Optimization of Power and Delay in VLSI Circuits Using Transistor Sizing and Input Ordering

by

Chin Hwee Tan

B.S. Electrical Engineering with highest honors
University of Illinois at Urbana-Champaign, 1992

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1994

Signature of Author [illegible]
Department of Electrical Engineering and Computer Science
March 11, 1994.

Certified by [illegible]
Jonathan Allen
Director, Research Laboratory of Electronics, MIT
Professor, Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by [illegible]
Frederic R. Morgenthaler
Chair, Department Committee on Graduate Students

Optimization of Power and Delay in VLSI Circuits Using Transistor Sizing and Input Ordering

by

Chin Hwee Tan

Submitted to the Department of Electrical Engineering and Computer Science on March 11, 1994 in partial fulfillment of the requirements for the Degree of Master of Science in Electrical Engineering.

ABSTRACT

Low-power design is becoming increasingly important in today's technology as wireless communication and mobility of equipment become increasingly desirable. In this thesis, a fast and efficient low power design method using cell libraries is developed. This optimization routine utilizes accurate and efficient statistical power estimation methods, transistor sizing, and input ordering. A delay model which takes into account input transition time is developed. An augmented cell library that contains cells that have been designed and sized to give good power and delay trade-offs is constructed, keeping area and input ordering in mind. Finally, an algorithm that selects the best sized versions to use for the circuit so that a given delay constraint is satisfied with minimal power dissipation is developed. Options that use the switching probabilities of a node, input ordering, and critical path analysis, are also provided to enhance the basic algorithm.

Thesis supervisor: Jonathan Allen

Title: Director, Research Laboratory of Electronics, MIT
Professor, Department of Electrical Engineering and Computer Science

Acknowledgments

Special thanks to my thesis supervisor, Jonathan Allen, for his help and advice throughout the development and completion of this thesis. His patience, understanding and encouragement has been invaluable.

Thanks to Jose Monteiro, Ignacio McQuirk, Ricardo Telichevesky, Mark Seidel and Chris Umminger for their technical assistance, and to rest of the VLSI group up on the 8th floor for their help in one way or another.

Thanks to the Singapore Economic Development Board for their support of my pursuit of a graduate degree at MIT, which is truly a great institution, both for its achievements and its people.

Thanks to all my friends in Massachusetts, who have made my stay here enjoyable and memorable.

Heartfelt thanks to my family, especially my parents, for all their love and support, for always.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Introduction

In VLSI circuit design, two major concerns in optimization have been delay and area. Research has been done on the circuit level to examine the trade-offs between them. In particular, transistor sizing has been well established as a good way to achieve reductions in the delay of circuits, while the resultant increase in rectangular area from transistor sizing can be minimized by special layout techniques.

As wireless communication and mobility of equipment become increasingly desirable, power dissipation of circuits has become a major concern in circuit synthesis. In performance driven synthesis of VLSI circuits, low-power design has joined the ranks of area and delay as major motivations in optimization.

Hence in today's VLSI circuit design, there is a need to ensure low power dissipation while satisfying delay constraints.

1.2 Background

1.2.1 Transistor Sizing

Transistor sizing is well established as an effective way to speed up circuits. Numerous studies have been done in this area [1 to 9].

For static CMOS, the delay of a transition can be modelled as dependent on RC, where R is the effective resistance of the transistors in the pull-up or pull-down circuitry and C is the capacitive load driven by these transistors. R is inversely proportional to the width of the transistors while C is proportional to the size of the transistors in the next stage. Hence by increasing the width of the transistor at the current stage delay can be reduced.

The effects of transistor sizing were studied by custom sizing a four-bit adder [38]. A speedup of 2.3:1 was achieved. The transistors were arranged such that the large p-transistors were above the small n-transistors and vice versa, so that no increase in rectangular area resulted from sizing.

In most of the literature on sizing, the length of each transistor is kept at a set value while the width is treated as a continuous variable. Linear Programming methods or other numerical simulation methods are then applied to find the optimal size for each transistor in a circuit [1, 17].

Delay has been shown to be reduced by 60% [2] and in certain cases up to 73% [1], hence proving that transistor sizing is a useful tool in delay optimization.

1.2.2 Power Estimation

As power dissipation becomes an increasingly important issue, accurate power estimation models are needed. Previous work on power, delay and area optimization such as Berkelaar's work, [17, 18] lack a technique for accurately estimating power dissipation.

Recent developments of probabilistic techniques have produced a fast and efficient way of estimating power [13], which is proportional to the average switching probability of a node. The power dissipation of a gate is approximated by the change in energy for charging and discharging the output capacitance of the gate. Since a gate does not necessarily switch at every clock cycle, the frequency of switching is estimated by the clock frequency multiplied by the expected number of switches per cycle.

Average power is given by:

$$P_{avg} = \left(\frac{1}{2} \times C_{load} \times V_{dd}^2\right) \times \left(\frac{E\,(transitions/cycle)}{T_{cyc}}\right) \qquad \text{(Eqn 1.1)}$$

where P_{avg} denotes average power, C_{load} is the load capacitance, V_{dd} is the supply voltage, T_{cyc} is the global clock period, and E(transitions) is the expected value of the number of gate output transitions per global clock cycle [13].

In [13], signal probabilities are estimated by a process of symbolic simulation. A general delay model is used so that the Boolean conditions that cause glitching in circuits can be correctly computed to be included in calculations of switching probabilities.

This statistical method of power estimation provides a simple way of examining power dissipation in terms of sizing, since gate capacitance is proportional to transistor width.

1.2.3 Input ordering

Delay through gates with multiple inputs is dependent on the arrival times of the input transitions. The time between the latest switching of the inputs to the switching of the output is minimal if the input that switches last is closest to the output node, due to body effects.[37]

The inputs to a gate can be ordered such that the latest arriving input is placed at the fastest pin of a gate, so as to achieve better speed performance. Previous studies have shown that input ordering can achieve a speedup of up to 60% decrease in delay [25].

Input ordering does not cause any increase in power dissipation due to capacitive load increase, and may actually decrease power due to shorter delay times and hence possible reduction in glitching. It provides yet another means of delay and power optimization.

1.3 Motivation

Previous work has attempted to size individual transistors in custom designed circuits. However, commercially, product to market times must be small, and hence much circuit design is done with standard cells as the target technology. Since standard cell libraries are widely used, it is feasible to have a library of gates with transistors that are previously sized to give good power and delay trade-offs, and then the problem of optimization is to choose the best version of each cell to use. This provides a method with larger granularity, and because of the early binding of transistor widths, a computationally simple method of optimization.

In many available standard cell libraries, cell transistors are not minimum-sized, and only one version of each gate is available. Examination of the power and delay curves of various gates show that delay can be reduced significantly with sizing without much increase in power. Hence, a library where several sized versions of a gate are available, where each gate is sized with area minimization and input ordering in mind such that they give good delay response without high power dissipation, would be very useful when designing for power and delay performance. Since the aim is to have low power dissipation, circuits are first mapped with minimum-sized gates and then changed to larger gates as necessary to satisfy delay constraints. In this way, we start with minimum power, and gates that are not dominant in determining the delay of the circuit remain minimum-sized, thereby ensuring low-power dissipation. Having mapped the circuit, the switching probability of each node can be calculated, and this information, as well as the delay parameters of each gate, can be utilized in optimization routines that select the version that is best suited for each node, such that a given delay constraint is satisfied with minimum power.

1.4 Outline of thesis

This thesis presents a fast and efficient low power design method using cell libraries. Transistor sizing, input ordering and accurate and efficient statistical power estimation methods are utilized in the optimization process. This work can be divided into three sections. The first is the development of an accurate delay model, which takes into account input transition times and the different delay times of each input pin of a gate. This is described in chapter 2.

The second section is the development of an augmented standard cell library, which contains cells that have good power and delay trade-offs. This involves designing the geometry of each cell so as to reduce area increases due to sizing, developing a sizing methodology for delay reduction, a selection process of cells from power and delay curves, and finally extracting the necessary parameters. All these processes are described in chapter 3.

The last section is the development of optimization strategies used in selecting the best versions of each gate function to use in a circuit so as to satisfy a delay constraint while minimizing power dissipation. Chapter 4 describes the basic traversal and selection process, including circuit delay calculations and critical path determination.

Chapter 5 describes several options implemented to enhance the basic algorithm. These options include the threshold option which uses the switching probabilities and output load of nodes to determine what cells to be changed first, a check-critical-path option and a maxtime option that allows more flexibility in cell selection. Chapter 6 describes how input ordering is implemented and used in optimizing circuits.

Results from testing the augmented library with the selection algorithms on several test circuits are presented in chapter 7, along with a discussion of the effectiveness of the cells in giving good power and delay trade-offs, as well as the contribution of the various options.

The thesis is concluded with a discussion of the contribution of this work and possible avenues for future work.

CHAPTER 2

Delay Modeling

2.1 Introduction

An accurate delay model is a necessity for speed optimization. Often, in optimization routines, gate delays are treated as a fixed quantity, regardless of input slope and output load. There is also a tendency to associate a gate with only one delay, ignoring the difference between output rise (pull-up) and fall (pull-down) times. These issues, however, have been shown to affect the speed of a gate [12, 27], and hence need to be taken into account in delay estimation.

2.2 Delay Model

In this thesis, a "pin delay" model for delay is used. This model is built upon that provided by the Sequential Interactive Synthesis (SIS) package [32]. The output delay is modeled for each pin of a gate when it is the last one to change, or the one that will cause a switching event. This delay model introduces block and drive values for each input pin into the cell library characterization. Separate block and drive values are derived for rise and fall delay. Delay is estimated by:

$$Gate\_delay = block\_delay + \frac{Output_{load}}{Output_{drive}} \qquad \text{(Eqn 2.1)}$$

However, this delay model does not take input transition time into account. To verify this delay model and to investigate the effects of input transition time on output delay, cells from a standard cell library were simulated. Cells were laid out with different output loads to examine the dependence of delay on output load, as well as with different input loads (or the load the fanin node sees), to vary the input transition time (Fig. 2.1).

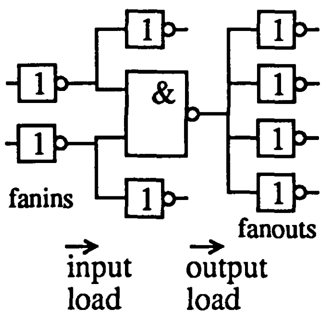


FIGURE 2.1 Nand gate with input load of two and output load of four.

All cells are designed in static CMOS. The output load seen at a node is the sum of gate capacitances of all the p and n transistors it is connected to. The input load of a gate pin is the output load seen by the fanin node connected to this pin, and it includes the gate capacitances of the p and n transistors in the library cell that this pin leads to, as well as the gate capacitances of the transistors in other cells that are connected to the fanin node. All capacitive loads are measured in terms of the load of an inverter, or the capacitive load of a p-transistor and an n-transistor. To vary the input transition time, cells were laid out with their input nodes connected to varying numbers of inverters, thereby varying the load of the fanins and hence varying the transition times of the fanin output, which is the input to the current gate. The output of the gate was also connected to varying numbers of inverters to vary the output load.

The delay values were obtained for each pin in the gate by simulating this gate with the pin input as the one that will cause a transition at the output, other inputs remaining constant during this transition. These cells were simulated with HSPICE and their delay plotted against output load (Fig. 2.2). For a given input transition time, delay is found to have a linear relation to output load, verifying the delay model used in SIS. However, as input transition time is varied by varying the input load, delay is seen to shift upward substantially. This shows that input transition time cannot be ignored in the delay model.

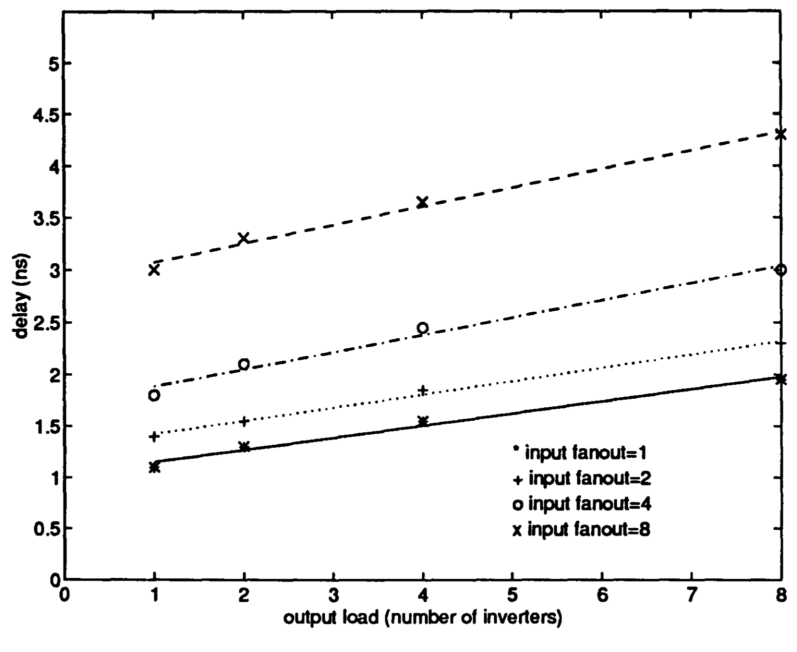


FIGURE 2.2 Fall delay of oai21 gate versus output load with different input transition times.

To investigate the dependency of output delay on input transition time, delay is plotted against the input load (Fig. 2.3). For a given output load, delay has a linear relation with input load as well. Recognizing the effect of input slope on output delay, a new parameter, the input drive, is added into the delay model. Delay is now estimated by:

$$Gate\_delay = \frac{Input_{load}}{Input_{drive}} + block\_delay + \frac{Output_{load}}{Output_{drive}} \qquad \text{(Eqn 2.2)}$$

The SIS delay package has been modified to take input drive into account.

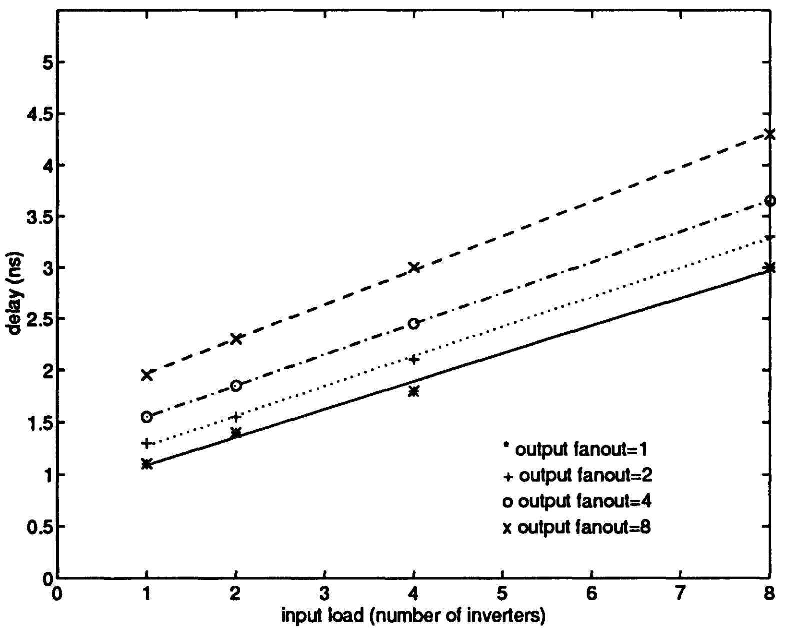


FIGURE 2.3 Fall delay of oai21 gate versus input load with different output loads

2.3 Data Point

To ensure that no overlap of delay calculations occurs between successive gates in a circuit, delay values are taken from a point in the input transition (e.g. when it reaches 50% of its final value) to a corresponding time in the output transition (e.g. when that output reaches 50% of its final value).

This method has been used in previous work to find delay. Brocco modelled gate delay as the time to reach 20% of final value after the input from the previous stage has reached 20% of its final value [12]. Kayssi used a similar method, using V_{il} as the data point [27]. Weste and Eshraghian modelled the time taken for a logic transition to pass from input to output as the time difference between the 50% level of the input transition to the 50% output level [35].

To investigate the best data point with which to obtain delay values, voltage transition graphs of several gates were analyzed. The input drive, block delay and output drive were obtained with different data points. The voltage transition graph of a node is found to be initially very dependent on the rate of change of the input. Taking delay values from 25% (close to threshold voltage) of input transition to 25% of output transition results in a small input drive value, comparable with that of the output drive. As the data point is moved upwards from 25% to 50% to 70%, the input drive increases, while the output drive decreases. This indicates that the last portion of the transition graph is heavily dependent on the output load.

Analyzing the delay among several gates, the 50% point was found to be most consistent. Taking fall delay from the 25% data point yields a much larger delay than when using the 70% data point for certain gates, such as the 2-input nand gate, whereas for rise delay the 70% data point yielded a much larger delay (Fig. 2.4). Delay values taken with the 50% point are in between the extremes and hence more reliable.

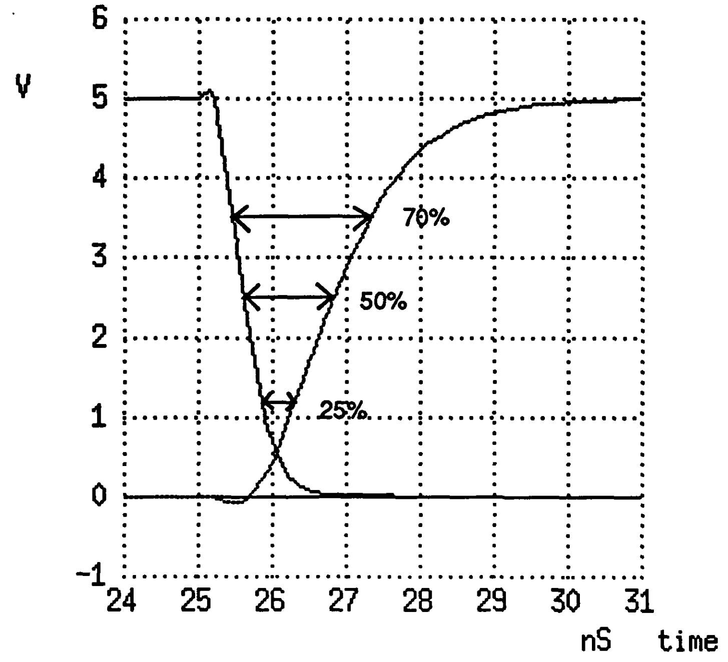


FIGURE 2.4 SPICE plots for nand2 rise delay to illustrate 25%, 50% and 70% data points.

The various data points were also tested on an adder circuit and a small nine-gate circuit. Circuit delays were obtained from SPICE outputs, using the three data points discussed. These delays were compared to that obtained by adding the gate delays, obtained from the corresponding data point, of all gates in the critical path. The best agreement between the two methods of calculating circuit delay was from the delays obtained with the 50% data point.

A problem that exists for such a data point as well as that for bigger levels, (e.g. 70%) is that for certain gates, the slope of the output transition is much steeper than that of the input transition, which means that output may reach 50% of its final value before input does, resulting in negative gate delay. Such negative gate delays will not occur if the data point is close to the trigger value (20% or V_{il}) [27].

For the 50% value point, negative delay were found to occur only for gates with very large input loads. Since large fanout loads cause large delays and are undesirable, many mapping algorithms disallow fanout to exceed a certain value or have options that utilize fanout optimization to improve delay. When large loads are rare, negative delays are rare when using the 50% data point.

The 50% data point is found to be the most consistent and accurate, hence delay values for the library cells are obtained using the 50% data point. Delay values are taken from the time input transition reaches 50% of its final value to the time when output transition reaches 50% of its final value.

2.4 Capacitive Load Extraction

Capacitive load has been shown by various work to be dependent on the width of transistors [1, 4, 5]. However, parasitic and wiring capacitances contribute to the capacitive load as well. To ensure accurate delay estimation, capacitive load values need to be accurate, in addition to accurate block and drive values.

The dependence of capacitive load on the width of a transistor were investigated by laying out a chain of two inverters with a sized cell at the output (Fig. 2.5). The layouts were extracted into SPICE decks and simulated. Delay values were taken from point A to O.

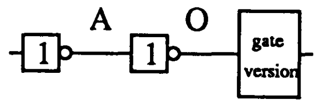


FIGURE 2.5 Capacitive load determination

The dependence of capacitive load on transistor width was found not to be directly proportional but linear. (Fig. 2.6) The constant offset is due to wiring capacitances, parasitic capacitances and other capacitive loads not associated with the gate, drain or source capacitances of a transistor. Capacitive input loads of each library cell are hence obtained by simulation of each sized gate with a chain of inverters as described above, to obtain accurate load values. This process, and the process of delay parameter extraction, are further described in section 3.4.

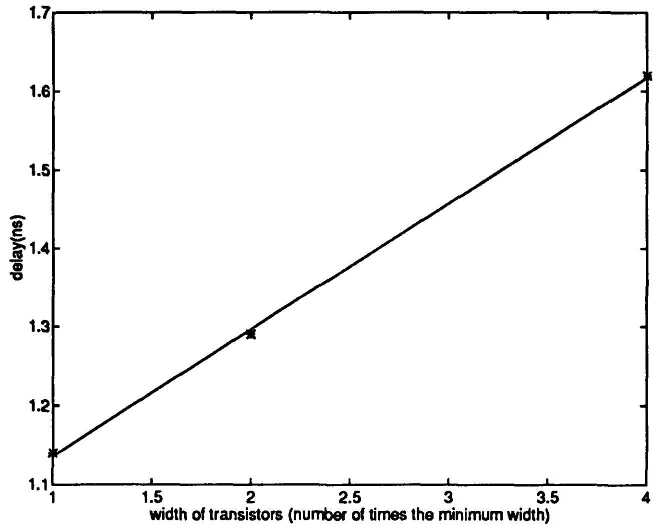


FIGURE 2.6 Delay of inverter versus width of transistors (p & n)

2.5 Circuit Delay

Circuit delay is estimated in the SIS delay package by calculating all gate arrival times (rise and fall calculated separately) using the pin delay model as described in section 2.2. The final circuit delay is obtained by the latest arriving primary output. Hence the delay of a circuit is defined by the delay of its largest delay path form input to output. This is the critical path. Decreasing the delay of the critical path will thus also decrease the delay of the circuit. Hence our approach to circuit delay reduction will be based on reducing the delay of the critical paths.

2.6 Summary

In this work, a detailed delay model that takes into account rise and fall times, varying pin delays and input transition times is used. Accurate capacitive load measurements are also done to ensure that delay calculations are accurate. Having a reliable delay model, we are now ready to proceed with establishing a good cell library and an optimization strategy to get the best power performance in a circuit for a given delay.

CHAPTER 3

The Augmented Standard Cell Library

3.1 Introduction

Standard cells are widely used in today's VLSI circuit design. The performance of such circuits are highly dependent on the performance of the standard cells used. In order to have low-power design with good speed performance, a standard cell library is needed where each gate is available in different sizes to cater to different output loads, with each version sized to give good delay response without high power dissipation, hence catering to different speed and power requirements.

This chapter describes the process whereby such an augmented standard cell library is developed. Cell versions of each logic function are chosen by careful experiment and analysis of the power and delay curves for each sized version. This involves laying out the circuit, sizing the transistors, simulating the cells, plotting the power and delay curves for each logic function, selecting the version and finally extracting the delay parameters (block, drive and capacitive load values) by simulation. The gate functions of the library developed are based on previously developed standard cell libraries.

3.2 Layouts

The layouts of standard cells require certain rules in geometry such as equal height so that the connection of cells can easily be done, without incurring wasted area and messy connecting wires. Since our aim is to size the transistors to achieve good delay and power performance, the layouts of the cells should be done so that the increase in rectangular area due to transistor sizing is minimal.

3.2.1 Standard Cell Geometry

Standard cells should be laid out in a way such that they can be easily abutted to neighboring cells. [34, 35]. Different layers should be continuous as far as possible when two cells are abutted. To achieve this, all cells are made the same height, with the diffusion, p-wells and metal lines at the same heights from the bottom of the cell.

CMOS two micron technology is used in this work. The height of certain cells are restricted by the diffusion to substrate contacts, making the minimum height achievable by these cells 35 lambda (1 lambda = 1 micron), or 76 lambda if the V_{dd} and G_{nd} rails are included. This height is maintained throughout the library. The V_{dd} and G_{nd} lines run horizontally and are 20 lambda wide to provide the needed conductivity in propagating the supply voltages throughout the circuit. The input nodes are placed on top and at the bottom of each cell for channel routing and the output nodes are connected by the second metal layer[1]. The layout cell of a 2-input nand gate is shown in Fig. 3.1.

1. Mentor Graphics lxcells were used as a reference in the layouts of standard cells. Automatic cell generators were not used in this work as such cell generators could not take into account our techniques for input ordering and sizing when deciding the geometry of the layout.

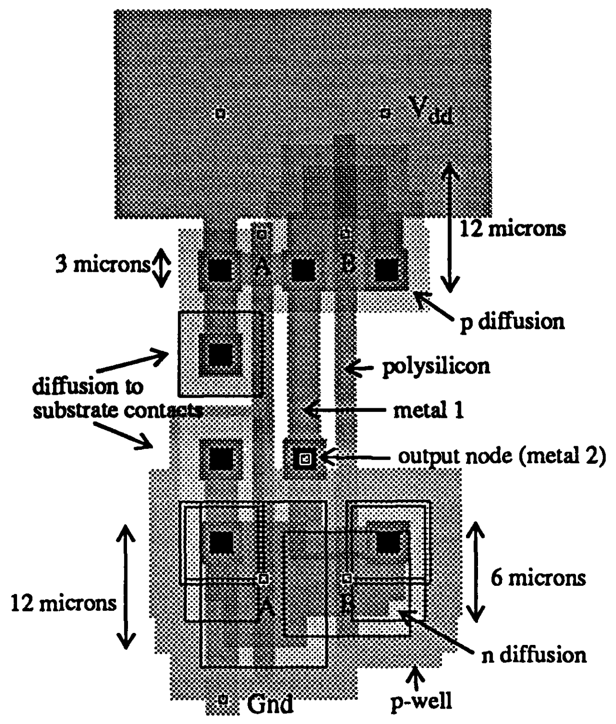


FIGURE 3.1 Layout of a two-input nand gate.

3.2.2 Area Minimization

To minimize the area, all cells are designed to have as few diffusion breaks as possible. Euler graphs and stick layout diagrams were used to find the minimally-sized layout for each cell. All cells in the library, except the xor and xnor gates, have no diffusion breaks.

All contacts for the cell are in the central part of the cell, and any sizing will expand upwards (for p transistors) or downwards (for n transistors) (See Fig. 3.1), hence giving the cell flexibility in sizing. Since metal layers are allowed to overlap with diffusion and polysilicon layers, the outward expansion of large transistors are done as an overlap under the bus lines (See Fig. 3.1). Since the bus lines are sized at 20 lamda, such increase in transistor sizes do not increase the rectangular area occupied by the cells.

To further minimize the area with sizing, one method is to place the p transistors that are sized bigger above the n transistors that are sized smaller and vice versa, giving the layout an inverted fit, thereby reducing unnecessary white spaces.

All library cells are sized at height 76 lamda (including the bus lines) with widths ranging from 15 lamda to 49 lamda. No increase in rectangular area resulted from sizing of transistors in any of the gate versions.

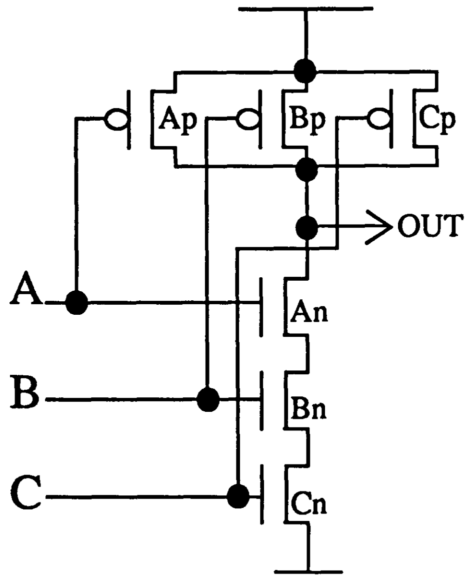
3.2.3 Sizing Methodology with Input Ordering

When transistors are sized larger in a gate, the drive for the next stage is increased, thereby reducing the output delay. However, the input load seen by the previous stage is increased, thereby slowing down the input transition and hence the switching of the current gate. This effect must be taken into account in sizing transistors.

Relative arrival times of inputs can affect the delay of a gate [37], as each pair of p and n transistors in a gate have different input transition to output transition times. In optimizing the delay of a gate, input ordering should be taken into account. The delay of a gate depends on the switching of the transistor associated with the input that will cause an output switching event. Worst case delay occurs when this input arrives latest. Delay can be reduced if the pin delay at a transistor that has the latest arriving input is made the fastest in the gate.

In selecting which transistor to size, the inputs are expected to be ordered such that the latest arriving input is placed at the transistor that has the shortest delay to output. This is, in most cases, the transistor closest to the output, due to body effect. [37]. Taking into account that the latest arriving input will be placed at this transistor, the transistors in the gate are sized such that the load on the latest arriving input is minimal, but the drive for the gate output is maximal.

Transistors in series with the one that has the latest input are sized first to increase the drive to the next stage, while not affecting the load on the latest arriving input. If necessary, the transistor with the latest input is then sized to further increase the drive. This process is illustrated in Fig. 3.2.



NAND3

Latest arriving input is placed at A.

Ap is enlarged to increase drive of pull-up.

Bn and Cn are sized larger than An to increase drive of pull-down but minimize load on previous stage of A.

FIGURE 3.2 Sizing of transistors in a three-input nand gate.

3.3 Power and Delay Curves

The layouts of different sized versions are extracted into SPICE decks for simulation. This is done to ensure that parasitic capacitances are taken into account. To have an accurate estimate of power versus delay, primary inputs applied to the cells take into account input switching probabilities and output switching probability. The inputs are assumed to have equal switching frequency (switching 50% of the time) and the number of output high's and low's are as expected by the gate function.

The power and delay values obtained from SPICE simulation for the various sized versions of a gate are plotted. As shown by Fig. 3.3 the shape of the curve shows that by sizing a cell, delay can be reduced from that of the minimum sized version by a significant amount without much increase in power.

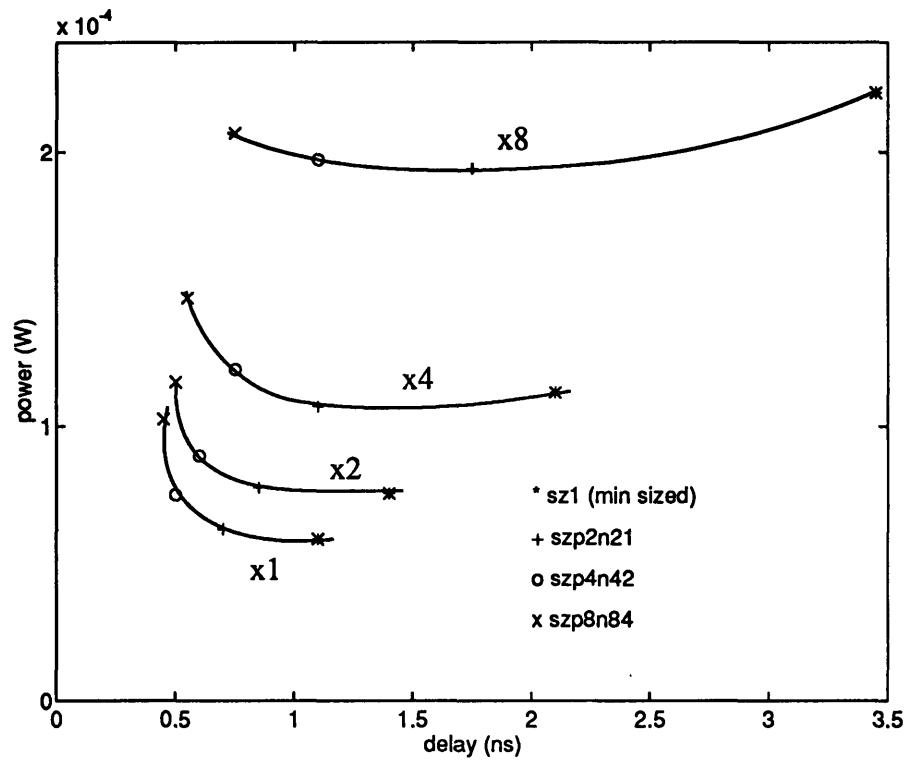


FIGURE 3.3 Power versus delay curve for a three-input nand gate with output loads of x1, x2 and x8.

Increased sizes mean increased capacitive loads for the inputs, which lead to increased power. However, the power dissipated by a larger sized gate can be less than that of the minimum sized gate when output load is large. As shown in Fig 3.3, when the output load is eight, the sized versions actually gave a lower power dissipation than the minimum sized one. This anomaly could be due to the fact that when output load is large, switching is slow, and the period when both n and p transistors are on is larger. This increase in current flow could lead to higher power dissipation. With large gates, however, switching time is reduced, and the short circuit current flows for a shorter period of time, thereby resulting in lower power. This reduction in power could offset the increase caused by increased capacitive load for the inputs, thereby resulting in lower power dissipated than that of the minimum-sized gate.

Power and delay points of several differently sized cells are plotted to obtain the general power and delay curve for the gate. Cell versions that yield good power and delay trade-offs such as those on the power and delay curve where the slope is changing the fastest are chosen. Three versions of each gate, each suitable for a different range of output load, were selected[1]. The advantage of having more versions is not apparent as computation time for the optimization algorithm would increase, and for most mapped circuits, the three versions can cater to the range of output load.

The augmented library constructed consist of 17 gate functions, each with three or four versions (see footnote), for a total of 54 cells.

Transistor widths were treated as continuous variables in previous work by Berkelaar [17] and in sizing algorithms such as TILOS [1]. In this case, we want to take into account sizing methods that also take into account relative delays of input arrivals, input ordering and area minimization. In addition, because of our use of cells as individual design elements, the sizes of the transistors are fixed before they are actually mapped onto a circuit, which means that the output load they will drive is not known at the time of sizing. Hence heuristic methods of sizing are required. Other optimization formalizations remain to be explored.

1. Four versions were found to be useful for the inverter, 2-input nand and 2-input nor gates.

3.4 Parameter Extraction

3.4.1 Delay Parameters

Once the cell versions are chosen, each cell is laid out with different numbers of inverters as its output load and also with different input loads (as described in section 2.2) to obtain the input drive and output drive parameters. All capacitive values of the cell library are specified in terms of the capacitance of an inverter, hence a output load of one means that the output load is equivalent to the capacitance of an inverter. Each cell is laid out with input loads of 1, 4, and 8 for each input pin (keeping output load constant at 1) and with output loads of 1, 4 and 8 (keeping input load at 1). The layouts are extracted into SPICE decks and simulated. Delay for the gate through each pin is measured by sensitizing the input to this pin. All other pin inputs are kept constant and non-deterministic of the output while this pin input is changed, thereby causing the output to switch. The pin delay is the time between the switching of this input to the switching of the output. (E.g. In Fig. 3.4, B is kept high while A switches, and the pin delay through A is the time that the input at A reaches 50% of its final value to the time the output at O reaches 50% of its final value.)

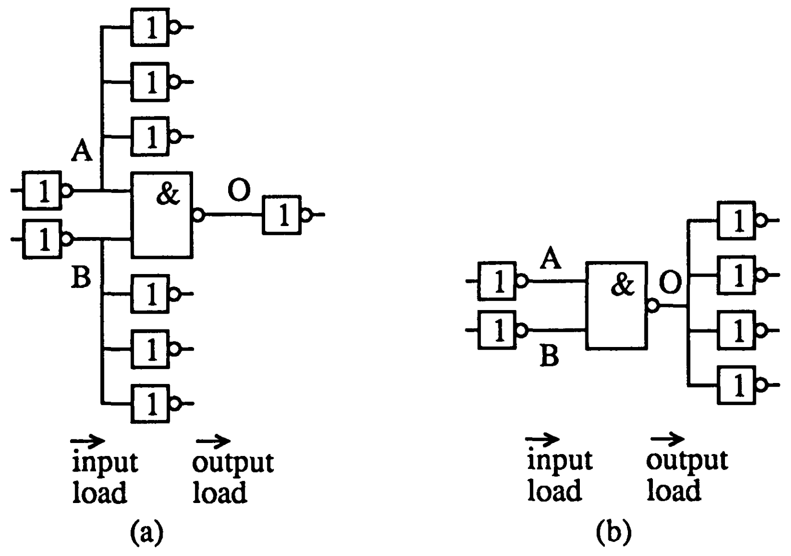


FIGURE 3.4 Delay parameters extraction. (a) Gate with input load of four and output load of one. (b) Gate with input load of one and output load of four.

Delay is plotted against input load as well as output load. The points are then curve fitted to find the linear relations, from which the block and drive values are derived (Fig. 3.5).

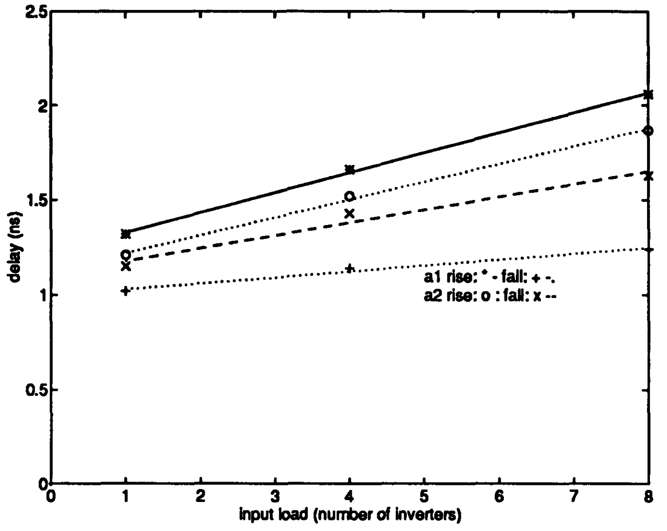


FIGURE 3.5 Input drive lines for both pin inputs of nand2.

The points for each pin input (rise and fall delay done separately) are line-fitted to an equation of the form (a + b x load). In chapter 2, it was established that delay through a gate can be modelled by the equation:

$$Gate\_delay = \frac{Input_{load}}{Input_{drive}} + block\_delay + \frac{Output_{load}}{Output_{drive}} \quad \text{(Eqn 3.1)}$$

For a plot of delay versus input load, the slope, b, is obviously the reciprocal of input drive. Since the cells were laid out with an output load of 1, the constant offset, a, is the sum of the block delay and the reciprocal of the output drive. Similarly, for the plots of delay versus output load, the slope is the reciprocal of the output drive while the constant offset is the sum of the block delay and the reciprocal of the input drive. Therefore, block delay is obtained by averaging the two values: constant offset of input graph - slope for output graph, and constant offset of output graph - slope of input graph. Table 3.1 shows the block and drive values of a minimum-sized 2-input nand gate.

gate name	pin	input pin load	rising output (ns)			falling output (ns)		
			1/ input drive	block delay	1/ output drive	1/ input drive	block delay	1/ output drive
nand2	a1	1.12	0.105	0.755	0.448	0.031	0.694	0.307
	a2	0.99	0.094	0.700	0.415	0.068	0.799	0.316

Table 3.1: Delay parameters for minimum-sized 2-input nand gate

3.4.2 Capacitive Load

To take into account all possible sources of capacitive load (including parasitic and wiring capacitances), the input load seen into each pin of a gate is obtained by simulation as well. A chain of two inverters, with the output of the second inverter connected to an input pin of a gate version, is laid out and extracted into a SPICE deck to be simulated (Fig. 3.6). The time between a transition at A to a transition at O is measured. This is the gate delay through Inv2.

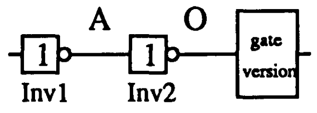


FIGURE 3.6 Capacitive load extraction

Considering equation 3.1, the input load of Inv2, which is the load driven by Inv1, is just an inverter (which is a load of 1). The delay parameters of an inverter have been obtained by the process described in the previous section. Hence the input drive, block delay and output drive values of an inverter can be plugged into equation 3.1. The only unknown left in the equation is output load, which is the input pin load of the gate version being examined, which can be obtained by simply solving the equation.

3.5 Summary

The augmented standard cell library is developed by a process of detailed layout, accurate simulation, and careful selection. All delay and capacitive parameters are also painstakingly obtained from layout and SPICE simulations to ensure accuracy. With a good cell library and accurate delay parameters, what remains in this low power design method is a good selection strategy to find the best sized versions to use in a circuit.

CHAPTER 4

Optimization Strategies

4.1 Introduction

In the last chapter, the development of an augmented standard cell library with cells designed to give good delay and power performance trade-offs was described. The effectiveness of this cell library is dependent on how the individual cells are used in a circuit. To utilize these cells effectively, we need good optimization strategies to select the best gate versions to use in circuits so as to achieve minimum power dissipation while satisfying delay constraints.

This chapter first describes the basics needed for our optimization strategies, namely critical path determination and the effects of changing a gate version. Then the basic selection algorithm is described, followed by a discussion of its limitations and the possible solutions.

4.2 Definitions

First, some definitions of terms that will be used throughout the chapter:

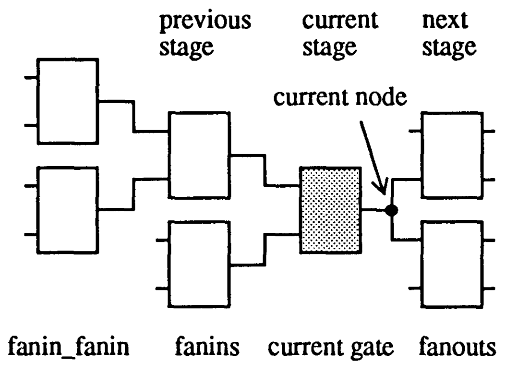


FIGURE 4.1 Definitions of terms in describing circuit topology

11. Current node: The output node whose gate version is being examined for optimization.

12. Current gate: The gate at the current node.

13. Current stage: All gates, including the current gate, which are at the same level from the circuit inputs.

14. Fanins: The gates whose outputs are connected to the input pins of the current gate.

15. Fanin_fanin: Fanins of the fanins of the current gate.

16. Fanouts: The gates which the output of the current gate is connected to.

17. Input pin load: The load at a pin of the current gate seen by a fanin.

18. Output load: The load driven by the current gate.

19. Pin delay time: The time from when the pin input switches to the time the output switches, in the case where the switching of this input determines the output.

20. Primary inputs and primary outputs: Inputs to the circuit and outputs of the circuit, respectively.

21. Arrival time of a node: The time from the switching of the primary inputs to the time the output of the gate at the node switches.

22. Required time of a node: The arrival time that the node has to satisfy in order for the primary outputs of a circuit to satisfy the delay requirement.

23. Slack time of a node: The difference between the required and arrival times at a node.

24. Required, arrival and slack times for a circuit: The delay requirement (rise and fall) for the circuit, the actual arrival times (rise and fall) of the latest arriving primary output of the circuit and the difference between these two, respectively.

25. Circuit arrival time: The larger of the rise and fall times of the arrival times of the circuit.

26. Critical slope and non-critical slope: Each node has two arrival times, one for rising, and the other for falling output transitions. For gates on the critical path, if the rising (falling) arrival time propagates to the circuit arrival time, then the critical slope of the node is rise (fall), and the non-critical slope is fall (rise).

4.3 Critical Path Determination

One of the aims of our optimization strategy is to satisfy a delay constraint, and in order to do that, we need to reduce the delay through the longest delay path of the circuit, which is the critical path.

The critical path is determined by first obtaining the latest arriving primary output of a circuit, and propagating from this node backwards toward the primary inputs.

Differences in rise and fall delays are taken into account as well as the phase and varying pin delays of gates.

The traversal is as follows. The latest arriving output node is determined. This is the last node on the critical path. The critical slope (rise or fall) of this node is determined by simply finding out which delay (rise or fall) is larger. Next, we scan through all fanins of this node, taking into account the phase or unateness of its pins. (e.g. An *inverting* phase for a pin means that when the input is falling, the output remains the same or rises, and when the input is rising, the output remains the same or falls).

If the critical slope of the node is *rise*, but the pin phase is *inverting*, the output arrival time due to this fanin is the sum of the fanin's falling arrival time and the rising pin delay of this pin. The critical slope of the fanin is *fall.* This calculation is done for all pins (and fanins). The critical fanin, or the fanin that causes the latest node output, is determined (see example below). (An alternate way is to do this calculation until the output arrival time due to a fanin is found to match the latest output of this node.) This latest fanin is on the critical path. Knowing which fanin caused the latest node output, we continue the traversal with this fanin, doing the same calculations as described above. This traversal is continued until we reach a primary input, thereby finding the first node on the critical path, and completing the determination of this path. Information on critical slope, critical path length, critical pin numbers (or the number of the pin that caused the latest output for the gate) are all retained for use in the optimization routines.

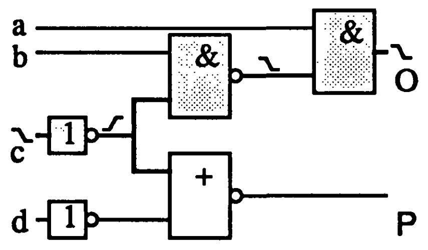


FIGURE 4.2 Critical path determination

This process is illustrated in Fig. 4.2. The latest arriving output is determined to be at O, where the fall delay is larger than the rise delay. The critical slope at O is *fall* and the *and* gate is on the critical path. Since an *and* gate has a non-inverting phase, the falling outputs of its fanins are considered. The fall arrival times of the two fanins are added to the respective fall pin delays through the *and* gate. The critical fanin, which caused the later arrival at O, is found to be the *nand* gate. The critical slope of this gate is *fall*, since the falling output propagates to the latest arriving output at node O. The *nand* gate has an inverting phase, so the rise time of its fanins are found, and added to the respective fall pin delays through the *nand* gate, to find the pin that caused the latest falling output at this gate. The highlighted *inverter* is found to be the critical fanin. Since the *inverter* has only one input and has an *inverting* phase, c is on the critical path and its critical slope is *fall*. Since c is a primary input, traversal is completed. The critical path is found to start from c, through the highlighted *inverter*, the *nand* gate and the *and* gate, which leads to the primary output O.

4.4 Impact of change of gate

4.4.1 Power Dissipation

When a gate version is changed to a differently sized version, the effect it has on power dissipation is due to two factors: the change in input load seen by its fanins, and differences in switching probabilities caused by reduced or increased glitching as a result of the new speed of the gate.

When a transistor is sized larger in a gate, the input driving this transistor sees a larger load than before, due to the dependence of gate capacitance on transistor width. The drive to the next stage (dependent on the resistive component of the transistors) is also increased, thereby leading to a faster transition at the output node of the gate (discussed in section 1.2.1). Power is proportional to load, and not dependent on the drive of a gate. When gate versions are changed, the change in load occurs at the inputs, not the output. Hence the change in power is dependent on the fanin probabilities and the input loads.

The change in power dissipation due to the change in input load can be calculated by multiplying the switching probability of a fanin node by the change in output load of the fanin due to the change in gate version. This change in load equals the new input pin load seen by the fanin minus the input pin load seen with the previous gate version. The sum of all such power changes of all fanins returns the change in power for the gate due to changes in load. Clock frequency and supply voltage remain the same, and need not be taken into account when comparing power dissipation among different versions (Eqn 1.1).

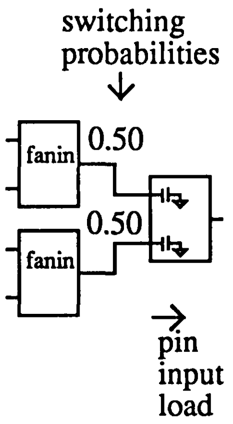


FIGURE 4.3 Changes in power dissipation due to changes in load.

To keep the computation simple and hence fast, the effects of a version change on the probability of glitching is not taken into account here. Comparison of final circuit power dissipation (calculated by recalculating all switching probabilities) to the original power dissipation of circuits shows that glitching effects caused by changes in gates are not substantial, hence this omission is justified.

4.4.2 Delay

When a gate version is changed, there are three main effects on delay. The first is a change in the pin delay through the gate, due to changes in delay parameters, which in turn is due to the different sizing of transistors in the versions. The second is a change in arrival times of the fanins of the gate, since each sees a different input pin load. The third is propagated delay effects to the rest of the circuit due to changes in arrival times of the current gate and its fanins.

To calculate the new arrival time of a node after a change in gate version, we must take the first two effects into account. First, we need to calculate the change in arrival times of the fanins. To do so, we need to calculate the pin-delays through the fanins, which have changed due to the change in input pin load. For each fanin, the pin delay through each input is calculated, and added to the arrival of the previous stage (fanin_fanin) (Fig 4.3). The phase of each pin, as discussed in section 4.3, must be taken into account to ensure correct delay calculations. The latest arrival time for each fanin is then determined. These fanin arrivals are added to the corresponding pin delays of the current gate, taking into account phase, to determine the latest arrival times for the output (both rise and fall).

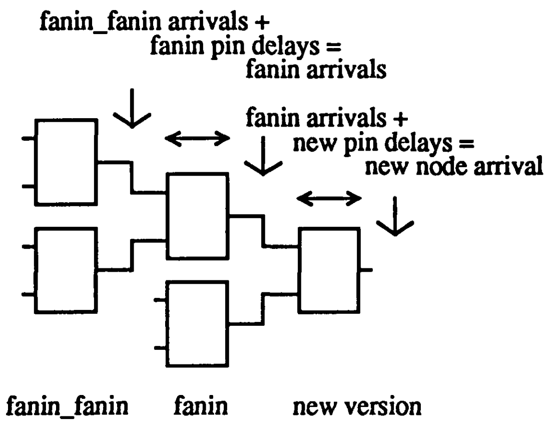


FIGURE 4.4 Impact of change of gate version on delay

When a gate version is changed, the arrival times of its fanin might increase or decrease due to the change in input pin load. If this fanin has multiple fanouts, the increase could be propagated to other paths. If the circuit has many long paths that have approximately the same delay, this increase could cause a different path to be critical and increase the delay of the circuit. However, if the critical path remains dominant, the decrease in the arrival times of the gate output would cause a decrease in the final arrival times of the circuit.

4.5 Basic Optimization Strategy

Since the purpose of the algorithm is to minimize power, the circuit is first mapped with minimum-sized gates[1]. Power is proportional to the load capacitance of the gate, so by using minimum-sized gates we begin with minimal power. Delay constraints are then satisfied by using bigger gate versions as necessary to reduce the delay.

1. Any mapping algorithm available to the user may be used. This optimization strategy starts with a circuit that has been mapped with minimum-sized gates. This particular mapping of gate functions is not changed in the optimization routine. Only gate versions (and not their logical functions) are changed.

A change of gate could have effects on many other gates, since the change in delay could propagate throughout the circuit. To update the delay of the whole circuit at every change of gate version would be computationally intensive. To avoid such continuous updates while keeping accurate knowledge of delay values as gate versions are changed, the critical path is traversed sequentially and delay values on the path updated as versions are changed. This ensures a fairly accurate update of critical delay values needed in the optimization process. The process is described in detail below.

After mapping the circuit with minimum-sized gates, a power estimation routine is run on the circuit to determine the switching probabilities of each node. Static timing analysis is also performed to determine the required, arrival and slack times at each node.

The critical path is then obtained. If the delay constraint is not satisfied, the algorithm traverses this path from input to output. At each node, delay values which may have changed due to previous changes in gate versions along the path are updated. Then each sized-gate version is tested out by calculating the new delay values, (taking into account change in arrival times of inputs due to changes in input pin loads, as well as changes in output arrival times due to new block and drive values) (See section 4.3.2) and the change in power from the original gate using the switching probabilities multiplied by the change in input load seen at the pins (Refer to section 4.3.1).

If the critical slope delay (rise or fall) of the node is reduced as a result of the change in versions, and the non-critical slope delay is not increased, the ratio of reduction in delay to increase in power, R_{dp}, is calculated. The version that gives the best ratio is selected.

To test the effectiveness of using this ratio in the determination of the best gate version, an option that selects gates based on their ability to reduce delay is implemented. The gate that gives the best reduction in delay is chosen, regardless of the increase in power. This is the *use_delay_only* option.

Once the gate version is selected, all arrival times at the node and at its fanins are updated. The slack of the critical path is also continually updated as each gate version is changed. The new slack value is calculated by subtracting, from the old slack, the change in delay obtained by changing the gate version. Both rise and fall slack times are updated. The smaller of the two slacks is taken as the new slack. Traversal of the critical path stops when this slack becomes positive.

If traversal is allowed to continue for the whole path regardless of the slack, the required delay may have been achieved somewhere along the path, but since traversal continues despite that, more gates are changed, resulting in unnecessary decrease in delay and increase in power. Furthermore, if the critical path is long, these unnecessary computations make the optimization routine more intensive than it need be.

Note that this method of updating the slack may not return the actual slack of the circuit. (Refer to Fig. 4.5) A and C are on the critical path. If A is sized larger, the critical arrival time of its output is decreased. Say this decrease in delay was *d*, then the slack will be updated with this value. However, this change in delay may not propagate to the output. If the largest arrival time through C is now via B, the arrival time of C is still less than before A was changed, but by an amount less than *d*. Then circuit slack is actually less than the estimated value.

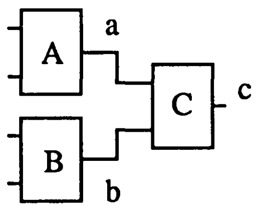


FIGURE 4.5 Slack determination

When a traversal is stopped due to a positive slack, a static timing analysis is redone on the whole circuit. From this analysis, the actual slack of the circuit is obtained, and if the slack is still negative, optimization is resumed.

This traversal on critical paths is done until all paths in the circuit satisfy the delay constraints or if all gate versions have been examined.

4.6 Problems and Solutions

In analyzing circuits optimized by the strategy described above, several interesting problems arising from circuit topology were found.

A problem that might arise with such a traversal is that, referring to Fig. 4.6, if gate A is first changed, followed by C, gate A was changed and selected on the basis that C is minimum-sized. When C is also changed, the input pin load seen by A has changed, and the version selected for A may no longer be the best version.

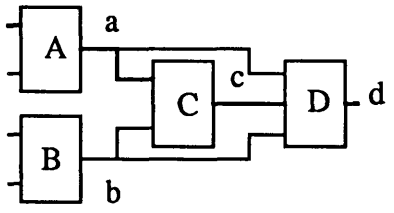


FIGURE 4.6 Propagation of effects of change of gate

Furthermore, the fanin of a current gate (D) may come from gates which feed into earlier gates on the critical path too (Gates A and B in fig.4.6) The increase in delay of these fanins due to the change in input pin load of the current node was not taken into account in analyzing the previous gate affected (gate C) and hence the arrival time of C previously calculated may not be accurate, which in turn means that we may not have the correct information when analyzing D.

However, in the next run, if the critical path is still the same path, then A could be changed again to reflect the right choice, and so could C. The number of changes is limited by the number of versions of a gate and the fact that a gate will not be changed unless the new version is faster, hence cyclical optimization will not occur, and in these special cases the best solution can still be achieved after several traversals.

Another problem is shifts in critical paths. A and C are on the critical path. After analyzing A and changing the version, the arrival time of A is reduced. The latest output at C is now through pin b, rather than a. The critical path has changed. B could be made faster to make C even faster.

To solve this problem, a new option, *check_critical_path* is employed to stop a traversal on a critical path once the path has changed, and resume optimization on the new critical path. This option is described in the next chapter.

4.7 Limitations

The critical path is traversed sequentially so that delay values can be accurately updated on the critical path. Hence gates are optimized in the order of their occurrence in the critical path from input to output. However, certain gates are better candidates than others for optimization, mainly those nodes with large loads but small switching probabilities, as these could give a large reduction in delay, given the large load, and have low power dissipation increase, due to the small probabilities. These gates should be optimized first to obtain better power and delay trade-offs.

Furthermore, the condition that a gate version is accepted only if the critical slope delay is decreased while the non-critical slope delay remains the same or decreases might be too restrictive. Certain gate versions might reduce critical slope delay significantly while increasing non-critical slope delay slightly. To give the circuit flexibility, the non-critical slope could be allowed to increase.

These two limitations are resolved in the options *threshold* and *maxtime* which are described in chapter 5.

4.8 Summary

Each node in a circuit is connected to others and effects of a change in version for a gate can be felt elsewhere in the circuit. To select the best version of each gate to use in a circuit without continuous circuit delay updates, only the critical path is traversed and updated. The selection of cells are based on their ability to give reductions in delay with little increase in power. While this optimization strategy provides a simple method of cell selection, it is nonetheless limited by the sequential traversal and by the possible effects on gates off the critical path. These limitations are explored in the next chapter.

CHAPTER 5

Enhancements to the Basic Strategy

5.1 Introduction

As discussed in the last chapter, while the basic optimization strategy makes use of all available information at a given node to make its decisions, it is nonetheless limited by the sequential scan of gates on the path, by the limited information of the circuit at each node, and by the rigidity in selection of gates in return for a guaranteed reduction in circuit delay.

This chapter describes three options that are implemented to overcome these shortcomings of the basic routine and hence enhance the capabilities of our selection process. These are the *threshold*, *check_critical_path* and *maxtime* options.

5.2 Threshold

Nodes with low input switching probabilities but large output loads have good potential for delay reduction, and given the low switching probability, increased transistor sizes may not increase power by much. To reduce computation complexity, the circuit is traversed sequentially along the critical path. Sequential traversal means that gates are optimized according to their order on the critical path. However, this does not make use of the fact that certain gates are better candidates than others for optimization, namely those with large ratio of load to switching probability, R_{lp}.

In the *threshold* option, the ratio of load to switching probability of a node, R_{lp}, is used to determine the priority with which the gate will be optimized. A threshold value of this ratio is used to determine which gates to change first. The threshold is determined by an estimated percentage of gates on the critical path that will need to be changed in order to achieve the delay constraint. When the critical path is traversed from input to output, only nodes with ratios higher than this threshold will be analyzed and changed. Other nodes will just have their delay values updated. If the delay constraint is still not satisfied after completing the traversal of the critical path, the threshold is lowered to allow more gates to be changed. If, however, the critical path has changed, then this threshold value is recalculated as before.

The threshold value is determined as follows:

First, the load to switching probability, R_{lp}, of all nodes on the critical path is determined and sorted in ascending order into an array. The load is the output load of a node, and the switching probability is the sum of all switching probabilities of its fanin nodes. As discussed in section 3.4.1, the change in power dissipation due to a change in gate version is mainly due to the change in input pin loads of the gate. Hence this change in power is obtained by summing, for all fanins, the product of the switching probability of each fanin node and the change in input pin load looking into the current gate. However, when determining the threshold, we have no knowledge of which gates and which transistors in the gates will be sized, so the sum of all input probabilities is used. By summing the switching probabilities, we are assuming that the change in load seen by each transistor in a gate is the same, which means that the change in power is the sum of the probabilities multiplied by some constant value, which is the change in load. When comparing gates, this constant value can be dropped since it has no effect on the relative magnitude of two gates. This assumes that all transistors will be sized equally and by the same amount for all cells in the library. While this is not the case, with the limited information on the change of load available when the threshold is determined, (which is before the traversal of the critical path begins), this method provides a simple and general gauge of the change in power dissipation in changing a gate version. At first glance it might seem as if the output switching probability should be used, but the load change in changing a gate version is not at the output but at the inputs. Another argument would be to use the switching probability of the critical fanin node, but the transistor that is sized may not be the one on the critical path (Refer to section 3.2.3). Furthermore, using the sum of all fanin switching probabilities reflects the fact that a gate with more inputs (e.g. a four-input nand compared to a two-input nand) would result in greater power dissipation as more transistors need to be sized (namely those in series in the pull-down) to reduce delay. Using the sum of probabilities of all fanins ensures that all input nodes are taken into account.

Next, the percentage of gates that need to be changed is determined. This is done by dividing the magnitude of the slack of the circuit (difference between the required arrival time and the actual arrival time) by the expected reduction in delay by changing a cell version. This gives us an estimate of the number of gates that need to be changed. The percentage of gates on the critical path that need to be changed is then obtained from dividing this number by the number of gates on the critical path.

Since the R_{lp} ratios of each node are already sorted into an indexed array in ascending order, the R_{lp} ratio that is the required percentage from the end of the array is chosen as the threshold. Only gates with R_{lp} ratios above this value will be allowed to be changed. With this calculation, the percentage of gates that satisfy this criterion is the parentage of gates that need to be changed.

The threshold value enables us to just look at the needed number of gates on the critical path, instead of traversing the whole path. Gates with the best potential for power and delay trade-offs are also picked out to be optimized first, thereby giving us better results compared to a sequential traversal from input to output.

5.3 Check_Critical_Path

When the gate version at a node is changed, the delay of the next stage could be caused by a different input than before. Referring to Fig. 5.1, the critical path is originally from node a to c. When a faster cell version of gate A is used, the arrival time at a is reduced, and the critical path may have shifted from node a to b. This means that the delay at output c could be further reduced if the arrival time at b is made smaller. This option checks if such a shift in critical path has occurred. It does so by simply checking if the current node causes the latest output at the next stage on the critical path. If so, critical path has not been changed by the change in gate version. Else, critical path has changed, and static timing analysis is redone on the whole circuit to update delay values and find the new critical path. Optimization is then continued by traversing the new critical path.

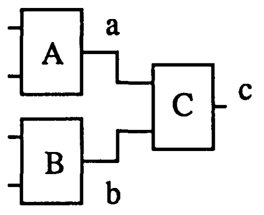


FIGURE 5.1 Change in critical path

This option detects shifts in the critical path so that all possible avenues for delay reduction are found and used. When this option is used, static timing analysis on all nodes is done more often, thereby updating all delay information more frequently, though at the expense of computation time. If the change in gate has reduced the delay of the critical path such that another path elsewhere in the circuit is now critical, this change would be also be detected whenever timing analysis is done on all nodes and the critical path determined.

5.4 Maxtime

To ensure a reduction in delay when a gate version is changed, a version is accepted only if the critical slope delay (rise or fall) decreases while the other slope delay remains the same or decreases. While this guarantees a reduction in final delay, there could be gate versions where the critical slope delay could be reduced further, but the delay of the non-critical slope would increase. Since the non-critical slope delay is invariably equal or less than the critical slope delay, the positive slack in the non-critical slope may be able to absorb the increases in non-critical slope delay caused by these gates and still result in smaller circuit delay.

To increase the flexibility in choosing gate versions, the non-critical slope could be allowed to increase. However, this might lead to increased circuit delay if the slack of the non-critical slope is not large enough to absorb the increases in delay. To compromise between the two effects, a gate is accepted if the critical slope delay is reduced, and the maximum of rise and fall critical path delay is reduced. This means that if the non-critical slope delay is larger than the critical slope delay at a node, it will not be allowed to increase. (Note that the critical path delay is the sum of all critical slope delays through the critical path, and at certain nodes, the critical slope delay could be larger than the non-critical slope delay, and smaller at other nodes, but the sum of all critical slope delays is larger than the sum of all non-critical slope delays on the critical path.)

While this method does not guarantee a reduction in circuit delay, it increases flexibility in the selection of gates for better results, while preventing large increases in non-critical path delay to reduce the possibility of increased circuit delay.

5.5 Summary

The three options described in this chapter, *threshold*, *check_critical_path* and *maxtime*, offer enhancements to the basic optimization routine by making use of node switching probabilities, critical path updates and non-critical slope slacks respectively. These options offer a larger number of tools in selecting the best gate versions for a circuit. The effectiveness of each of these options will be discussed and analyzed in chapter 7.

CHAPTER 6

Input Ordering

6.1 Introduction

Inputs to multiple input gates usually have different arrival times. Pin delays through a gate also differ among pins. Since delay is dependent on both input arrival times and pin delays through a gate, for input pins that have the same logical function, the inputs can be ordered such that the resultant arrival time of the gate output is minimal. Since the gate version is not changed, the delay of the gate can be reduced without any increase in power dissipation. This makes input ordering an attractive option.

6.2 Dual Process

In this work, input ordering is implemented in two ways. Firstly, it is done on the cell level. To increase the effectiveness of input ordering, the transistors of each cell are sized so that the fastest pin in the gate is made even faster, sometimes at the expense of the other pins. This is because circuits often have a dominant critical path, and to decrease the delay through this path, we make the pin delays of gates on this path as small as possible. The process whereby the transistors are sized assuming inputs are ordered is described in section 3.2.3. This process binds the optimal arrangement of inputs when the cells are laid out, and increases the potential gains in delay performance by input ordering at the circuit level.

In order to capitalize on such transistor sizing, the inputs of the gates should be ordered correspondingly. Input ordering on the circuit level takes into account the function of each pin, input arrival times and input load changes in reordering inputs, among other things. This dual process of input ordering binds the optimal order of inputs at two very different time frames. The first is done when the cells are laid out, and the second during the run time of the optimization routine, when the circuit is already mapped and the actual arrival times of the inputs are known. The dual process on both cell and circuit levels maximizes the effect of input ordering.

6.3 Input Ordering at a Gate

First, the inputs of the gate that are logically the same (e.g. the inputs to nand and nor gates), and hence can be reshuffled, are identified.

At first glance, the next step is just to sort these inputs such that the latest arriving one is matched with the fastest pin of the gate. If so, input ordering is just a matter of sorting the inputs according to their arrival times, and placing the latest one at the fastest pin and the second latest on the second fastest pin and so on, with the fastest input at the slowest pin. The delay of the critical fanin is minimized, and the gaps between the arrival times due to other fanins are lessened too.

However, this simplistic method overlooks the changes in drive and load values for each pin and fanin, when the inputs are reordered. Several effects are felt when inputs are changed. Firstly, for a fanin, when the pin it is attached to is changed, the load it sees into the pin has changed too, causing fanin arrival times to change. Secondly, pin delay through a gate is dependent on the load that the previous stage drives (input load in Eqn 2.2), since this load determines the input transition time. When the fanin connected to a pin is changed, so is the load of the pin (which is the output load of the fanin). This causes the pin delay through a gate to vary according to the fanin it is connected to.

To cater to such varying fanin arrival times and pin delays, all possible permutations of input ordering are tried out and the arrival times of the output calculated for each. For n inputs, there are n! possible permutations, but since the largest number of permutable pins for a gate in a library is four (in the case of 4-input nand and 4-input nor gates), the maximum number of permutations is 4!=24. Hence the complexity of testing out each permutation is not excessive. The permutation of gate inputs that gives the fastest arrival times at a gate is used. In the case where two solutions give the same optimal result, the one where the arrival times of the next latest pin output (caused by a different fanin) is smaller is chosen, so that the general circuit delay is reduced, as well as the worst case delay.

6.4 Input Ordering Options

6.4.1 Input Ordering at All Nodes

Input ordering can be done at all nodes before the other optimization routines are run to reduce the general delay of all nodes in the paths. Input ordering of all nodes has the advantage of firstly, reducing the circuit delay (or critical path delay), and secondly, reducing the general delays of non critical paths, thereby giving more space for possible implementations of path relaxation for further power reduction. (A sized gate on a non-critical path can be sized smaller to reduce power. The increase in delay through this gate can be absorbed by the positive slack at the node.)

After input ordering is done on all nodes, the critical path is found, and input ordering is again done on the critical path, this time taking into account the critical slope. When input ordering is done on all nodes, the permutation of inputs that gave the minimum delay (in terms of the larger of rise and fall delay) is chosen. In ordering the gates on the critical path, the pin that gives the minimum critical slope delay (rise or fall) is chosen. This should further reduce the delay on the critical path.

6.4.2 Input Ordering at each stage

As gate versions are changed along the critical path, delay values are changed. When a gate is changed to a faster version, the output delay is decreased, hence for the next stage, input ordering can be done again to achieve the minimal delay for this new set of fanins. This option allows input ordering to be carried out at each stage as gate versions are changed. The slack of the circuit is also updated as input ordering is done by subtracting, from the old slack, the change in critical delay caused by the reordering, taking into account both rise and fall times.

In this option, the permutation that gives the fastest critical slope arrival time, on the condition that the non-critical slope delay is not increased, is chosen. This is to ensure that the reordering of inputs will decrease the final circuit delay.

6.5 Summary

The use of input ordering in sizing the individual cells and in the optimization algorithm enables us to make full use of the effectiveness of reordering inputs. Input ordering on the circuit level takes into account load and drive value changes in both pin and fanin delays. The implementation of two options for input ordering offers a variety of choices in optimizing the circuit. The results from utilizing this option are shown in the next chapter.

CHAPTER 7

Implementation and Results

7.1 Introduction

The basic optimization strategy and all options, as well as the augmented standard cell library, were implemented and tested on several test circuits. This chapter first describes the implementation of the design process, and then presents the results of the tests, followed by an analysis and discussion of the efficiency and effectiveness of the cell library, the basic optimization strategy, each of the options, as well as input ordering.

7.2 Implementation

The low-power design process developed in this thesis is implemented using C language and utilizes various packages in the Sequential Interactive Synthesis (SIS) system. SIS is an interactive tool for synthesis and optimization of combinational and sequential circuits developed at the University of California at Berkeley [32]. Power estimation routines developed in [13] and [39] are also used. Such routines have been implemented in SIS.

The mapping of circuits were done using the mapping package in SIS. Network and node packages in SIS were used to define the circuit. Power estimation routines were used to obtain the switching probabilities of nodes. The routines for static timing analysis (the delay package) in SIS were modified to take into account input transition times.

The augmented standard cell library was designed using Mentor Graphics GDT layout tool. Simulation of cells were done with HSPICE. The extracted parameters of the augmented library (PAL) are written in *pal.genlib* in the *genlib* format described in [32]. This format is accepted by the various routines in SIS.

The optimization strategies developed in this thesis were implemented in the optimization program for power and delay using sizing (OPPADUS). This program was written in C. OPPADUS optimizes circuits to satisfy a delay constraint with minimal power. The program includes routines of critical path determination, input ordering, threshold determination, check_critical_path; routines for calculating the effects of a change of gate version on power and delay; a brute force routine that finds all possible combinations of versions that will satisfy the delay constraint; the optimize_delay routine which implements the basic optimization strategy with maxtime, and calls the various routines for other options; and the main routine that puts all these routines together. The total length of the program is 2557 lines.

OPPADUS is implemented in SIS, and can be called interactively with the command *opy*. The various options implemented in OPPADUS can be called using flags: "-a" for *use delay only*, "-t" for *threshold*, "-m" for *maxtime*, "-p" for *input ordering at all nodes*, "-e" for *input ordering at each stage*, "-c" for *check critical path* and "-b" for the brute force routine. The brute force routine is a recursive program that tries out all combinations of gate versions possible in a circuit to find the optimal solution (least power) under the delay constraint. It is obviously very computationally intensive. Small circuits like ex1 took hours to complete. This further supports the need for a heuristic routine.

OPPADUS outputs the final arrival time of the circuit, the average power dissipation and a list of the nodes and the gate versions used.

OPPADUS was compiled and ran on a DEC Alpha machine (DEC AXP 3000/500, running at 150MHz, with 256kB of RAM).

7.3 Test Set

A test set of five circuits were used to test the optimization program. These circuits include a simple circuit with just 11 gates (ex1), another combinational circuit consisting of 101 gates (ex2), a four-bit adder, a multiplexer and x4, a large circuit with 503 nodes. These circuits were chosen to reflect different circuit configurations. For example, the four-bit adder has just one dominant critical path, while the multiplexer and x4 circuits have a number of dominant paths.

The circuits were optimized under a delay constraint using the basic algorithm and one or more options. The delay constraints are chosen to be in the mid-range between the initial delay using minimum-sized gates and the minimum delay achievable using the augmented library. This gives the algorithm sufficient flexibility for optimization. If the constraints are too close to either of the end values mentioned above, then the number of possible combinations of the circuits that have delay values close to the constraint becomes limited and hence the effectiveness of the routines will not be apparent. The sensitivity of the constraint value on the results is high when the constraint is close to the two extreme values. As the constraint value move towards the center of the two extremes, sensitivity is lessened, since the solution set enlarges. The effectiveness of the algorithm in selecting the best solution from this set can then be better tested.

7.4 Results

The basic algorithm, combined with one or more of the options, was tested on the six circuits. While each option can be used independently, a combination of options often yields better results. For example, *threshold* and *input ordering* each proved to be effective options, but when used together, the final power and delay trade-offs were even more significant. *Input ordering at each stage* is a good supplement to *input ordering for all nodes*, and hence is tested with the latter. The results of running the algorithm and its options on the five circuits are shown in Tables 7.1 to 7.5.

Table 7.1 shows the number of gates in each circuit and the delay and power dissipation of the circuits when initially mapped with minimum-sized versions. Table 7.2 shows the results from running the basic algorithm, first using only the reduction in delay as a guideline for the selection of versions (the *use delay only* option, discussed in section 4.5), and secondly utilizing the switching probabilities by using R_{dp}, the ratio of decrease in delay to rise in power.

Table 7.3 shows the results from running the algorithm with the *threshold* option, and then with *maxtime* and *threshold.* Table 7.4 shows the results of running *input ordering at all nodes* in addition to *threshold,* and again using *maxtime* as well. Finally Table 7.5 shows the results of running *input ordering at each stage* and at all nodes, with *threshold,* and then running *check critical path* in addition to *threshold* and *input ordering at all nodes.*

The best results obtained from the various combinations of options are summarized in Table 7.6 under the section "Overall Effectiveness" (section 7.6).

circuit	number of gates	initial mapping with minimum-sized gates	
		delay (ns)	power (uW)
ex1	11	9.13	25.2
ex2	101	21.06	218.5
adder4	55	17.98	155.0
mux	93	22.40	279.4
x4	503	18.19	1469.5

Table 7.1: Results for minimum-sized mapping

circuit	required time (ns)	a) Basic algorithm using delay only			b) basic algorithm using switching probabilities		
		delay (ns)	power (uW)	cpu time (s)	delay (ns)	power (uW)	cpu time (s)
ex1	7.8	7.71	25.9	< 0.1	7.71	25.9	< 0.1
ex2	16.8	16.58	237.5	0.1	16.62	232.0	0.1
adder4	14.0	13.92	185.8	0.1	14.11	180.1	0.2
mux	15.7	15.66	390.2	0.3	15.91	357.9	0.6
x4	14.6	14.67	1597.8	6.4	14.59	1548.6	5.9

Table 7.2: Results for basic algorithm

circuit	required time (ns)	a) threshold			b) threshold, maxtime		
		delay (ns)	power (uW)	cpu time (s)	delay (ns)	power (uW)	cpu time (s)
ex1	7.8	7.71	25.9	< 0.1	7.71	25.9	< 0.1
ex2	16.8	16.74	225.5	0.2	16.74	225.5	0.2
adder4	14.0	14.11	180.1	0.2	13.98	180.8	0.1
mux	15.7	15.91	357.7	0.9	15.68	342.6	0.5
x4	14.6	14.58	1545.8	9.1	14.59	1528.6	7.0

Table 7.3: Results for using threshold with and without maxtime

circuit	required time (ns)	a) threshold, input ordering at all nodes			b) threshold, maxtime, input ordering at all nodes		
		delay (ns)	power (uW)	cpu time (s)	delay (ns)	power (uW)	cpu time (s)
ex1	7.8	7.57	25.9	< 0.1	7.57	25.9	< 0.1
ex2	16.8	16.63	221.4	0.2	16.63	221.4	0.2
adder4	14.0	13.81	181.9	0.2	13.98	170.3	0.2
mux	15.7	15.68	343.2	1.2	15.62	334.3	0.8
x4	14.6	14.60	1515.5	5.9	14.60	1506.8	4.9

Table 7.4: Results for using input ordering and threshold with and without maxtime

circuit	required time (ns)	a) threshold, input ordering at all nodes and at each stage			b) threshold, input ordering at all nodes and at each stage, check critical path		
		delay (ns)	power (uW)	cpu time (s)	delay (ns)	power (uW)	cpu time (s)
ex1	7.8	7.47	25.9	< 0.1	7.47	25.9	< 0.1
ex2	16.8	16.70	222.5	0.2	15.90	230.6	0.5
adder4	14.0	13.82	172.9	0.3	14.10	167.1	0.3
mux	15.7	15.70	304.8	0.6	15.53	325.0	1.2
x4	14.6	14.59	1538.3	8.2	14.92	1520.3	6.2

Table 7.5: Results for using threshold, input ordering at all nodes and at each stage with and without check critical path.

7.5 Effectiveness of the optimization tools

7.5.1 The Augmented Standard Cell Library

A circuit is first mapped with minimum sized cells. Cells are then replaced by larger versions as needed by our optimization program to satisfy a delay constraint. The main purpose of the augmented cell library is to provide cells that give good delay and power trade-offs so that the increase in power from the minimum-sized mapping will not be large. From the results in Tables 1 to 5 (and summarized in Table 7.6 in section 7.6), it can be seen that delay has been reduced by 22% to 43% with only 1% to 8% increase in power using the basic algorithm with various options (see Fig. 7.1). This indicates that the cells in the augmented cell library are very effective in providing good power and delay trade-offs.

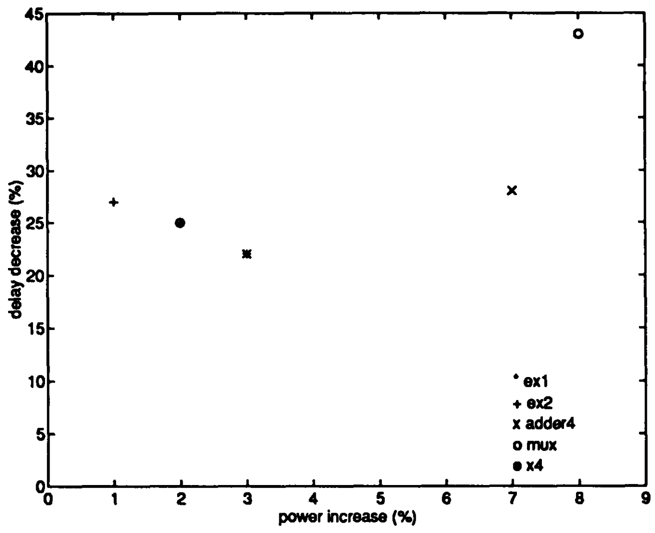


FIGURE 7.1 Plot of power and delay trade-offs achieved

7.5.2 The Basic Algorithm

The basic algorithm makes use of the ratio of change in delay to change in power, R_{dp}, to determine the best gate version to use for delay reduction with minimal power increase. The change in power is estimated with the switching probability estimates. The version that gives the best R_{dp} ratio is chosen. The effectiveness of utilizing this criterion is tested by comparing this method to that where the gate version that gives the best reduction in delay, regardless of increase in power, is chosen. From Tables 7.2a and 7.2b, power dissipation reduces by up to 9% by using power estimation with switching probabilities. Hence this method is effective in minimizing power increase.

7.5.3 Threshold

Since the basic algorithm traverses the critical path sequentially, gates were optimized according to their order on the critical path. The *threshold* routine avoids this sequential traversal to select gate versions according to their potential in giving good delay and power trade-offs. The *threshold* routine worked well in giving better power values when compared to just using the basic algorithm.

In most circuits, power dissipation was reduced by this option, though the effects of using this option are not as significant in cases where the required arrival time of the circuit is a lot less than the arrival time of the circuit with minimum-sized gates. This is because in calculating the initial threshold, the percentage of gates on the critical path that need to be changed to satisfy the delay constraint is large, perhaps 100%, due to the large negative circuit slack. If this percentage is 100%, then using the *threshold* option is redundant, as all gates are allowed to change. The effect of using a threshold is felt only as the magnitude of the circuit slack reduces with iterations of the algorithm. However, if part of the critical path had been traversed before, (i.e. it overlaps with a previous critical path), many gates on the current critical path have already been changed to the most preferred version, and less gates can be changed to reduce delay further, thereby reducing the effectiveness of using a threshold value.

In general, the *threshold* routine worked well in reducing power, though the degree of effectiveness varied between circuits.

7.5.4 Maxtime

The *maxtime* option allows more flexibility in the selection of versions by allowing versions that decrease the critical slope delay but increase the non-critical slope delay to be used. This increased flexibility enabled the power dissipation in the adder4, mux and x4 circuits to be less than if this option was not used (Tables 7.3a and 7.3b and Tables 7.4a and 7.4b). Reductions of up to 7% were achieved. For the adder circuit

(Tables 7.3a and 7.3b without *input ordering*), power dissipation increased slightly. This could be due to two factors. Firstly, the delay constraint was not satisfied when just using the *threshold* option. In using the *maxtime* option over the *threshold* option, the delay constraint was satisfied, and in order for this reduction in delay to occur, certain gates could have been sized bigger, others smaller, but the overall effect was an increase in power, which is expected given the reduction in delay. The second possible reason for this slight increase may be that the final circuit delay might have been increased in an iteration, and hence to bring down the circuit delay, more gates could have been changed, or larger versions used, leading to the increase in power.

Overall, *maxtime* proved to be effective in reducing power dissipation. (Referring to Table 7.3 and Table 7.4, power dissipation was less when *maxtime* is used for most circuits except ex1 and ex2. The only increase occurred with adder4 without *input ordering*, but this increase was less than 0.4%.)

7.5.5 Input Ordering at all Nodes

In input ordering, pin inputs of a gate are ordered such that the latest gate output arrival time is reduced as far as possible. Reordering inputs does not change the gate version, hence the reduction in delay is achieved without any increase in power. Therefore, input ordering is expected to be very effective in minimizing power dissipation. Comparing Tables 7.3a with 7.4a and Tables 7.3b with 7.4b, it can be seen that input ordering reduced power is almost all cases. The reduction in delay due to input ordering meant fewer gates needed to be changed for the constraint to be satisfied and hence the power dissipation decreased. Hence *input ordering at all nodes* is definitely a useful option.

7.5.6 Input Ordering at each stage

As gate versions are changed, the arrival times of the output nodes are changed, and hence the inputs previously ordered may no longer be the optimal permutation. *Input ordering at each stage* ensures that the optimal permutation is always used. From the results in Table 7.5a, the dual use of *input ordering at all nodes* and *at each stage* reduced power dissipation by as much as 17%, (in the case of the mux circuit), when compared to not using the input ordering options (Table 3a). Comparing Tables 7.4a and 7.5a, it can be seen that *input ordering at each stage* is effective in further reducing power dissipation after input ordering is done at each node. While this option reduced the power dissipation in adder4 and mux significantly, and reduced the delay in ex1, the power for ex2 and x4 was increased slightly (less than 2%). This could be due to the constraint that both rise and fall times of the gate output must decrease for a permutation to be accepted (as opposed to the constraints in *input ordering at all nodes*, where either the maximum of the rise and fall times must decrease or the critical slope delay must decrease) (discussed in 6.4). This disallows some gates that might have given better power and delay trade-offs from being used. Nonetheless, this increase in power is small compared to the decrease in power obtained in the other circuits. Hence *input ordering at each stage* is another useful option in the optimization process.

7.5.7 Check_Critical_Path

The option *check_critical_path* checks for any changes in the critical path as gate versions are optimized, and searches for all possible avenues for delay reduction in a circuit. A new critical path is found whenever the original path is no longer critical, and traversal is resumed on the new path. The results from check_critical_path were somewhat varied. While this continuous update method reduced power dissipation for the adder4 and x4 circuit, delay went up (comparing Tables 7.5a and 7.5b). For the mux and ex2 circuits, this option reduced the delay but increased the power.

Continuous updates of critical paths meant that more gates would be changed as each new path is found. This might lead to a decrease in final delay. On the other hand, these gates might not be in the final critical path, hence their optimization might have been redundant, leading to increased power. For example, in Fig. 7.1, gate A was sized to be the largest version to reduce the delay, and in doing so, the gate is off the critical path and gate B is now on the critical path. If at the end of the optimization gate B is still on the critical path, a smaller version of gate A could have been used without changing the delay of gate C.

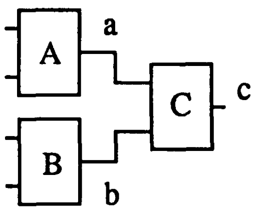


FIGURE 7.1 Redundant Optimization

Furthermore, each of the critical paths found may be interlaced with each other, or have inter-dependencies. A larger version used in one path might be a fanout gate to gates on the final critical path, thereby leading to increased delay.

7.6 Overall Effectiveness

The results showed that the effectiveness of each of the options depended on the topology of the circuit being optimized and on the delay constraint. For example, the simplicity of ex1, with a dominant critical path where each gate fans out to only one gate in the next stage, meant that the minimum sized versions might be the best versions to use, given the small output loads. Hence few gates were changed by the algorithm, and the options did not make much of a difference in this case. However, delay for this circuit was still reduced by 22% with only an increase in power dissipation of 3%.

Comparing the minimum-sized mapping with the best solution from the different combinations of options (Tables 7.6a and 7.6b), delay was reduced by up to 43% with less than 8% increase in power. For ex2, delay was reduced by 27% with only a 1% increase in power, while the delay of x4 was reduced by 25% with only a 2% increase in power.

circuit	a) initial mapping with minimum-sized gates		b) Summary of best results from combinations of options			c) Basic algorithm using delay only		
	delay (ns)	power (uW)	delay (ns)	power (uW)	cpu time (s)	delay (ns)	power (uW)	cpu time (s)
ex1	9.13	25.2	7.47	25.9	< 0.1	7.71	25.9	< 0.1
ex2	21.06	218.5	16.63	221.4	0.2	16.58	237.5	0.1
adder4	17.98	155.0	14.10	167.1	0.3	13.92	185.8	0.1
mux	22.40	279.4	15.70	304.8	0.6	15.66	390.2	0.3
x4	18.19	1469.5	14.60	1506.8	4.9	14.67	1597.8	6.4

Table 7.6: Summary and comparisons of best results

Combinations of options produced up to a 28% decrease in power dissipation when compared to the basic algorithm which did not take into account switching probabilities (Tables 7.6b and 7.6c). Each option had its own unique contributions, and the effectiveness of each routine also depended on other options. For example, input ordering reduced the arrival times at gate outputs, and due to this change in delay values, different gate versions might be chosen by the basic algorithm or other options, thereby leading to further changes in delay or power dissipation of the circuit as a whole.

All routines took little computation time to run. The worst case scenario in traversing a critical path is to change all gates on the path, and the complexity is simply the number of gates on the critical path multiplied by the number of versions of each gate.

7.7 Summary

The tests from utilizing the augmented library and running the various algorithms developed for cell selection on five circuits returned good results in terms of delay reduction with minimal power increase. Circuit delays were reduced by up to 43% with up to 8% increase in power, while the options were able to reduce 28% of power dissipation when compared to using the basic algorithm. All options turn out to be beneficial to reducing power in one way or another.

In conclusion, the basic algorithm with the various options provided a computationally simple method of selecting cell versions for optimal performance. Coupled with a library with cells sized to give good power and delay trade-offs, the whole optimization process returned results that indicate that this design process is indeed effective and efficient.

CHAPTER 8

Conclusion

8.1 Introduction

In view of the increasing importance of low-power design, a fast and efficient low power design method is developed in this thesis. This chapter summarizes the unique contributions of this work and possible future work that could be done.

8.2 Unique Contributions

This work combines several optimization tools that have been previously studied separately along with new ideas to produce a low power design method that is both effective and efficient.

First of all, cell library design is used, for efficiency and ease in circuit design. An augmented cell library is developed with each cell designed to give good power and delay trade-offs. Transistor sizing and input ordering are used in the design of such cells. In addition, area minimization is also taken into account. Transistors in each cell are sized so as to provide a large reduction in delay with little increase in power for the cell.

An accurate delay model that takes into account input transition time is developed, and delay parameters for each cell are carefully obtained from layout extraction and SPICE simulations to ensure accuracy.

Algorithms that select the best cell versions to use in a circuit are developed. Circuits are first mapped with minimum-sized versions to ensure low power, and then gate versions are replaced by larger versions as necessary to satisfy the delay constraint with minimal power increase. Statistical power estimation methods are used to accurately estimate power in a circuit, and the switching probabilities of each node in the circuit, obtained from such an analysis, are used to make decisions regarding the use of sized cell versions at a local level.

In addition to accurate power estimation, input ordering is also used in the algorithm to further optimize the circuit. The effectiveness of input ordering is fully utilized by applying this technique in sizing the transistors in each library cell, as well as on the circuit level by reordering gate inputs for optimal delay performance.

Several unique circuit traversal algorithms are developed, each utilizing different aspects of circuit topology and node characteristics. Switching probabilities, output load, different rise and fall times, and critical path analysis are all used in several different options to the main algorithm for further enhancement of the selection process. The heuristic methods developed have produced an optimization routine that takes little computation time, but produces circuits with desirable delay and power performance.

8.3 Future directions

Possible extensions to the algorithm include a similar but separate algorithm to optimize delay under a power constraint. Given a power constraint that is above the power dissipation of a minimum-sized mapping, power could be increased until the constraint is reached while reducing delay significantly. Other possible improvements include developing a technique to relax the delay of the cells that are not on the critical path, and not in the dependency chains for that path, by changing the versions at these nodes to smaller versions, so as to reduce power further. This can also eliminate redundant optimization. Critical path analysis could also include false path detection.

Theoretical characterization of the algorithm so that some provable results may be derived could also be investigated, though it may be difficult to characterize such heuristic methods. Possible non-heuristic methods of gate selection could also be attempted, though it is not clear at this stage how this can be achieved, given all the factors involved.

Other possible future research could involve establishing a more extensive library to include latches for sequential circuits, and finding other optimization formalizations for determining the sizes of the cells. Circuit forms other than static CMOS could also be examined and the influence of circuit styles could also be studied.

8.4 Summary

The low power design method developed in this thesis has utilized a number of tools, including statistical power estimation, transistor sizing, and input ordering. Further work could be done to enhance the usefulness of this design method in allowing different constraints, or in extending the types of circuits that can be designed with this method.

In conclusion, the basic algorithm, with the various options, provided a fast and efficient way to determine the best combination of gate versions to use. Together with the usage of mainly minimum-sized cell versions coupled with cells sized to give good delay and power trade-offs, this package provides a fast and convenient way to design standard cell circuits for low-power.

Bibliography

[1] J.P. Fishburn and A.E. Dunlop, "TILOS: A Posynomial Programming Approach to Transistor Sizing," in IEEE International Conference on Computer Aided Design, pg. 326-328, November 1985.

[2] D. Marple, "Transistor Size Optimization in the Tailor Layout System," in IEEE Design Automation Conference, Pg. 43-48, 1989.

[3] D. Marple, "Performance Optimization of Digital VLSI Circuits", Technical report: CSL-TR-86-308, Stanford University, 1985.

[4] J. Yuen and C. Svenson, "CMOS Circuit Speed Optimization based on Switch Level Simulation," LSI Design Center, Linkoping University, Sweden.

[5] K. Hedlund, "Models and Algorithms for Transistor Sizing in MOS Circuits," in IEEE International Conference on Computer Aided Design, pages 12-14, October, 1984.

[6] K. Hedlund, "Aesop: A Tool for Automated Transistor Sizing," in Proceedings of the 24th Design Automation Conference, pg. 114-120, June, 1987.

[7] H. Hsieh and D. Ostapko, "Size Optimization for CMOS Basic Cells of VLSI", ISCAS, 1992.

[8] M. Crit, "Transistor Sizing in CMOS Circuits," in Proceedings of the 24th Design Automation Conference, pg. 121-123, June, 1987.

[9] M. Shoji, "FET Scaling in Domino CMOS Gates," IEEE Journal of Solid-State Circuits, vol. sc-20, No. 5, October 1985.

[10] M. Matson and L. Glasser, "Macromodeling and Optimization of Digital MOS VLSI Circuits," IEEE Transactions on Computer-Aided Design, Vol. CAD-5, No. 4, October 1986.

[11] M. Matson, "Macromodeling and Optimization of Digital MOS VLSI Circuits," Ph.D.

dissertation, Massachusetts Institute of Technology, Feb. 1985.

[12] L. Brocco, S. McCormick and J. Allen, "Macromodeling CMOS Circuits for Timing Simulation", IEEE Transactions on Computer-Aided Design, Vol. 7, No. 12, December 1988.

[13] A. Ghosh, S. Devadas, K. Keutzer and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," Proceedings of the 29th Design Automation Conference, pg. 253-259, 1992.

[14] S. Devadas, K. Keutzer and J. White, "Estimation of Power Dissipation in CMOS Combinational Circuits Using Boolean Function Manipulation,", IEEE Transactions on Computer-Aided Design, Vol. 11, No. 3, March 1992.

[15] A. Shen, A. Ghosh, S. Devadas, k. Keutzer, "On Average Power Dissipation and Random Testability of CMOS Combinational Logic Networks," in IEEE International Conference on Computer-Aided Design, pg. 402-407, November 1992.

[16] A. Chandrakasan, S. Sheng and R. Brodersen, "Low-Power CMOS Digital Design," IEEE Journal of Solid-State Circuits, Vol. 27, No. 4, April 1992.

[17] M. Berkelaar and J. Jess, "Gate Sizing in MOS Digital Circuits with Linear Programming," Proceedings of the European Design Automation Conference 1990, pg. 217-221.

[18] M. Berkelaar, "Area-Power-Delay Trade-off in Logic Synthesis," Ph.D. dissertation, Technische Universiteit Eindhoven, September 1992.

[19] V. Tiwari, P. Ashar, S. Malik, "Technology Mapping for Low Power," in Proceedings of the 30th Design Automation Conference, pg. 74-79, 1993.

[20] C. Tsui, M. Pedram, A. Despain, "Technology Decomposition and Mapping Targeting Low Power Dissipation," in Proceedings of the 30th Design Automation Conference, pg. 68-73, 1993.

[21] E. Detjeus, G. Gannot, "Technology Mapping in MIS," IEEE International Conference on Computer Aided Design, pg. 116-119, 1987.

[22] K. Keutzer, K. Kolwicz, M.Lega, "Impact of LIbrary SIze on the Quality of Automated Synthesis," IEEE International Conference on Computer Aided Design, pg. 120-123, 1987.

[23] K. Keutzer, "DAGON: Technology Binding and Local Optimization by DAG Matching," in Proceedings of the 24th Design Automation Conference, pg. 341-347, 1987.

[24] F. Obermeier and R. Katz, "An Electrical Optimizer that Considers Physical Layout," in Proceedings of the 25th Design Automation Conference, pg. 453-459, 1988.

[25] B. Carlson and C. Chen, "Performance Enhancement of CMOS VLSI Circuits by Transistor Reordering," Design Automation Conference, pg. 361-366, 1993.

[26] H. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits," IEEE Journal of Solid-State Circuits, Vol. sc-19, No. 4, August 1984.

[27] A. Kayssi, K. Sakallah and T. Mudge, "The Impact of Signal Transition Time on Path Delay Computation," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 40, No. 5, May 1993.

[28] J. Silva, K.Sakallah, L. Vidigal, "FPD - An Environment for Exact Timing Analysis," IEEE International Conference on Computer Aided Design, pg. 212-215, 1991.

[29] S. Huang, T. Parng, J. Shyu, "A New Approach to Solving False Path Problem in Timing Analysis," IEEE International Conference on Computer Aided Design, pg. 216-219, 1991.

[30] P. McGeer and R. Brayton, "Efficient Algorithms for Computing the Longest Viable Path in a Combinational Network," in Proceedings of the 26th Design Automation Conference, pg. 561-567, 1989.

[31] H. Chen and D. Du, "Path Sensitization in Critical Path Problem," IEEE International Conference on Computer Aided Design, pg. 208-211, 1991.

[32] E. Sentovich, K. Sing, L. Lavagno, C. Moon, R. Murgai, "SIS: A System for Sequential Circuit Synthesis," Electronics Research Laboratory Memorandum No. UCB/ERL M92/41, University of California at Berkeley, 1992.

[33] D. Hill, D. Shugrad, J. Fishburn and K. Keutzer, *Algorithms and Techniques for VLSI Layout Synthesis*, Kluwer Academic Publishers, 1989.

[34] R. Mazaiasz and J. Hayes, *Layout Minimization of CMOS Cells*, Kluwer Academic Publishers, 1992.

[35] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley, 1985.

[36] V. Chvatal, *Linear Programming*, W.H. Freeman and Company, 1983.

[37] C. T. Gray, W. Liu and R. Cavin, *Wave Pipelining: Theory and CMOS Implementation*, Kluwer Academic Publishers, 1993. Section 4.2.2, "Delay Models", pg. 65ff.

[38] C.H. Tan, "Speed Optimization of a Four-Bit CMOS Full-Adder by Transistor Sizing," final project report for 6.371: Introduction to VLSI Systems, MIT, Dec 1992.

[39] J. Monteiro, S. Devadas and B. Lin, "A Methodology for Efficient Estimation of Switching Activity in Sequential Logic Circuits", Design Automation Conference, 1994.

4735-40