. 84-197, August 1984.

[3] M. A. Horowitz,"*Timing Models for MOS Circuits*", Technical Report No. SEL 83-003, Stanford University, December 1983.

[4] E. B. Lee and L. Markus,**Foundations of Optimal Control Theory**, Wiley, New York, 1967.

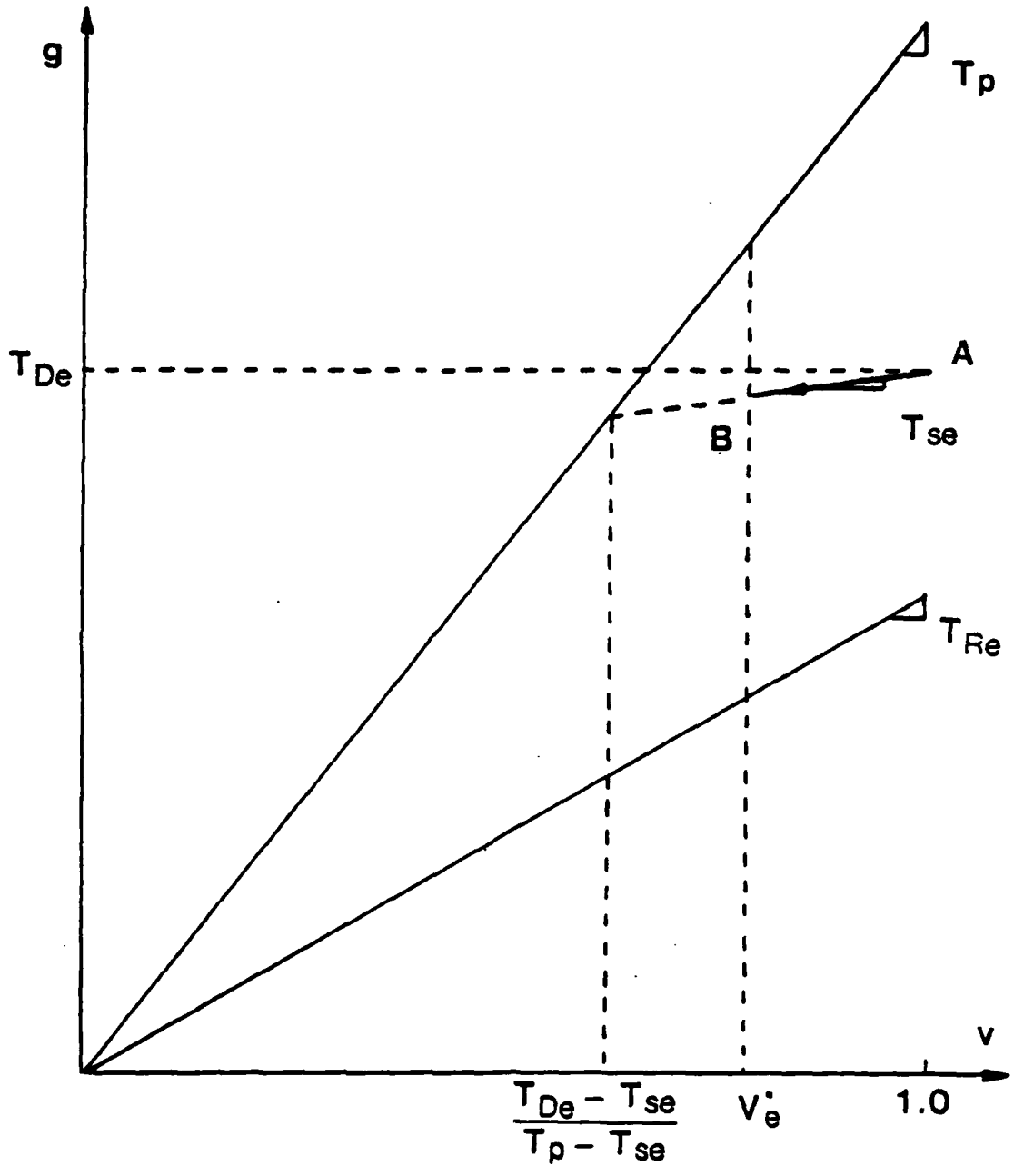


Fig. 2.2(i) : AB is optimal.

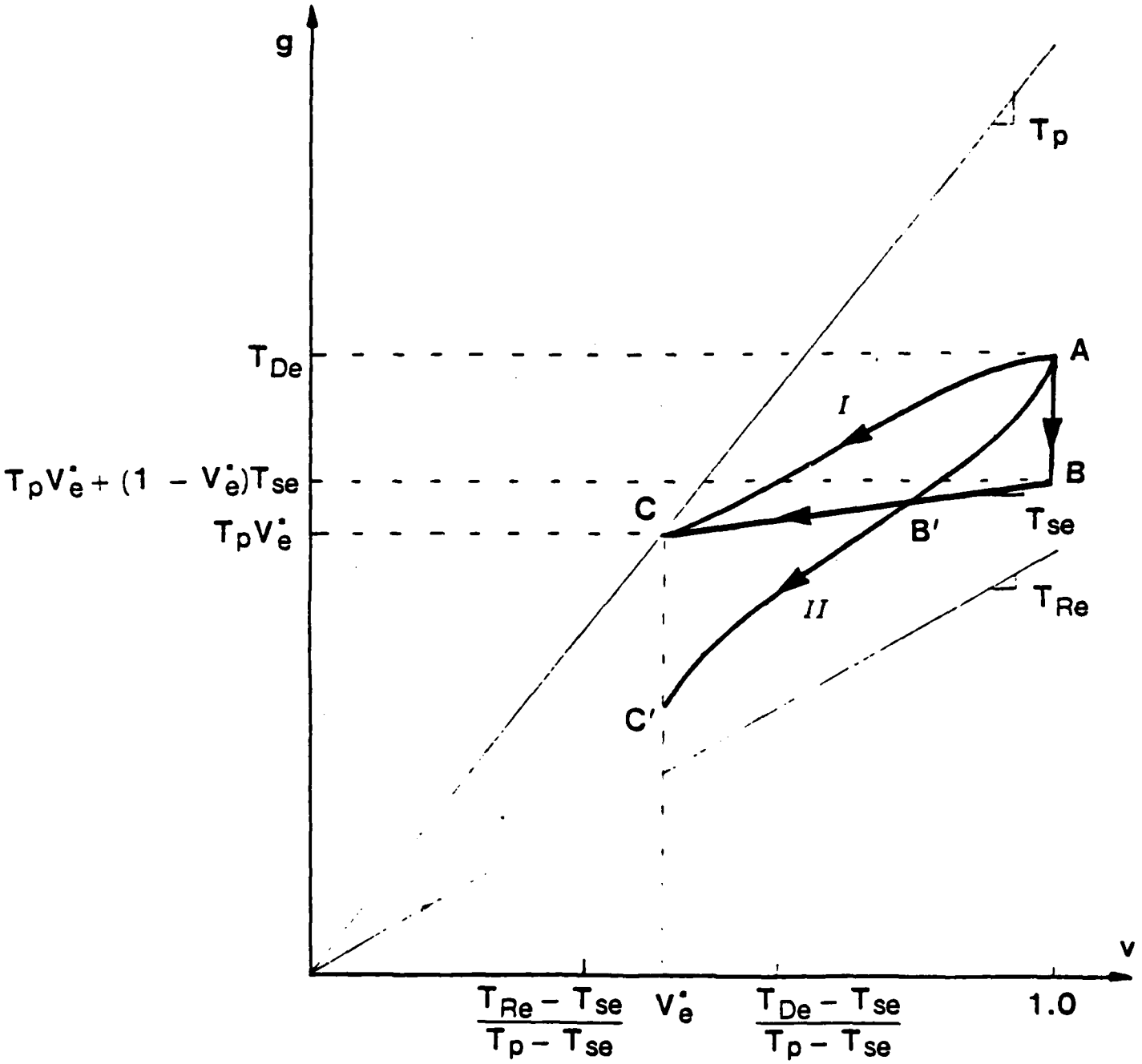


Fig. 2.2(ii) : ABC is optimal.

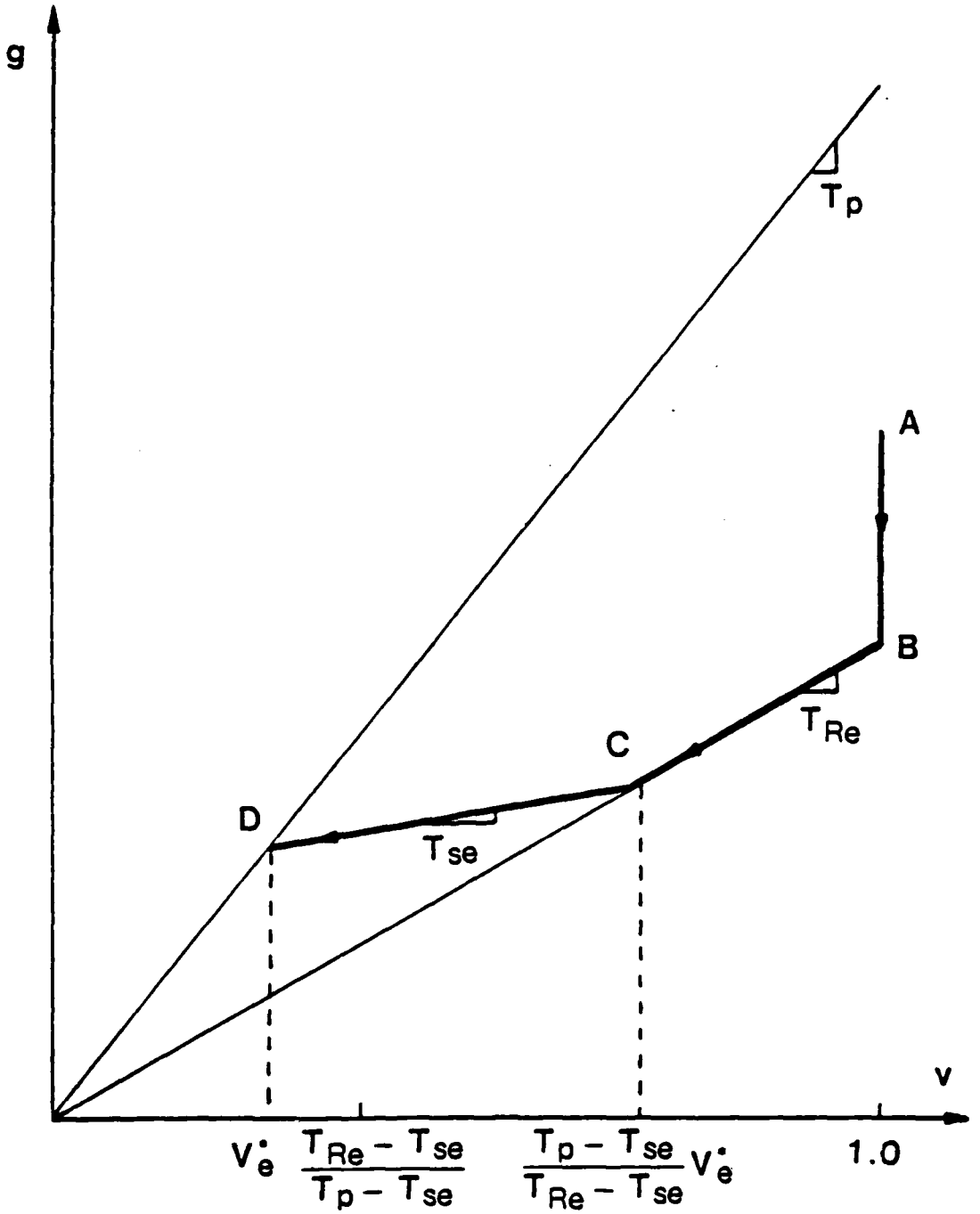


Fig. 2.2(iii) : ABCD is optimal.

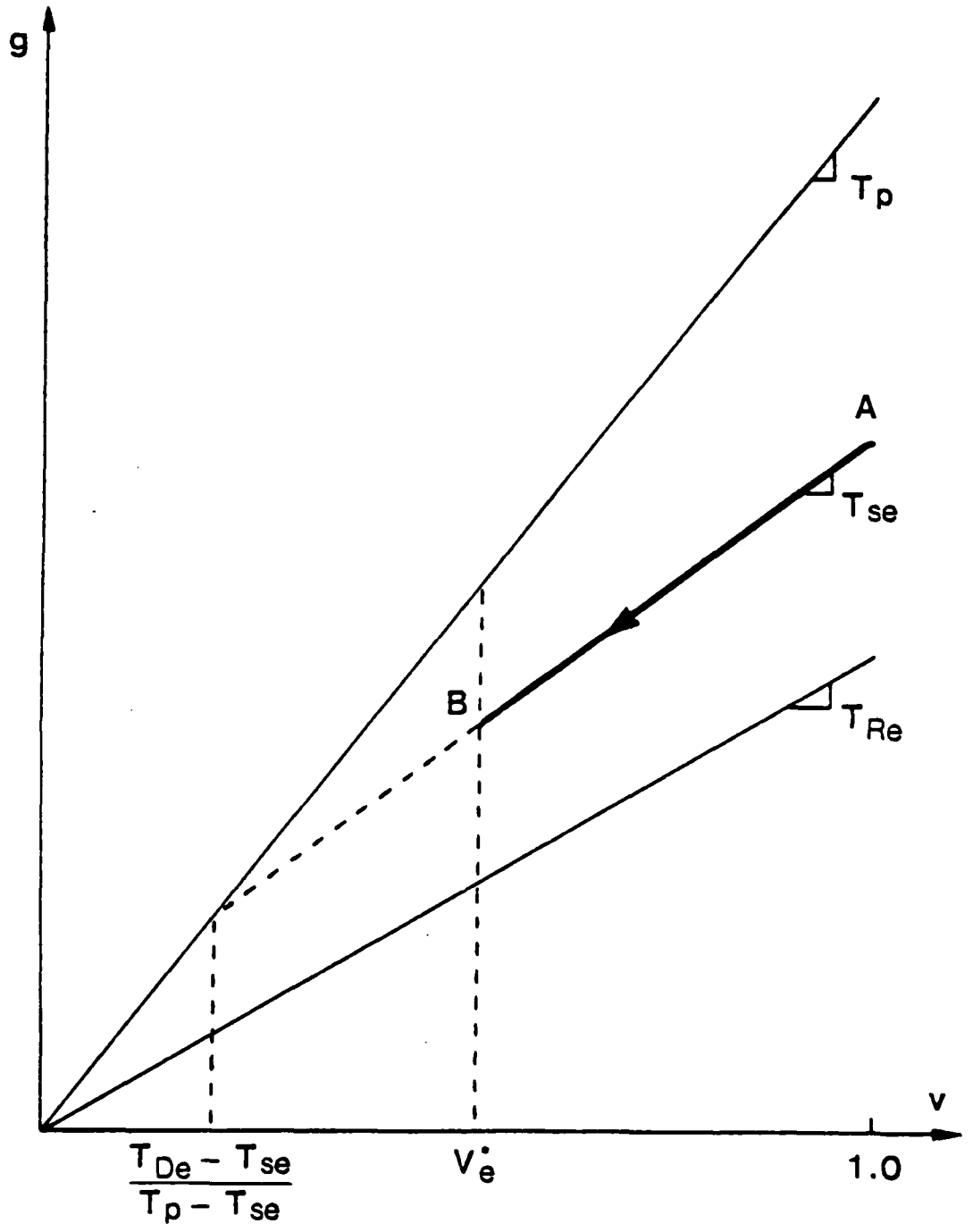


Fig. 2.3(i) : AB is optimal.

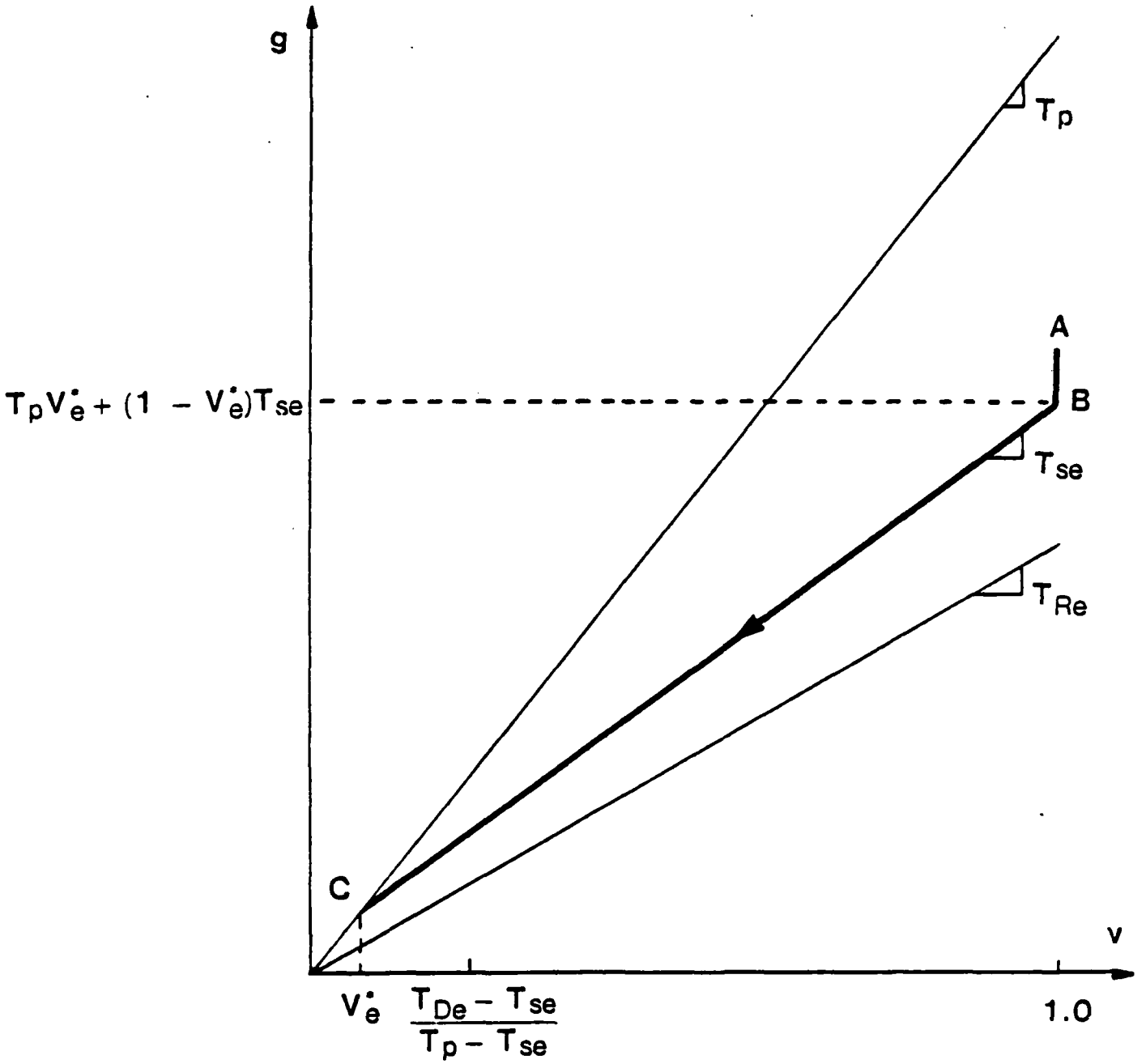


Fig. 2.3(ii) : ABC is optimal.

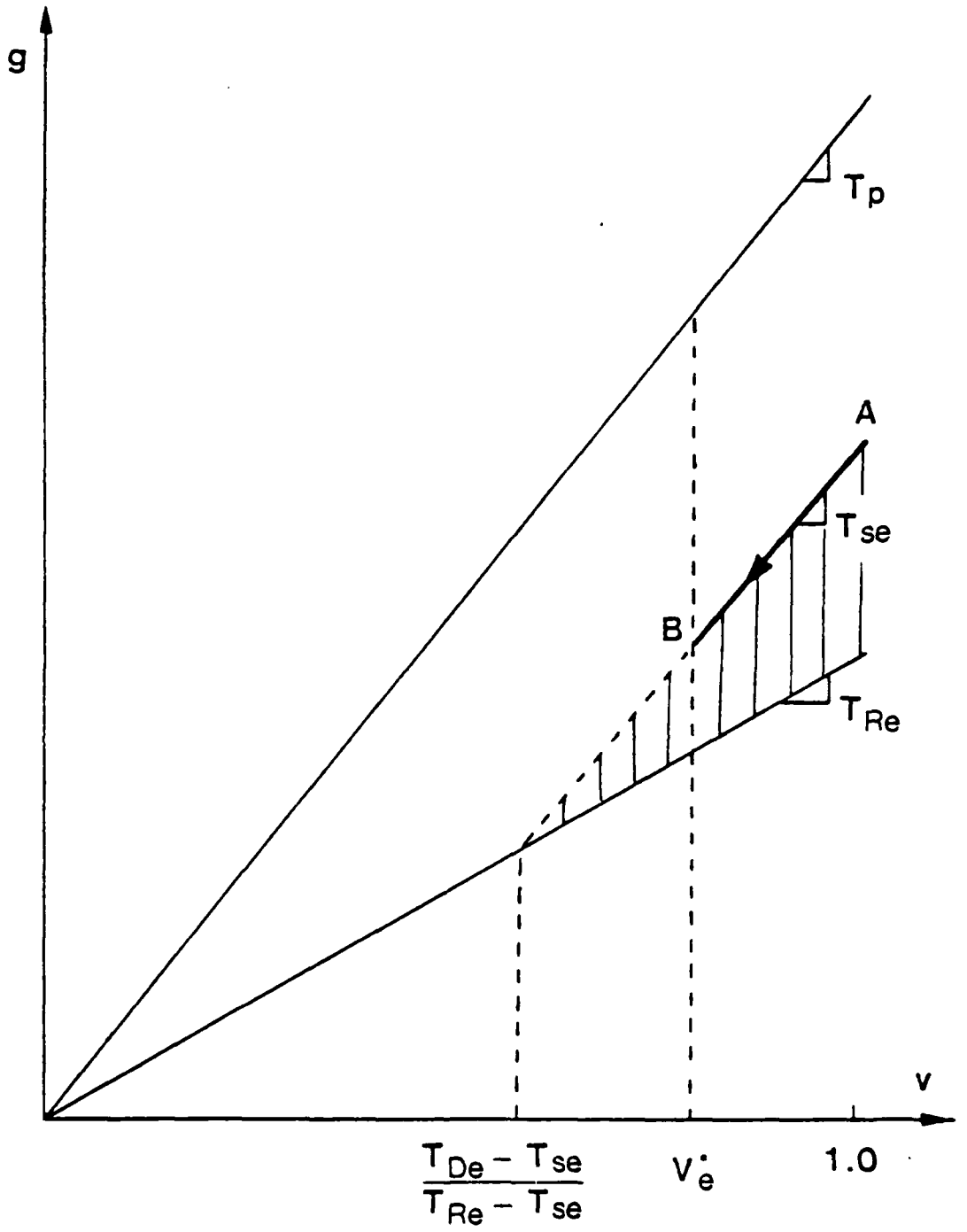


Fig. 2.4 : When $T_{se} > T_{De}$, only the shaded region can be reached without violating the constraints, and every trajectory eventually leaves the shaded region.

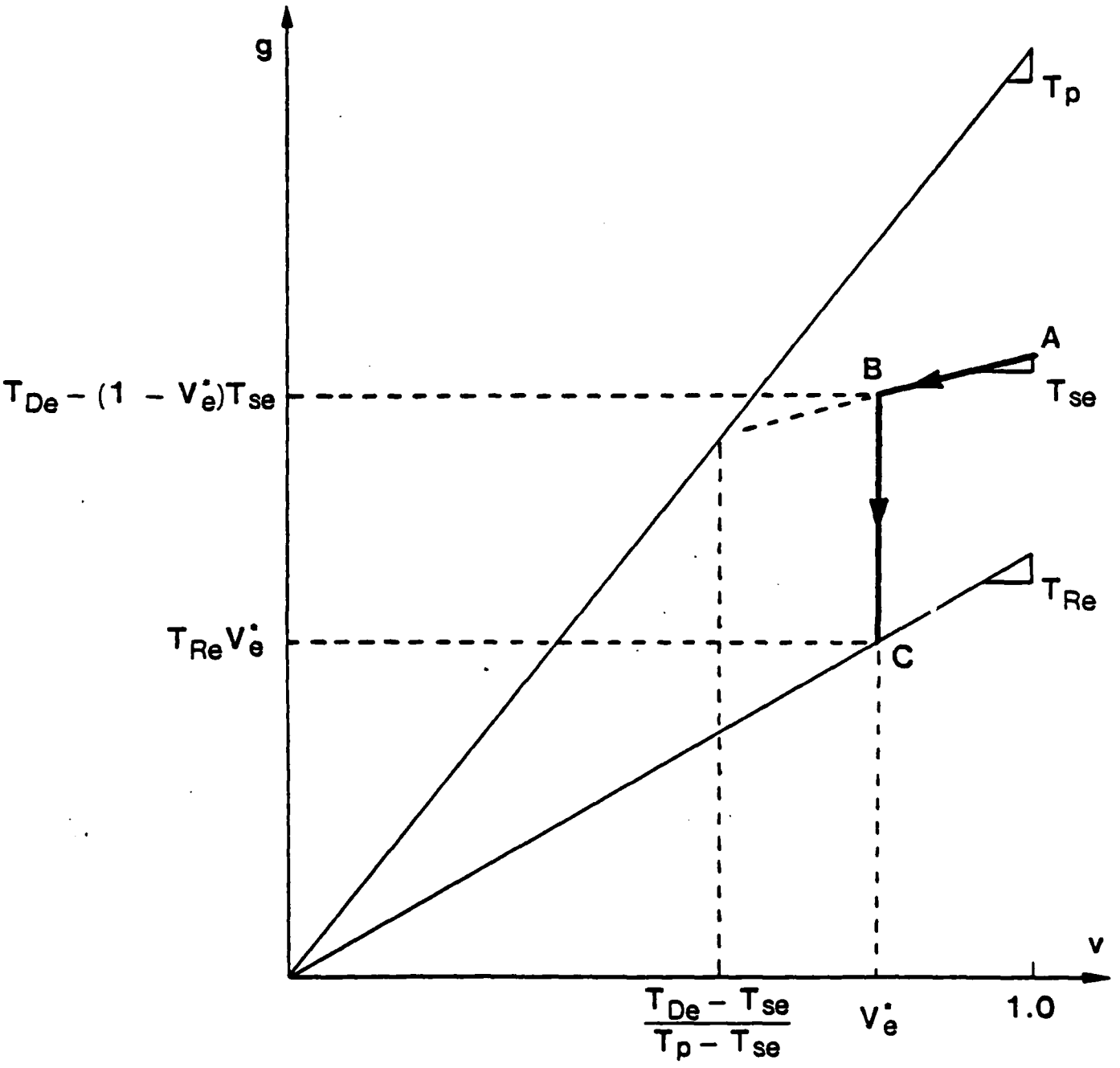


Fig. 3.1(i) : ABC is optimal.

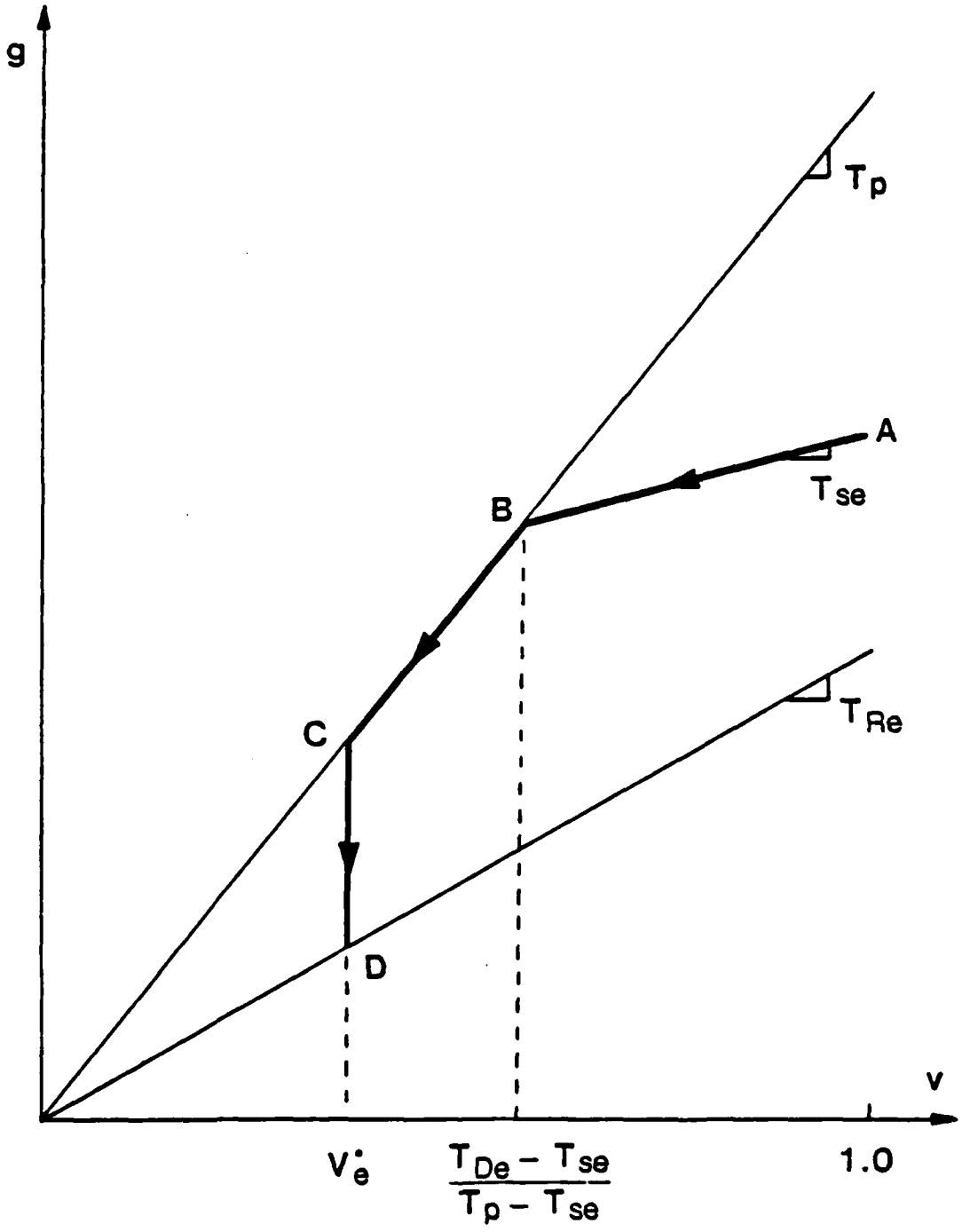


Fig. 3.1(ii) : ABCD is optimal.

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CAMBRIDGE, MASSACHUSETTS 02139

VLSI Memo No. 84-211 November 1984

A Circuit Grammar for Operational Amplifier Design*

Andrew Lewis Ressler**

Electrical circuit designers seldom create really new topologies or use old ones in a novel way. Most designs are known combinations of common configurations tailored for the particular problem at hand. In this thesis I show that much of the behavior of a designer engaged in such ordinary design can be modeled by a clearly defined computational mechanism executing a set of stylized rules. Each of my rules embodies a particular piece of the designer's knowledge.

A circuit is represented as a hierarchy of abstract objects, each of which is composed of other objects. The leaves of this tree represent the physical devices from which physical circuits are fabricated. By analogy with context-free languages, a class of circuits is generated by a phrase-structure grammar, of which each rule describes how one type of abstract object can be expanded into a combination of more concrete parts.

Circuits are designed by first postulating an abstract object which meets the particular design requirements. This object is then expanded into a concrete circuit by successive refinement using rules of my grammar. There are in general many rules which can be used to expand a given abstract component. Analysis must be done at each level of the expansion to constrain the search to a reasonable set. Thus the rules of my circuit grammar provide constraints which allow the approximate qualitative analysis of partially instantiated circuits. Later, more careful analysis in terms of more concrete components may lead to the rejection of a line of expansion which at first looked promising. I provide special failure rules to direct the repair in this case. As part of this research I have developed a computer program, CIROP, which implements my theory in the domain of operational amplifier design.

---

*This thesis was submitted to the Department of Electrical Engineering and Computer Science in January 1984 in partial fulfillment of the requirements for the degree of Doctor of Philosophy. The research was done at the Artificial Intelligence Laboratory at M.I.T. and was supported in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract No. N00014-80-C-0622.

**Currently: 5758 Flag Flower Place, Columbia, MD 21045; (301) 992-9654.

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CAMBRIDGE, MASSACHUSETTS 02139

VLSI Memo No. 84-212 November 1984

Macromodeling of Digital MOS VLSI Circuits*

Mark D. Matson**

ABSTRACT

This paper presents a method for modeling MOS combinational logic gates. Analyses are given for power consumption, output response delay, output response waveshape, and input capacitance. The models are both computationally efficient and accurate, typically lying within 5% of SPICE estimates. They are pertinent to simulation and optimization applications. A general macromodeling software support package is described. A companion paper discusses a circuit optimizer based on these models.

---

*This work was supported in part by an RCA fellowship, in part by the Defense Advanced Research Projects Agency of the Department of Defense under Contract No. N00014-80-0622, and in part by the Air Force Office of Sponsored Research under Contract No. F49620-84-C-0004.

**Department of Electrical Engineering and Computer Science, M.I.T., Room 36-575, Cambridge, MA 02139; (617) 253-8169.

Macromodeling of Digital MOS VLSI Circuits

Abstract

This paper presents a method for modeling MOS combinational logic gates. Analyses are given for power consumption, output response delay, output response waveshape, and input capacitance. The models are both computationally efficient and accurate, typically lying within 5% of SPICE estimates. They are pertinent to simulation and optimization applications. A general macromodeling software support package is described. A companion paper discusses a circuit optimizer based on these models.

1. Introduction

This paper discusses accurate, computationally efficient models for MOS logic gates. The models are well suited for simulation and optimization of high performance VLSI circuits. The models are based on device equations, and acquire much of their accuracy through careful consideration of waveshape effects.

The significance of waveshape effects has been investigated by other workers. Crystal [1], a timing simulator, models transistors as resistors, but uses different values for transistor resistances depending on input waveform. While this leads to good accuracies (typically within 10% of SPICE predictions), the approach does have some limitations. For example, the tables of effective transistor resistances depend on a uniform trigger voltage (the point on a logic gate's transfer curve where $v_{OUT} = v_{IN}$) and can produce substantial errors if this restriction is removed, for instance by varying beta ratios. Moreover the table interpolations can generate jagged delay functions; this can make the optimization task more difficult.

For these reasons we chose to base our models entirely on device equations. Horowitz [2] pursued a similar strategy in modeling the delay of a MOS inverter. He derived equations for the gate's response and then obtained estimates of parameters from the gate's drive curves (curves of v_{OUT} versus v_{IN} for different values of load current).

In this paper we describe more general and sophisticated models. We develop equations for power consumption, output waveform, and input capacitance of a general MOS logic gate. To obtain high accuracy in the model, we wrote a macromodeling support package to determine the equations' parameters. The package curve fits the model equations to SPICE simulation results and finds the parameter set which provides the highest accuracy.

Section 2 discusses the basic principles of the macromodeling approach. Section 3 presents models for MOS inverters. We begin with a resistor-capacitor model and discuss its limitations. We then develop a more elaborate model, one accounting for waveform shape effects. The analysis is extended to more general logic gates in section 4. The theory gives us the form of the macromodel equations. In section 5 we describe how the equations' parameters are determined with a sophisticated macromodeling support package.

2. Motivation and Intent

Circuit optimization is a computationally expensive process. It is an iterative procedure, requiring multiple simulations at each step to evaluate delays and their gradients. Moreover, high performance circuit design requires fairly accurate delay estimates, but using a device level simulator would be out of the question for all but small circuits.

Since it is too time consuming to compute circuit responses during the optimization, we instead pursue an approach where much of the work is performed prior to the optimization. We divide a large circuit into many small pieces. This partitioning is done such that the pieces have limited, well understood interactions, while the elements inside the pieces have strong, complex interactions. Thus computing the interactions among elements within a piece would be very expensive, and it behooves us to characterize the behavior of the pieces beforehand to avoid having to compute it during the optimization.

This approach is called macromodeling. In the digital MOS domain, candidates for pieces would be cells such as logic gates and storage elements. We model the attributes of the cells as functions of the cell's internal description and boundary conditions. In particular, we are concerned with a cell's power, input load, and output waveform attributes. The cell's internal description consists of its transistor sizes, layout parasitics, and process parameters. Boundary conditions are imposed on the cell by external agents. These include input waveforms from drivers and output loading from receivers and wiring capacitances. We characterize waveforms as time-shifted ramps with exponential tails. This waveshape is representative of those found in digital MOS circuits.[1] Figure 1 displays an example. The chain of inverters is driven by a falling input waveform; the figure shows the output waveform of each gate. Here T_{BE} denotes the time shift, and T_{SW} the time constant of the exponential portion. Conceptually T_{BE} is the time until the output begins to move in response to an input transition, and T_{SW} is a measure of how quickly the output switches once it does begin to change. We curve fit actual circuit waveforms to the time-shifted ramps with exponential tails. From the figure we see that the output waveform of the chain of inverters is described by

$$chain\,T_{BEout} = \sum_{i=1}^{n} T_{BEi}$$
$$chain\,T_{SWout} = T_{SWoutn}$$

We characterize output loads in terms of an effective capacitance, dividing charge transferred by

---

[1] Actual circuit waveforms begin more smoothly than our approximation. However the error is negligible because the logic gate driven by the waveform does not really begin to switch until the waveform until the waveform reaches $V_{trigger}$ (the point on the dc transfer curve where $V_{in} = V_{out}$) and is therefore insensitive to the shape of the first part of the waveform.

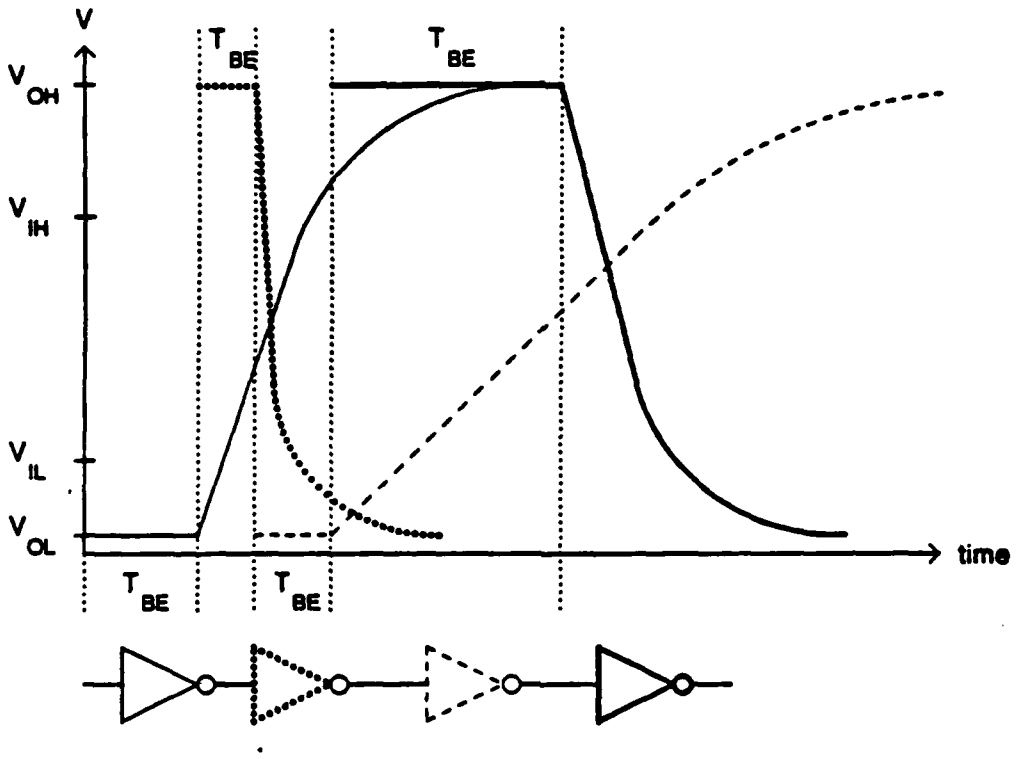


Figure 1: Waveform Characterization

change in voltage. This allows us to model RC interconnection networks, since the effective capacitance can be a function of waveform slope.

We "black-box" the cell as shown in Figure 2. The cell is affected by its environment via the boundary conditions T_{SWin} and C_L. It interacts with its neighbors via its interface attributes C_{in} and T_{SWout}. The internal attributes power and T_{BEout} are isolated from the environment and have no influence on the attributes of the cell's neighbors.

3. Inverters

We begin our macromodeling analysis with the ubiquitous inverter, illustrated in Figure 3. The results will be extended to more general gates in a subsequent section. For the sake of conciseness, our analysis is only shown for rising input, falling output nMOS gates. The macromodel equations for the opposite transition and for CMOS are similar. We will present the actual macromodel equations (for both transitions) in the section on general logic gates.

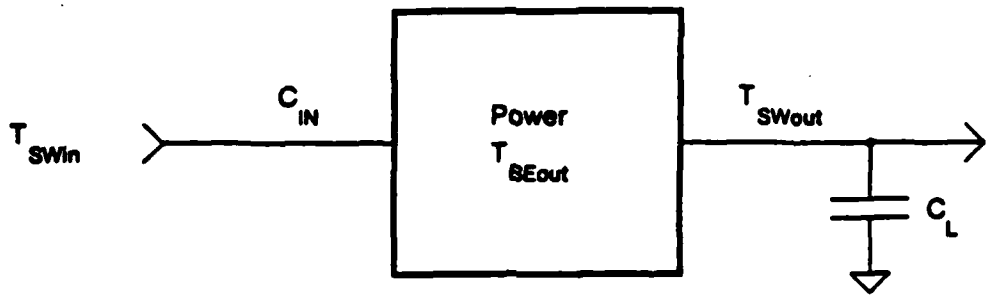


Figure 2: Macromodel Representation

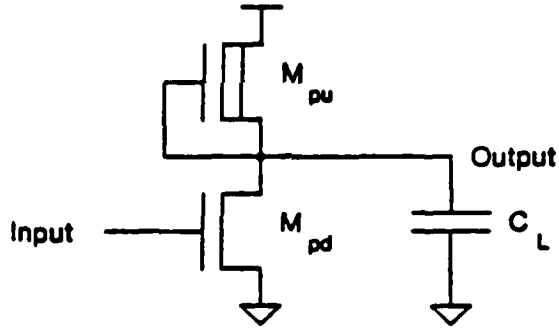


Figure 3: A Depletion Load nMOS Inverter

3.1. Objective Function

The practicing engineer typically must design circuits such that they satisfy delay specifications. The engineer also desires to minimize some objective function subject to those delay constraints. Power dissipation is a major concern in ratioed nMOS technology. We accordingly choose to minimize power dissipation, which for nMOS is dominated by static power consumption. The static power consumed by an nMOS inverter is roughly proportional to the shape factor of the pullup; that is,

$$Power = a_1 + a_2 S_{pu}$$

where a_1 and a_2 are constants that depend on the fabrication process and power supply voltage.

The choice of an objective function for CMOS circuits is not as clear. Typically a designer wishes to minimize area, power dissipation, or some combination of the two. Characterizing the area consumption is difficult because it is highly dependent on layout style. However we can easily describe the contribution of the transistors. This is simply

$$Area = Poly\ Pitch \times \sum_{i=1}^{n} stack\ width_i$$

where we have omitted the transistor lengths because for CMOS they are set to the minimum channel length.

3.2. Output Waveform

3.2.1. Resistive Model

Computational limitations mandate the use of a simple delay model. The simplest transistor representation that provides tolerable accuracy is a switched resistor. The MOS transistor is modeled with a capacitor from the gate to ground and a switched linear resistor from drain to source. The gate to source voltage controls whether the resistor is switched on or off. The delay characteristics of the model, along with their implications for circuit optimization, have been analyzed in [3] and [4]. The principal advantage of the model is its simplicity, which allows one to derive closed form expressions for the optimal transistor sizes, leading to fast run times. Unfortunately the model can be alarmingly inaccurate. Moreover the errors can be exacerbated by the optimization. For a chain of similar gates where the capacitive loading on each stage is dominated by the input capacitance of the next stage (rather than the wiring capacitance), pushing the chain for speed results in equal stage delays. For nMOS this virtually guarantees that while for rising outputs the stage is insensitive to input waveshape, for falling outputs the stage is highly sensitive to input slope. This sensitivity means that the transistor cannot be accurately modeled as a resistor, and the effect on total chain delay is significant because the stage delays are equal. Rising output stage delays, for which the resistive model tends to be valid, do not dominate the total delay. The model exhibits errors of up to 70%, clearly unacceptable for serious circuit design.

3.2.2. Extended Model

Faced with the inability of the resistive model to account for waveshape effects, we are compelled to derive a more elaborate model. Ever mindful of computation time limitations, we pursue the simplest possible extensions that will provide the needed accuracy. We begin by studying the inverter's response to different input waveform slopes, paying particular attention to the different regions of transistor operation.

As the inverter's input rises and its output falls, the pullup and pulldown transistors sequence through different regions of operation. These regions are summarized in Table 1. For the fast input response[2] the bulk of the delay accrues from the last states where the pulldown is in its linear region. Hence the pulldown can be approximated by a resistor, and the resistive model works well here. However for slow inputs the pulldown is saturated for a significant portion of the transition, causing the inverter to behave like an amplifier. In this mode the inverter is highly sensitive to the input waveform and consequently the resistive model breaks down.

Fast Input Response		Slow Input Response	
pullup	*pulldown*	*pullup*	*pulldown*
linear	off	linear	off
linear	sat	linear	sat
linear	linear	sat	sat
sat	linear	sat	linear

Table 1: Pullup/Pulldown Regions of Operation

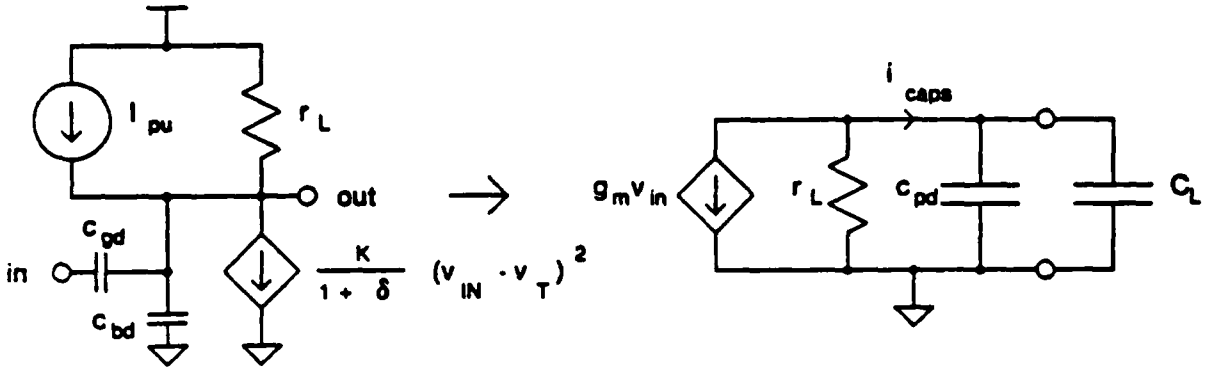
We seek a simple model that includes both amplifier and resistor behavior. We are especially concerned with the middle and latter parts of the input transition, for it is here that the inverter's output is sinking the most current. The beginning of the transition is not as crucial. For slow inputs the inverter can be modeled as an amplifier when the pulldown is saturated, and as a resistor to V_{OL} when the pulldown is in its linear region. As the input transition becomes faster, the inverter spends proportionately less time as an amplifier and more as a resistor. The mapping of the inverter to an amplifier and resistor is shown in Figure 4. For continuity of v_{OUT} and i_{OUT} when the model changes from an amplifier to a resistor, we use the resistor model when $0 \geq r_{pd}\, i_{caps} + v_{OUT}$.

The ac model used for the amplifier behavior clashes with traditional engineering philosophy.

---

[2] "Fast" means that the input transition time is fast relative to the output transition time.

Amplifier



Resistor

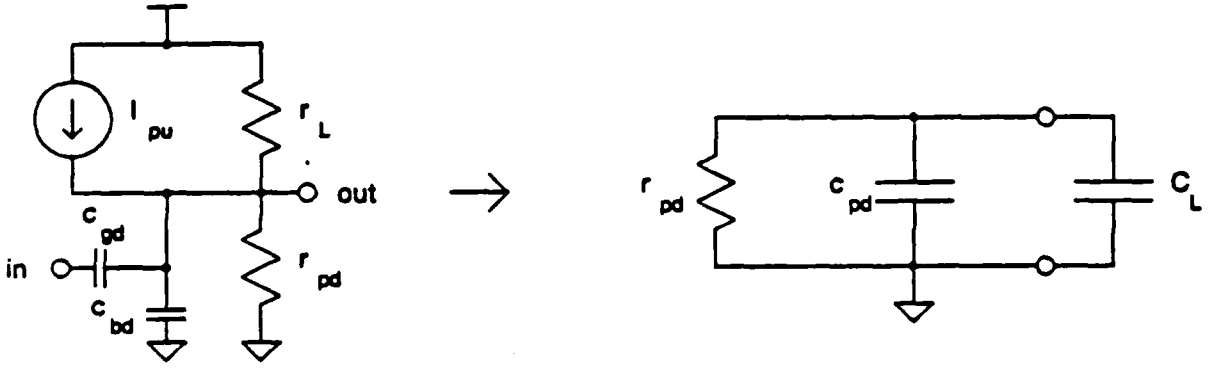


Figure 4: Delay Mapping

Normally one creates an ac model by linearizing a circuit about a quiescent operating point. Here however we are interested in large signal behavior. Consequently, while we can perturb the system from an initial point (in this case $(v_{IN}, v_{OUT}) = (V_{IL}, V_{OH})$), we have no easy method to calculate the model's parameters such as g_m and r_L. We cannot simply evaluate the parameters at a quiescent operating point because we have none. We instead view the problem at a more objective-oriented level, and seek to determine which values of g_m, r_L, etc., will provide the closest approximation to observed response times. Moreover, rather than using the same set of parameter values for the rising input and falling input responses (which would correspond to using a single group of drive curves to characterize the inverter), we desire to acquire additional accuracy by using distinct sets for each transition's begin and switch responses. This leads us to the following strategy: analytically derive expressions for the form of the macromodel equations, then curve fit to observed data to solve for the

constants in the equations.

Closed form expressions for the model's response to different input waveforms can be derived. Here we will outline the basic concepts; the actual equations will be given at the end of section 4. For very fast inputs the inverter changes from an amplifier to a resistive form immediately. In other words, the first order resistive model is valid. Figure 4 shows the model. For fast inputs the inverter model does not change immediately, but does change before the output transition completes. We use the amplifier model of Figure 4 but omit the resistor r_L. The output switches quickly enough that the current in the total capacitance C_{total} ($= C_L + c_{pd}$) dominates that of r_L, allowing us to neglect the resistor. For moderate inputs the input waveform is slow enough that the model changes after the output has fallen. The current in C_{total} still dominates that of r_L. For slow inputs, the current through r_L can no longer be neglected. Unfortunately this leads to equations which cannot be solved for closed form expressions for T_{BE} and T_{SW}. For very slow inputs, the input and output waveforms have slowed to the point where the current through C_{total} is almost negligible compared to that through r_L. The amplifier system reaches steady state, exhibiting a constant tracking error to the ramp input, being entirely limited by the speed of the input [5].

Having described a method for determining the inverter's response to various input slopes, we now seek a means of combining the results into one conglomerate expression. It is common to use smoothing functions to effect this combination. However many workers fail to consider the computational overhead incurred with these functions. We instead create simple functions that exhibit the desired behavior in each of the input slope regimes. To avoid placing any unnecessary burdens on the optimization algorithms, we choose functions that are twice continuously differentiable. Although optimization algorithms exist for solving problems with ill-behaved (eg. discontinuous) functions, because of their added generality these algorithms tend to be slower. Moreover we prefer functions that do not contain multiple maxima or minima; ie. that are unimodal. This helps eliminate cusps that could trap an optimizer's iterative solution technique. The resulting inverter equations are fully described and analyzed in [6].

3.3. Input Capacitance

Calculating a gate's delay requires knowledge of the input capacitances of the gates that it drives. In this section we study an inverter's input capacitance. Our results will be extended to more general logic gates in a later section.

We begin by considering the components of the input capacitance. Figure 5 shows our model. The input capacitance has two constituents: the gate to drain and gate to source transistor capacitances.

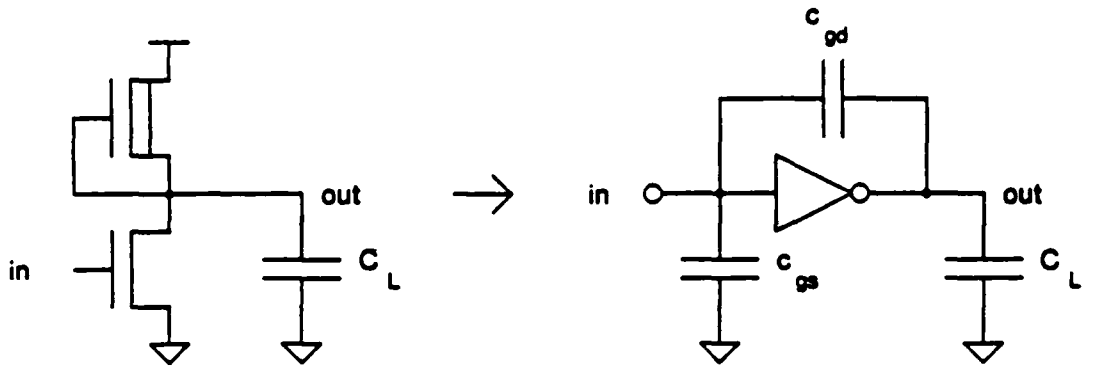


Figure 5: Input Capacitance Model

The input capacitance presented to the driver can change during the course of the input transition. This effect is largely due to the input to output coupling capacitance c_{gd}. Consider a rising input transition of moderate speed. During the beginning portion of the input waveform—that is, before the input voltage has reached V_{IL}—the inverter's output has not yet moved significantly. The input capacitance is therefore simply $c_{gs} + c_{gd}$. Both terms are proportional to the pulldown transistor's width.

As the input voltage passes V_{IL}, the inverter begins to pull its output low. Consequently the driver must supply more current to charge c_{gd} than it would have had the output voltage remained fixed. This is called the Miller effect [7]. The effective input capacitance has increased. Note that the total voltage switch across c_{gs} is always $V_{OH} - V_{OL}$, while that across c_{gd} is 2 $(V_{OH} - V_{OL})$, but we are only interested in the capacitance seen during the beginning and switching portions of the input waveform. For very fast input transitions the output will not have moved until after the input has fully switched; hence the driver will not have seen any Miller capacitance during the actual transition. As we slow the speed of the input transitions, more of the output's switching time overlaps with the input's and we see more Miller capacitance. Eventually all of the output's switching time overlaps with the input's and C_{SWin} reaches a plateau. The expected behaviors of the effective input capacitances C_{BEin} and C_{SWin} appear in Figure 6.

The analysis is complicated slightly by the fact that since c_{gd} and c_{gs} are functions of v_{GS} and v_{GD}, they not only vary as the gate switches, but their average value during the input transition changes as the input transition slows down. The outcome of this change in c_{gd} is that for rising inputs the Miller effect is significant, while for falling inputs the Miller effect is scarcely noticeable.

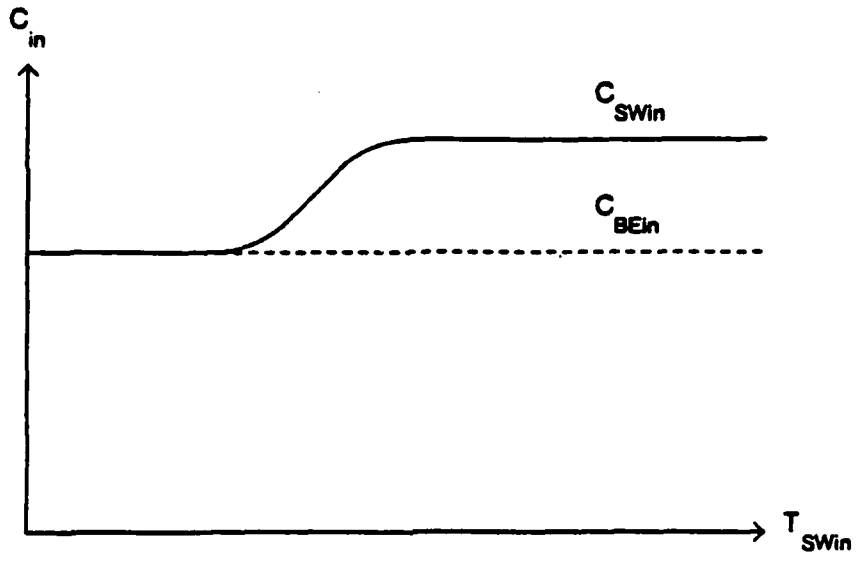


Figure 6: Expected Input Capacitance

4. General Logic Gates

Inverters are but one of a myriad of logic gates found in circuit designs. In this section we will extend our discussion to cover a more general class of logic gates. We limit our analysis to logic gates with a single active input, as shown in Figure 7. Transitions at multiple inputs are not supported by our abstract model; accurate evaluation of their effects requires a low level simulator that computes node voltages and branch currents. We feel that this represents an excessive computation cost and therefore choose a worst case gate state with a single active input to model multiple input transitions.

We will derive macromodel equations for the general logic gate by extending our inverter equations. As regards the objective function—be it power or area—the equations are basically unchanged. The power consumption of an nMOS gate is still proportional to the shape factor of the pullup, and the power or area consumption of a CMOS gate is still dependent on the stack widths. However the equations for the output waveform and input capacitance require moderate extensions.

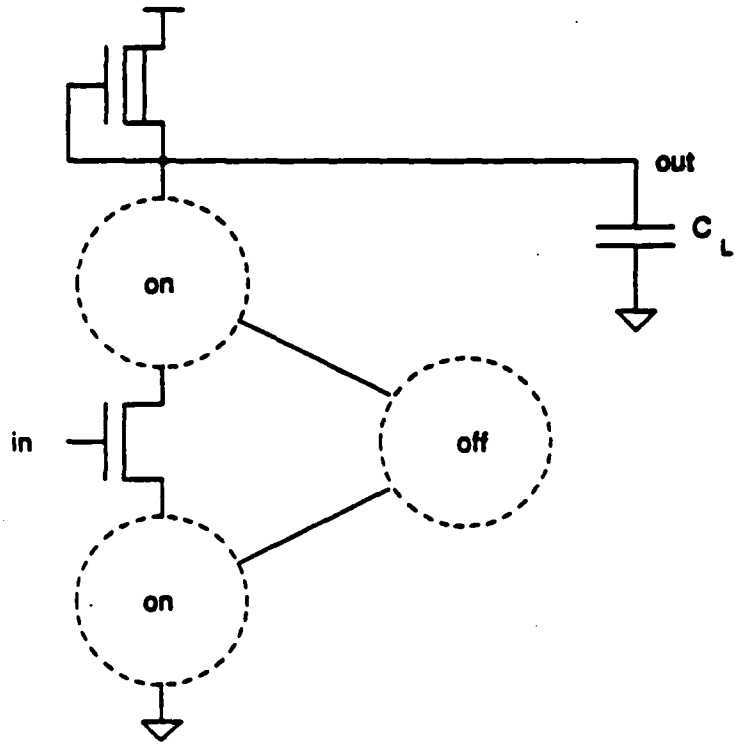


Figure 7: General Logic Gate

4.1. Output Waveform

Additional transistors in a logic gate introduce two complications. If they are part of the path that switches the output by forming a path to V_{DD} or ground, their resistance and capacitance impedes the output transition. If they are included in a side path that does not connect the output to a supply rail, their channel capacitance could add to the load capacitance and hinder the output transition. During the output switching transient, transistors with high inputs are predominantly in their linear region. Hence we model them as RC lines formed of their drain to source resistance and channel to gate and substrate capacitance.

Figure 8 contains an example. Note that while the top transistor in the right pulldown branch is not in the conducting path, its capacitance adds to the total load. The general situation is depicted in Figure 9.

The additional transistors affect the gate's response in two ways. For fast inputs, the switching transistor can still be modeled as a resistor with its c_{gs} and c_{gd}, but a closed form expression for the

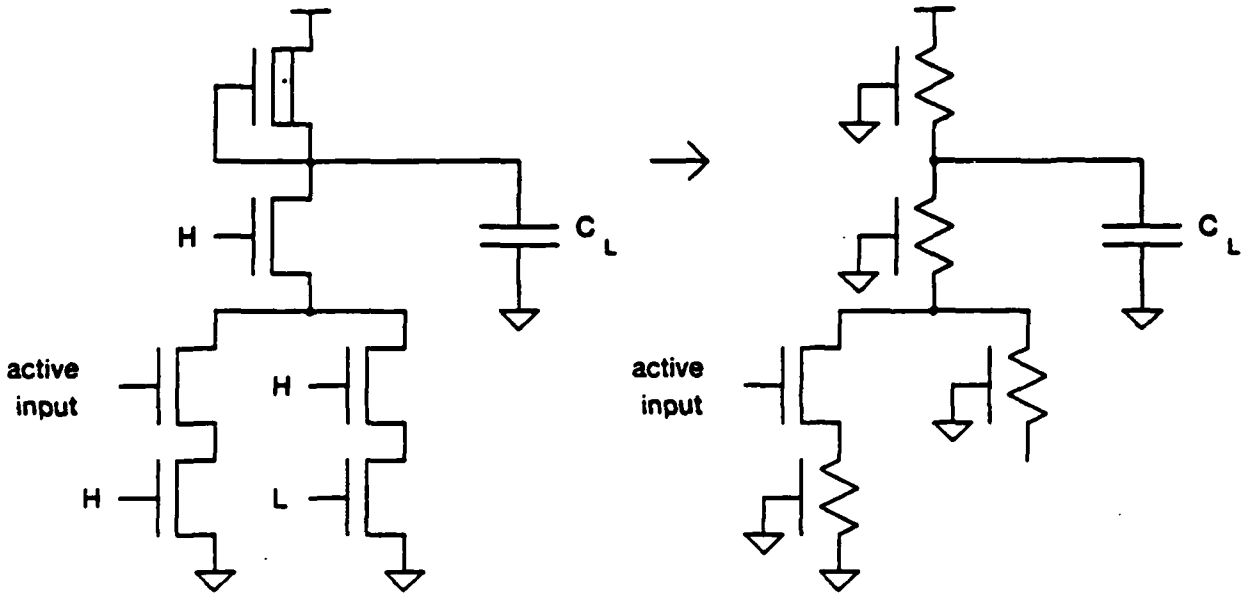


Figure 8: Example of a General Logic Gate

output waveform cannot be derived. We instead find the approximate response by using an approach first proposed by Elmore [8] and now used in waveform estimation and bounding work in MOS circuits [9, 10]. This approximates the true response as a single time constant exponential. For slower inputs the speed of the output transition is limited by the slope of the input and transconductance of the switching transistor. Consequently, transistors in the conducting path which are electrically after the switching transistor have small v_{DS} and we can neglect their voltage drops. However we must add their capacitances (along with those of any transistors connected to them) to the total load capacitance. Transistors which are before the switching transistor do not impose any additional load since their capacitances are already discharged; nonetheless their resistance will decrease the switching transistor's effective g_m if they are in the conducting path, impeding the output transition. This effect is illustrated in Figure 10. The effective g_m has been reduced to $g_m / (1 + g_m r_b)$.

Combining our results, we obtain the following equations for the output response of the general nMOS logic gate. For each equation, the macromodeling support package finds the parameters a_i, b_i, etc. that provide the closest agreement with SPICE data.

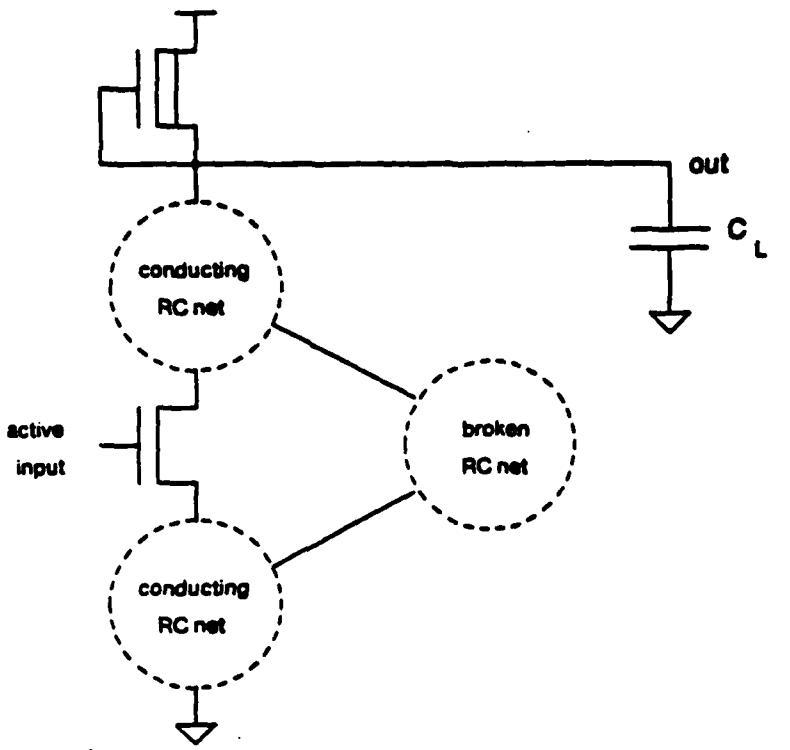


Figure 9: Circuit Model for a General Logic Gate

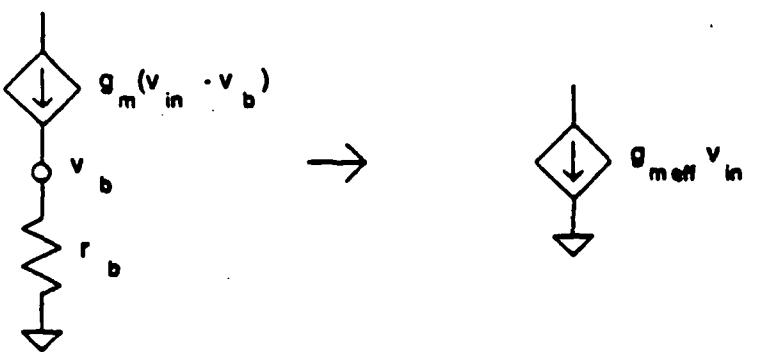


Figure 10: Reduction of Effective Transconductance

Let w_{pd} = width of the switching device

$w_{pd\,total}$ = effective total w_{pd} of devices in the conducting path (treat as if w_{pd}'s were conductances)

$w_{pd\,before}$ = effective total w_{pd} of devices before the switching transistor in the signal path

$$C_{total} = c_{pd\,driver} + c_{pd\,after} + C_L$$

$$c_{pd\,driver} = c_1 w_{pd}$$

$$c_{pd\,after} = \sum_{after} c_2 w_{pdi}$$

Delay until output begins to fall

$$T_{BEFout} = T_{BEFO} + m T_{SWRin}$$

$$T_{BEO} = a_1 + b_1 \frac{1}{w_{pd}} C_{total}$$

$$m = d_1 + d_2 \frac{S_{pu}}{w_{pd\,total}} + d_3 \frac{C_{total}}{w_{pd}}$$

Switching time of falling output transition

$$T_{SWFout} = T_{SWFO} + m T_{SWRin}$$

$T_{SWFO} = a_1 +$ Elmore delay approximation with $r_{pd} = \frac{b_1}{w_{pd}}$ for each conducting pulldown

$$m = d_1 + d_2 \frac{S_{pu}}{w_{pd}} + C_{total}\left(\frac{d_3}{w_{pd}} + \frac{d_4}{w_{pd\,before}}\right)$$

Delay until output begins to rise

$$T_{BERout} = T_{BERO} + m T_{SWFin}$$

$$T_{BEO} = a_1 + b_1 \frac{1}{S_{pu}} C_{total}$$

$$m = d_1 + d_2 \frac{S_{pu}}{w_{pd\,total}} + d_3 \frac{C_{total}}{w_{pd}}$$

Switching time of rising output transition

$$T_{SWRout} = \frac{(T_{SWRO} - \bar{T}_{SWRO})^2}{(T_{SWRO} - \bar{T}_{SWRO}) + mT_{SWFin}} + (\bar{T}_{SWRO} + mT_{SWFin})$$

$T_{SWRO} = a_1 +$ Elmore delay approximation with $r_{pu} = \frac{b_1}{S_{pu}}$

$$m = d_1 + d_2\frac{S_{pu}}{w_{pd}} + C_{total}\left(\frac{d_3}{w_{pd}} + \frac{d_4}{w_{pd\,before}}\right)$$

$$\bar{T}_{SWRO} = e_1 + e_2 T_{SWRO}$$

4.2. Input Capacitance

As we have seen, the addition of extra transistors to form more complicated logic functions has an effect on a gate's output response. We have examined the effective capacitance of an nMOS inverter during the beginning and switching phases of the rising and falling input. Of these four modes, only one depended on the output waveform. The input capacitances during the beginning portion of the input transition had no dependence because the output was still stationary. The capacitance during the switching portion of a falling input had none because the average input to output coupling capacitance dropped as the input waveform slowed down, leading to no net Miller effect. Hence the input capacitance for these three modes depends only on the pulldown transistor's size, being proportional to the transistor's width (assuming a fixed channel length). Only the switching portion of the rising input possesses a significant output waveform dependence. To account for this dependence we must analyze the conducting path containing the switching transistor. Figure 11 shows an abstract gate model along with its circuit level representation. We model 'on' transistors as resistors (linear region approximation) and have added the appropriate capacitances from nonconducting paths to the total load capacitance.

We find that r_b causes a significant drop in the input capacitance. This fact has been exploited for many years by amplifier designers to raise input impedance and thereby improve the transfer characteristic. For very fast inputs, the contribution of c_{gd} is reduced by a factor $(1 + g_m r_b)$. For very slow inputs node *a* will have dropped to V_{OL} by the completion of the input transition; hence the input capacitance is identical to that of an inverter. For moderate inputs the input capacitance is in the transition region from fast to slow input behavior. Since the total capacitance increase during the transition region from fast to slow inputs grows, the slope of the transition increases.

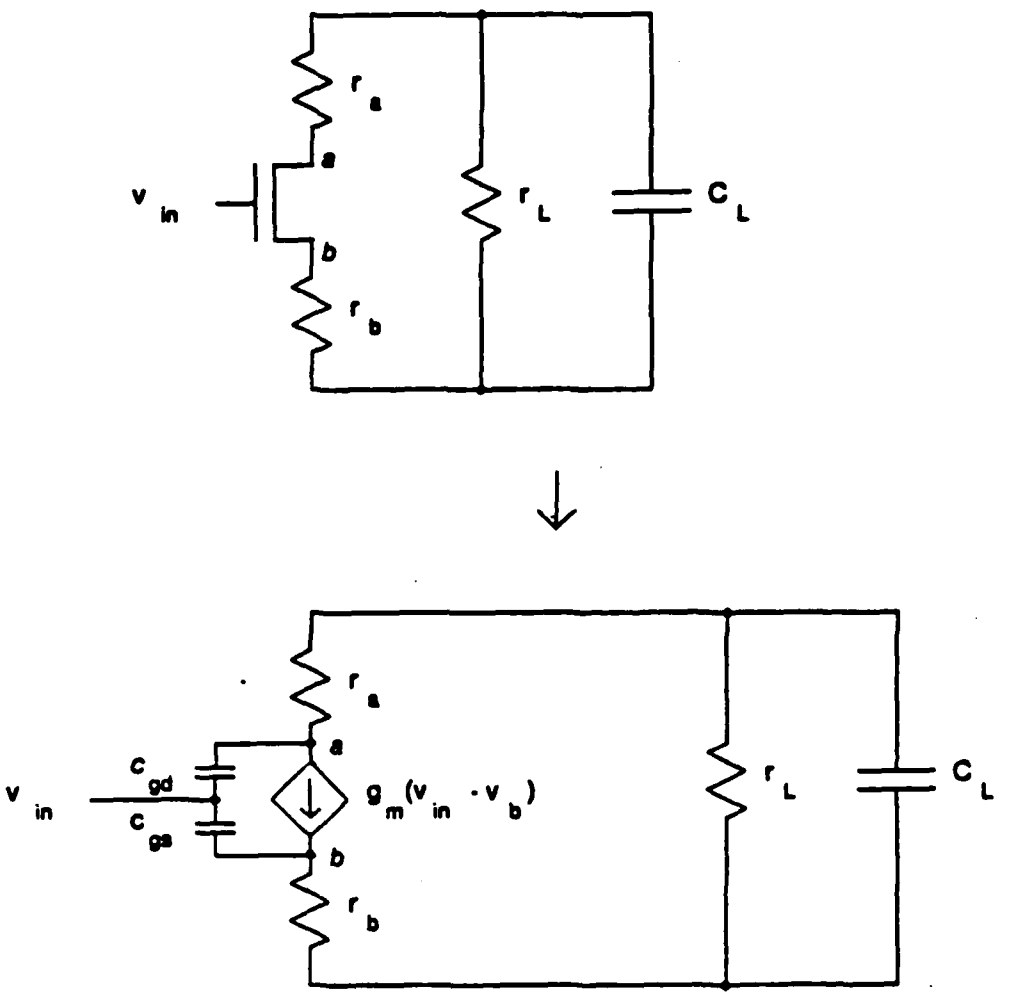


Figure 11: Circuit Model for Input Capacitance

Our analysis yields the following equation for CSWRin:

$$C_{SWRin} = C_{SWROin} + \Delta C_{SWRin}\left(\frac{m T_{SWRin}}{\Delta C_{SWRin} + m T_{SWRin}}\right)$$

$$\Delta C_{SWRin} = C_{SWR\infty in} - C_{SWROin}$$

$$C_{SWROin} = \frac{a_1 w_{pd}}{1 + a_2 w_{pd} / w_{pdbefore}}$$

$$C_{SWRcoin} = b_1 w_{pd}$$

$$m = w_{pd}\left(c_1 + \frac{w_{pd}}{C_{total}}\left[c_2 + c_3 \frac{w_{pd}}{w_{pdbefore}}\right]\right)$$

5. Implementation

We have developed a general purpose macromodeling software package. The modeler takes as input cell template files, a macromodel control file, and macromodel equations. Each cell template file contains a logic cell's general topology. The macromodeler inserts values for device sizes, capacitive loads, and input waveforms into the template, and then runs SPICE on the resulting circuit. The values of the input capacitance and output waveform are extracted from the SPICE output and stored. This process repeats for every combination of device sizes, loads, and input waveforms specified in the control file. At present 216 SPICE runs are performed for the general logic gate analyzed in this section. The particular logic cells used are inverters and NAND gates. Owing to the simplicity of the cells, the SPICE simulations are quite fast, each requiring about ten cpu seconds on a DEC 20/60.

Once the data points have been obtained the macromodeler solves for the constants in the macromodel equations by using nonlinear curve fitting algorithms. We minimize the sum of squared error; minimizing the maximum error might also be acceptable but it is too sensitive to noise in the data. The curve fitter uses a Davidon-Fletcher-Powell algorithm [11] with modifications to accept upper and lower bounds on the parameters [12]. This is essential for ensuring that the final equations make physical sense. Otherwise local minima in the error function could draw the curve fitter toward nonphysical values for the constants. Local minima in the error function also mandate that higher order effects be successively included in the model equations. That is, we solve for the first order terms in the equations first and then progressively solve for higher order terms. For example, when curve fitting the macromodel equation for output switching times, we first start with the simple RC model. We select a subset of data points with fast inputs and large capacitive loads — those points for which the model is most accurate — and solve for the RC terms in the equation. We lock these parameters and then solve for the waveshape terms. Next we solve for self capacitance terms. Finally we unlock all parameters and curve fit again. This technique helps to ensure that we reach the global minimum of the error function and adds very little to the total computation time because the time is

dominated by the SPICE runs.

The modeler is written in a computer language called CLU [13]. It consists of SPICE interface, minimization, and matrix manipulation program modules. These modules contain 3200, 1800, and 1000 lines of code, respectively. All told, the modeling support routines comprise about 6000 lines of CLU code; the modeling programs specific to the general logic gate discussed in this section represent an additional 1700 code lines.

Pertinent curve fit statistics are shown in Table 2. The macromodel equations are typically within several percent of the SPICE predictions, a major improvement over RC models. These benefits come at a small price in computational overhead because we have modeled the response of the entire cell, rather than using a more sophisticated transistor model and then having to compute the transistors' interactions to obtain the cell's response. The accuracy and computational speed of the macromodels make them well suited for both simulation and optimization applications.

Rising Input, Falling Output			*Falling Input, Rising Output*		
Model Eqn	% Error ave	max	Model Eqn	% Error ave	max
C_{BERin}	1.5	5.6	C_{BEFin}	1.5	6.9
C_{SWRin}	3.7	12.3	C_{SWFin}	1.3	9.7
T_{BEFout}	5.7	18.3	T_{BERout}	4.6	13.2
T_{SWFout}	8.6	27.8	T_{SWRout}	3.0	10.6

Table 2: Macromodel Curve Fit Accuracies

6. Acknowledgments

Lance Glasser was very helpful throughout the course of this research and in the preparation of this paper. I am also pleased to acknowledge enlightening discussions with Charles Zukowski, John Wyatt, Paul Penfield, Jr., Dan Dobberpuhl, and Mark Horowitz. Jack Hilibrand and Larry Rosenberg provided a good deal of technical insight and moral support. This work was supported by an RCA fellowship, the Defense Advanced Research Projects Agency under contract number N00014-80-0622, and the Air Force under contract number F49620-84-C-0004.

References

[1] J. Ousterhout, "Switch-Level Delay Models for Digital MOS VLSI," *Proceedings 21st Design Automation Conference*, IEEE, June 1984, pp. 542-548.

[2] M. Horowitz, *Timing Models for MOS Circuits*, PhD dissertation, Stanford, 1983.

[3] L. P. J. Hoyte, "Automated Calculation of Device Sizes for Digital IC Designs," Master's thesis, MIT, 1982.

[4] L. Glasser and L. Hoyte, "Delay and Power Optimization in VLSI Circuits," *Proceedings 21st Design Automation Conference*, IEEE, June 1984, pp. 529-535.

[5] J. Roberge, *Operational Amplifiers: Theory and Practice*, Wiley, 1975, pp. 97-104

[6] M. Matson, *Macromodeling and Optimization of Digital MOS VLSI Circuits*, PhD dissertation, MIT, 1985.

[7] P. Gray and C. Searle, *Electronic Principles: Physics, Models, and Circuits*, Wiley, 1969.

[8] W. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics*, Vol. 19, No. 1, January 1948, pp. 55-63.

[9] P. Penfield, J. Rubinstein, and M. Horowitz, "Signal Delays in RC Tree Networks," *IEEE Transactions on Computer Aided Design*, Vol. CAD-2, 1983, pp. 202-211.

[10] T.-M. Lin and C. Mead, "Signal Delay in General RC Networks with Application to Timing Simulation of Digital Integrated Circuits," *Proceedings, Conference on Advanced Research in VLSI*, MIT, January 1984, pp. 93-99.

[11] R. Fletcher and M. Powell, "A Rapidly Convergent Descent Method for Minimization," *Computer Journal*, Vol. 6, 1963, pp. 317-322.

[12] D. Bertsekas, "Projected Newton Methods for Optimization Problems with Simple Constraints," *SIAM Journal on Control and Optimization*, Vol. 20, 1982, pp. 221-246.

[13] B. Liskov, A. Snyder, R. Atkinson, and C. Schaffert, "Abstraction Mechanisms in CLU," *Communications of the ACM*, Vol. 20, No. 8, August 1977, pp. 564-576.

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CAMBRIDGE, MASSACHUSETTS 02139

VLSI Memo No. 84-213 November 1984

Optimization of Digital MOS VLSI Circuits*

Mark D. Matson**

ABSTRACT

Power consumption and signal delay are crucial to the design of high performance VLSI circuits. This paper presents a CAD tool for optimizing digital MOS designs. The tool determines the transistor sizes that minimize circuit power consumption subject to constraints on signal path delays. Computational efficiency is obtained through macromodeling techniques (described in a companion paper) and a specialized optimization algorithm.

The macromodels are based on device equations, and encapsulate logic gate behavior in a set of simple yet accurate formulas. The optimization algorithm exploits properties of the digital MOS domain to convert the primal optimization problem into a dual form which is much easier to solve. The result is a CAD tool which can optimize a circuit in roughly the amount of time needed to perform a transistor level simulation of the circuit.

---

*This work was supported in part by an RCA fellowship, in part by the Defense Advanced Research Projects Agency of the Department of Defense under Contract No. N00014-80-0622, and in part by the Air Force Office of Sponsored Research under Contract No. F49620-84-C-0004.

**Department of Electrical Engineering and Computer Science, M.I.T., Room 36-575, Cambridge, MA 02139; (617) 253-8169.

Optimization of Digital MOS VLSI Circuits

Abstract

Power consumption and signal delay are crucial to the design of high performance VLSI circuits. This paper presents a CAD tool for optimizing digital MOS designs. The tool determines the transistor sizes that minimize circuit power consumption subject to constraints on signal path delays. Computational efficiency is obtained through macromodeling techniques (described in a companion paper) and a specialized optimization algorithm. The macromodels are based on device equations, and encapsulate logic gate behavior in a set of simple yet accurate formulas. The optimization algorithm exploits properties of the digital MOS domain to convert the primal optimization problem into a dual form which is much easier to solve. The result is a CAD tool which can optimize a circuit in roughly the amount of time needed to perform a transistor level simulation of the circuit.

1. Introduction

The design of a VLSI circuit is an enormous task. Sophisticated CAD tools are essential if designers are to take full advantage of the power offered by fabrication technology. We describe a tool for optimizing the performance of digital MOS circuits. This tool finds the transistor sizes that minimize power consumption subject to constraints on signal path delays. The principle advantage is an increase in designer productivity. At present, designers size transistors based on intuition and numerous SPICE simulations. This process is so time consuming—for both man and machine—that designers are hard pressed to arrive at any circuit that meets delay specifications and can rarely afford the extra effort needed to minimize power consumption as well. This not only hinders the design of the circuit at hand, it makes it difficult to compare alternate topologies for implementing functional blocks, as the performance benefits offered by different topologies cannot be truly ascertained unless the corresponding circuits have been optimized.

Another application is automatic module generation for silicon compilers. The module's transistors must be properly sized in order to meet system performance specifications, but it would be unthinkable to have a human perform the sizing. The task could involve thousands of transistors, making it too mundane and complicated. A special purpose optimizer can accomplish the chore far more efficiently.

Several authors have studied optimization work of this nature. General purpose optimization packages such as DELIGHT [1] and APLSTAP [2] perform much of the work in the optimization process. They iteratively improve the design solution as a designer would, but by employing nonlinear optimization algorithms, choose the next solution point more accurately and efficiently than a human could. The key advantage is that an optimal solution is reached. However the optimization process tends to be computationally expensive for a number of reasons. First, since the optimization package is general purpose in nature, it cannot exploit properties of digital MOS logic and use algorithms which would be more problem specific and hence potentially faster. Second, because the optimization package is isolated from the circuit's data base, communicating solely via the simulator, there is no provision to embed additional information in the data base which could assist the optimization, either to allow one to access information more readily or to apply a more efficient algorithm. Third, the circuit's signal path delays must be determined fairly accurately; this generally entails the use of a device level simulator such as SPICE, which is rather expensive computationally. The consequence of these three factors is that general purpose optimizers are typically restricted to circuits with at most about thirty design parameters.

In an effort to address larger designs, other workers have investigated more specialized techniques

[3]. By using a resistive model for transistors and neglecting the changes in a logic gate's input capacitance induced by sizing its transistors, these workers were able to greatly simplify the optimization problem. They reformulated the original problem, a minimization subject to nonlinear constraints, as an unconstrained minimization. This allows for much simpler optimization algorithms, leading to fast convergence times. Nonetheless, the simplifications needed to reformulate the problem seriously reduce the accuracy of both the power minimization and the satisfaction of the delay constraints, making the approach inappropriate for high performance circuit designs.

Other authors have aimed for fast computation times by simplifying both the logic gate models and the optimization techniques. Examples are TV [4] and Andy [5]. These tools use resistor models for transistors instead of the computationally expensive device level models. Heuristics, rather than nonlinear optimization algorithms, guide the sizing of transistors in critical paths. In particular, TV speeds up paths by widening the transistors of slow logic gates, while Andy uses a fixed sizing ratio from gate to gate when a chain drives a large capacitive load. Although these approaches are computationally fast enough to be applied to large circuits, our problem domain requires more accuracy and efficiency. The resistor model is not accurate enough for high performance design, and iteratively applying heuristics is not as efficient as nonlinear optimization algorithms that simultaneously consider all critical paths.

2. Overview of Paper

This paper presents a novel approach to the transistor sizing problem. We attack the competing needs for accuracy, computational speed, and a nearly optimal solution by combining the benefits of the previous approaches we examined. Like TV and Andy, we work at a higher level of abstraction than SPICE, transcending the details of actual transistor operation. However we acquire additional computational speed by modeling entire logic cells rather than just individual transistors. Like the general purpose optimizers, we employ nonlinear optimization techniques. This helps to assure that we reach an optimal solution in an efficient fashion. We exploit properties of digital MOS circuits and apply a specialized algorithm to the problem, yielding striking improvements in computational speed.

Section 3 outlines the special features of the optimization problem, describing the properties of the objective and constraint functions. Section 4 presents the theory of the optimization algorithms. We choose a method particularly suited to our problem, taking advantage of the properties of the digital MOS domain, and of our ability to create a circuit data base customized for the transistor sizing problem. Our approach, called duality, allows us to partition the problem into many simpler, smaller subproblems, and to transform the nonlinear delay constraints into box constraints (eg. constraints of

the form $x \geq 0$). Section 5 discusses the implementation of the optimizer. We describe the organization of the software and study the optimizer's performance on some example circuits.

3. Properties of Our Problem

We begin by choosing an optimization technique which is appropriate to our problem. Selection of the technique is highly problem dependent, as "appropriateness" in nonlinear optimization is nearly synonymous with fast computation times, requiring that the optimization technique be closely matched with the problem's characteristics. We therefore commence by considering the properties of our optimization problem.

We desire to minimize a circuit's power consumption subject to constraints on signal path delays and transistor sizes. The objective function, power, is the linear sum of the power consumptions of each circuit cell. For nMOS the power consumption of each cell is linear in the shape factor of the pullup transistor. For CMOS the power consumption is linear in the capacitive loads which must be driven, which are due to the area of the transistor gates and interconnect capacitance, but also depends somewhat on the input waveforms. Hence for nMOS, and nearly for CMOS, the power consumption of a circuit is a separable function of the form

$$P_{total} = \sum_{i=1}^{n} P_i$$

where P_i is a function of cell i only.

The problem's constraints are of two varieties: delay specifications and transistor size design rules. The delay along a signal path is a nonlinear function of the circuit's transistor sizes, and is nonlocal, being composed of contributions from each cell along the signal path. Fortunately transistor sizing is very nearly a separable operation, because both waveshape and capacitive loading effects die off rapidly with electrical distance. Consider an inverter chain. Whether the input signal is slow or fast, by the time the waveform has propagated to the chain's output its shape will be predominantly determined by the last gate in the chain. Fast inputs put the gate in an RC response mode where the output waveform's switching time is governed by the gate's effective output resistance and capacitive load. Slow inputs place the gate in a gain limited mode where the gate's gain increases the sharpness of the waveform's transition. Thus a gate behaves as a crude wave shaper.

Capacitive loading effects also attenuate quickly with electrical distance. Suppose the chain is driving a large capacitive load. The last gate will have to be fairly wide in order to drive the load. The second to last gate will in turn have to be somewhat large to drive the wide pulldown transistor of the

last gate. We need progressively less widening as we work our way backwards from the load. Within a few gates we reach a point where we are fully shielded from the size of the load.

Design rules restrict the minimum transistor size. For nMOS there is also a minimum beta ratio (ratio of pulldown to pullup shape factors) requirement. The former is a box constraint; the latter is a linear constraint. The constraints on a circuit's transistors are entirely local to each cell, and are therefore separable.

Accuracy requirements are also important, and exhibit a peculiar ambivalence in our problem. The delay specifications on the signal paths must be met to the full accuracy afforded by the macromodels. However the power minimization is less critical. We can tolerate a fair amount of error in minimizing the circuit's power consumption, especially if the inaccuracies are minor and are accompanied by large savings in computation time. In fact, at present designers use only crude heuristics or mostly ignore the power consumption issue.

In summary, the problem embraces characteristics ranging from the trivial to the extremely difficult. The objective function is a simple summation of contributions from each logic cell, each contribution being linear in the cell's transistor sizes. On the other hand, we anticipate hardship with the delay constraints, since they are global and nonlinear. Fortunately there are only a few of them; typically a designer will specify delays for only about ten critical paths through a functional block. In contrast, the transistor size constraints are quite simple, consisting of linear and box constraints. However there are a large number of them, at least one for every transistor in the circuit, carrying the potential for huge run times. The objective and constraint functions are essentially separable, linearly composed of nearly independent contributions from the circuit's cells. We would prefer a nonlinear optimization algorithm that can exploit this separability, pursuing a divide and conquer strategy where the problem is partitioned into many smaller subproblems. This segmentation is beneficial because with most optimization algorithms, run times grow superlinearly with the number of design variables. Thus by breaking up a large problem, faster run times can be achieved. In particular, if the problem could be partitioned down to the cell level, the size of the vector space for each subproblem would be the number of transistors in each cell. Small vector spaces usually imply fast run times.

4. Duality

We carefully studied several optimization techniques, including feasible directions and penalty methods. Neither of these methods is capable of exploiting the near separability of the problem. Since we felt that partitioning was vital to achieving fast run times, we chose a technique called duality, a fairly exotic approach in comparison to the other two methods. We shall see that the

EMBEDDING GRAPHS IN BOOKS: A LAYOUT PROBLEM WITH APPLICATIONS TO VLSI DESIGN

Fan R. K. Chung *
Frank Thomson Leighton **
Arnold L. Rosenberg ***

ABSTRACT

We study the graph-theoretic problem of embedding a graph in a book with its vertices in a line along the spine of the book and its edges on the pages in such a way that edges residing on the same page do not cross. This problem abstracts layout problems arising in the routing of multilayer printed circuit boards and in the design of fault-tolerant processor arrays. In devising an embedding, one strives to minimize both the number of pages used and the "cutwidth" of the edges on each page. Our main results (1) present optimal embeddings of a variety of families of graphs; (2) exhibit situations where one can achieve *small* pagenumber only at the expense of large cutwidth, and conversely; and (3) establish bounds on the minimum pagenumber of a graph based on various structural properties of the graph. Notable in this last category are proofs that (a) every n-vertex valence-d graph can be embedded using $O(dn^{1/2})$ pages, and (b) for every valence $d>2$, for all large n, there are n-vertex valence-d graphs whose pagenumber is at least

$$\left\lceil \frac{n^{1/2-1/d}}{\log^2 n} \right\rceil .$$

computational efficiency that it affords more than compensates for its conceptual complexity. The basic idea of duality is to form a so-called dual problem which can be significantly easier to solve than the original, or primal, problem. In our case the primal is difficult to solve because of the global, nonlinear delay constraints and large number of transistors to size.

Duality offers several major advantages. First, the primal problem need not be feasible. This is a strong possibility because high performance designs often push circuit topologies to the limits of their performance. It is likely that a designer will specify delays for some signal paths which cannot be met. In this event we desire that our CAD tool do its best to meet those speed specifications while optimizing the power consumption of the other paths whose delay constraints can be met. Duality achieves this goal. Second, inactive constraints pose no difficulty for duality. A designer specifies maximum delays along signal paths. Due to paths sharing common portions, it is possible that one path's delay specification will be exactly met while a companion path will be faster than required, and yet this situation minimizes power consumption. This is essentially a recasting of the critical path problem; the first path is one of the circuit's critical paths. Third, and perhaps most importantly, duality can be extremely efficient computationally. This is due to two factors. Duality converts the nonlinear delay constraints into simple box constraints, allowing us to apply fairly simple optimization algorithms (which implies robustness as well) with quasi-Newton methods. The quasi-Newton methods lead to fast convergence. Also, the dual approach permits us to exploit the separability of the power and delay functions, enabling us to use a divide and conquer strategy where each cell is optimized separately. Partitioning affords significant computation speed advantages.

Like any nonlinear optimization approach, the advantages are balanced by drawbacks. Duality is not applicable to all problems; it works best for those satisfying a certain convexity requirement, a property which digital MOS circuits possess. Another drawback is due to our partitioning approach rather than duality itself. Although exploiting separability provides run time improvements, it necessitates the maintenance of additional data in the circuit's data base, along with a close interaction between the control structure and the data base. Partitioning the circuit into cells implies incremental optimization of each cell in succession. This mandates a sophisticated data base, and places profound requirements on the programming language used to implement the optimizer.

4.1. Lagrange Multipliers

Lagrange multipliers are the key to understanding duality. We shall explain their use and significance through a simple example. Consider a chain of two inverters. The input is driven by a source v_{IN} through a resistor R_S; the output connects to a load capacitance C_L. We wish to constrain the maximum delay of the chain. If we fix the width of the pullup transistor, the length of the pulldown,

and the beta ratio of the inverters, then we can treat the power consumptions of the inverters as the only free variables because specifying the power consumption of either logic gate determines the gate's transistor sizes. Let p_1 be the power consumed by the first gate and p_2 be that consumed by the second, and suppose we desire the total delay $T_{total} = T_1 + T_2$ to be less than or equal to some T^*.

This maximum delay specification places restrictions on the allowable power consumptions of the gates. Certain regions in (p_1, p_2) space will not meet the speed specification. For instance if the shape factors of the transistors in the second inverter are too small, the inverter will not be able to charge the capacitor C_L quickly enough to satisfy the delay constraint. On the other hand, if the shape factors are too large, implying a wide pulldown and hence a large input capacitance, the first inverter will not be able to drive the second quickly enough. Of course the first inverter's shape factors can be made larger to drive the extra load, but after a certain point the first inverter's input capacitance becomes so large that the delay through R_S precludes meeting the delay specification. Since power consumption is linearly related to shape factor, the bounds on the shape factors imply bounds on the power consumption. Similar reasoning applies to the power consumption of the first inverter, giving us the forbidden zones (dashed lines) shown in Figure 1.

We can more precisely characterize the feasible set of power consumptions. We do this by employing a simple RC model for the inverters, allowing us to derive an analytic expression for the delay through the gates as a function of their power consumptions. The resulting constraint surface $T_{total} = T_1 + T_2 = T^*$ is elliptical as depicted in the figure.

Figure 1 also shows the correlation between total power and path delay. The dotted lines are contours of constant power. To meet the delay constraint we must stay within the circular region, but the total power dissipation varies with position in the region. As we move toward the upper right of the feasible set, the power dissipation increases. At the point *Max* we have reached the maximum power consumption that will still allow us to satisfy the delay constraint. Here the delay and power contours are tangent, and their gradients point in the same direction. If we instead work our way toward the lower left of the feasible set, the total power dissipation decreases. When we reach the point *Min* the dissipation will be at its lowest level that will still satisfy the delay constraint. Here the delay and power contours are again tangent, but now their gradients point in opposite directions; mathematically this can be expressed as

$$\nabla P = -\mu \nabla T$$
or $\nabla (P + \mu T) = 0$
where $\mu > 0$

The variable μ is called a Lagrange multiplier, and offers the key to solving our nonlinear optimization problem.

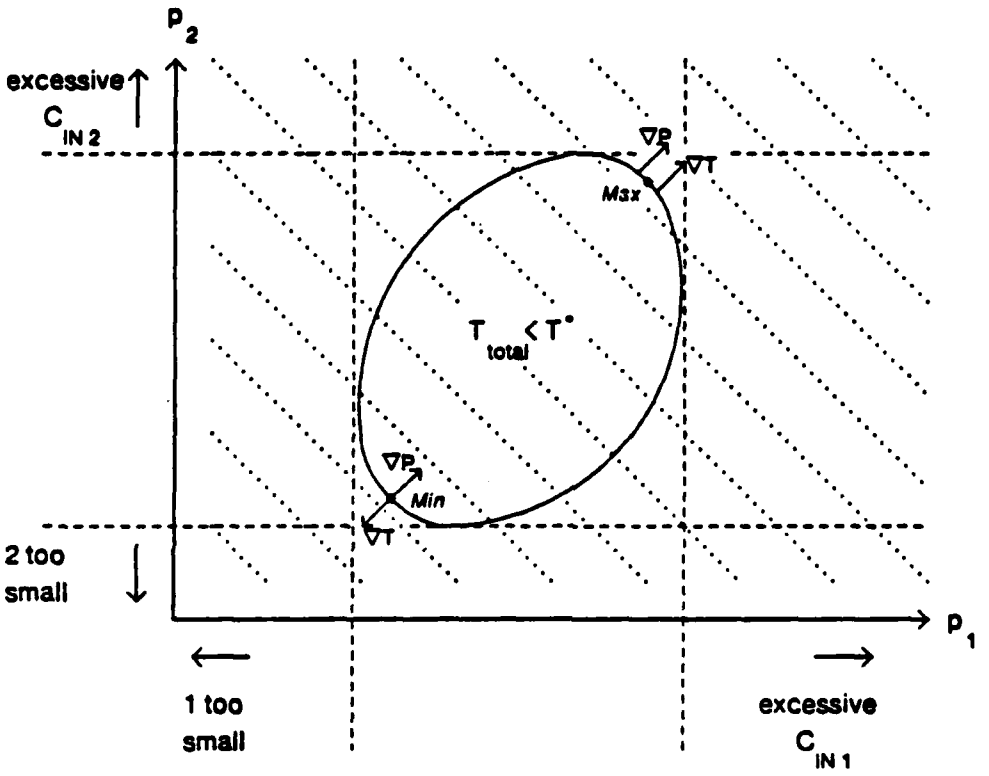


Figure 1: Contours of Delay and Power

4.2. Finding the Optimum

We can acquire an understanding of how to find the optimum by applying a graphical approach to the inverter chain. We are interested in the possible total power and total delay combinations that the circuit can exhibit. In other words, we desire the locus of points (T_{total}, P_{total}) that will be generated if we substitute all valid transistor size combinations into the circuit. This locus of points is denoted the set of all possible pairs, $\mathcal{P}$, and is displayed in Figure 2. The set's lower left boundary is the classic power-delay tradeoff curve (bold line); it represents designs that offer propagation delays with the lowest possible power consumption for those delays. Points toward the left of the curve are in the high speed, high power region. As we move down the curve to the right, we trade off speed for

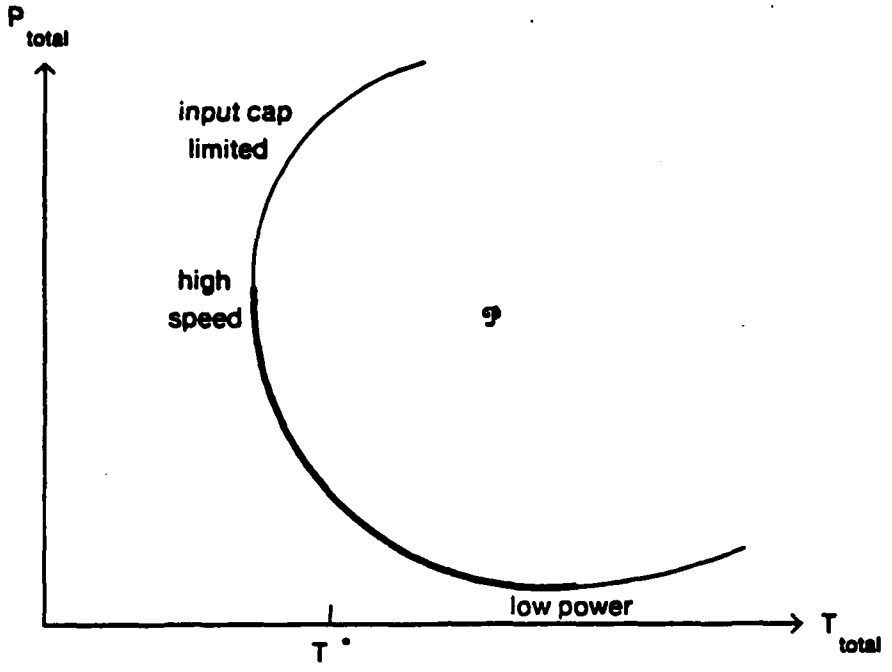


Figure 2: Set of Possible Points

reduced power consumption, and eventually enter the low power, low speed region.

Points that are not on the tradeoff curve correspond to nonoptimal circuits. These circuits either consume more power than an optimal circuit with the same delay, or are slower than an optimal circuit with the same power consumption. For example, suppose the inverter chain is driving a large capacitive load. We should make the second inverter's shape factors relatively large in order to drive the load, and then make the first inverter slightly large to drive the wide pulldown of the second inverter. If we reverse the ordering, making the first inverter very large rather than the second, the circuit will still consume the same amount of power as the optimal one, but will be considerably slower.

Our delay specification restricts the points that we can accept to those having a total delay less than or equal to T^*. We can focus our attention on this subset by shifting the vertical axis as shown in Figure 3. Points to the left of the axis have delays which are faster than T^*; this subset is called the *feasible region*. The optimum is the point in this region with the lowest power consumption, and is located at $(0, P^*)$ in the figure.

We must somehow reach this optimum point starting from an arbitrary point in the set $\mathcal{P}$. The

approach duality takes can be thought as a two step process, illustrated in Figure 3. The first step is to move to and remain on the lower boundary of $\mathcal{P}$. The second is to walk along this boundary to the optimum. Note that while conceptually this process may be interpreted as two steps, it must be implemented as an inner loop embedded in an outer loop. Step one corresponds to the inner loop, and step two to the outer. This forces the search to follow along the lower boundary of the set.

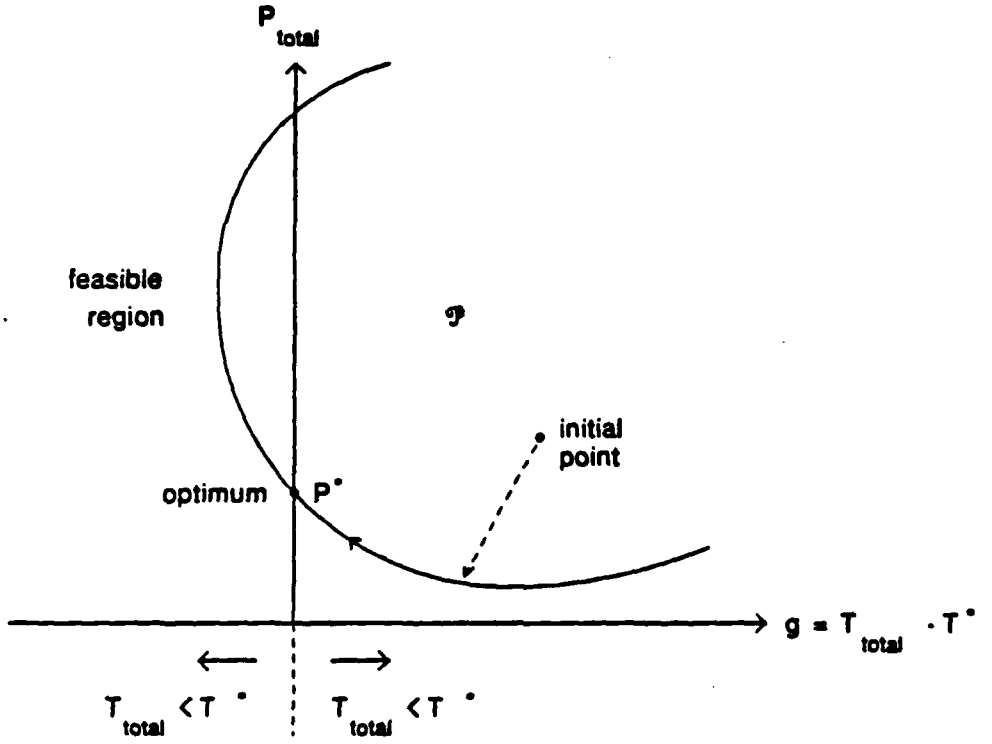


Figure 3: Reaching the Optimum

We will now describe the implementation of each loop. Figure 4 gives a graphical representation of the inner loop. Suppose that we begin at some arbitrary assignment x of transistor sizes, with some arbitrary nonnegative Lagrange multiplier vector μ. The transistor sizes x map to point (g(x), P(x)) in g-P space. We can move from this point to the lower boundary of the achievable set by sliding the solid line down until it is tangent to the bottom of $\mathcal{P}$, while preserving the slope of the line. By geometry we know that a line through a point (g(x), P(x)) with normal (μ, 1) intersects the vertical axis at $P(x) + \mu g(x)$. The multiplier μ fixes the slope of the line. Hence this sliding operation is equivalent to bringing the vertical intercept down while holding μ fixed. We must perform the minimization

$$\min \{P(x) + \mu g(x)\}$$
subject to $x \in \mathfrak{D}$., the set of valid transistor sizes

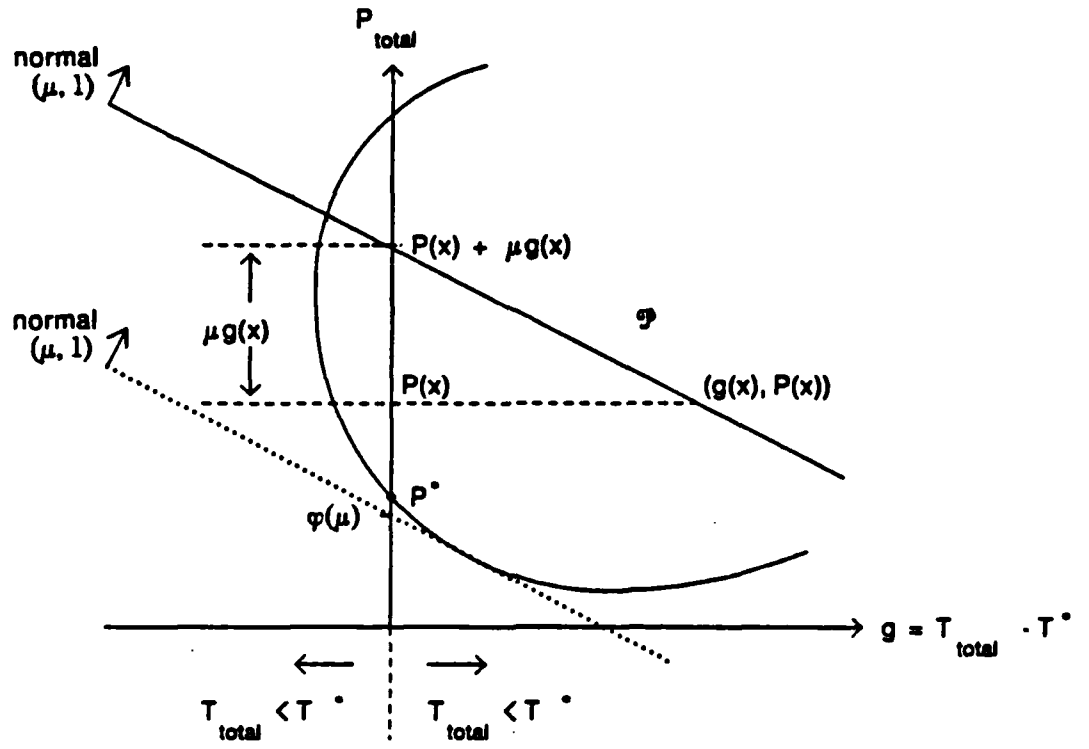


Figure 4: Inner Loop Minimization

We shall denote this intercept as $\varphi(\mu)$, the dual functional.

The outer loop walks along the lower boundary toward the optimum. We can gain insight into how this might be accomplished by contemplating the effect of different Lagrange multipliers on the inner loop's minimization. Figure 5 provides an illustration. We see that as we move toward the optimum point $(0, P^*)$ the intercepts $\varphi(\mu_i)$ increase in value until they reach P^*. Conversely, if we move away from the optimum in either direction, the intercepts $\varphi(\mu_i)$ decrease. This is a maximization:

$$P^* = \max \varphi(\mu)$$
$$\text{subject to } \mu \geq 0$$

This maximization gives us the Lagrange multiplier μ of the optimum, while the inner loop minimization provides the optimal transistor size assignments.

We can now grasp the intuitive significance of the Lagrange multiplier. From Figure 5 it is apparent that as μ increases, the line becomes more vertical, and we move up and toward the left. Power consumption increases whereas delay decreases. We are generating transistor size assignments that

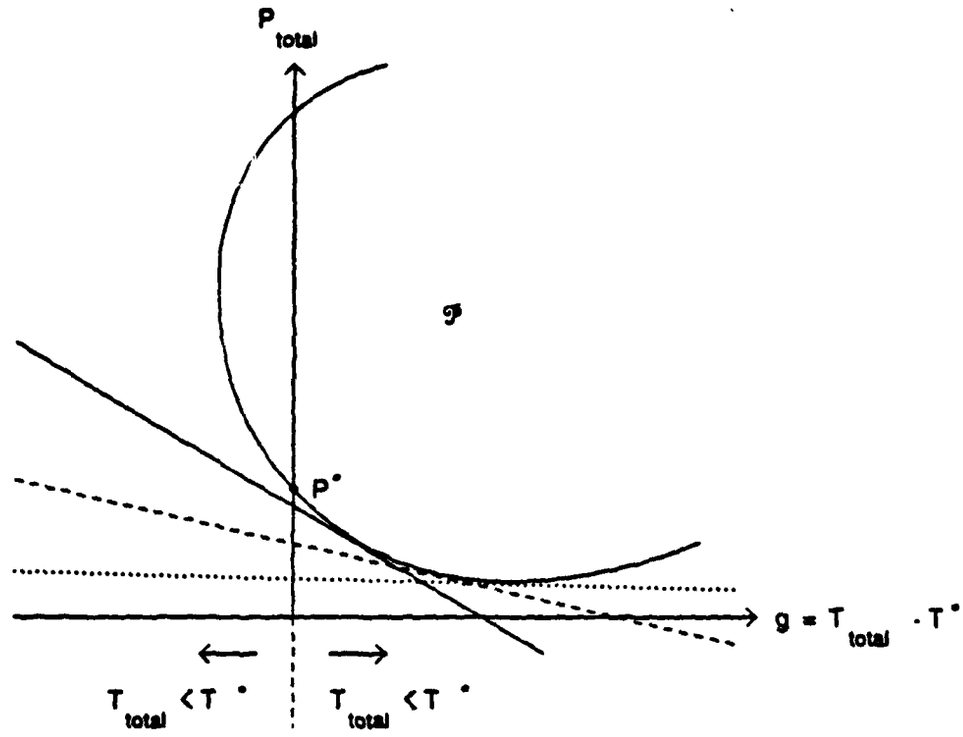


Figure 5: Outer Loop Maximization

push the circuit topology harder for speed. The fact that the multiplier has a concrete, practical meaning is quite important, because it allows a designer to follow our CAD tool's "intent" as it optimizes a circuit, showing the signal paths that are the most troublesome in meeting the delay specifications. This knowledge is vital for directing efforts to improve the circuit, such as reduction of interconnect capacitance and modification of circuit topologies (eg. the insertion of super buffers).

4.3. Degenerate Cases

It is crucial that optimization algorithms perform properly even when faced with certain degenerate conditions in the delay constraints, such as inactive or infeasible constraints. Inactive constraints can come from one of two sources: (1) a delay specification on a signal path that is so loose that minimum size transistors along the path will satisfy it, or (2) interactions among paths give rise to a situation where meeting one path's constraint causes another's to be inactive. Of these two possibilities, the second is the most likely, and occurs frequently in practice. An inactive constraint arises because the point of minimum power lies to the left of the constraint's vertical axis in power-constraint space; the dual algorithm will converge to the optimum by driving the constraint's Lagrange multiplier to zero.

Another important degenerate condition comes from infeasible delay constraints, where the designer requests a maximum signal path delay that cannot possibly be met. These cases occur frequently in high performance circuit design, as the designer pushes a circuit topology and fabrication process to the limits of their performance. Under these situations we desire that the optimizer do the best that it can, sizing those paths with infeasible constraints such that they switch as fast as possible, and sizing paths with feasible constraints such that their power consumption is optimized. The dual algorithm will drive the former paths' Lagrange multipliers towards infinity, sizing the transistors for maximum speed. Thus the algorithm gives useful feedback to the designer, indicating the maximum speed the circuit topology can provide.

4.4. Restrictions

As we mentioned at the beginning of our discussion, although duality does offer significant advantages over other optimization methods, it is limited in the scope of objective and constraint functions that it can solve. In particular, certain objective and constraint functions can produce a condition known as a duality gap. These functions give rise to nonconvexities in the lower left boundary of the set $\mathcal{P}$, leading to a gap between the solution found by the dual algorithm and the true optimum P^*. We have never encountered a duality gap for any of the circuits we have optimized. The power and delay equations describing digital MOS gates, and the separability inherent in the digital MOS domain, make the occurrence of a gap unlikely and imply a small gap even if one should appear. If a gap ever occurs the circuit will still meet delay specifications, with a bounded amount of excess power dissipation.

5. Implementation

We have seen that duality maps the nonlinear delay constraints into simple box constraints on Lagrange multipliers. This mapping leads to simple, computationally efficient control structures. The outer loop maximization uses a Davidon-Fletcher-Powell quasi-Newton method [6] with modifications for the box constraints [7]. The inner loop minimization is more complicated since it must handle linear as well as box constraints; it uses an algorithm due to Bard [8]. Since both loops work in small vector spaces and use second derivative information, the optimizer runs very fast.

The language chosen to implement the optimizer embodies many of the principles of data abstraction and object oriented programming. These features were essential owing to our incremental optimization approach and the hierarchical nature of VLSI design. We needed a language that supported automatic dynamic data structure allocation, abstract data types, implicit pointers, and recursive procedure calls and data structures. We chose the CLU programming

language [9], which runs on DEC 20's and VAX's. The language system has extensive compile time type checking and an outstanding interactive debugger. Both greatly facilitate program development. Another good choice would have been Zetalisp on a Symbolics 3600. The optimizer consists of generic nonlinear minimization and maximization routines, a circuit optimization support package, and routines for optimizing generic nMOS logic gates. These program modules represent 3500, 7400, and 2700 lines of CLU code, respectively.

We have applied our optimizer to many circuits; here we present two representative cases. Our first example is a chain of three inverters. (The circuit is simple in order to allow a comparison with DELIGHT.) We began with minimum size transistors and requested maximum rise and fall delays of 8.0 ns. Optimization statistics appear in Table 1. In the table, T_{BEout} is the time until the output begins to move in response to an input transition and T_{SWout} is a measure of how quickly the output switches once it does begin to change. The optimizer reached a solution in slightly over 15 cpu seconds on a DEC System 20/60.

Optimization Accuracy:

	cpu time [sec]		
optimizer	*set up*	*optimization*	*power*
DELIGHT (VAX 750)	133.7	3018.0	2.02
Present Work (DEC 20/60)	1.1	15.2	2.08

Delay Accuracies:

	predicted [ns]	*SPICE [ns]*		
Path	T_{BEout} T_{SWout}	T_{BEout} T_{SWout}	*error [%]*	*μ [mW/ns]*
in → out, rise	4.06, 3.85	3.80, 3.76	+5	0.403
in → out, fall	5.23, 0.64	5.53, 0.77	+7	0.000

Total SPICE verification time (DEC 20/60 running FORTRAN): 16.5 cpu sec

Table 1: Optimization Statistics for the Inverter Chain

An attempt was made to run DELIGHT on the inverter chain and compare its results to those of our optimizer, but the effort met with only partial success. The complex interactions among objective and constraint functions overwhelmed DELIGHT's direction finding routine, causing the program to hang up in infinite loops.[1] This illustrates how general purpose optimization algorithms can fail when faced

[1] Bill Nye, the author of DELIGHT, believes that the problem lies in the direction finder's quadratic programming subroutine. He is investigating more robust routines.

with a problem of this nature; special purpose algorithms are essential. To assist the direction finder, we eliminated the maximum beta ratio and minimum shape factor constraints, and started DELIGHT at an initial set of transistor sizes that was fairly close to the optimal solution. The problem was also simplified by not evaluating the chain's rising input, falling output response. This did not affect the final solution since this transition's delay constraint was not active, but it halved the number of SPICE runs needed and reduced the strain on DELIGHT's direction finder. DELIGHT required five iterations to converge to within five percent of the optimum, consuming 3018 cpu seconds on a VAX 11/750. Table 1 gives the statistics.

The performances of the circuits produced by the two optimizers are quite similar. Both have falling input, rising output delays of 8.0 ns as requested, with power consumptions of 2 mW. The power consumption of DELIGHT's circuit is less than ours by about three percent, but this is mainly due to the removal of the minimum S_{pu} constraint on the second inverter.

Our optimizer runs considerably faster than DELIGHT with SPICE. It is difficult to make an exact comparison of how fast DELIGHT would run on the DEC 20/60, had it been able to handle the inverter chain without simplifications, but we can make fairly accurate estimates. A DEC 20/60 will run Fortran code about three or four times faster than will a VAX 750. DELIGHT only evaluated one path transition, with fewer transistor size constraints and an initial set of sizes that was fairly close to the optimum. These simplifications halve the number of SPICE simulations per iteration and reduce the number of iterations needed to reach the optimum, leading to about a factor of five improvement in run time. Hence we believe that DELIGHT would require about 4000 cpu seconds to size the inverter chain on our DEC 20. This is about 300 times slower than our optimizer. We also feel that our run times scale better than DELIGHT's as circuit complexity increases. The partitioning scheme used by our optimizer leads to approximately linear growth, while the growth rates of DELIGHT's feasible directions algorithms and SPICE's simulation algorithms are more rapid.

We now discuss the optimization of a more complicated example, a four bit adder. One bit of the adder is shown in Figure 6. The adder is comprised of sixteen logic cells having a total of 72 transistors. Path delay constraints were placed on five of the signal paths. Table 2 gives the optimization statistics. Starting with minimum size transistors, the optimizer required only 520 cpu seconds to optimize the adder. In contrast, considerably more time was needed for SPICE runs to just verify the accuracy of the predicted delays.

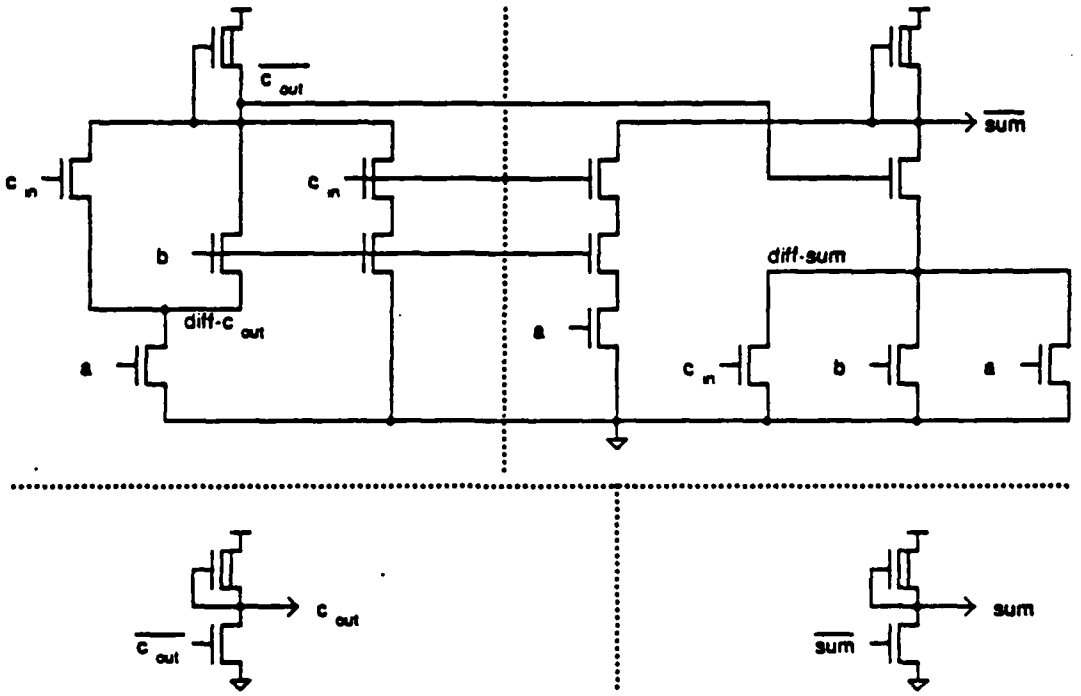


Figure 6: A Full Adder Module

6. Acknowledgments

I am very grateful to Lance Glasser for his support in this research and help in the preparation of this paper. I also thank Dimitri Bertsekas and Kevin Tsai for teaching me the art and science of nonlinear optimization, and Jack Hilibrand and Larry Rosenberg for encouragement and guidance. This research was supported by an RCA fellowship, the Defense Advanced Research Projects Agency under contract number N00014-80-0622, and the Air Force under contract number F49620-84-C-0004.

References

[1] W. Nye, E. Polak, A. Sangiovanni-Vincentelli, and A. Tits, "DELIGHT: An Optimization-Based Computer-Aided Design System," *Proceedings International Symposium on Circuits and Systems*, IEEE, April 1981, pp. 851-855.

[2] R. Brayton, G. Hachtel, and A. Sangiovanni-Vincentelli, "A Survey of Optimization Techniques for Integrated-Circuit Design," *Proceedings of the IEEE*, Vol. 69, No. 10, October 1981, pp. 1334-1362.

Path	predicted [ns] T_{BEout}, T_{SWout}	SPICE [ns] T_{BEout}, T_{SWout}	error [%]
$A_0 \rightarrow Sum_0$, rise	6.20, 1.66	6.76,1.73	-7
$A_0 \rightarrow Sum_0$, fall	5.53, 0.49	6.20, 0.62	-12
$A_1 \rightarrow Sum_1$, rise	6.26, 1.66	6.67, 1.73	-6
$A_1 \rightarrow Sum_1$, fall	5.53, 0.49	5.96, 0.59	-8
$A_2 \rightarrow Sum_2$, rise	6.28, 1.66	6.72, 1.73	-6
$A_2 \rightarrow Sum_2$, fall	5.52, 0.49	5.92, 0.61	-8
$A_3 \rightarrow Sum_3$, rise	6.19, 1.66	6.78, 1.74	-8
$A_3 \rightarrow Sum_3$, fall	5.44, 0.49	5.86, 0.60	-8
$c_{in} \rightarrow c_{out}$, rise	17.43, 2.46	15.14, 2.55	+12
$c_{in} \rightarrow c_{out}$, fall	18.30, 0.55	18.99, 0.61	-4

Optimization time (DEC 20/60 running CLU):
set up: 2.7 cpu sec optimization: 519.6 cpu sec

Total SPICE verification time (DEC 20/60 running FORTRAN):
1468.6 cpu sec

Table 2: Optimization Statistics for the Four Bit Adder

[3] A. Ruehli, P. Wolff, and G. Goertzel, "Analytic Power/Timing Optimization Technique for Digital System," *Proceedings 14th Design Automation Conference*, IEEE, June 1977, pp. 142-146.

[4] N. Jouppi, "Timing Analysis for nMOS VLSI," *Proceedings 20th IEEE Design Automation Conference*, IEEE, June 1983, pp. 411-418.

[5] S. Trimberger, "Automated Performance Optimization of Custom Integrated Circuits," *Proceedings International Symposium on Circuits and Systems*, IEEE, May 1983, pp. 194-197.

[6] R. Fletcher and M. Powell, "A Rapidly Convergent Descent Method for Minimization," *Computer Journal*, Vol. 6, 1963, pp. 317-322.

[7] D. Bertsekas, "Projected Newton Methods for Optimization Problems with Simple Constraints," *SIAM Journal on Control and Optimization*, Vol. 20, 1982, pp. 221-246.

[8] Yonathan Bard, *Nonlinear Parameter Estimation*, Academic Press, 1974.

[9] B. Liskov, A. Snyder, R. Atkinson, and C. Schaffert, "Abstraction Mechanisms in CLU," *Communications of the ACM*, Vol. 20, No. 8, August 1977, pp. 564-576.

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CAMBRIDGE MASSACHUSETTS 02139

VLSI Memo No 84-216 December 1984

A Mixed-Integer Linear Programming Problem Which is Efficiently Solvable*

Charles E. Leiserson and James B. Saxe**

ABSTRACT

Efficient algorithms are known for the simple linear programming problem where each inequality is of the form $x_i - x_j \leq a_{ij}$. Furthermore, these techniques extend to the pure integer programming variant of the problem where all the unknowns are required to be integers. This paper gives an efficient solution to the mixed-integer linear programming variant where some, but not necessarily all, of the unknowns are required to be integers. The algorithm we develop is based on a graph representation of the constraint system and runs in $O(|V||E|\lg|V|)$ time. It has several applications including optimal retiming of synchronous circuitry, VLSI layout compaction in the presence of power and ground busses, and PERT scheduling with periodic constraints.

*Presented at the 21st Annual Allerton Conference on Communication, Control and Computing, October 1983. This research was supported in part by the Defense Advanced Research Projects Agency under Contract No. 0014-80-C-0622 and by the Office of Naval Research under Contract No. 0014-76-C-0370.

**Leiserson: Laboratory for Computer Science, M.I.T., Room NE43-835, Cambridge, MA 02139. (617) 253-5833; Saxe, current address: Department of Computer Science, Carnegie-Melon University, Pittsburgh, PA 15213.

1. Introduction

Much research has centered on the problem of finding shortest paths in graphs. It is well known that there is a direct correspondence between the *single-source shortest-paths problem* and the following simple linear programming problem.

> *Let S be a set of linear inequalities of the form $x_j - x_i \leq a_{ij}$, where the x_i are unknowns and the a_{ij} are given real constants. Determine a set of values for the x_i such that the inequalities in S are satisfied, or determine that no such values exist.*

This paper considers the *mixed-integer* linear programming variant of this problem in which some (but not necessarily all) of the x_i are required to be integers. The problem arises in the context of synchronous circuit optimization [8], but it has applications to PERT scheduling and VLSI layout compaction as well.

Before formally defining the mixed-integer programming problem, we restate the linear programming problem above in another form.

> **Problem L.** *Let $G = (V, E, a)$ be an edge-weighted, directed graph, where $V = \{1, 2, \ldots, |V|\}$ is the vertex set, the set E of edges is a subset of $V \times V$, and for each edge $(i, j) \in E$ the edge weight a_{ij} is a real number. Find a vector $x = (x_1, x_2, \ldots, x_{|V|})$ satisfying the constraint that*
>
> $$x_j - x_i \leq a_{ij}$$
>
> *for all $(i, j) \in E$, or determine that no feasible vector exists.*

The graph G is called a *constraint graph* for the linear programming problem. There are three advantages in adopting a graph representation of the problem. First, an adjacency-list representation [1, p. 200] of the constraint graph G is more economical than, for example, a linear programming tableau or, when the graph has relatively few edges, a matrix of the a_{ij}. Second, Problem L frequently arises in situations that are naturally described by a graph. Finally, the graph-theoretic formulation helps in understanding the algorithms that solve this kind of problem.

A method for solving Problem L was discovered in the late 1950's by Ford and Bellman [7, p. 74]. Yen [12] gave some improvements to the Bellman-Ford algorithm as well as a cogent analysis showing that its running time is $O(|V|^3)$. This bound is easily improved to $O(|V||E|)$ by using an adjacency-list representation for the constraint graph.

The Bellman-Ford algorithm can also be used to solve the *integer* linear programming variant of Problem L, in which all the x_i are required to be integers. If the edge weights a_{ij} all happen to be integers, the Bellman-Ford algorithm will produce integer values for the x_i. If the a_{ij} are not integers, however, but the x_i are required to be integers, each edge weight a_{ij} may be replaced by $\lfloor a_{ij} \rfloor$ without affecting the satisfiablity of the inequalities.

The focus of this paper is the *mixed-integer* variant of Problem L.

> **Problem ML.** *Let $G = (V, V_I, E, a)$ be a "mixed-integer," edge-weighted, directed graph, where $V = 1, 2, \ldots, |V|$ is the vertex set, the set V_I is a subset of V, the set E of edges is a subset of $V \times V$, and for each edge $(i, j) \in E$ the edge weight a_{ij} is a real number. Find a vector $x = (x_1, x_2, \ldots, x_{|V|})$ satisfying the constraints that*
>
> $$x_j - x_i \leq a_{ij}$$

for all $(i,j) \in E$ *and that* $x_i \in \mathbb{Z}$ *for all* $i \in V_I$*, or determine that no feasible vector exists.*

The vector $x = (x_1, x_2, \ldots, x_{|V|})$ is called a *solution* to graph G, and if graph G has a solution, we say that G is *satisfiable*. When it is clear from context, we use the same terminology for Problem L.

In addition, we shall refer to the vertices in V_I as the *integer* vertices of G and the vertices in $V_R = V - V_I$ as the *real* vertices of G. We also partition the set of edges into two sets depending on whether the vertex at the head of the edge is integer or real:

$$E_I = \{(i,j) \in E \mid j \in V_I\},$$
$$E_R = \{(i,j) \in E \mid j \in V_R\}.$$

This paper presents two algorithms to solve Problem MI. The first, which runs in $O(|V|^2|E|)$ time, is a straightforward extension of the Bellman-Ford algorithm. The second is more sophisticated and has a running time of $O(|V||E|\lg|V|)$ for arbitrary graphs and $O(|V||E|)$ for dense graphs. We conjecture that the $O(|V||E|)$ running time achieved by the Bellman-Ford algorithm for the pure linear programming and pure integer programming versions of the problem is not achievable in general for Problem MI.

The remainder of this paper is organized as follows. Section 2 reviews the Bellman-Ford algorithm. Section 3 presents a simple relaxation algorithm for solving Problem MI. Section 4 discusses two techniques– Dijkstra's algorithm and *reweighting*—which are used in Section 5 to construct an asymptotically efficient algorithm for Problem MI. We discuss applications and present some concluding remarks in Section 6.

2. Shortest paths and the Bellman-Ford algorithm

This section reviews how the *Bellman-Ford algorithm* solves Problem L. Although the results of this section are well known and can be found in most textbooks on combinatorial optimization (see, for example, [7, p. 74]), we repeat the material here for the reader's convenience.

There is a natural correspondence between Problem L and the graph-theoretic *single-source shortest-paths* problem. Let $G = (E, V, a)$ be an instance of Problem L. Suppose that for each vertex $i \in V$, there is a path to i from vertex 1, and let d_i be the weight of shortest (least-weight) path from vertex 1 to vertex i. (At the end of the section, we shall discuss the case in which some vertices are not reachable from vertex 1.) Then for any edge $(i,j) \in E$, we have $d_j - d_i \leq a_{ij}$ since the edge (i,j) can be appended to a shortest path from vertex 1 to vertex i to produce a path from vertex 1 to vertex j of weight $d_i + a_{ij}$. Thus the shortest-path weights d are a solution to G.

Whenever G is satisfiable, there are infinite number of solutions. For example, if x is a solution to G, then uniformly adding any constant k to each x_i yields another solution y, where $y_i = x_i + k$ for each $i \in V$. The assignment $x_i \leftarrow d_i$ gives each x_i its largest possible value subject to the constraint that $x_1 = 0$. To see this, consider any path p of weight d_i from vertex 1 to vertex i. If the inequalities associated with the edges of p are summed, the unknowns associated with the intermediate vertices cancel and the result is the inequality $x_i - x_1 \leq d_i$.

Whenever the graph G contains some cycle c whose weight is negative, the shortest path weight from vertex 1 to any vertex i on cycle c is undefined because the weight of any path

to vertex i can be diminished by appending a traversal of c. In this case the graph G is not satisfiable. If the inequalities associated with the edges of c are summed, all the unknowns x_i cancel, and the resulting inequality asserts that 0 is less than or equal to the weight of c, which is false.

The Bellman-Ford algorithm, which is given below, solves Problem L by finding the weight of the shortest path to each vertex from vertex 1. Should the graph contain a negative-weight cycle, the algorithm reports that the graph is unsatisfiable by calling the procedure *Fail*, whose semantics we leave unspecified.

Algorithm BF *(Bellman-Ford algorithm).*

```
BF1.  x_1 ← 0;
BF2.  for i ← 2 to |V| do x_i ← ∞;
BF3.  for ind ← 1 to |V| − 1 do
BF4.      foreach (i, j) ∈ E do
BF5.          if x_j > x_i + a_ij then x_j ← x_i + a_ij;
BF6.  foreach (i, j) ∈ E do
BF7.      if x_j > x_i + a_ij then Fail
```

For each vertex $j \in V$, the Bellman-Ford algorithm iteratively updates the weight x_j of a tentative shortest path from vertex 1 to vertex j. After initialization, the algorithm makes $|V|-1$ passes through the edges in E and *relaxes* each edge (i,j) by computing $x_j \leftarrow \min(x_j, x_i + a_{ij})$.

A simple analysis due to Yen [12] indicates why the Bellman-Ford algorithm works. The weight x_j converges to the weight d_j of a shortest path from vertex 1 to vertex j if the edges on the path are relaxed in order along the path. The sequence of edges relaxed by the Bellman-Ford algorithm consists of $|V|-1$ copies of some ordering of E, and therefore contains every vertex-disjoint path as a subsequence. If there are no negative-weight cycles in G, then every shortest path is vertex disjoint, so each x_i converges to the shortest-path weight d_i. On the other hand, if there is a negative-weight cycle in the graph, the algorithm detects this condition by iterating once more through all edges to see whether any of the inequalities remain unsatisfied.

The Bellman-Ford algorithm as given above determines the weight of the shortest path from vertex 1 to each vertex, and therefore solves Problem L whenever all vertices of G are reachable from vertex 1. The code can be adapted to solve Problem L on arbitrary graphs by simply changing the initialization step (lines BF1–BF2). In particular, if each x_i is assigned a finite initial value u_i, the relaxation in lines BF3–BF5 sets each x_i to its maximum value subject to the constraints that $x_j - x_i \le a_{ij}$ for each edge $(i,j) \in E$ and that $x_i \le u_i$ for each vertex $i \in V$. Notice that whenever the constraint graph G is satisfiable, it is satisfiable subject to the additional constraints $x_i \le u_i$. Should the inequalities be inconsistent because there is a negative-weight cycles in the graph, the relaxation will not converge to a solution, and the inconsistency will be detected by the test in lines BF6–BF7.

3. Simple relaxation algorithms for Problem MI

As was mentioned in the introduction, Problem MI can be solved directly by the Bellman-Ford algorithm when all unknowns are real (Problem L) and when all unknowns are integer. The combination of integer and real unknowns, however, seems to make the problem harder.

In this section, we gain some intuition about the structure of Problem MI by introducing two algorithms that solve it in much the same way as the Bellman-Ford algorithm solves Problem L. The asymptotically efficient algorithm from Section 4 is derived from the second of these algorithms.

A natural approach to solving Problem MI is to see whether the Bellman-Ford relaxation approach can be made to work. Since we have both integer and real vertices in the graph, however, we must modify the relaxation step BF5 in the Bellman-Ford algorithm to produce an integer value whenever j is an integer vertex (line R6). This approach does in fact work, but it requires more iterations than the simple Bellman-Ford algorithm. The next algorithm solves Problem MI. The number of iterations n in line R2 will be determined in the analysis following the algorithm.

Algorithm R. ***(Relaxation.)***

R1. **foreach** $i \in V$ **do** $x_i \leftarrow 0$;
R2. **for** $ind \leftarrow 1$ **to** n **do**
R3. **foreach** $(i,j) \in E$ **do**
R4. **begin**
R5. $x_j \leftarrow \min(x_j, x_i + a_{ij})$;
R6. **if** $j \in V_I$ **then** $x_j \leftarrow \lfloor x_j \rfloor$;
R7. **end**;
R8. **foreach** $(i,j) \in E$ **do**
R9. **if** $x_j > x_i + a_{ij}$ **then** *Fail*;

In order to determine a value of n such that Algorithm R works, we introduce the notion of a *reducing path*. Let p be a path starting at some vertex k, and suppose that x_k is initially set to 0 and that all the remaining x_i are initialized to ∞. Suppose the edges in path p are traversed in order starting from k, and each edge (i,j) along the path is relaxed as in statements R5–R6. If each relaxation of an edge (i,j) reduces the value x_j, the path p is called a *reducing path*.

Whenever a sequence of edges contains all reducing paths as subsequences, the relaxation of each edge in the sequence in order yields a solution. (The proof is analogous to Yen's analysis [12] of the Bellman-Ford algorithm.) The Bellman-Ford algorithm solves Problem L because in a satisfiable graph with only real vertices, each vertex occurs at most once on any single reducing path. (And in fact, every shortest path is a reducing path.)

When some unknowns are integer and some are real, however, it is possible for a reducing path to visit the same vertex more than once, even if the graph is satisfiable. For example, in the graph shown in Figure 1, the reducing path $p = 3 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 3 \rightarrow 2$ visits vertices 2 and 3 three times each. If all the x_i are initially set to 0, the edges of p must be relaxed in their order along the path to achieve convergence. Moreover, relaxing the entire edge set in some arbitrary order only $3 = |V| - 1$ times might not achieve convergence. Since the value of n in line R2 must be at least the maximum number of edges in any reducing path, the value $|V| - 1$, which was used in Algorithm BF, will not suffice.

Fortunately, reducing paths are never very long in satisfiable graphs because of the following lemma.

Lemma 1. *Suppose $G = (V, V_I, E, a)$ is satisfiable. If p is a reducing path in G, then*

1. *p visits no integer vertex more than once, and*
2. *p never visits the same real vertex twice without visiting some integer vertex in between.*

Proof. If either condition is violated, then the reducing path p can be extended indefinitely by repeating the cycle that causes violation. ∎

Lemma 1 allows us to determine a value for n in line R2 of Algorithm R such that the x converges to a solution whenever G is satisfiable. Any reducing path contains each integer vertex at most once and each real vertex at most $|V_I| + 1$ times. Since the number of edges in a path is one less than the number of vertices, any reducing path for a satisfiable graph can have no more than $|V_I| + (|V_I| + 1)|V_R| - 1 = |V_I||V_R| + |V| - 1$ edges. Thus the limit n of the outer loop in Algorithm R should be set to $|V_I||V_R| + |V| - 1$.

This analysis suggests the following algorithm which is slightly more efficient than Algorithm R, and which forms the basis of the asymptotically efficient algorithm presented in the next section.

Algorithm M. *(Modified relaxation.)*

```
M1.   foreach i ∈ V do x_i ← 0;
M2.   for ind ← 1 to |V_R| do
M3.       foreach (i,j) ∈ E_R do
M4.           x_j ← min(x_j, x_i + a_ij);
M5.   for ind2 ← 1 to |V_I| do
M6.       begin
M7.       foreach (i,j) ∈ E_I do
M8.           x_j ← min(x_j, ⌊x_i + a_ij⌋);
M9.       for ind ← 1 to |V_R| do
M10.          foreach (i,j) ∈ E_R do
M11.              x_j ← min(x_j, x_i + a_ij);
M12.      end;
M13.  foreach (i,j) ∈ E do
M14.      if x_j > x_i + a_ij then Fail;
```

The only difference between this algorithm and Algorithm R is that it treats the edges in E_I separately from the edges in E_R. In lines M7–M8 of Algorithm M, each edge in E_I is relaxed once. There are $|V_I|$ such passes over E_I which are preceded, followed, and separated by *exhaustive relaxations* of the edges in E_R (lines M2–M4 and M9–M11). In each exhaustive relaxation of E_R, edges are relaxed until no further changes in the values of x_j are possible for $j \in V_R$. (Actually, the relaxations in lines M2–M4 and M9–M11 are only guaranteed to be exhaustive if there are no negative-weight cycles in E_R. If there are cycles of negative weight, however, this condition is detected at the end by the convergence test in lines M13–M14.)

4. Dijkstra's algorithm and reweighting

Section 5 gives a more efficient algorithm to solve Problem MI than either Algorithm R or Algorithm M. Two important techniques are used in the algorithm. The first is Dijkstra's

algorithm which finds shortest paths in a graph from a single source in the case when all the edge weights are nonnegative. The other is *reweighting*, which is a technique due to Edmonds and Karp [3] and used by Johnson [6] in his efficient algorithm for solving the *all-pairs shortest-paths problem*.

Given a graph $G = (V, E, a)$ such that all edge weights a_{ij} are nonnegative, Dijkstra's algorithm computes for each vertex i, the weight d_i of the shortest path from vertex 1. Because each edge is relaxed exactly once, this algorithm is faster than the Bellman-Ford algorithm which solves the same problem for arbitrary edge weights. Dijkstra's algorithm derives its efficiency from the observation that along any shortest path from vertex 1, the shortest-path weights d_i form a nondecreasing sequence if all the edge weights are nonnegative. Thus, a sequence consisting of all edges $(i, j) \in E$ in nondecreasing order of the distances d_i contains as subsequences shortest paths from vertex 1 to all vertices in V. Furthermore, such a sequence of edges can be computed on the fly using a priority queue. (The textbook [1] gives a proof of correctness for this algorithm.)

Algorithm D *(Dijkstra's algorithm).*

D1. $x_1 \leftarrow 0$;
D2. **for** $i \leftarrow 2$ **to** $|V|$ **do** $x_i \leftarrow \infty$;
D3. $Q \leftarrow V$;
D4. **while** $Q \neq \emptyset$ **do**
D5. **begin**
D6. *Choose* $i \in Q$ *such that* $x_i = \min_{j \in Q} x_j$;
D7. $Q \leftarrow Q - \{i\}$;
D8. **foreach** $j \in V_R$ *such that* $(i, j) \in E_R$ **do**
D9. $x_j \leftarrow \min(x_j, x_i + a_{ij})$;
D10. **end**;

If the set Q in the algorithm is implemented as a standard priority queue, each extraction (lines D5–D6) and update (line D8) costs $O(\lg|Q|) = O(\lg|V|)$ time. Thus the total running time of Dijkstra's algorithm is $O(|E| \lg|V|)$. Johnson [6] shows that by implementing Q as a fixed-height heap [5], the running time can be brought down to $O(h|E| + h|V|^{1+1/h})$, where h is an integer constant that may be chosen after the input is presented. The choice $h = \lceil \lg|V| \rceil$ gives the bound $O(|E| \lg|V|)$. For families of *dense* graphs where $|E| = \Omega(|V|^{1+\epsilon})$ for some constant $\epsilon > 0$, the choice $h = \lceil 1/\epsilon \rceil$ gives an $O(|E|)$ bound.

Since Dijkstra's algorithm is equivalent to the Bellman-Ford algorithm on graphs with nonnegative edge weights, it can be used to solve Problem L on such graphs. This is not very interesting in itself, since any graph $G = (V, E, a)$ in which all edge weights are nonnegative can be trivially satisfied by setting x_i to 0 for each $i \in V$. Our interest in Dijkstra's algorithm comes from a stronger property of the solutions it finds. Suppose the initialization step (lines D1–D2) is changed so that each variable x_i is initialized to a finite value u_i. Then the relaxation procedure in lines D3–D10 will set each x_i to its largest possible value consistent with the constraints that $x_j - x_i \leq a_{ij}$ for each edge $(i, j) \in E$ and that $x_i \leq u_i$ for each vertex $i \in V$. In other words, lines D3–D10 of Dijkstra's algorithm are functionally equivalent to lines BF3–BF5 of the Bellman-Ford algorithm provided that all the edge weights a_{ij} are nonnegative. Since a graph with only nonnegative edge weights can never contain a negative-weight cycle, no test for convergence is necessary in this case.

The efficient algorithm we shall present to solve Problem MI is a modification of Algorithm M. Notice that lines M9–M11 of Algorithm M exhaustively relax the edges in E_R in a manner similar to lines BF2–BF4 of the Bellman-Ford algorithm. In Algorithm M, however, this code is executed many times. The efficient algorithm to solve Problem MI uses a trick to replace this code with code based on the more efficient relaxation procedure in lines D3-D10 of Dijkstra's algorithm. This trick is the technique of *reweighting* due to Edmonds and Karp [3].

Lemma 2. *Let $G = (V, E, a)$ be an edge-weighted graph, for each $i \in V$ let r_i be a real number, and let $H = (V, E, b)$ where $b_{ij} = a_{ij} + r_i - r_j$ for each edge $(i, j) \in E$. For each vertex $i \in V$ let x_i be a real number and let $y_i = x_i - r_i$. Then $x_j - x_i \leq a_{ij}$ for all $(i, j) \in E$ if and only if $y_j - y_i \leq b_{ij}$ for all $(i, j) \in E$ (that is, x is a solution to G if and only if y is a solution to H.)*

Proof. Trivial. ∎

We call the vector $r = (r_1, r_2, \ldots, r_{|V|})$ a *reweighting* of the graph G.

5. An asymptotically efficient algorithm for solving Problem MI

This section shows how Dijkstra's algorithm and reweighting can be incorporated into Algorithm M to yield a faster algorithm for solving Problem MI. Given a graph $G = (V, V_I, E, a)$, the idea is to find a reweighting r such that the reweighted graph $H = (V, V_I, E, b)$ has edge weights $b_{ij} = a_{ij} + r_i - r_j \geq 0$ for all edges $(i, j) \in E_R$. Lemma 2 guarantees that G is satisfiable if and only if H is satisfiable and also that a solution y to H can be converted into a solution x to G by setting $x_i = y_i + r_i$ for each $i \in V$. The advantage gained by transforming the problem on G to a problem on H is that the relaxation portion of Dijkstra's algorithm (lines D3–D10) can replace the Bellman-Ford relaxation (lines M9–M11), which is the most expensive part of Algorithm M.

The first stage of the algorithm is to determine the reweighting values r_i for all $i \in V$ and the new edge weights $b_{ij} = a_i + r_i - r_j$ for all $(i, j) \in E$. We must choose the values r_i such that $b_i \geq 0$ for all $(i, j) \in E_R$. Since this is equivalent to requiring that $r_j - r_i \leq a_{ij}$ for all $(i, j) \in E_R$, values for the r_i can be found by applying the Bellman-Ford algorithm to the graph (V, E_R, a). The first few lines of the algorithm are:

Algorithm T. *(Efficient algorithm.)*

```
T1.  for i ∈ V do r_i ← 0;
T2.  for ind ← 1 to |V_R| do
T3.      for (i, j) ∈ E_R do
T4.          r_j ← min(r_j, r_i + a_ij);
T5.  for (i, j) ∈ E_R do
T6.      if r_j > r_i + a_ij then Fail
T7.  for (i, j) ∈ E do
T8.      b_ij ← a_ij + r_i − r_j;
```

If the algorithm fails in line T6, then there is a cycle of negative weight among the edges in E_R, and hence graph G is unsatisfiable even in the absense of integer constraints. Otherwise, the values b_{ij} computed in line T8 are nonnegative for all $(i, j) \in E_R$.

The next stage of Algorithm T is to solve the mixed-integer problem on the graph $H = (V, V_I, E, b)$ by alternately relaxing the edges in E_I and the edges in E_R as in Algorithm M. We begin by initializing the values y_i, which will converge to a solution to H if H is satisfiable.

T9. **for** $i \in V$ **do** $y_i \leftarrow 0$;

This initialization has the added fortune of subsuming the first exhaustive relaxation of E_R (lines M2–M4 in Algorithm M). After the execution of line T9 we have $y_j - y_i = 0 - 0 \leq b_{ij}$ for all $(i,j) \in E_R$, which means that the edges in E_R are already exhaustively relaxed.

The next portion of Algorithm T parallels lines M5–M11 of Algorithm M and is where most of the computing gets done.

```
T10.  for ind ← 1 to |V_I| do
T11.      begin
T12.      for (i,j) ∈ E_I do
T13.          y_j ← min(y_j, ⌊y_i + b_ij⌋);
T14.      Q ← V;
T15.      while Q ≠ ∅ do
T16.          begin
T17.          Choose i ∈ Q such that y_i = min_{j∈Q} y_j;
T18.          Q ← Q − {i};
T19.          for j ∈ V_R such that (i,j) ∈ E_R do
T20.              y_j ← min(y_j, y_i + b_ij);
T21.          end;
T22.      end;
```

This code solves the problem on graph H in almost exactly the same way that Algorithm M would. The only difference is the method by which the edges of E_R are exhaustively relaxed. Whereas lines M9–M11 of Algorithm M perform the exhaustive relaxation using the Bellman-Ford algorithm, lines T14–T21 of Algorithm T take advantage of the nonnegativity of the b_{ij} for $(i,j) \in E_R$ and use Dijkstra's algorithm.

The final part of Algorithm T is to check the convergence of the y and to apply Lemma 2 to produce a satisfying assignment x for the original graph G.

```
T23.  for (i,j) ∈ E_I do
T24.      if y_j > y_i + b_ij then Fail;
T25.  for (i,j) ∈ E do
T26.      x_i ← y_i + r_i;
```

Lines T23–T24 check the convergence of y by testing the inequalities associated with the edges in E_I. The inequalities resulting from edges in E_R need not be checked because the relaxation in lines T14–T22 is guaranteed to be exhaustive. (If there were negative-weight cycles in E_R, we would have detected this in lines T5–T6.)

Lines T25–T26 convert the solution y to graph H into a solution x to graph G. Lemma 2 ensures that the inequalities $x_j - x_i \leq a_{ij}$ are satisfied, but we must also show that the x_i are integers for all $i \in V_I$. For each $i \in V_I$ the value y_i is an integer, however, and furthermore, the

values of the r_i produced in lines T1–T4 are zero for all $i \in V_I$. Thus for all the integer vertices, the x_i are integers.

In summary, we have proved the following theorem.

Theorem 3. *Algorithm T solves Problem MI.*

The running time of Algorithm T is $O(|V||E|\lg|V|)$. (Johnson's techniques [6] [5] can be used to reduce the actual running time to $O(|V||E|)$ for dense graphs by implementing the priority queue Q as a fixed-height heap.) Tighter analysis in terms of the sizes of the sets V_I, V_R, E_I, and E_R is possible, however. In particular, the closer bound $O(|V_R||E_R| + |V_I||E_I| + |V_I||E_R|\lg|V|)$ indicates that the algorithm performs even better when the number of integer vertices is small.

6. Applications, extensions, and conclusions

The solution to Problem MI was demanded by a problem concerning optimization of sychronous circuitry by retiming [8]. This section briefly reviews this application, and gives two other problems—compaction of VLSI circuits in the presence of power and ground busses and PERT scheduling with periodic constraints—which can be reduced to Problem L. We also consider an extension of Problem MI where multiple sets of periodic constraints must be satisfied. (For example, some of the x_i are required to be integers, and others to be exact multiples of a constant c.) This section is abbreviated in the extended abstract.

Circuit optimization by retiming

This application is omitted. (The interested reader is refered to [8].)

PERT scheduling

Suppose we have a constraint graph representing milestones in a project, the edge-weights indicate the timing constraints between milestones. Generally, the Bellman-Ford algorithm can be used to provide an optimal scheduling of the milestones. If a work day is from 9:00 a.m. to 5:00 p.m., however, we may not wish to schedule a one-hour job to start at 4:30 p.m. Advancing the job to the next day, however, may cause another job to be advanced as well if the two jobs are constrained to fall near each other. The problem of PERT scheduling with periodic constraints can be cast as Problem MI.

Intuitively, the mixed-integer formulation allows one to include for each job 1. a (real) variable representing the starting time of the job, and 2. an (integer) variable representing, say, noon on the day the job occurs. Thus one can include constraints which say, "This job must finish before 5:00 p.m. on the day it occurs," and "These two jobs must start on the same day."

We also can solve certain problems when there are additional periodic constraints using an algorithm that runs in $O(|V|^3)$ time. As an example, we may wish to have not only variables representing noon on the day that a job starts, but also variables representing the week that a job starts. Thus constraints involving weekends could be taken into consideration.

Circuit compaction

Optimal (one-dimensional) compaction of VLSI circuit layouts [4] is another application of the Bellman-Ford algorithm. Each layout feature is given a variable representing an x-coordinate, and the design rules are enforced using constraints of the form $x_j - x_i \leq a_{i,j}$. It may be desirable, however, to allow feature i to be to the left of feature j or vice versa, but not to allow them

to occupy the same position. Unfortunately, if one wishes to allow this kind of transposition of layout features, either optimality or performance must be sacrificed because the problem becomes NP-complete [9]. But for certain compaction problems arising in practice, transposition of layout features can be allowed.

Some design methodologies enforce the placement of power, ground, and clock to be at regular intervals. For example, one signal processing system [10] requires that these wires be repeated every 200λ, and that the width of all cells in the system be integer multiples of this distance. The designer is then constrained to build a new cell so that the layout features are tightly packed among the global wires. In this context, where some layout features may go on one side or the other of some global wire but may not overlap, the compaction problem can be formulated as Problem MI.

Acknowledgments

We would like to acknowledge the contributions by Flavio Rose of MIT when we first studied this problem. The three of us originally produced a $O(|V|^3)$ time solution to Problem MI which is described in Flavio's master's thesis [11]. Thanks to Ron Rivest of MIT for reading an earlier draft of the paper. Thanks also to Don Johnson of Penn State, Dick Karp of Berkeley, Gene Lawler of Berkeley, and Nimrod Megiddo of CMU for helpful discussions.

References

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Reading, Massahusetts, 1983.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, Vol. 1, 1959, pp. 269–271.

[3] Jack Edmonds and Richard M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the Association for Computing Machinery*, Vol. 19, No. 2, April 1972, pp. 248–264.

[4] Min-Yu Hsueh, "Symbolic layout and compaction of integrated circuits," Memorandum No. UCB/ERL M79/80, University of California, Berkeley, December 1979.

[5] Donald B. Johnson, "Priority queues with update and finding minimum spanning trees," *Information Processing Letters*, Vol. 4, No. 3, December 1975, pp. 53–57.

[6] Donald B. Johnson, "Efficient algorithms for shortest paths in sparse networks," *Journal of the Association for Computing Machinery*, Vol. 24, No. 1, pp. 1–13, January 1977.

[7] Eugene L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[8] Charles E. Leiserson, Flavio M. Rose, and James B. Saxe, "Optimizing synchronous circuitry by retiming," *Third Caltech Conference on Very Large Scale Integration*, Randal Bryant, ed., Computer Science Press, Rockville, Maryland, March 1983, pp. 87–116.

[9] Thomas Lengauer, "On the solution of inequality systems relevant to IC-layout," *Proceedings of the 8th Conference on Graphtheoretic Concepts in Computer Science*, Carl Hanser Verlag,

Munich, West Germany, 1982.

[10] Richard F. Lyon, "A bit-serial VLSI architectural methodology for signal processing," *VLSI '81*, John P. Gray, ed., Academic Press, New York, 1981, pp. 131–140.

[11] Flavio M. Rose, *Models for VLSI Circuits*, Masters Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, March 1982. Also available as MIT VLSI Memo No. 82–114.

[12] Jin Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quarterly of Applied Mathematics*, Vol. 27, No. 4, 1970, pp. 526–530.

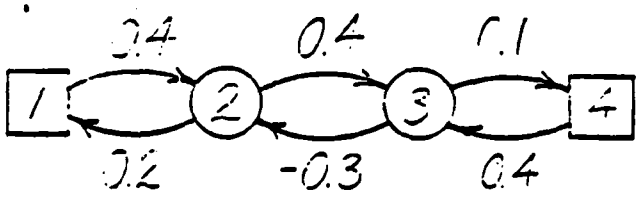


Figure 1

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CAMBRIDGE MASSACHUSETTS 02139

VLSI Memo No. 85-221 January 1985

Improved Bounds on Signal Delay in MOS Interconnect*

John L. Wyatt, Jr., Qingjian Yu, Charles Zukowski,
Han-Ngee Tan, Peter O'Brien**

ABSTRACT

Computationally simple bounds for signal propagation delay in linear RC tree models for MOS interconnect have been derived and have proved useful in timing analysis of digital MOS IC's. We show that these bounds can be derived quite simply as the payoff functions for a certain linear optimal control problem and that they apply not only to RC trees but to more general RC meshes as well. Finally, two methods are given by Rubinstein et al.

---

*To appear in Proceedings, IEEE Conference on Circuits and Systems, Beijuing, People's Republic of China, June 1985. This work was supported by the National Science Foundation under Grant No. ECS-8310941 and by the Air Force Office of Sponsored Research under contract No. F49620-84-C-000

**Wyatt, Zukowski, and O'Brien: Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA 02139, Wyatt: Room 36-865, (617) 253-6718, Zukowski: (617) 253-8169, O'Brien: (617) 253-4619. Yu, current address: Electrical Engineering Teaching Office, East China Institute of Technology, Nanjing, Jiangsu, People's Republic of China. Tan, current address: School of Electrical and Electronic Engineering, Nanyang Technological Institute, Upper Jurong Road, Singapore 2263.4

Improved Bounds on Signal Delay in MOS Interconnect

John L. Wyatt, Jr.[1], Qingjian Yu[2], Charles Zukowski[1],
Han-Ngee Tan[3], Peter O'Brien[1]

[1]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

[2]Electrical Engineering Teaching Office, East China Institute of Technology, Nanjing, Jiangsu, People's Republic of China.

[3]School of Electrical and Electronic Engineering, Nanyang Technological Institute, Upper Jurong Road, Singapore 2263.

Abstract

Computationally simple bounds for signal propagation delay in linear RC tree models for MOS interconnect were derived in [1] and have proved useful in timing analysis of digital MOS IC's [2-4]. We show that these bounds can be derived quite simply as the payoff functions for a certain linear optimal control problem and that they apply not only to RC trees but to more general RC meshes as well. Finally, two methods are given for tightening the original bounds given in [1].

I. Introduction

In digital integrated circuits, signal propagation delay through conducting paths with distributed resistance and capacitance is frequently a significant part of the total delay and grows in relative importance as feature sizes shrink. Timing analysis of digital IC's can be speeded up by using approximate delay formulas, e.g., the "Elmore delay" [5], in place of detailed numerical simulation for interconnect paths. Bounds on the delay, applicable to those paths that can be modelled as linear, nonuniform branched RC ladder networks, i.e., "RC trees," were derived in [1]. But, as discussed in [6-8], certain circuits used in MOS logic cannot be modelled as RC trees because they contain one or more loops of resistors, as shown in Fig. 1. Several examples of such circuits, called "RC meshes," arising in MOS logic networks are given in [6-7]. As used in this paper, the term RC mesh includes RC trees as a special case.

This paper is concerned with bounds on signal propagation delay in linear, lumped RC tree and mesh networks driven by an ideal voltage source. Since the meaning of "delay" is somewhat application-dependent, we bound the delay by bounding the zero-state step response at any output node of interest.

II. Network Differential Equations for RC Meshes

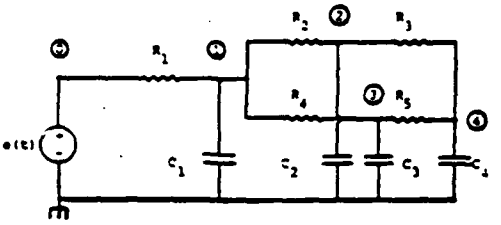


Fig. 1: This linear RC mesh differs from an RC tree because of the resistor loop.

Node Numbering Convention

The ground node is not numbered. The node connected to the voltage source is numbered 0. The remaining nodes are numbered in any order from 1 to N, where N is the total number of capacitors, as in Fig. 1.

We isolate the resistor subnetwork R containing all the resistors and assign reference directions to the capacitor currents $i_1, \ldots, i_N$ as shown in Fig. 2. Let node 0 serve as the datum node of R. The node voltages with respect to datum are given in terms of the capacitor currents by the resistance matrix R as shown below.

$$\begin{pmatrix} v_1-e \\ v_2-e \\ \vdots \\ v_N-e \end{pmatrix} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1N} \\ r_{21} & r_{22} & \cdots & r_{2N} \\ \vdots & \vdots & & \vdots \\ r_{N1} & r_{N2} & \cdots & r_{NN} \end{bmatrix} \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{pmatrix} \qquad (1)$$

Of course R is symmetric since R is reciprocal, and R is positive definite because all resistors are assumed positive.

Consider the step response of the network with zero initial conditions. Substituting $e=1$ and $i_k = -C_k\dot{v}_k$ into (1), we obtain the network differential equations

$$1 - v_i(t) = \sum_{j=1}^{N} r_{ij}C_j\dot{v}_j(t), \quad i=1,\ldots,N, \; t > 0, \qquad (2)$$

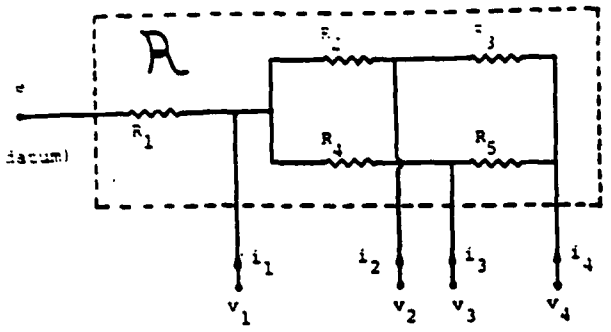


Fig. 2: The resistor subnetwork R extracted from the circuit in Fig. 1 is described by a resistance matrix as in (1).

which are identical in form to eq. (9) of [1]: the only difference is that in [1] certain resistances R_{ij} appear in place of the r_{ij}'s above. The resistances R_{ij} were defined specifically in terms of the topology of a tree in [1], while the r_{ij}'s here are defined for an arbitrary mesh as the elements of the resistance matrix. The reader can easily verify that the two definitions agree in the special case of a tree.

III. Optimal Control Method for Determining Bounds on the Step Response

The original derivation of step response bounds in [1], though entirely correct, is somewhat obscure and applies only to RC trees. The alternate derivation outlined below yields essentially the same results, but it applies to meshes as well and also affords a natural way to incorporate additional information and thereby obtain tighter bounds, as shown in Section IV. Facts 1-4 below parallel the development in [1] and are given here for completeness. Fact 3 will be examined more closely in Section V.

Fact 1

For any three nodes, i, j, k of an RC mesh,

$$r_{ii}\, r_{kj} \geq r_{ki}\, r_{ij} \ . \tag{3}$$

The proof of (3), given in [7], generalizes the argument for the special case of a tree in [1].

Fact 2

The zero-state step response of an RC mesh is completely monotone, i.e.,

$$\dot{v}_j(t) \geq 0, \ j=1,\ldots,N, \ \forall t \geq 0 \ . \tag{4}$$

The proof is in [9].

Fact 3

For any two nodes i and k of an RC mesh and any instant t during the step response,

$$r_{ii}(1-v_k(t)) \geq r_{ki}(1-v_i(t)) \tag{5}$$

$$r_{ki}(1-v_k(t)) \leq r_{kk}(1-v_i(t)) \ . \tag{6}$$

Given Facts 1 and 2, the derivation of (5) and (6) is identical to that in Appendix D of [1], i.e., for (5) note that $r_{ii}(1-v_k) - r_{ki}(1-v_i) =$ (using (2)),

$$\sum_{j=1}^{N} (r_{ii}r_{kj} - r_{ki}r_{ij})\, C_j \dot{v}_j \geq 0,$$

where the last inequality follows from Facts 1 and 2. The proof of (6) is similar.

At this point the strategy becomes one of reduced order modelling with time-domain error bounds. Choosing a distinguished node i as the output node of interest, we seek to describe the system in terms of only two state variables, the distance to equilibrium $(1-v_i(t))$ and its integral

$$f_i(t) \triangleq \int_t^{\infty} [1-v_i(t')]dt' = \sum_k r_{ik}C_k[1-v_k(t)], \tag{7}$$

where the last equality follows upon substituting (2) into the integral and evaluating. Using (5) and (6) in (7) yields the following inequality between these two state variables:

$$\underbrace{\left[\sum_k r_{ik}^2 C_k / r_{ii}\right]}_{\triangleq T_{R_i}} (1-v_i(t)) \leq f_i(t) \leq \underbrace{\left[\sum_k r_{kk}C_k\right]}_{\triangleq T_P} (1-v_i(t)), \ \forall t \geq 0 \ . \tag{8}$$

From (7) one initial condition is

$$f_i(0) = \sum_k r_{ik}C_k \triangleq T_{D_i} \ . \tag{9}$$

It was shown in [1] that step response bounds can be obtained by appropriate manipulations of (4,5) and (7-9) above, but the methodology is somewhat obscure. We believe a clearer view emerges from recasting the calculations into the form of a linear minimum- (and maximum-) time optimal control problem with state constraints, in which an input u(t) is introduced to represent the unknown waveform $v_i(t)$:

Minimize (or maximize) T

for the dynamical system

$$\dot{f}_i(t) = -(1-v_i(t)) \tag{10}$$

$$\frac{d}{dt}(1-v_i(t)) = u(t) \ , \tag{11}$$

with initial conditions

$$f_i(0) = T_{D_i}, \ (1-v_i(0) = 1 \ , \tag{12}$$

state constraints

$$T_{R_i}(1-v_i(t)) \leq f_i(t) \leq T_P(1-v_i(t)), \ \forall t \geq 0 \ , \tag{13}$$

input constraint: $u(t) \leq 0, \ \forall t \geq 0,$ (14)

and terminal condition

$$(1-v_i(T)) = (1-v_i^*), \quad 0 < v_i^* \leq 1 \ . \qquad (15)$$

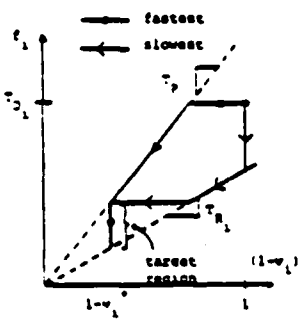


Fig. 3: Fastest and slowest trajectories from the initial state to the target region, subject to the state constraints indicated by dotted lines.

The optimal trajectories can be determined by inspection without recourse to Pontryagin's maximum principle, since the time duration of any path in the $(1-v_i)$ - f_i plane can be found by rearranging and integrating (10) to yield

$$T = \int_{f_{final}}^{f_{init}} \frac{1}{1-v_i} \, d f_i \ . \qquad (16)$$

Thus the fastest trajectory from the initial point to the target interval is the one for which both the region of integration $[f_{final}, f_{init}]$ and the integrand $(1-v_i)^{-1}$ are minimized, and the slowest trajectory is found similarly. See Fig. 3. The minimum and maximum times depend on the "target" voltage v_i^* and are denoted $T_{min}(v_i^*)$ and $T_{max}(v_i^*)$. The inverse functions, denoted respectively $\bar{v}_i(t)$ and $\underline{v}_i(t)$, are readily seen to be the upper and lower bounds, respectively, for all feasible solutions to the optimal control problem and hence for the step response of the mesh. Furthermore, these are the best possible bounds we could construct using the information contained in (10)-(15), since they are attained by feasible trajectories. The algebraic form of $\underline{v}_i(t)$ and $\bar{v}_i(t)$ obtained in this way can be easily read off from Fig. 3 and agrees with the results in [1]: the exact expressions are omitted for the sake of brevity. They approach a well-defined limit in the case of a distributed network, e.g., the simple example in Fig. 4, for which they are plotted in Fig. 5.

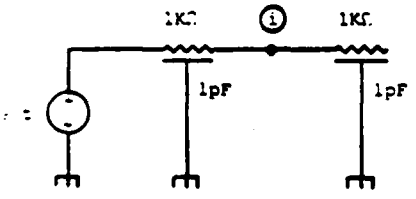


Fig. 4: The bounds approach a well-defined limit for a distributed network such as this one, for which T_{R_i} = 1.33 ns., T_{D_i} = 1.5 ns., T_P = 2.0 ns, and T_{R_i} = 0.33 ns.

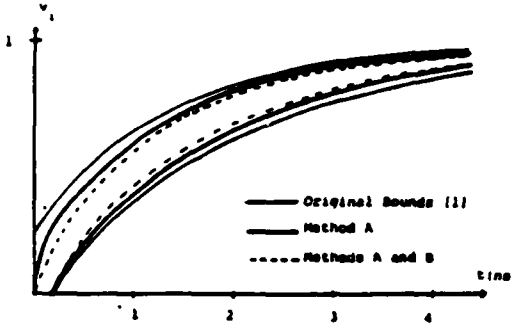


Fig. 5: Step response bounds for the network in Fig. 4, with output taken at node i.

The reader can check that the "Elmore time constant" T_{D_i} is the first moment of the impulse response and therefore a reasonable estimate of the delay. The step response estimate $v_{i,est}(t) \triangleq 1 - \exp(-t/T_{D_i})$, discussed in [8,10], corresponds to a straight line trajectory from the initial condition to the origin in Fig. 3 and is therefore a feasible (but not optimal) solution to the optimal control problem, i.e., $\underline{v}_i(t) \leq v_{i,est}(t) \leq \bar{v}_i(t)$, $\forall t \geq 0$, for every mesh. It is readily seen that $T_{R_i} \leq T_{D_i} \leq T_P$ always and that the estimate and bounds represent an effort to approximate the dynamics of a higher order network by one with a single time constant T_{D_i}: they are exact only in that case. Whenever $(T_P - T_{R_i}) \ll T_{D_i}$, the wedge-shaped region in Fig. 3 is quite narrow and the bounds will be quite tight. Chapter 3 of [8] gives examples of networks for which the bounds are good and others where they are poor.

IV. Method "A" for Bounds Improvement: Limits on the Maximum Slew Rate of Node Voltages

The optimal trajectories shown in Fig. 3 include horizontal segments along which v_i changes while f_i remains constant. Since $\dot{f}_i = -(1-v_i) < 0$, these segments correspond to instantaneous jumps in v_i and cannot occur in practice. We can tighten the bounds by adding constraints eliminating such trajectories. The simplest form for such a constraint is a "minimum slope bound" in the $(1-v_i)$-f_i plane of the form

$$\frac{d f_i}{d(1-v_i)} \geq \tau_i > 0 \ . \qquad (17)$$

This rules out both trajectories in Fig. 3 as feasible solutions. The new optimal trajectories are as shown in Fig. 6, and the corresponding algebraic form for $\underline{v}_i(t)$ and $\bar{v}_i(t)$ is given in [11].

The inequality (17) corresponds to a "slew-rate bound," i.e., a bound on the derivative, for v_i, since

$$\frac{\dot{v}_i}{(1-v_i)} = -(1-v_i)^{\cdot}/(1-v_i) = (1-v_i)^{\cdot}/f_i = \frac{d(1-v_i)}{d f_i} \leq \frac{1}{\tau_i} \ . \qquad (18)$$

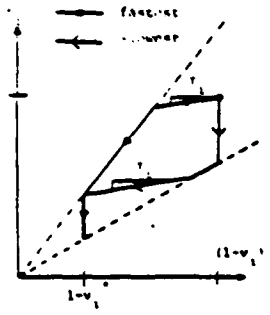


Fig. 6: The slope constraint (17) alters the optimal trajectories in Fig. 3 as shown.

For any mesh we know that $\tau_i \geq r_{ii}C_i$, since $1-v_i = \sum_{j=1}^{N} r_{ij}C_j\dot{v}_j \geq r_{ii}C_i\dot{v}_i$, from (2) and (4). Using $\tau_i = r_{ii}C_i$ can significantly tighten the bounds whenever the mesh contains only a small number of lumped capacitors, as is commonly the case in reasonably accurate circuit models for distributed interconnect [12]. But as progressively more R's and C's are used to model a given section of interconnect, $C_i \to 0$ and (17) becomes useless with $\tau_i = r_{ii}C_i$.

Fortunately, values of τ_i greater than $r_{ii}C_i$ can be found for many RC trees. Space constraints limit us to mentioning only one of the results in this direction obtained in [13]. Consider an RC line with the nodes numbered in increasing order as one moves away from the source. It was first noted in [8] that for such a network

$$\frac{\dot{v}_j(t)}{1-v_j(t)} \geq \frac{\dot{v}_i(t)}{1-v_i(t)}\ , \quad \forall j \leq i,\ \forall t \geq 0\ , \tag{19}$$

a rigorous proof was given in [14], and the result extended in [13] to include all nodes of an RC tree up to the first branch point. Using (19) and (5) in (2), one can show that if i is any node of an RC line, or any node of an RC tree between the source and the first branch point, then

$$\tau_i \geq \sum_{j=1}^{i} r_{ij}^2 C_j / r_{ii} \triangleq \hat{\tau}_{R_i}\ . \tag{20}$$

The dark curve in Fig. 5 shows the original bounds in [1] for the network in Fig. 4, along with the improvement one obtains from using the slew-rate bound (20).

V. Method "B" for Bounds Impovement: Spatial Convexity of Node Voltages

At any instant during the step response of an RC line or tree, the node voltages are a convex function of distance from the source. This is a consequence of the monotone charging of capacitors, indicated in (2). For the network in Fig. 4, a characteristic voltage profile is plotted in Fig. 7, where the "distance" from a point x to the source is represented by the resistance r_{xx}. The

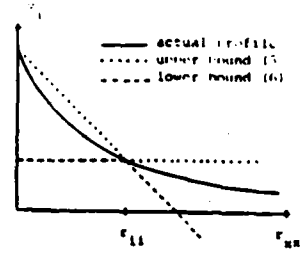


Fig. 7: Typical convex spatial voltage profile for the network in Fig. 4, along with the bounds (5) and (6).

inequalities (5) and (6) bound the node voltages elsewhere in the network in terms of the node voltage v_i of interest and are also plotted in Fig. 7 for this RC line. They are quite different in character: (5) gives a spatially convex profile in this case but (6) does not. Considerable improvement over (6) is possible since a convex curve is bounded below by any tangent line, i.e.,

$$1-v_k \leq (1-v_i)\left[1 + \lambda\left(\frac{r_{kk}}{r_{ii}} - 1\right)\right] \tag{21}$$

for some $\lambda \in [0,1]$. Substituting (21) into the right hand side of (7) and taking the maximum over λ yields

$$f_i \leq (1-v_i)\ \max\left\{T_{D_i}\ ,\ \sum_k \frac{r_{ik}r_{kk}}{r_{ii}}C_i\right\} \leq T_P(1-v_i)\ , \tag{22}$$

thus reducing the effective value of T_P (from 2.00 ns. to 1.83 ns. for the network in Fig. 4) and further improving the voltage bounds as shown in Fig. 5. Current research includes extending this technique to trees.

VI. Concluding Remarks

The research in this paper was stimulated by recent work that appeared in [1,8]. The new developments reported here are 1) two lemmas [7,9] that provide a rigorous basis for extending the theory from RC trees to RC meshes, 2) the optimal control formulation of the problem [13], 3) an extension and rigorous proof [13] of a bound on node voltage slew rates, 4) a systematic method [11] for finding tighter step response bounds using slew rate limits, and 5) method "B" for bounds improvement.

Acknowledgement

It is a pleasure to acknowledge helpful conversations with Professors Lance Glasser and Paul Penfield of MIT, Professor Mark Horowitz of Stanford University, and Dr. Bernard Murphy of the SDA Corporation, Santa Clara, CA. This work was supported by the National Science Foundation under Grant No. ECS-8310941 and the Air Force Office of Sponsored Research under contract No. F49620-84-C-0004.

References

[1] Rubinstein, J., P. Penfield, Jr., and M.A. Horowitz, "Signal Delay in RC Tree Networks," IEEE Trans. Computer-Aided Design, vol. CAD-2, no. 3, pp. 202-211, July 1983.

[2] Putatunda, R., "Autodelay: A Second-Generation Automatic Delay Calculation Program for LSI/VLSI Chips," Proc. IEEE Int. Conf. on Computer-Aided Design, Santa Clara, CA, November, 1984, pp. 188-190.

[3] Jouppi, N., "Timing Analysis for NMOS VLSI," ACM IEEE 20th Design Autom. Conf. Proc., June 1983, pp. 411-418.

[4] Tamura, E., K. Ogawa, and T. Nakano, "Path Delay Analysis for Hierarchical Building Block Layout System," ACM IEEE 20th Design Autom. Conf. Proc., June 1983, pp. 403-410.

[5] Elmore, W.C., "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," Jour. Appl. Physics, vol. 19, pp. 55-63, January 1948.

[6] Wyatt, J.L., Jr., and Q. Yu, "Signal Delay in RC Meshes, Trees and Lines," Proc. IEEE Int. Conf. on Computer-Aided Design, Santa Clara, CA, November 1984, pp. 15-17.

[7] Wyatt, J.L., Jr., "Signal Delay in RC Mesh Networks," to appear in IEEE Trans. Circuits and Systems, 1985.

[8] Horowitz, M.A., "Timing Models for MOS Circuits," Technical Report No. SEL 83-003, Integrated Circuits Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305, December 1983.

[9] Wyatt, J.L., Jr., "Monotone Behavior of Nonlinear RC Meshes," VLSI Memo No. 82-128, November 1982, p. 10. All VLSI Memos are available from the Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139.

[10] Lin, T.-M., and C.A. Mead, "Signal Delay in General RC Networks," IEEE Trans. Computer-Aided Design, vol. CAD-3, no. 4, October 1984, pp. 331-349.

[11] Tan, H.N., and J.L. Wyatt, Jr., "Time-Optimal Trajectories Associated with Voltage Bounds in RC Tree Networks," VLSI Memo No. 84-205, October 1984.

[12] Sakurai, T., "Approximation of Wiring Delay in MOSFET LSI," IEEE J. Solid-State Ckts., vol. SC-18, no. 4, August 1983, pp. 418-426.

[13] Yu, Q. and J.L. Wyatt, Jr., "Improved Bounds on Signal Delay in RC Trees Using Inequalities on the Derivatives of Node Voltages," VLSI Memo 84-197, August 1984.

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CAMBRIDGE, MASSACHUSETTS 02139

VLSI Memo No. 85-223 January 1985

A Simple Cost Model For Multilayer Integrated Circuits*

A. L. Robinson, L. A. Glasser and D. A. Antoniadis**

ABSTRACT

This paper describes a simple model for the cost of implementing a circuit in a multilayer integrated circuit (MLC) technology relative to the cost required for a conventional single-plane version. The model indicates that MLC technologies can be used to cost-effectively implement circuits that have significantly more transistors than can be obtained with a single-plane technology only if the availability of the third dimension for device placement and interconnect results in a significant reduction in the total silicon area used. A potential application for MLC technology is improving the speed of circuits of moderate size by using the third dimension to reduce the length and associated resistance and parasitic capacitance of interconnect lines. Examples of these types of applications are discussed, as is the applicability of the model to other three-dimensional integrated circuit technologies currently under investigation.

---

*This work was supported in part by DARPA contract no. N00014-80-C-0622 and in part by SRC contract no. 83-01-033. A. L. Robinson acknowledges the support of the General Electric Company during the preparation of the manuscript. INTERNAL DISTRIBUTION ONLY UNTIL 5/1/85

**Glasser and Antoniadis: Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA 02139. Glasser: Room 36-587, (617) 253-4677, Antoniadis: Room 39-227 (617) 253-4693, Robinson, current address: General Electric Corporate Research and Development Center, Schenectady, NY 12345.

Generalized Planar Matching

Fran Berman[1]
Tom Leighton[2]
Peter W. Shor[2]
Larry Snyder[3]

[1] Computer Science Department
Purdue University
West Lafayette, Indiana 47907

[2] Mathematics Department and
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

[3] Computer Science Department
University of Washington
Seattle, Washington 98195

Abstract: In this paper, we prove that *maximum planar H-matching* (the problem of determining the maximum number of node-disjoint copies of the fixed graph H contained in a variable planar graph G) is NP-complete for any connected planar graph H with three or more nodes. We also show that *perfect planar H-matching* is NP-complete for any connected outerplanar graph H with three or more nodes, and is, somewhat surprisingly, solvable in linear time for triangulated H with four or more nodes. The results generalize and unify several special-case results proved in the literature. The techniques can also be applied to solve a variety of problems, including the optimal tile salvage problem from wafer-scale integration. Although we prove that the optimal tile salvage problem and others like it are NP-complete, we also describe provably good approximation algorithms that are suitable for practical applications.

Key Words: Approximation Algorithm, Covering, Matching, NP-Complete, Optimal Tile Salvage, Packing, Planar Graph, Wafer Scale Integration.

Fran Berman was supported by a Purdue Research Foundation Summer XL Grant and NSF Grant MCS-80-05387. Larry Snyder was supported by ONR Contract N00014-8-K-0360. Tom Leighton and Peter Shor were supported by Air Force contract OSR-82-0326, DARPA contract N00014-80-C-0622, and the Bantrell Foundation.

END

FILMED

10-85

DTIC