

10-1-1993

Circuit Optimization by Transistor Reordering for Minimization of Power Consumption under Delay Constraint

Sharat C. Prasad
Texas Instruments

Kaushik Roy
Purdue University School of Electrical Engineering

Follow this and additional works at: <http://docs.lib.purdue.edu/ecetr>

Prasad, Sharat C. and Roy, Kaushik, "Circuit Optimization by Transistor Reordering for Minimization of Power Consumption under Delay Constraint" (1993). *ECE Technical Reports*. Paper 244.
<http://docs.lib.purdue.edu/ecetr/244>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

CIRCUIT OPTIMIZATION BY
TRANSISTOR REORDERING FOR
MINIMIZATION OF POWER
CONSUMPTION UNDER DELAY
CONSTRAINT

SHARAT PRASAD, TEXAS INSTRUMENT
KAUSHIK ROY, PURDUE UNIVERSITY

TR-EE 93-35
OCTOBER 1993



SCHOOL OF ELECTRICAL ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

Circuit Optimization by Transistor Reordering for Minimization of Power Consumption under Delay Constraint

Sharat C. Prasad

Integrated Systems Laboratory
Texas Instruments Inc.

Kaushik Roy

1285 Electrical Engineering Building
Purdue University
West Lafayette, Indiana 47907-1285

Topics : Technology Mapping (4.2) and Timing Analysis and Verification (1.3)

Please direct all correspondence to :

Sharat C. Prasad

Integrated Systems Laboratory
Texas Instruments Incorporated

P. O. Box 655474, MS 446

Dallas, TX 75265

(214)-995-7296

Fax : (214)-995-6190

prasad@hc.ti.com

Circuit Optimization by Transistor Reordering for Minimization of Power Consumption under Delay Constraint

Abstract

In this paper we address the problem of optimization of VLSI circuits to minimize power consumption while meeting performance goals. We present a method of estimating power consumption of a basic or complex CMOS gate which takes the internal capacitances of the gate into account. This method is used to select an ordering of series-connected transistors found in CMOS gates to achieve lower power consumption. The method is very efficient when used by library based design styles. We describe a multi-pass algorithm which makes use of transistor reordering to optimize performance and power consumption of circuits, which has a linear time complexity per pass and which converges to a solution in a small number of passes. Transformations besides transistor reordering can be used by the algorithm. The algorithm has been benchmarked on several large examples and the results are presented.

1 Introduction

In order to meet the rapidly advancing functionality, performance, cost-efficiency and other requirements, automatic synthesis tools have become indispensable. The designs are described at Register-Transfer or higher level. A technology independent logic gate level realization is generated. It is then mapped to a specific technology library. Finally the technology mapped circuit is optimized to ensure that all requirements have been met. The process has multiple steps as the problem of realizing the specified functionality with gates in a given library to meet all the requirements is too complex to be solved in one step. The third step, namely optimization of a technology mapped circuit, is the subject of this paper.

Technology mapping algorithms try to minimize the area [1, 2] and power consumption [3, 4, 5] while ensuring that the delay does not exceed an upper limit. Real circuits are Directed Acyclic Graphs (DAGs). The process of mapping for DAGs, being similar to optimal code generation with common subexpressions, is NP-complete [6]. So, to make it possible to use the mapping algorithms, which inherently have exponential complexity, the DAG is broken up into small trees. Needless to say, the critical path delay of the circuits after such technology mapping is not the smallest that it can be. Tools which use results of a timing analysis to guide application of transformations to the mapped logic [7, 8, 9, 10] achieve further reduction in delay.

A review of existing circuit optimization algorithms is presented in section 2. The existing algorithms do not take power consumption into account and/or require too long a run-time to be used on the large circuits of today. The work described here is a part of an effort to develop a circuit optimization algorithm for minimizing power consumption and area under delay constraint. The algorithm is capable of using several transformations, has a linear time complexity per pass and requires a small number of passes to converge. In this paper we present this algorithm and describe the use of a sample transformation - transistor reordering. We present a method of estimating power consumption of a basic or complex CMOS gate which takes the internal capacitances of the gate into account. This method is

used to select an ordering of series-connected transistors found in CMOS gates to achieve lower power consumption. The method is very efficient when used by library based design styles.

Most performance driven circuit optimization algorithms are based on iterative improvement. Any such algorithm has four components -

1. a method for determining which gates in the circuit to examine next,
2. a set of transformations to apply to the gates being examined,
3. methods for computing the overall improvement due to transformations, and
4. a method for updating the circuit after each transformation.

In case of an algorithm for power minimization which is to work with more than one transformation, the methods referred to in (1) and (4) above should be independent of the individual transformations. Some of the transformations referred to in (2) should be able to influence power consumption of the circuit. Methods referred to in (3) above should be able to evaluate the effect of the transformations on power consumption of the circuit.

Our algorithm is based upon the algorithm described in [11]. The latter is similar to the algorithm presented in [12] and is essentially an algorithm for computing the longest path in the circuit by computing the longest path through each gate in the circuit. Additionally it also optimizes the circuit as it is carrying out the computation. Our algorithm, described in section 6, specifies the gate to be examined next as the potential candidate for a transformation and updates the timing data of the circuit after a transformation is applied. At present the algorithm uses only one transformation - transistor reordering. The selection of transistor reordering is motivated by the results reported by Sakurai et. al. in [13] and summarized in section 5.

In a digital CMOS circuit, power is dissipated only when there is a transition (a ZERO to ONE or ONE to ZERO in logic value) at a circuit node. The problem of determining how often transitions occur at a node in a digital circuit is difficult because transitions depend on

the applied input vectors and the time instants at which they are applied. During the course of normal operation these vary widely. A review of reported methods of estimating power consumption of a digital circuit is presented in section 2.2. Our estimation method meets the requirement that if a transformation introduces a new gate, it is possible to compute the contribution of the new gate to the total power consumption of the circuit *incrementally*.

Our model for estimating power consumption in a CMOS basic or complex gate and its application in reordering the series-connected transistors for low power consumption are the subjects of sections 4.1 and 5.2. Finally in section 7 experimental results for MCNC benchmark circuits are presented.

2 Prior Work

2.1 Delay Constrained Circuit Optimization

Circuit optimization algorithms which work with a technology-mapped circuit have an inherent advantage over algorithms which work with technology independent circuit — knowledge of precise delays. This makes the former more effective. Consequently a number of such algorithms have been reported in the literature. In [7] each gate which is on at least one timing path with delay larger than the specified value is examined. The transformation they use is gate resizing. Each time a gate is resized the longest timing path through every gate in the circuit is recomputed.

In [10] each node on the current critical path is examined. Several transformations, which include gate resizing, buffer insertion, local restructuring, etc., are used. Here too the timing analysis is updated after each transformation. Typically this means carrying out the timing analysis again for the entire **fanin** and **fanout** cones of the transformed gate or set of gates.

In [9] a systematic breadth-first traversal is used to visit each gate in the circuit and examine it. They, like us, make use of transistor-reordering. Each individual gate is simulated using SPICE to determine, among other things, delay through the gate. The breadth-first traversal enables them to ensure that delays to only those gates are made invalid which are no longer required during the current traversal.

The first two, [7] and [10], exhibit quadratic or worse worst-case time complexities. The third, [9], cannot be used for large circuits due to its use of SPICE.

Only in [9] power consumption is considered. They state that simply choosing the input order to maximize the propagation delay through the gate minimizes power dissipation. They do not take into consideration the switching characteristics of the input signals. To decide whether to connect the latest arriving signal to the transistor next to the output or to the transistor next to V_{dd} /ground, they carry out two SPICE simulations of each gate.

2.2 Power Estimation

All reported methods of estimating power consumption of digital circuits take one of the following two approaches -

1. simulate the circuit with input vectors [14], or
2. use statistical switching properties of signals [15, 16].

Approach (1) is clearly not suitable for the application at hand. In approach (2) power consumption of the circuit is obtained by summing the power consumption of all the gates in it. If leakage currents, direct-path short-circuit currents and gate internal node capacitances are ignored, the average power drawn by a CMOS gate is given by

$$W_{av} = \frac{1}{2} C_Y V_{dd}^2 \left\{ \lim_{T \rightarrow \infty} \frac{n_y(T)}{T} \right\} \quad (1)$$

Here V_{dd} is the supply voltage, C_y is the capacitance at the output of the gate, and $n_y(T)$ is the number of transitions of $y(t)$, the logic signal at the output of the gate, in the time interval $[-T/2, T/2]$. The last term in equation (1) is the average number of transitions per unit time. Some researchers assume that the value of a logic signal in one time frame is independent of that in the previous time frame and equate the last term in above equation to $f p_y (1 - p_y)$, where $1/f$ is the duration of a time frame, and p_y and $(1 - p_y)$ are the probabilities that at a given instant of time t , $y(t) = \text{ONE}$ and $y(t) = \text{ZERO}$, respectively. p_y is called the signal probability of $y(t)$. This assumption is not correct and the probability

of a transition cannot be computed from static signal probabilities [16]. In [15] the author has presented a probabilistic measure of the average number of transitions per unit time for any logic signal and called it *transition density*. Under assumptions that logic signals can be modeled as *strict-sense stationary mean-ergodic stochastic processes* and logic modules have zero delay the transition densities and signal probabilities of signals at circuit nodes can be computed using a breadth-first traversal of the circuit if those of primary input signals are known.

3 Preliminaries and Definitions

3.1 Circuits

Our circuits are *synchronous* and are composed of *gates* from a library. Each gate has one or more input pins and one output pin. Several pins are electrically tied together by a signal. Each signal connects to the output pin of exactly one gate, called the driver gate.

3.2 Gate delay model

The delay characteristics of all the gates in the library are known and are in the form of a pair $\{T_{i,j}^i(G), R_{i,j}(G)\}$ for every pair of an input terminal I_i and an output terminal O_j of every gate G . $T_{i,j}^i(G)$ is the fanout load independent delay and is called the *intrinsic delay*. The super-script "i" is for "intrinsic." $R_{i,j}(G)$ is the additional delay per unit fanout load. Each input terminal I_i of each gate G has a capacitance $C_i(G)$ associated with it. The total propagation delay through a gate G from a given input terminal I_i to a given output terminal O_j is given by

$$T_{i,j}^i(G) + R_{i,j}(G)C_j(G),$$

where $C_j(G)$ is the total fanout load capacitance at O_j . Separate delays are associated with rising and falling transitions.

3.3 Switching event probabilities

The signal probability p_y of the logic signal $y(t)$ at node y is the probability that $y(t) = \text{ONE}$ at a given instant of time t . Then $(1 - p_y)$ is the probability that it is a ZERO. Let us assume, without any loss of generality, that all signal transitions occur at the leading edge of the clock. Let y be the node in the circuit which has been monitored for a large number N of clock cycles. Let the node be observed at a point in the clock cycle which is separated from the leading edge of the clock by a long enough interval to allow the logic level at node to reach its stable value. Then it follows that we would have observed a ONE during $p_y N$ clock cycles and a ZERO during $(1 - p_y)N$ of the clock cycles.

We normalize all transition densities by dividing them by the transition density of the clock ($= 2f$, where f is the clock frequency). The normalized transition density d_y , a real number between 0 and 1, of the node y is the probability that if we select a clock cycle at random, there was a transition at the node at the beginning of the clock cycle. Then $(1 - d_y)$ is the probability that there was no transition. We denote the probability of a rising or a falling transition at node y by p_y^\uparrow . So,

$$p_y^\uparrow = d_y \tag{2}$$

Since a rising transition at any node (including an internal nodes) has the same probability of occurring as a falling transition (because a rising transition has to be followed by a falling transition and vice-versa)

$$p_y^\uparrow = p_y^\downarrow = \frac{1}{2} p_y^\uparrow, \tag{3}$$

where p_y^\uparrow is the probability of a rising transition at node y and p_y^\downarrow is the probability of a falling transition at node y .

3.4 Computing transition probabilities

Computation of signal probabilities and hence of signal transition probabilities is made complicated by the fact that even if primary input signals are mutually independent, due to

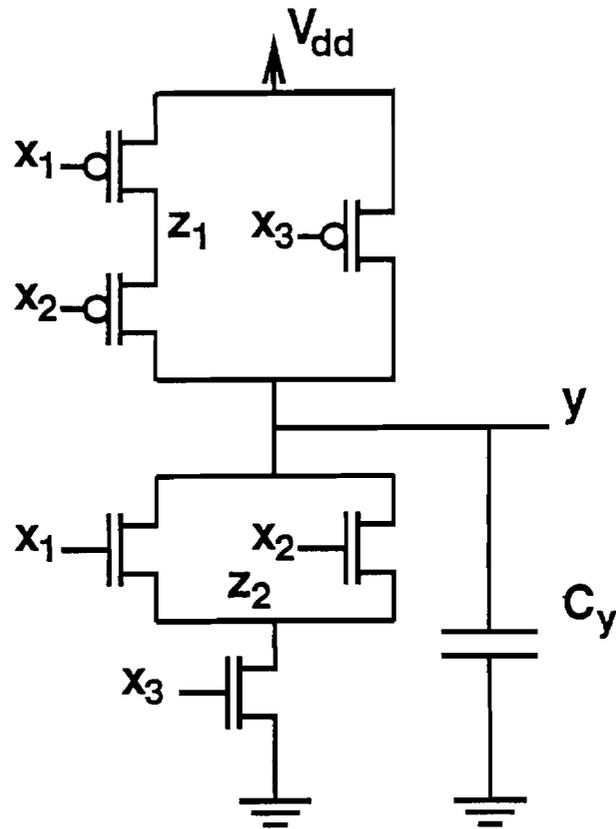


Figure 1: CMOS gate $y = \overline{(x_1 + x_2)x_3}$

reconvergent fanout, signals at circuit nodes become correlated. In [22] an efficient implementation of a general algorithm to compute signal probabilities has been presented. This algorithm performs well in cases of small number of primary inputs even if the circuit is large. We make use of this algorithm.

4 CMOS gates

As submicrometer feature sizes become commonplace and the push to shrink feature sizes further continues, the changing electrical properties of the circuit structures affect their usefulness. An example is CMOS gates with series connected transistors. Because the delay scales linearly with number of transistors in series, large NAND/NOR/complex gates are used infrequently. However Sakurai et. al. in [13] report a result which should encourage more extensive use of NAND/NOR/complex gates. They show that the ratio of the delay of NAND/NOR to the delay of inverter becomes smaller in the submicrometer region. This is because the V_{ds} and V_{gs} of each transistor in a series connection of several transistors are

smaller than those of a transistor in an inverter. The smaller voltages make carrier velocity saturation less severe of a problem.

Another reason for infrequent use of complex gates in particular is the lack of broad availability of good technology mapping tools. In the very recent past a number of technology mapping algorithms have been reported in the literature which are able to explicitly minimize delay [1, 2].

If and when larger NAND/NOR/complex gates begin to be used, it will become even more important to order the series-connected transistors in them properly. For example in case of the 4-input CMOS NAND gate in Texas Instruments' BICMOS gate-array library, for a load of thirteen inverters (ignoring interconnect), the delay varies by 20 % and power dissipation by 10 % between the good and the bad transistor order. For this reason we have selected transistor reordering or gate-input reordering as the sample transformation to use with our optimization algorithm. It is to be noted that this transformation has the nice feature that it has negligible impact on area.

4.1 Power consumption of CMOS gates

Consider the CMOS gate in figure 1, input pattern $x_1 = \uparrow, x_2 = 0, x_3 = 1$ causes a falling transition at the output and so does the input pattern $x_1 = 0, x_2 = \uparrow, x_3 = 1$. But in case of the first pattern, the capacitance being discharged equals $C_y + C_{z_1}$ which is more than C_y , the capacitance being discharged in case of the second pattern. This simple example points out two things,

1. The gate presents a variable capacitance to the power/ground rails. The magnitude of this capacitance depends on the logic values at the inputs to the gate.
2. Given two signals, A and B which are to be connected to the two equivalent inputs x_1 and x_2 of the gate in figure 1 and which are such that very often A has a transition and B stays ZERO, then A should be connected to x_2 and B to x_1 as this results in lower power consumption than the other case.



Capacitances at the internal nodes of a CMOS gate charge and discharge resulting in additional power consumption. Some of these, but not all, happen at the same time instants as the charging and discharging of the capacitance at the gate output. For example, when the input to the gate in figure 1 is $(x_1 = \downarrow, x_2 = 1, x_3 = 1)$ there may be a rising transition at z_1 though y remains at ZERO. On the other hand when the input is $(x_1 = \uparrow, x_2 = 0, x_3 = 1)$ there is a falling transition at both z_1 and y . Had the occurrence of transitions at internal nodes had a simpler relation to occurrence of transitions at the output node, equation 1 could have been used for the total power consumption of a CMOS gate with C_y replaced with a *net effective output capacitance*. But it is not so and hence internal nodes of a gate need to be treated as circuit nodes distinct from (though related to) the output node of the gate. Hence, from equation (1),

$$W_{av} = \frac{1}{2} V_{dd}^2 f(C_y d_y + \sum_i C_{z_i} d_{z_i}), \quad (4)$$

where z_i 's are the internal nodes of the gate and d_{z_i} is the normalized transition density of "signal" at node z_i .

In [17] the authors show that determining $p_{z_i}^\uparrow$ ($= d_{z_i}$) is NP-hard. The difficulty in determining $p_{z_i}^\uparrow$ arises from the fact that for a rising transition to occur at an internal node at the beginning of a clock cycle, (i) the logic signal at the node must have had value ZERO in the immediately previous clock cycle, and (ii) conduction state of at least one path from node to V_{dd} must have changed from OFF to ON at the beginning of the clock cycle. Probability of a conducting path existing between any two nodes of a gate can be determined under the assumption that inputs to the gate are mutually independent. But determining the logic value at an internal node is difficult because an internal node may have been isolated from both V_{dd} and ground during the immediately previous clock-cycle and possibly in all of the zero or more clock-cycles immediately preceding that. In effect it is required to keep track of the past transitions at the node potentially going back to $-\infty$ — an impossible task. In [18] this problem is resolved by assuming, whenever state of an internal node cannot be determined, that it is such that a transition occurs. This is done by using, in place of $p_{z_i}^\uparrow$,

the probability that the number of conducting paths from z_i to V_{dd} changes from zero to a number larger than zero. The latter probability is an upper limit on $p_{z_i}^\uparrow$. This suits their problem of determining the worst case peak supply current. As we need to be able to make a comparison between power dissipated in the same gate for two different input orderings we need the errors to be small and comparable in two cases.

A more accurate upper limit on $p_{z_i}^\uparrow$ can be obtained from the observation that for a pair of complimentary transitions to occur at an internal node, the number of conducting paths from the node to V_{dd} must change from zero to a number larger than zero followed by a similar change in that to ground. If the number of conducting paths from a node to V_{dd} changes from zero to a number larger than zero once every 40 clock-cycles but that to ground only once every 100 clock-cycles, then the node cannot have more than 2 transitions every 100 clock-cycles. Therefore,

$$p_{z_i}^\uparrow = \begin{cases} p_{z_i, V_{dd}}^\uparrow & \text{if } p_{z_i, V_{dd}}^\uparrow \leq p_{z_i, V_{ss}}^\uparrow \\ p_{z_i, V_{ss}}^\uparrow & \text{otherwise} \end{cases}$$

where p_{z_i, z_j}^\uparrow is the probability that the number of conducting paths from z_i to z_j changes from zero to greater than zero.

The p_{z_i, z_j}^\uparrow 's can be determined by applying a reduction procedure to a graph obtained from the schematic of the gate. To each node of the gate there corresponds a vertex in the graph. Hence there is a vertex for ground, V_{dd} , y , and each of the z_i 's. To each transistor with its drain and source connected to two nodes, there corresponds an edge between the corresponding vertices. Each edge e has two labels p_e and p_e^\uparrow . p_e denotes the probability that edge e is ON or conducting and p_e^\uparrow denotes the probability that edge e turns from OFF to ON. Note p_e equals p_{x_i} if e corresponds to an NMOS transistor with logic signal x_i connected to its base and to $1 - p_{x_i}$ otherwise. p_e^\uparrow is simply $d_{x_i}/2$ in both the cases. These graphs are a restricted class of graphs called series-parallel graphs [19].

The required path conduction probabilities can now be determined using an operation termed graph reduction in [18]. Let e_p (e_p) denote the single edge equivalent to the parallel

(series) combination of two edges e_1 and e_2 , then

$$p_{e_p} = p_{e_1} + p_{e_2} - p_{e_1}p_{e_2} \quad (5)$$

$$p_{e_s} = p_{e_1}p_{e_2} \quad (6)$$

$$p_{e_p}^\dagger = p_{e_1}^\dagger(1 - p_{e_2}) + p_{e_2}^\dagger(1 - p_{e_1}) - p_{e_1}^\dagger p_{e_2}^\dagger \quad (7)$$

$$p_{e_s}^\dagger = p_{e_1}^\dagger p_{e_2}^\dagger + p_{e_2}^\dagger(1 - p_{e_1}) - p_{e_1}^\dagger p_{e_2}^\dagger \quad (8)$$

Let the path from node x_i to node x_j be reduced to edge e_r , then

$$p_{x_i, x_j}^\dagger = p_{e_r}^\dagger$$

4.2 Characterization of gates in the library

For any given n , the total number of functions with n inputs is 2^{2^n} , a very very large number even for small n . Most typical semi-custom libraries will have gates corresponding to a very small subset of these and usually limited to $n \leq 6$. In that case, it is efficient to analyze each of these in advance. Subsequently the results of the analysis are used when processing a circuit containing these gates.

The characterization process derives symbolic expressions for $p_{z_i, V_{dd}}^\dagger$ and $p_{z_i, V_{ss}}^\dagger$ for each internal node z_i of each gate in the library. The symbols in these expression are the p_{x_i} 's and $p_{x_i}^\dagger$'s, where x_i 's denote the input signals to the gate.

In addition the node capacitances C_{z_i} for each internal node z_i of each gate are also computed.

5 Transistor Reordering

5.1 Delay

Since the load capacitance at the output of every gate in the circuit is known, given a gate, the propagation delays from each of its inputs to its output is known. As a result of timing analysis the total delay for the longest paths through each input of the gate is also known. When the gate is a NAND or a NOR gate, to reorder its inputs to reduce

delay, the latest arriving signal is connected to the input with the smallest delay and so on. When the gate is a complex gate, the set of input terminals is divided into permutable sets. For example, for the complex gate in figure 1 the results is $\{\{x_1, x_2\} \{x_3\}\}$ and only input signals connected to x_1 and x_2 may be interchanged. The set of input signals is also analogously divided. Now each subset of input terminals and the corresponding subset of input signals is taken and reordering is carried out. In full-custom design styles, it is possible to swap a transistor or a parallel connection of transistors with another transistor or a parallel connection of transistors connected in series with the first. But in semi-custom design styles, one is restricted to using the gates available in the library without being able to modify their internal connections. Hence in case of semi-custom design styles, gate-input reordering is a more appropriate name for this transformation than transistor reordering.

5.2 Power Consumption

Because of the complex dependence of the power consumption of a gate on signal transition probabilities of its inputs, it is not possible to tell with small computational effort as to which input order is the best. Hence we make use of an exhaustive enumeration method as well as a heuristic method. As most gates only have a small number of transistors in series, we found exhaustive enumeration viable. All possible orderings are enumerated and the power consumption in case of each is computed (section 4).

The heuristic we developed consists of computing, for each signal, the probability of the event that it will be switching and all other signals in the same permutable set will have the "on" value (i.e. a ONE for NMOS and a ZERO for PMOS). Then the signal with the largest value is connected to the input closest to the output terminal. To understand the reasoning behind the heuristic, consider the case when the signal with the largest value is connected to the input closest to V_{dd} /ground. Each time the signal has an OFF-to-ON transition with the other transistors in series in ON state, all the internal node capacitances discharge/charge. When the signal has an ON-to-OFF transition with the other transistors in series in ON state, internal node capacitances charge/discharge causing power to be consumed.

6 Optimization algorithm

The algorithm is based on breadth-first traversals of the circuit. A traversal going from primary inputs and register outputs to primary outputs and register inputs is referred to as a *forward* traversal and the one in reverse direction is called a backward traversal. Each forward traversal allows us to compute for each gate G the delay of the longest path from a primary input or register output to this gate output which we denote by $T^f(G)$. Similarly the backward traversal allows us to compute or recompute for each gate the delay of the longest path from this gate output to a primary output or a register input which we denote by $T^b(G)$. Obviously the total length of the longest path through a gate G is given by $T^t(G) = T^f(G) + T^b(G)$. Both traversals require time which is a linear function of the size of the circuit [12].

First we try to meet the performance goal. To do this we begin by performing a forward traversal and then a backward traversal. Hence when a gate G is reached during the backward traversal $T^t(G)$ is known as $T^f(G)$ was computed earlier and $T^b(G)$ has just been computed. If $T^t(G)$ is smaller than the specified delay, nothing is done. If $T^t(G)$ is greater than the specified delay, we try to reorder the inputs of G to reduce $T^t(G)$ (section 5.1). If we succeed in doing that the delay through G would have changed and the $T^f(G')$'s for each gate G' in the fanout cone of G and $T^b(G'')$'s for each gate G'' in the fanin cone of G become invalid. But we do not need either of them during the current traversal! The current backward traversal has already finished with gates G' in the fanout cone of G and is in the process of computing $T^b(G'')$'s for each gate G'' in the fanin cone of G . When this backward traversal is completed, there exist valid $T^b(G)$'s for each gate G but if any reordering took place then not all $T^f(G)$'s are valid. If no reorderings took place then either performance goal has been met or the performance of the circuit cannot be improved further. In both cases we proceed to power minimization.

If some reorderings took place, then we perform a forward traversal next. Once again when a gate G is reached $T^t(G)$ is known as $T^b(G)$ was computed during the backward

<i>Circuit</i>	<i>Initial</i>		<i>Optimized</i>			
	Delay	Power	Delay		Power	
				%		%
vg2	10.4	414	9.7	9.5	384	7.3
rd84	8.0	581	7.4	6.9	543	6.5
rd73	6.3	627	5.8	7.3	584	6.8
clip	7.2	653	6.6	7.7	602	7.8
duke2	10.6	1771	9.6	9.3	1659	6.3
misex3c	10.6	2560	9.9	6.0	2416	5.6
apex6	9.6	4200	8.9	7.0	4001	4.7
alu4	24.4	4823	22.4	8.2	4618	4.3
apex4	18.3	13977	17.0	7.1	13124	6.1

Table 1: Experimental results

traversal just completed and $T^f(G)$ has just been computed. Just as during the backward traversal described above we continue the forward traversal trying to reorder gate inputs when $T^f(G)$ is larger than specified.

The alternating forward and backward traversals are continued until during a traversal no reorderings took place. When that happens either performance goal has been met or the performance of the circuit cannot be improved further. In both cases we proceed to power minimization.

Power minimization is also carried out using alternating forward and backward traversals. But now each gate is evaluated differently. We determine the increase in delay for the input order corresponding to least estimated power dissipation (section 5.2). If the increase in delay is less than the available slack, the inputs are reordered. Slack is defined as the difference in delay between the longest path in the circuit and the longest path through the gate.

7 Experimental Results

The algorithms described here have been implemented in Common Lisp Object System [20] on Texas Instruments' Explorer workstation. We report the results of experiments with several MCNC benchmark circuits. These circuits were translated to the input language of DROID [21] design system. Two-level optimizations and multi-level optimizations [22]

were carried out and the circuit was finally mapped to Texas Instruments' BICMOS gate-array library. The delay and estimated power consumption of the circuits at this stage are reported in columns two and three of the table in figure 7. The delays are in nS and the power consumption is in units of power consumption of a gate driving one inverter and experiencing 10^6 transitions per second. All inputs were assigned a signal probability of .5. The transition densities were random numbers in the range from .001 to 100 million transitions per second.

The technology mapped circuit were then input to the optimization tool described here. A low enough delay goal (5 nano-second) was specified forcing optimization for performance until the delay could not be reduced further. Thereupon the circuits were optimized for lower power consumption without increasing the delay. The final delay and power consumption and the percentage improvements from corresponding initial values are reported in columns 4 through 7. It is seen that the average improvement in delay is 8 % and the same in power consumption. is 7 %.

8 Conclusions

In this paper we presented a method of estimating power consumption in CMOS gates which takes the capacitances at internal nodes of the gates into account. We used these estimates to guide gate input reordering to reduce power consumption. We also described an efficient algorithm which can use gate input reordering and other transformations to optimize power consumption of circuits under a delay constraint.

9 Acknowledgment

The authors would like to express their thankfulness to Susan Hric, Sanjive Agarwala, Pat Bosshart, Brock Barton, Bob Hewews, and Pallab Chatterjee.

References

- [1] K. Chaudhary and M. Pedram, "A Near Optimal Algorithm for Technology Mapping Minimizing Area under Delay Constraints," *Proc. 29th ACM/IEEE Design Automation*

Conf., pp. 492-498, 1992.

- [2] H. J. Touati, C. W. Moon, R. K. Brayton and A. Wang "Performance-oriented Technology Mapping," *Proc. 6th MIT Conf, Advanced Research in VLSI*, E. J. Dally ed., pp. 79-97, 1990.
- [3] B. Lin and H. de Man, "Low-Power Driven Technology Mapping under Timing Constraint," *International Workshop on Logic Synthesis*, pp. 9a.1-9a.16, 1993.
- [4] Chi-Ying Tsui, M. Pedram and A. M. Despain, "Technology Decomposition and Mapping Targeting Low Power Dissipation," *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 68-73, 1993.
- [5] V. Tiwari, P. Ashar and S. Malik, "Technology Mapping for Low Power," *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 74-79, 1993.
- [6] A. V. Aho, S. C. Johnson and J. D. Ullman, "Code Generation for Expressions with Common Subexpressions" *Journal of the ACM*, 24(1), pp. 146-160, 1977.
- [7] Wen-Ben Jone and Chen-Liang Fang, "Timing Optimization By Gate: Resizing And Critical Path Identification," *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 135-139, 1993.
- [8] H. J. Touati, et. al., "Delay Optimization of Combinational Logic Circuits by Clustering and Partial Collapsing," *IEEE International Conf. on CAD*, pp. 188-191, 1991.
- [9] B. S. Carlson and C. Y. R. Chen, "Performance Enhancement of CMOS VLSI Circuits by Transistor Reordering," *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 361-366, 1993.
- [10] J. P. Fishburn, "LATTIS: An Iterative Speedup Heuristic for Mapped. Logic," *Proc. 29th ACM/IEEE Design Automation Conf.*, pp. 488-491, 1992.
- [11] S. Agarwala and P. Bosshart, "A Linear Time Algorithm for Timing Directed Optimizations," *In preparation*.
- [12] R. B. Hitchcock, G. L. Smith and D. D. Cheng, "Timing Analysis of Computer Hardware," *IBM J. Res. Develop.*, vol. 26, no. 1, pp. 100-105, Jan. 1982.
- [13] T. Sakurai and A. R. Newton, "Delay Analysis of Series-Connected MOSFET Circuits," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 2, pp. 122-131, February 1991.

- [14] R. Burch, F. Najm, P. Yang, and T. Trick, "McPOWER: A Monte Carlo Approach to Power Estimation," *IEEE International Conf. on CAD*, pp. 90-97, 1992.
- [15] F. N. Najm, "Transition Density, A Stochastic Measure of Activity in Digital Circuits," *28th ACM/IEEE Design Automation Conf.*, pp. 644-649, 1991.
- [16] A.Ghosh, S.Devdas, K.Keutzer, and J.White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," *29th ACM/IEEE Design Automation Conf.*, pp. 253-259, 1992.
- [17] F. N. Najm and I. Hajj, "The complexity of Test Generation at Transistor Level," Rep. UILU-ENG-87-2280, Coordinated Science Lab., Univ. of Illinois at Urbana Champaign, Dec. 1987.
- [18] F. Najrn, R. Burch, P. Yang, and I. Hajj, "Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 4.,pp. 439-450, April 1990.
- [19] F. Harary, "Combinatorial problems in graphical enumeration," in *Applied Combinatorial Mathematics*, E. F. Beckenbach, Ed. New York: Wiley, 1984.
- [20] D. G. Bobrow, L. G. Demichiel, R. G. Gabriel, S. E. Keene, G. Kiczales, and D. A. Moon, "Common Lisp Object System," in *Common Lisp*, Digital Press, pp. 770-864, 1990.
- [21] P.Kollaritsch, S.Lusky, D.Matzke, D.Smit, and P.Stanford, "A Unified Design Representation Can Work," *26th ACM/IEEE Design Automation Conf.*, pp. 811-813, 1989.
- [22] K. Roy and S. Prasad, "Circuit Activity Based Logic Synthesis for Low Power Operations," *IEEE Trans. on VLSI Systems*, to appear in December 1993.