Published in *Integration, the VLSI Journal*, Volume 20, Issue 3, 1996.

Technology Mapping for Low Power in Logic Synthesis

Vivek Tiwari
Dept. of EE, Princeton Univ.
Princeton, NJ

Pranav Ashar
C&CRL, NEC USA
Princeton, NJ

Sharad Malik
Dept. of EE, Princeton Univ.
Princeton, NJ

Abstract

Traditionally, three metrics have been used to evaluate the quality of logic circuits – size, speed and testability. Consequently, synthesis techniques have strived to optimize for one or more of these metrics, resulting in a large body of research in optimal logic synthesis. As a consequence of this research, we have today very powerful techniques for synthesis targeting area and testability; and to a lesser extent, circuit speed. The last couple of years have seen the addition of another dimension in the evaluation of circuit quality – its power requirements. Low power circuits are emerging as an important application domain, and synthesis for low power is demanding attention.

The research presented in this paper addresses one aspect of low power synthesis. It focuses on the problem of mapping a technology independent circuit to a technology specific one, using gates from a given library, with power as the optimization metric. We believe that the difficulty in obtaining accurate models of power at the technology independent level makes it difficult to optimize for power at this level, and thus feel that the technology mapping step offers the most direct way of power optimization during logic synthesis.

Several issues in modeling and measuring circuit power, as well as algorithms for technology mapping for low power are presented here. Empirically it is observed that a significant variation in the power consumption is possible just by varying the choice of gates selected. In fact, our experiments over a large set of benchmark circuits show that compared to mapping for power, mapping for area or delay can lead to circuits that have significantly higher power consumption: up to 32% higher in case of mapping for area, and up to 153% higher in case of mapping for delay.

1 Introduction

There is a growing demand for low power circuits today. This demand comes from two distinct sources. The first of these is circuits that have significantly high power requirements, which necessitate expensive packaging and cooling. An example of this is the 21064 processor from DEC, implementing the Alpha architecture. This consumes approximately 25 Watts of power. The second, and probably more important, is portable applications for which a long battery life is desirable. A common example of this is portable notebook computers. Cellular phones are another example of a growing market of hand held applications that need low power circuits.

The need for low power circuits is being actively addressed on several fronts. At the micro-architecture level, several techniques have been presented that reduce the power requirements (e.g. [4, 3, 5, 17]). At the circuit level, a popular technique is to turn off the system clock for parts of the circuit that are not active. At the device level, work is being done to reduce the peak voltage needed for switching, thus directly resulting in reduced power consumption. Some work has recently been done in the area of low power logic synthesis (automatic design of logic) [12, 19, 20]. Several interesting ideas are presented in these works. However, one problem common to them is that they are applied to technology independent circuits, where gate models for power are very inaccurate. As a result it is difficult to predict if any of the reduction seen at this level will hold up after a final technology mapping step. Similar problems have been faced in logic synthesis in the past in the context of area and delay optimization. Partly as a consequence of this experience, the focus of our research in low power logic synthesis is the technology mapping stage where, we believe, it is possible to have reasonable power models for gates in the library.

A related topic, which has also been the focus of considerable research in the recent past [11, 18, 15, 7, 16], is one of accurately and efficiently measuring the power dissipated in a circuit. The first issue we examine in this paper is that of modeling the power requirements of gates. Traditionally the active area of gates has been considered to be proportional to the power. Given the fact that in CMOS, power is consumed only during gate switching, active area will track power only if all the gates switch equally often. There is no reason to believe this, and in fact it is easy to show otherwise. Section 2 considers a gate power model that incorporate the switching characteristics of the gate. Given a gate power model, the next step is to use this in measuring the power consumption of a given mapped circuit. Efficient power analysis is imperative since it will be needed in any algorithm that claims to optimize for power. An efficient power analysis technique akin to [11] is also presented in Section 2. The cost function for technology mapping for low power is derived from this power analysis technique.

The importance of considering power consumption during technology mapping has led to efforts that are in some measure parallel to ours. The preliminary results of our study were presented at a conference [23] where Tsui et al. [8] also presented their work. The extensions of their work are in a separate reference [9]. We have also extended our work since then, and the results are presented here. While the basic problem being considered by Tsui et al. [9] is the same as the one being considered here, there is a significant difference in the approach and the presentation of the research. The technology mapping algorithm used by Tsui et al. is adapted from an earlier work that examines the area-delay tradeoff in technology mapping [6], by replacing the area metric with power. Emphasis is placed on technology decomposition, and on considering the drain capacitance of internal nodes of a complex gate. The emphasis of our work is on fully understanding the nature of the problem of considering power during technology mapping. This is also reflected in the presentation of this paper. The mapping methodology that we use is different from the one used by Tsui et al.. It uses the area and delay based technology mapping used in SIS [22] as a starting point. It is instructive to understand the differences in traditional technology mapping and mapping for power, and we have illustrated these through examples. It is important to quantify the maximum potential of power reduction during technology mapping and experimental results for this are presented here. The interaction between the power metric and the other metrics, area and delay is of prime importance from the synthesis point of view. Due to its importance, we deal with the tradeoffs between power and area, and power and delay, individually and in great detail. The mapping problem is further studied through comparison between tree based and DAG based mapping and the effect of library size on the result of mapping.

The most successful techniques in technology mapping for the traditional metrics of area and delay have followed a similar paradigm. Exact and efficient algorithms, using dynamic programming, were first developed for tree circuits. These were then extended to take care of internal fanout. We follow a similar paradigm. We first present an exact algorithm for tree mapping. This algorithm is novel in as much as it has the flavor of both area and delay mapping. Power consumption, like area, is additive; and like delay, it depends on the load. For non-tree circuits, the mapping problem is NP-hard. Thus, recourse has to be taken to efficient heuristics that will work well for most practical instances. One of the techniques used in area and delay mapping has been the use of tree mapping for parts of the circuits that are trees and considering some matches across the fanout points. We adopt a similar strategy, with some variations in the heuristics. Tree mapping for low power, and its extensions to handle non-tree circuits is the subject of Section 4. We believe that the effectiveness of the mapping algorithm can be improved by an appropriate choice of the starting point for low power mapping. We present some ideas in that direction.

Experimental results shown in Section 5 on a large set of benchmark circuits indicate a potential for reduction in the power requirements by optimal technology mapping. Comparison of power mapping with area and delay mapping reveals that the latter can lead to circuits with significantly higher power consumption: up to 32% higher in case of mapping for area, and up to 153% in case of mapping for delay. The penalty incurred by power mapping in terms of area and delay and the tradeoff between power and delay is also explored in Section 5.

Finally, Section 6 summarizes the contributions of this paper and points out promising extensions of this work.

2 Gate-Level Power Dissipation Models

The need for an accurate model for power is imperative, given that our goal is to reduce this very metric. Since we are dealing with the circuit at the gate level, we need to have an abstraction for the power dissipation at the gate level. Using this abstraction we can quantitatively measure the power consumption in the circuit.

Power is consumed in a circuit element only when there is current flow. We are specifically dealing with CMOS circuits. In a CMOS gate, the most significant part of the power consumption occurs *only during transitions at the output.* Only when the output is charging/discharging is there a significant current flow from the output to the V_{dd} or ground terminals. There is a very small static or DC component of this current, which can be ignored. This means that power is only consumed for the charging and discharging of the capacitance at the output. In effect, the average power consumed by a CMOS gate is given by $\frac{1}{2} \times C_{output} \times V_{dd}^2 \times N \times f$, where C_{output} is the output capacitance of the gate, V_{dd} is the supply voltage, f is the frequency of the clock and N is the expected number of transitions at the output, per clock cycle.

Besides V_{dd} and f, which we assume are fixed, there are two different parameters in the above expression, C_{output} and N. Let us consider models for each.

2.1 Transition Probabilities

If we only consider transitions between *steady states* (i.e. we ignore the hazards or glitches), we can think of N as the probability of a transition occurring at the output of a gate during one clock cycle. This probability depends upon the Boolean function being computed and the the probabilities of the primary inputs.

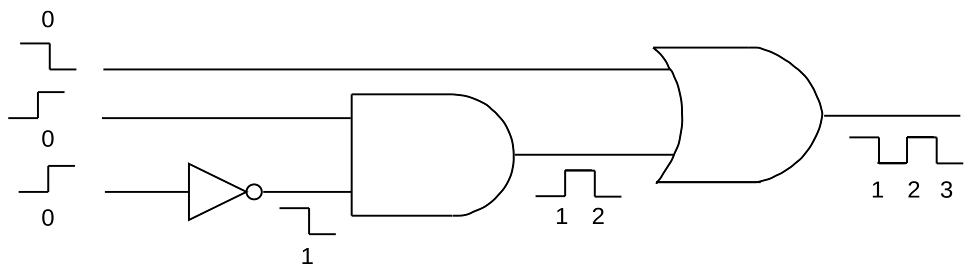


Figure 1: An Example of Glitching

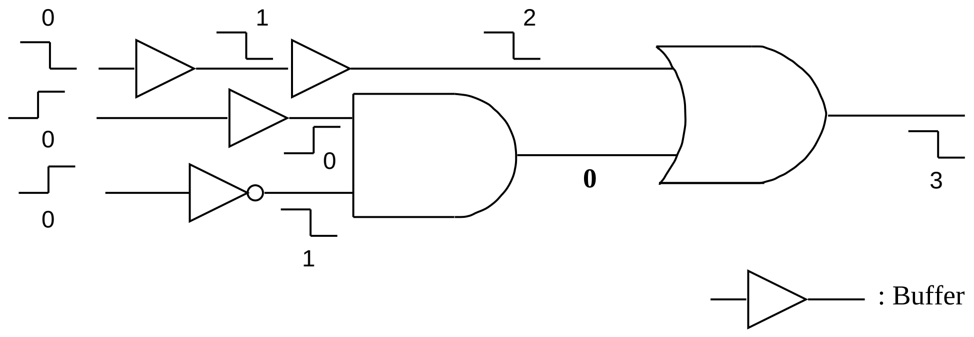


Figure 2: Path Balancing with Buffers

We define *signal probability* at the output of a gate as the probability of a logical 1 at the output. Let us denote it by p_g. Computation of p_g is best illustrated by an example. Suppose the minterms for which the function computed at the gate g evaluates to a 1 are $a_1a_2'a_3$, $a_1'a_2a_3$ and $a_1a_2a_3$. Further let p_{a_1}, p_{a_2} and p_{a_3} be the mutually independent signal probabilities of the primary inputs a_1, a_2 and a_3 respectively. Then the probability of the output of g being 1 in the steady state is given by: $p_g \;=\; p_{a_1}(1-p_{a_2})p_{a_3} \;+\; (1-p_{a_1})p_{a_2}p_{a_3} \;+\; p_{a_1}p_{a_2}p_{a_3}$

p_g is the probability of a 1 occurring at the output of the gate, once it has stabilized. Therefore, the probability of 0 being the stable value is $(1-p_g)$. Thus the probability of a change in the stable state of the output as a result of change in the primary inputs is $p_t \;=\; 2 \times p_g \times (1-p_g)$ corresponding to the sum of the probabilities of the changes from 1 to 0 and 0 to 1. This assumes that the consecutive input vectors to the circuit are uncorrelated.

p_t can also be viewed as the *signal transition probability*, *i.e.* the probability of a transition occurring at the output of the gate, if all the gates had zero delay. In that case we are only dealing with transitions between stable states and filtering out the *glitches*. Glitches have to be considered if we assume non-zero delays at gates. Figure 1 illustrates this. Assuming a delay of one for each gate we have six transitions in all. A zero delay model would yield no transitions at the output of the AND gate and only one at the OR gate.

This leads us to the conclusion that the total power consumed in a circuit is the sum of two components, a logical switching power P_l and a glitching power P_g. The logical switching power is the same as the total power in a zero delay model.

For the purposes of this paper, we consider the zero delay power to be a valid approximation of the total power. This is motivated by the following reasons:

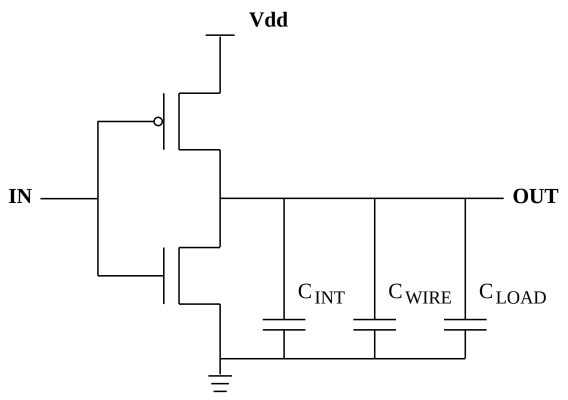


Figure 3: Model of a CMOS Inverter

- In practice, glitching power forms a relatively small component of the total power dissipation in most circuits [12]. This is especially true when one is interested in average rather than peak power dissipation.
- Glitching is difficult to model accurately. For example, if adjacent transitions occur sufficiently close in time, only a small hump occurs at the output instead of a complete swing to V_{dd} or ground, due to the inertial gate delay. In this case, what is logically a glitch consumes little power. Accurate low level information about physical gate delays is required to model this effect correctly.
- The glitches that occur in a non-zero delay model are a direct consequence of unequal path lengths from the primary inputs to the outputs of the gates. An appropriate insertion of buffers, possibly as a post-processing step, can mitigate this effect to some extent. For example, consider Figure 2. All gates including buffers have unit delays. The total number of transitions comes down to five. In fact, for this circuit there will be no glitches for any input transition if all inputs change at the same time.

Note that under the zero-delay approximation, the signal transition probability at a node in the network is dependent only on the Boolean function at the node and the transition probabilities at the primary inputs. The method used to derive the signal transition probabilities is described in Section 4.2.

2.2 The Output Capacitance

Refer to Figure 3 for the model of a CMOS gate. As shown in the figure, the capacitance at the output of a CMOS gate is the sum of three components C_{INT}, C_{WIRE}, C_{LOAD} [25]. C_{INT} represents the internal capacitance of the gate. This basically consists of the diffusion capacitance of the drain. C_{WIRE} represents the capacitance due to the physical interconnection, and C_{LOAD} represents the sum of gate capacitances of the transistors fed by the output. The values of C_{INT} and C_{LOAD} can be obtained from the data provided for each gate in the technology library. C_{WIRE} is routing dependent and is ignored at the gate level.

Once we have the capacitance values, the power consumed by a gate g, is given by $P_g = k \times (C_{INT} + C_{LOAD}) \times p_g$. Here p_g is the transition probability at the output of g and k is a technology dependent scaling factor. Using this model it is easy to compute the power consumed by a circuit once it has been mapped. This computation involves a straightforward traversal of the circuit, tallying up the power consumed at each gate.

3 Review of Technology Mapping for Area/Delay

Technology mapping involves the optimal implementation of a Boolean function using gates from a given library. It serves as the final step in the logic synthesis pipeline. In this section, we review the graph-covering based technology mapping algorithms for area and delay proposed in [13, 21, 24]. This background is helpful in understanding the algorithms in subsequent sections for technology mapping targeting low power.[1]

3.1 Formulation as DAG Covering

In its most general form, technology mapping can be formulated as a DAG[2] covering problem. The Boolean function to be mapped is represented in canonical form as a DAG (called the *subject DAG*) using a chosen basis function. Similarly, gates in the library are represented as DAGs (called *patterns*) using the same basis function. Each pattern has associated with it a *cost* based on the optimization criterion. Technology mapping is then the problem of finding an optimal cost covering of the nodes of the subject DAG using available patterns under the constraint that the inputs to a match must be available as the outputs of other matches.

Two criteria are involved in the choice of the basis function, namely, number of patterns and the optimization potential. The basis function consisting of two-input NAND/NOR gates and inverters has been found to be a good compromise among the various possible choices [21]. A heuristic effective in increasing the optimization potential has been the addition of a pair of cascaded inverters in the subject DAG, and the corresponding addition of a no-cost pattern consisting of a pair of cascaded inverters in the library [21].

It was shown in [21] that this form of DAG covering can be formulated as a column covering problem with each row of the covering matrix representing a node in the subject DAG, and each column representing a pattern match. A 1 in a matrix element signifies that the node associated with the row is covered by the pattern match associated with the column. The goal is to find a set of columns so that all rows are covered, the columns have minimal cost, and each match has its inputs available from the outputs of other matches. The problem is NP-hard for any basis function, and no heuristics are available to solve such a constrained column-covering problem effectively. Consequently, this formulation of DAG covering is not viable for circuits with more than a few nodes.

An alternative approach to technology mapping, proposed by Keutzer [13], is the tree-covering[3] approximation to DAG-covering. In this approach, the subject DAG is partitioned into a forest of trees, and the covering problem is solved on each of the trees. This approach is motivated by the existence of efficient dynamic programming algorithms for optimum tree covering [1]. While there is no unique way of partitioning the DAG into trees, usually the simplest solution of breaking the

[1]For an explanation of the terminology used in this section, the reader is referred to [21].

[2]Directed Acyclic Graph.

[3]A *tree* is a directed acyclic graph in which each node has a single fanout.

DAG at each fanout point is employed. A number of heuristics have been proposed [21] to reduce the effect of the initial partitioning of the subject DAG into trees on the optimality of the solution.

This basic approach of DAG partitioning prior to pattern matching has been found to be acceptable for cost functions based on circuit area, delay, and combinations thereof. In the next three sections, we review the tree covering algorithms at the heart of this approach to technology mapping for area and delay.

3.2 Tree Covering for Area

The key property that makes efficient tree covering algorithms possible is that given a match, m, at a node in the subject graph, the costs of the best matches at the inputs to m are independent of each other. It follows that the minimum cost match, m, at the root of a tree is the match that minimizes $area(m) + \sum_{v_i \in inputs(m)} min\_area(v_i)$, where the v_i are nodes in the subject graph input to the match m and $min\_area(v_i)$ is the cost of the minimum cost match at the node v_i. An algorithm to accomplish this would traverse the tree once from the leaves to the root, visiting each node exactly once. When a node is visited, all the matches at that node are enumerated and from among those the minimum cost match is stored. Since the cost function evaluation at each node is simple, the complexity of this approach is the cost of generating all the pattern matches at the nodes of the subject graph.

3.3 Tree Covering for Delay

The delay cost of a pattern is the propagation delay of the associated gate. To a first order, the propagation delay of a gate, g, is given by the formula $\alpha + \beta \times \gamma$, where α is the intrinsic (load independent) delay of g, β is the delay per unit load factor, and γ is load being driven by g. γ is dependent on the number, type, and size of gates being driven by g. In general, larger gates have higher intrinsic delay, but a lower value of the load dependent delay. The delay cost of a tree cover is the maximum propagation delay through any path from a leaf to the root.

While tree covering for delay is similar to tree covering for area in that the costs of the best matches at the inputs to a match, m, at a node in the subject graph are independent of each other, there is a key difference. The difference lies in the fact that the cost of a match cannot be determined until matches have been obtained for its fanout nodes, since that determines the load factor. Consequently, optimum tree covering for delay cannot be accomplished by a single pass through the tree. Two passes are required. In the first pass through the tree from the leaves to the root, each node in the subject graph is visited once and the minimum cost match at the node is stored *for each possible load value* (the cost of a match is the maximum delay from any leaf to the output of the match). In the second pass from the root to the leaves, the best match is picked for each node visited based on the matches stored in the first pass.

An approximate solution to this two pass procedure was proposed in [21]. Rather than store the best possible solution at a node for every possible load value, it was suggested that load values be discretized and that the actual load values encountered during tree covering be mapped to their nearest discrete values. In later work [24], it was realized that such discretization was not necessary because the same match at a node results in the minimum cost for an *interval* of load values rather than a single load value. It follows that it is only necessary to store a piecewise function at every node in order to store the minimum cost solution for each load value. Each section of the piecewise function corresponds to the minimum of the cost functions in the corresponding range of load values for each of the matches at the node. Since the cost function for each match is a linear function of the load, the piecewise function is piecewise linear.

3.4 Tree Covering for Area Under a Delay Constraint

The approach to this problem has the same flavor as tree covering for delay. The delay constraint is specified as a required time at the root of the tree. The required time cannot be propagated towards the leaves unless the load being driven by each node in the subject graph is known. Therefore a two pass method is again required. In the first pass through the tree from the leaves to the root, each node is visited once and the minimum area solution for that node is stored for each possible combination of required time and load value at that node. In the second pass from the root to the leaves, the best match (the minimum-area match corresponding to the actual load value, that satisfies the required time) is picked for each node visited. As soon as a match is picked, the required time can be propagated to the inputs of the match. Note while this approach is theoretically optimal, it has a prohibitively high computational expense. Therefore discretization of required times and load values is used to reduce the number of solutions that need to be stored. Details can be obtained from [21].

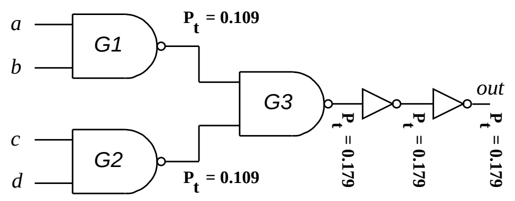
4 Technology Mapping Algorithm for Low Power

The overall flow of our algorithm for technology mapping for low power follows along the lines described in the previous sections for technology mapping for area and delay. Specifically, a canonical representation (called the subject DAG) is created for the Boolean function to be mapped using a basis function consisting of two-input NAND/NOR gates and inverters. Canonical representations (called patterns) are also obtained for each of the gates in the library using the same basis function. The technology mapping problem is then formulated as one of optimally covering the nodes in the subject DAG with patterns so that the cost function modeling power dissipation is minimized under the constraint that the inputs to a match are available as the outputs of other matches.

4.1 Relationship to Technology Mapping for Area/Delay

We now give an intuitive feel for the key differences and similarities between the cost function for power and cost functions for area and delay during technology mapping. As described in Section 2, the average power dissipated at a gate is proportional to $C_g \times P_t$, where C_g is the total capacitance at the gate output and P_t is the probability of a transition at the gate output. This clearly points to a basic difference that the cost of a match is dependent on the Boolean function at the output of a node in the subject graph unlike in the case of technology mapping for area and delay.

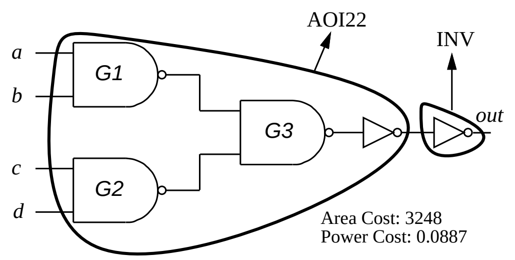
Consider the circuit in Figure 4(a) as an example where the difference between mapping for area and power is manifested. The signal transition probabilities at the outputs of gates G_1, G_2, and G_3 are 0.109, 0.109, and 0.179, respectively. Figure 4(b) shows a sample library with four gates. The areas and capacitances in standard units associated with these gates are also shown. The minimum-area mapping for this circuit is shown in Figure 4(c), and minimum-power mapping in Figure 4(d). The area and power cost of each mapping is shown in the corresponding figure. The area mapped circuit has lesser area than the power mapped circuit but it has 22% higher power consumption. To understand this we need to look at the values of the transition probabilities and



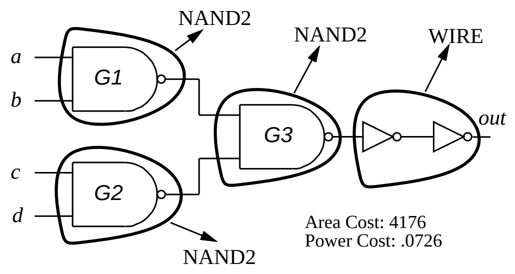
(a) Circut to be mapped

Gate Type	Area	Intrinsic Cap.	Input Load Cap.
INV	928	0.1029	0.0514
NAND2	1392	0.1421	0.0747
NAND3	1856	0.1768	0.0868
AOI22	2320	0.3410	0.1033

(b) Characteristics of the Library

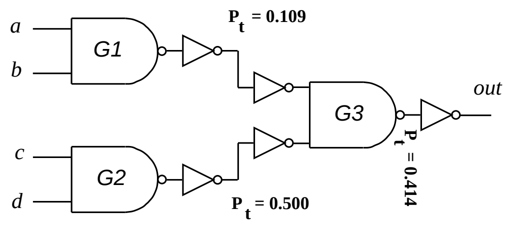


(c) Minimum−Area Mapping



(d) Minimum−Power Mapping

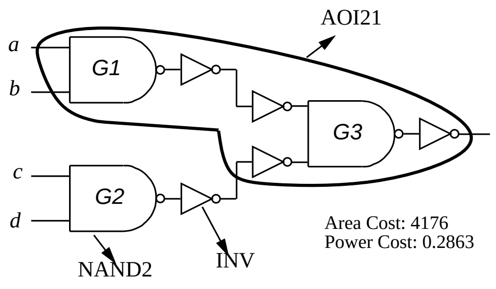
Figure 4: Example Illustrating the Difference Between Technology Mapping for Power and Area



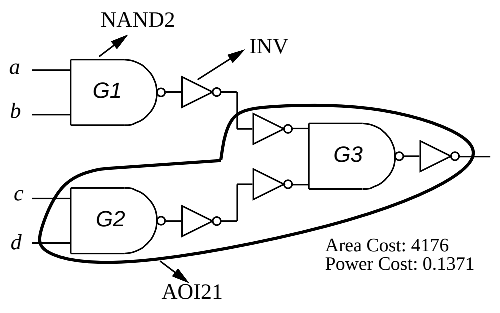
(a) Circut to be mapped

Gate Type	Area	Intrinsic Cap.	Input Load Cap.
INV	928	0.1029	0.0514
NAND2	1392	0.1421	0.0747
NAND3	1856	0.1768	0.0868
AOI21	1856	0.2310	0.0850

(b) Characteristics of the Library



(c) Minimum−Area Mapping



(d) Minimum−Power Mapping

Figure 5: Another Example Illustrating the Difference Between Technology Mapping for Power and Area

the capacitive loads at the output of gates. In the case of area mapping, the AOI21 , a large gate with a large capacitance, is driving the node which has the higher transition probability 0.179. In the case of power mapping, this high transition probability point is driven by a much smaller gate, a NAND2 . Thus even though the power mapped circuit has more gates, its overall average power consumption is lower.

Another mechanism that can lead to differences in power costs is illustrated in Figure 5. In this example, the area and power mapped circuits have the same area, but the area mapped circuit has a much higher power cost. The reason is that, in the case of the area mapped circuit, the NAND2 and NOT gates are driving nodes with have a high transition probability value, 0.5. The nodes with the lower transition probability value, 0.109, are hidden within the AOI21 gate. Exactly the opposite happens in the case of power mapping. The high transition probability nodes are hidden within the AOI21 gate, while the nodes with the lower transition probability are exposed at the outputs of the NAND2 and NOT gates. It should be noted that only those nodes of the subject graph that get mapped to outputs of the CMOS complex gates, contribute to the overall power consumption.[4]

The similarity between technology mapping for power and delay is that the cost of a match is, in part, dependent on the load driven by the match. In the most general case of the use of a non-zero-delay model in the computation of signal transition probabilities, (equivalent to considering power dissipation due to glitching), the best match at a node cannot be determined until the matches at the fanout nodes and the matches at their transitive fanins have been chosen. In the case of the zero-delay model which we use in this paper, the signal transition probability at the output of a node is independent of the match chosen for that node (*cf* Section 2). Therefore, the power dissipated in the load capacitance is the same for all matches at the node. Consequently, the best match at a node can be determined before the matches at the fanout nodes have been chosen as in the case of technology mapping for area. Even so, the actual cost of the best match at a node can only be determined once the matches at the fanout nodes are known. This point will be elaborated in subsequent sections.

Finally, as in the case of technology mapping for area, and unlike technology mapping for delay, fanout optimization is a non-issue in technology mapping for low power under the zero-delay model. Essentially, there is no notion of slack which can be distributed between the fanout branches of a node. An interesting observation is that if a non-zero delay model was used to calculate the signal probabilities, the insertion of the fanout tree might be beneficial since it can be used to balance the path lengths, and thereby reduce glitching. Therefore, fanout optimization could play a role in the technology mapping for power once glitching power is incorporated into the model.

4.2 Computing Signal Transition Probabilities

In the zero-delay model used in this paper, the signal transition probability at the output of a node is dependent only on the Boolean function at the node output in terms of the primary inputs. The nature of implementation of the logic feeding the node does not affect the signal transition probability. Therefore, the signal transition probabilities at the node outputs in the subject graph only need to be computed once in the beginning. We use the same approach for computing signal transition probabilities as proposed in [11, 18]. The probabilities are computed

[4]Certain CMOS standard cell libraries have multilevel gates, e.g., non-inverted gates like AND and OR . These consist of more than one CMOS complex gate. The power model described earlier has to be applied to each of the component gates to get the overall power cost.

under the assumption that the primary inputs are mutually and temporally uncorrelated. This is typically true for datapath circuits, but may not be true for control circuits. While these simplifying assumptions lead to a more efficient implementation, the technology mapping algorithms described here do not depend on these assumptions. During calculation of the signal transition probabilities at node outputs, the mutual correlation between primary inputs can be accounted for by considering conditional probabilities between all subsets of inputs. An efficient approximation of this is the use of pairwise conditional probabilities [10]. The temporal correlation at the primary inputs, i.e., correlation between the values at each input across clock cycles, can be dealt with by directly using transition probabilities at primary inputs for calculating the transition probabilities at node outputs.

Given the above assumptions, however, the signal transition probabilities are computed as follows. The probability that a signal makes the transition from 1 to 0 ($p_{1\to 0}$) or from 0 to 1 ($p_{0\to 1}$), is given by $p_1 \times (1 - p_1)$, or equivalently by $p_0 \times (1 - p_0)$. p_0 and p_1 are the probabilities that the signal takes the value 0 and 1, respectively. p_0 (p_1) can be computed efficiently by a traversal of the Binary Decision Diagram [2] constructed for the Boolean function of the node. For the standard benchmark circuits, the signal probability values at primary inputs are not specified. Thus, for the purpose of our experiments, these are assumed to be 0.5. It should be noted that the same algorithm can also be applied to compute the signal transition probabilities when the signal probabilities of the primary inputs are different from 0.5.

4.3 Tree Covering for Low Power

As in the case of area and delay, the DAG covering problem is approximated by posing it as a composition of a number of tree covering problems by first partitioning the subject DAG into a forest of trees. As in the case of technology mapping for area and delay, this approach is motivated by the existence of efficient tree covering algorithms for low power. The algorithm for optimum tree covering targeting low power is described below.

The key property that makes efficient tree covering algorithms for area/delay possible holds in the case of tree covering for power also. Given a match, m, at a node in the subject graph, the power costs of the best matches at the inputs to m are independent of each other. It follows that the match, m, with the minimum power cost at the root of a tree is the match that minimizes $power(m) + \sum_{v_i \in inputs(m)} min\_power(v_i)$, where the v_i are nodes in the subject graph input to the match m and $min\_power(v_i)$ is the cost of the minimum-cost match at the node v_i. $power(m)$ is proportional to $p_t \times (C_{INT} + C_{LOAD})$, where p_t is the signal transition probability at the node, C_{INT} is the intrinsic capacitance of the match, and C_{LOAD} is the load capacitance seen by the match. Since p_t is the same for all matches, the $p_t \times C_{LOAD}$ component is the same for all matches at the node. Therefore, the best match at the node can be obtained without knowing the value of C_{LOAD}. The actual cost (which includes the $p_t \times C_{LOAD}$ term) of the best match at a node is needed in order to determine the best match at the fanout of the node. This can be computed once the match at the fanout node is known.

An algorithm for tree covering for low power would traverse the tree once from the leaves to the root, visiting each node exactly once. When a node is visited, all the matches at that node are enumerated and from among those, the match with the minimum power cost is stored. As in the case of tree covering for area and delay, the complexity of this approach is the cost of generating all the pattern matches at the nodes of the subject graph since the cost function evaluation at each node is simple.

The pseudo-code for the tree-covering algorithm is given below. The argument to it is the root of the tree to be covered.

```
optimal-power-cover(node)
{
    if (node has been visited before) {
        /* Optimal cover for node is known. */
        return optimal cover for node ;
    }

    /* Find the optimal cover at node */
    min_cost = $\infty$ ;
    foreach match at node {
        cost = cost of match ;
        foreach input to match {
            optimal-power-cover(input) ;
            cost = cost + cost of optimal cover at input ;
        }
        if (cost < min_cost) {
            min_cost = cost ; node_match = match
        }
    }
    return optimal cover for node ;
}
```

The following theorem, stated here without proof, provides assurance regarding the optimality of the tree cover:

Theorem 4.1 *The procedure* **optimal-power-cover**(*node*) *is guaranteed to find the tree cover with minimum power dissipation for the tree rooted at node.*

4.4 Tree Covering for Low Power Under a Delay Constraint

The algorithm for tree covering for low power (for the zero-delay model) under a delay constraint proceeds in much the same manner as tree covering for area under a delay constraint. The delay constraint is specified as a required time at the root of the tree. Since the required time cannot be propagated towards the leaves unless the load being driven by each node in the subject graph is known, the solutions with minimum power cost for each possible combination of required time and load are stored at every node during the first pass through the tree from the leaves to the root. In the second pass from the root to the leaves, the best match (the minimum-power match corresponding to the actual load value that satisfies the required time), is picked for each node visited. As soon as a match is picked, the required time can be propagated to the inputs of the match. Discretization of required times and load values is used to reduce the number of solutions that need to be stored, as in the case of tree covering for area under a delay constraint.

4.5 DAG Covering for Low Power

Most real life circuits have nodes with multiple fanout, and are therefore not trees. For non-tree circuits, the technology mapping problem becomes NP-hard. Intuitively, the complexity increases because the matches at the inputs to a match are, in general, no longer independent of each other. For instance, if a match requires that a multiple-fanout node be internal to it, then the logic for the multiple-fanout node must be duplicated if all its fanout points are not contained in the match.

As stated in Section 3, a good approximation to DAG covering is to pose it as a composition of a large number of tree covering problems. This approximation has proved effective for area and delay optimization in the past. In this section, we give a flavor for the type of problems encountered in DAG covering for low power. Some heuristics to handle these problems are also presented. First, we revisit some heuristics for DAG covering for area and delay.

Partitioning the subject DAG into a forest of trees by breaking it up at each fanout point is in itself a simple, but effective, heuristic. In the actual implementation, it is not necessary to partition up the subject graph explicitly. The same effect can be obtained by ensuring that only tree patterns are used for library gates, and that no matches be allowed to span fanout points. This way, the algorithms from the previous sections can be applied directly to the subject graph. We just have to ensure that the cost of each multiple-fanout node is initialized to zero and that each node of the subject graph is visited only once. This approach has some problems. Firstly it allows no matches across fanout points, and thus no tree overlap is allowed. Usually, overlapping of trees can give better results for the three metrics, area, delay, and power. Using the strictly tree based approach means that this part of the search space will never be explored. Figures 6(a) and (b) illustrate this. If tree overlap is not allowed, the solution shown in Figure 6(a) would never be seen even though it is potentially better than a solution like Figure 6(b), which the algorithm would see.

Another problem is related to phase assignment. The root and the leaves of a tree have a preferred phase depending on the structure of the tree. While mapping a tree, these preferred phases are fixed and cannot be changed. If we enforce tree boundaries while covering the subject DAG, we are limited by the preferred phases since they are fixed. However, the best solution may actually involve exchanging phases between trees at multiple-fanout points. One way to solve this problem is to postpone it until after technology mapping. A global phase optimization algorithm may be then run on the graph. As an alternative, a cross-tree-phase heuristic is described in [21] to solve this problem. It has been implemented as an option in the technology mapping package in SIS [22].

An alternative approach, which can partly alleviate both the previous problems is to not restrict the algorithm to trees. The library is allowed to have non-tree patterns, and the subject graph can be a general DAG. Starting from the primary inputs, subject DAG nodes are traversed in a depth first[5] manner. All patterns (including ones which span multiple-fanout nodes) match at a node are enumerated. Like in tree mapping, the minimum-cost match is stored for a node.

The issues to be addressed now are the evaluation of the cost of matches that are rooted at, or go across, multiple-fanout points and the propagation of costs beyond such points. To understand the motivation for the latter problem, which occurs in the case of area, consider Figure 6(c). The area cost of the graph rooted at node C is C_C, the cost of the match at A is C_A and the cost of the match at B is C_B. Here the cost of the match at node C has to be propagated to both nodes A and B without it being considered twice. A reasonable heuristic is to propagate half the cost to

[5]By depth first we mean that a node is seen only after all the nodes in its fanin have been seen.

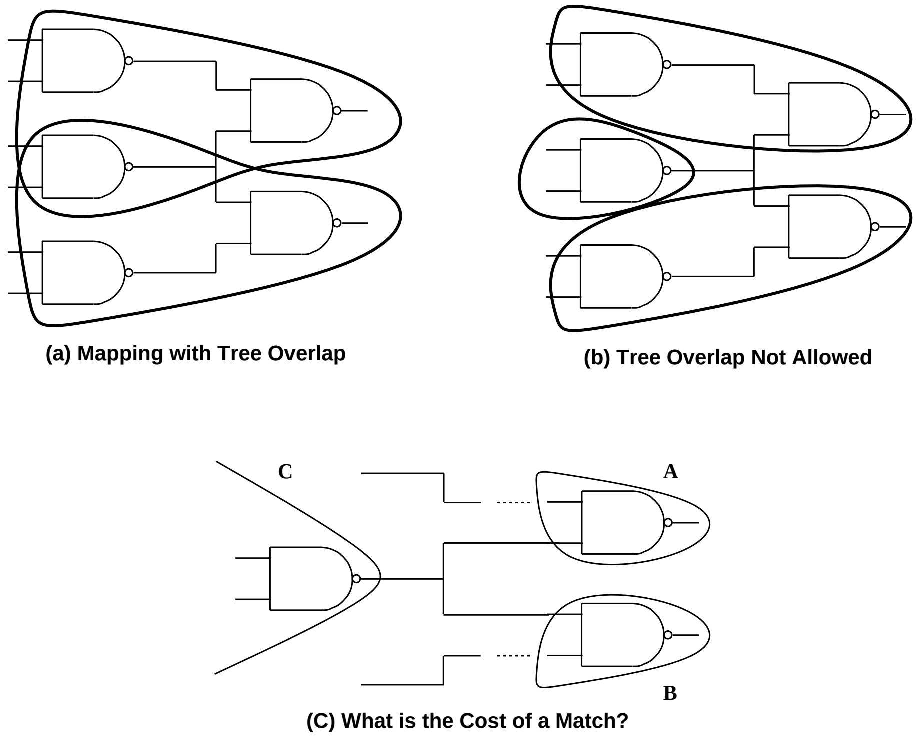

(a) Mapping with Tree Overlap
(b) Tree Overlap Not Allowed
C
A
B
(C) What is the Cost of a Match?

Figure 6: Effect of Multiple-Fanout Points

both A and B. Thus, the cost of the match rooted at A is taken to be $C_A + C_C/2$, and the cost of the match rooted at B to be $C_B + C_C/2$. This heuristic is known to give good results in most cases.

When the optimization metric is delay, evaluation of the cost of matches that are fed by multiple-fanout points is a problem, as illustrated by Figure 6(c). Suppose we are trying to evaluate matches at node A, and that B has not been seen yet. The arrival time at node C cannot be known until node B is mapped. However, the arrival time at node C is needed right away to evaluate the cost of matches at node A. Thus, A cannot be mapped before B, and conversely, B cannot be mapped before A. Touati [24] discusses this problem and proposes several heuristics to handle it.

Another problem is that if tree overlaps are allowed, the number and location of multiple-fanout points can be modified arbitrarily by the covering algorithm. Touati [24] recognizes this effect but ignores it.

General DAG covering for low power faces all of the above problems because of the nature of the cost measure used. Since the power cost at a node is in part the summation of the power costs of its input nodes, the problem faced by DAG covering for area is faced here also. In addition, since the cost at a node depends on the load values of the fanouts, the problems faced in mapping for delay are faced here also. The experience with generalized area and delay mapping, however, suggests that looking across tree boundaries can be beneficial. Therefore, generalized DAG mapping was implemented for power in spite of the potential problems. To handle multiple-fanouts, several heuristics, some of them similar to the ones used in area and delay were tried out. For instance, one heuristic was to divide the power at a multiple-fanout node by the number of fanouts in order

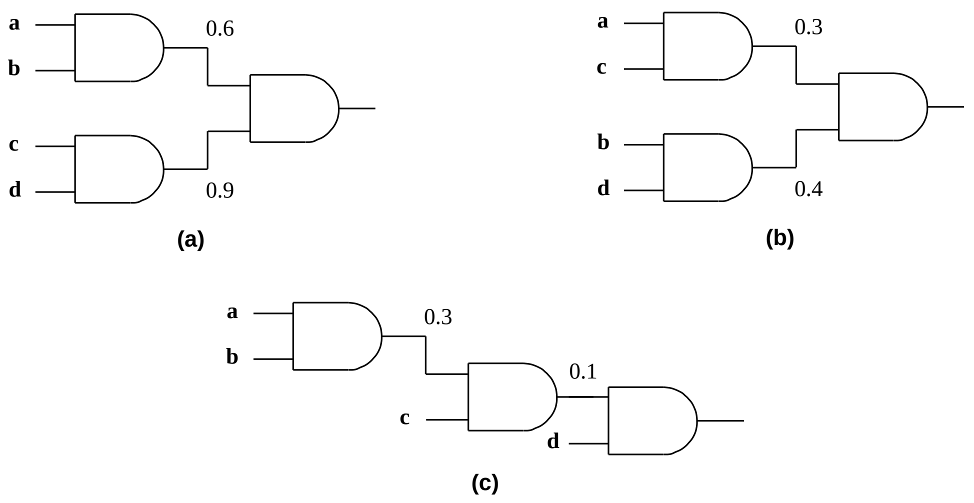


Figure 7: Effect of Decomposition on Transition Probabilities

to propagate the cost to the fanout nodes. Another heuristic initialized the cost contribution of the inputs of a match to be zero, if the match spanned across a fanout point. In the heuristics that used a non-zero cost at a multiple fanout point, a default value was used for the load at the fanouts that had not yet been mapped. The default used depends on the load values in the library.

The effectiveness of the DAG covering approach will be evaluated in Section 5.2.

4.6 Subject DAG Generation

The subject DAG corresponding to a technology independent circuit is not unique. Several different "technology decompositions" exist that result in internal signals having differing transition probabilities. Consider Figure 7, which shows three different decompositions into two input AND gates of the same function. The numbers on the internal signals are the transition probabilities. We see that even with the same topology (Figures 7(a) and 7(b)), assignments of variables to pins can make a difference in the transition probabilities. The range of possibilities is further enlarged by several different topologies being possible (For example, Figure 7(c) has a different topology than Figures 7(a) and (b)). Thus, it is possible that the technology decomposition step can lead to a significant difference in the power cost of the final mapped circuit. We are exploring this issue further. This problem is also investigated in [8].

5 Experimental Results

The algorithms and heuristics described in the previous sections have been implemented and integrated with the technology mapping package in SIS [22]. Several experiments were conducted on circuits from the MCNC and ISCAS benchmark suites. These experiments can be classified into two types. Those dealing strictly with tree covering, and those dealing with the generalized DAG covering. The experiments were carried out on a SUN SPARC 2 platform. Except for the additional time required to form BDDs and compute signal transition probabilities, the CPU time requirements were of the same order as for technology mapping for area.

Circuits	Min Power	Max Power		Area Power	
		Power	% Higher	Power	% Higher
5xp1	10.15	15.80	55.66	11.01	8.43
5xp1-hdl	5.39	8.42	56.14	6.07	12.61
9sym	16.20	27.87	72.07	17.79	9.84
9symml	13.53	25.96	91.83	15.05	11.20
9sym-hdl	6.65	10.13	52.38	7.72	16.17
alu2	19.61	35.21	79.56	22.78	16.19
alu4	34.53	58.11	68.30	40.26	16.60
alupla	8.30	16.31	96.45	9.31	12.08
apex6	43.19	80.14	85.56	49.92	15.60
apex7	14.50	29.26	101.78	17.09	17.85
b9	8.10	17.19	112.17	9.02	11.38
bw	17.12	30.18	76.29	18.97	10.85
c8	11.90	22.46	88.72	13.76	15.61
con1	1.24	2.74	120.68	1.34	7.48
duke2	36.31	75.25	107.23	41.77	15.03
f2	1.71	2.19	27.54	1.87	9.10
f51m	9.94	14.36	44.52	10.74	8.09
f51m-hdl	5.29	8.31	57.06	6.27	18.51
frg2	71.51	171.56	139.90	88.48	23.73
misex1	5.43	9.87	81.81	5.93	9.23
misex2	6.75	18.44	173.43	7.61	12.84
misex3	48.12	100.85	109.58	53.10	10.34
misex3c	38.76	77.85	100.86	42.98	10.90
pair	95.81	183.50	91.52	109.04	13.80
rd53	3.62	6.25	72.79	4.11	13.54
rd53-hdl	2.71	4.41	62.98	3.15	16.40
rd73	11.04	21.78	97.35	12.01	8.85
rd73-hdl	4.38	7.10	62.21	5.06	15.70
rd84	22.04	38.31	73.81	24.41	10.75
rd84-hdl	5.41	8.44	55.95	6.30	16.51
rot	42.38	83.38	96.76	47.75	12.67
sao2	9.59	19.48	103.24	10.57	10.30
sao2-hdl	13.54	24.49	80.81	15.37	13.50
vda	31.50	96.42	206.14	35.40	12.39
vg2	12.81	26.58	107.46	15.02	17.19
x1	22.78	50.62	122.25	25.33	11.22
x3	61.05	124.97	104.70	65.34	7.03
x4	32.25	71.47	121.61	36.70	13.80
z4ml	4.74	7.23	52.52	5.11	7.82
z4ml-hdl	3.25	5.49	69.07	3.77	16.10
C1355	31.31	47.08	50.33	35.13	12.17
C17	0.59	1.43	141.62	0.93	56.51
C1908	35.07	61.68	75.89	40.73	16.15
C432	12.56	29.11	131.70	15.30	21.80
C499	27.91	50.35	80.41	32.50	16.45
Average			90.24		14.23

Table 1: Results of Technology Mapping Using Pure Tree Mapping

The library used was `lib2.genlib`, a large, relatively accurately calibrated library with a wide variety of gates, distributed along with SIS. The libraries that are available do not specify the internal capacitance of the gates explicitly. We estimated the capacitances associated with a gate from the α, β, and γ provided for each gate in `lib2.genlib`.[6] If explicit capacitance values were supplied with the libraries, they could be used directly.

5.1 Tree Covering Results

The first experiment we carried out was designed to estimate the optimization potential of technology mapping for low power. The benchmark circuits were first mapped for minimum power, and then for maximum power. The variation in the power consumption in these two cases provides a measure of the flexibility in the circuit structure towards the power metric.

For the results of this experiment to be useful, the experiment had to be conducted under a rigid and well understood framework. Since DAG covering heuristics have varying behavior, strict tree covering was used to map the circuits. Strict tree covering (*cf* Section 4.5) means that the circuit is conceptually divided into trees at each multiple fanout point. Each tree is then mapped optimally using the tree covering algorithm described in Section 4.3. Care has to be taken that the trees are isolated from each other and do not influence each other. The inverter-pair heuristic (*cf* Section 3) was used for each tree for minimum power. Since the insertion of extra inverters is not fair for maximum power, the subject graph used for maximum power was the initial NAND decomposition of the circuit with no insertion of redundant inverters.

The results for maximum and minimum power are tabulated in the second and third column of Table 1. The fourth column shows the percent by which the maximum power is higher than the minimum power. The results show an average difference of around 90%, with individual differences going as high as 206%. The results indicate that power variations during technology mapping are significant, and thus, this justifies further exploration of power minimization during this phase of the design process.

Traditionally area has been used as a rough, first order estimate of the power consumption. To determine the validity of this assumption during technology mapping, the next phase of the experiment involved comparing the minimum power results with the power dissipation observed in circuits mapped for minimum area. For a fair comparison, strict tree mapping was used for area minimization also. These results are shown in the fifth column of Table 1. The last column shows the percent by which the power consumption of area optimized circuits is greater than that for the power optimized circuits. The results show an average difference of about 14%.

5.2 DAG Covering Results

The results of technology mapping using DAG covering are shown in Table 2. The second column shows the minimum power obtained using a variety of heuristics, some of which are mentioned in Section 4.5. The third column shows the power for the circuits that have been mapped for the

[6]Given a gate g, α is the intrinsic (load independent) delay of g, β is the delay per unit load factor, and γ is load being driven by g. The intrinsic capacitance of g was estimated as being proportional to α/β, and the input capacitance was estimated as being proportional to γ. This is justified as α has the dimensions RC and β has the dimensions R. Thus, their ratio has the dimensions of capacitance. Since α is the intrinsic delay of the gate and β is also internal to the gate, their ratio can be looked upon as the intrinsic capacitance of the gate. The power values given in the result tables are meant for relative comparison rather than as absolute measures of power.

Circuit	**Power Comparison**			**Area Comparison**		
	Min Power	**Area Power**	**% Higher**	**Power Area**	**Area Area**	**% Lower**
5xp1	8.559	9.70	13.30	145696	135488	7.01
5xp1-hdl	4.771	5.32	11.44	73776	71456	3.14
9sym	14.068	15.42	9.58	234320	199520	14.85
9symml	11.848	13.28	12.07	210656	187456	11.01
9sym-hdl	5.937	6.18	4.09	123424	115072	6.77
alu2	17.348	19.89	14.66	452864	376768	16.80
alu4	31.326	38.13	21.73	824528	722912	12.32
alupla	7.118	7.84	10.16	134096	122496	8.65
apex6	35.949	45.55	26.72	908512	717344	21.04
apex7	13.060	15.28	17.03	299744	253808	15.33
b9	7.371	8.38	13.68	139200	131312	5.67
bw	15.037	16.14	7.37	264944	226896	14.36
c8	10.675	11.82	10.70	206480	176784	14.38
con1	1.135	1.50	32.16	22272	20416	8.33
duke2	30.764	35.38	15.00	703888	600880	14.63
f2	1.527	1.77	15.72	24128	24128	0.00
f51m	8.429	8.96	6.25	141984	129920	8.50
f51m-hdl	4.663	5.08	8.96	71920	67744	5.81
frg2	63.402	75.91	19.73	1623536	1361376	16.15
misex1	4.708	5.32	13.02	83520	72848	12.78
misex2	5.587	6.78	21.42	162400	138736	14.57
misex3	41.510	46.37	11.70	972544	846800	12.93
misex3c	33.645	37.66	11.94	697856	589280	15.56
pair	86.235	96.60	12.02	1932096	1629568	15.66
sao2	8.334	9.39	12.63	161936	153120	5.44
sao2-hdl	12.052	12.89	6.92	257520	238960	7.21
rd53	3.130	3.35	6.90	50112	44544	11.11
rd53-hdl	2.446	2.45	0.16	44080	40368	8.42
rd73	9.486	10.42	9.81	178176	154976	13.02
rd73-hdl	4.098	4.12	0.51	78416	68208	13.02
rd84	19.696	21.22	7.73	381872	313664	17.86
rd84-hdl	4.917	4.89	-0.63	103472	90480	12.56
rot	37.667	42.48	12.78	762816	665840	12.71
vda	28.614	31.86	11.35	1325184	1147008	13.45
vg2	10.483	12.22	16.55	261696	193952	25.89
x1	19.790	23.74	19.98	411568	362848	11.84
x3	55.053	61.45	11.62	1071376	922432	13.90
x4	27.412	32.71	19.33	548448	541952	1.18
z4ml	4.108	4.57	11.25	64496	67280	-4.32
z4ml-hdl	2.901	3.09	6.69	50112	45936	8.33
C1355	27.498	28.26	2.77	735904	713632	3.03
C17	0.466	0.47	0.00	8352	8352	0.00
C1908	29.806	31.43	5.47	840304	610160	27.39
C432	11.210	12.69	13.18	239888	234784	2.13
C499	23.336	26.36	12.98	403680	458432	-13.56
Average			11.74			10.37

Table 2: Results of Comparison Between Technology Mapping for Power and for Area

minimum area using the SIS technology mapper. The power cost of the circuits optimized for area is, on the average, higher than the power of the power optimized circuits by more than 11%, with individual cases showing a difference of up to 32%. Note that for one case (`rd84-hdl`), the power of the area optimized circuit is slightly lower than the power of the circuit optimized for power. This shows the non-optimality of the heuristics, and suggests that there is scope for further improvement.

The comparison between the areas of the power optimized circuits and the area optimized circuits is also useful. It is shown in the last three columns of Table 2. Compared to the power mapped circuits, the area mapped circuits have an on the average about 10% lesser area. Since the power mapped circuits are seen to be sub-optimal with respect to area, it shows that a certain area penalty is paid to achieve the reduction in power. Again it is interesting to see that in some cases the area of the power optimized circuits is lower than the area of the area optimized circuits, pointing to the non-optimality of the generalized DAG covering for area as well.

We judged the effectiveness of the DAG covering heuristics for power minimization by comparing the results obtained with the results for strict tree covering as illustrated in Table 3. Strict tree covering was worse, leading to circuits which had on the average 14% higher power consumption.

5.3 Comparison with Delay Mapping

Besides area, the other traditional metric considered during technology mapping is delay. Experiments were performed to compare delay mapping with power mapping. The results are shown in Table 4. The second column shows minimum power obtained using DAG covering and third column shows the power cost of circuits obtained using the minimum delay option in the SIS technology mapper.[7] It is seen that delay optimized circuits consume much more power than the power optimized circuits. Their power cost is on the average, higher by more than 58%, with individual cases showing a difference of up to 153%. But the delay optimized circuits are faster than the power optimized circuits as shown in Columns 5 and 6. On the average they have about 30% lower delay.

The large disparity between power and delay mapping has to do with their conflicting demands for parameters of the gates chosen. Delay mapping tends to use gates with larger drive capability, that is a lower value of β. However larger drive capability implies larger capacitance and thus a higher power cost. On the other hand, during power mapping, gates with lower capacitance are preferred. But since these gates have lower drive capability, a penalty in terms of greater delay is incurred.

The results in Table 4 indicate that a huge tradeoff space is available between power and delay. One possible method to span this tradeoff space is to do technology mapping for power under a delay constraint as described in Section 4.4. The results for this are shown in Table 5. This method is expensive in terms of memory requirements and execution time. Due to limited computational resources we performed this experiment on a subset of the benchmark circuits. Columns 3 and 6 shown the power and delay, respectively, for minimum delay mapping while Columns 2 and 5 show the same for power mapping under delay constraint. The delay constraint chosen for each circuit was the delay of the delay mapped circuit, i.e., the values in column 6. For these circuits, minimum delay mapping had 83% higher power consumption and about 25% lower delay as compared to unconstrained minimum power mapping. Compared to delay constrained power mapping, minimum delay mapping still has a much higher power consumption, about 61%. But

[7]For minimum delay, two options were tried, -f3 -n1.0 and -f0 -n1.0. The circuit with lesser delay was chosen.

Circuit	DAG Power	Tree Power	% Higher
5xp1	8.56	10.15	18.62
5xp1-hdl	4.77	5.39	13.02
9sym	14.07	16.20	15.13
9symml	11.85	13.53	14.20
9sym-hdl	5.94	6.65	11.98
alu2	17.35	19.61	13.04
alu4	31.33	34.53	10.22
alupla	7.12	8.30	16.63
apex6	35.95	43.19	20.13
apex7	13.06	14.50	11.05
b9	7.37	8.10	9.92
bw	15.04	17.12	13.84
c8	10.68	11.90	11.48
con1	1.14	1.24	9.52
duke2	30.76	36.31	18.04
f2	1.52	1.71	12.47
f51m	8.43	9.94	17.89
f51m-hdl	4.66	5.29	13.42
frg2	63.40	71.51	12.79
misex1	4.71	5.43	15.25
misex2	5.59	6.74	20.73
misex3	41.51	48.12	15.93
misex3c	33.65	38.76	15.19
pair	86.23	95.81	11.11
rd53	3.26	3.62	11.04
rd53-hdl	2.45	2.71	10.67
rd73	9.49	11.04	16.35
rd73-hdl	4.10	4.34	6.81
rd84	19.70	22.04	11.90
rd84-hdl	4.92	5.41	10.03
rot	37.67	42.38	12.51
sao2	8.33	9.59	15.02
sao2-hdl	12.05	13.54	12.39
vda	28.61	31.50	10.08
vg2	10.48	12.81	22.23
x1	19.79	22.78	15.09
x3	55.05	61.05	10.89
x4	27.41	32.25	17.64
z4ml	4.11	4.74	15.46
z4ml-hdl	2.90	3.25	12.00
C1355	27.50	31.31	13.88
C17	0.47	0.59	26.82
C1908	29.81	35.07	17.65
C432	11.21	12.56	12.07
C499	23.34	27.91	19.59
Average			14.26

Table 3: Comparing Power Optimization using Tree Covering and DAG Covering

Circuit	Power Comparison			Delay Comparison		
	Min Power	Delay Power	% Higher	Power Delay	Delay Delay	% Lower
5xp1	8.56	9.94	16.16	13.69	10.06	26.52
5xp1-hdl	4.77	6.88	44.23	18.48	13.53	26.79
9sym	14.07	17.32	23.12	22.38	13.67	38.92
9symml	11.85	14.50	22.36	18.14	12.62	30.43
9sym-hdl	5.94	7.20	21.34	22.96	18.32	20.21
alu2	17.35	24.45	40.92	52.68	40.06	23.96
alu4	31.33	41.05	31.05	51.69	43.27	16.29
alupla	7.12	7.90	11.01	16.92	14.10	16.67
apex6	35.95	49.40	37.42	20.33	15.44	24.05
apex7	13.06	17.28	32.34	16.65	14.54	12.67
b9	7.37	9.13	23.86	11.04	8.19	25.82
bw	15.04	17.68	17.58	26.97	12.72	52.84
c8	10.68	13.29	24.46	11.45	7.57	33.89
con1	1.14	1.47	29.34	6.22	3.92	36.98
duke2	30.76	41.54	35.03	40.19	17.81	55.69
f2	1.52	1.77	15.94	4.51	3.65	19.07
f51m	8.43	10.62	26.03	13.02	10.02	23.04
f51m-hdl	4.66	6.64	42.31	17.04	11.71	31.28
frg2	63.40	115.00	81.39	50.43	26.84	46.78
misex1	4.71	5.60	19.05	12.71	7.19	43.43
misex2	5.59	8.38	50.06	19.41	7.93	59.14
misex3	41.51	63.65	53.33	45.30	23.78	47.51
misex3c	33.65	44.40	31.96	36.06	18.36	49.08
pair	86.23	150.63	74.68	43.66	26.04	40.36
rd53	3.14	4.24	34.78	8.74	7.12	18.54
rd53-hdl	2.45	4.70	91.99	11.89	10.97	7.74
rd73	9.68	12.29	26.93	14.25	12.27	13.89
rd73-hdl	4.10	9.10	122.13	15.58	13.63	12.52
rd84	19.70	24.50	24.41	30.58	20.77	32.08
rd84-hdl	4.92	11.35	130.75	18.07	16.14	10.68
rot	37.67	67.36	78.84	35.10	26.75	23.79
sao2	8.33	11.04	32.41	16.15	12.29	23.90
sao2-hdl	11.62	50.16	331.62	45.05	31.31	30.50
vda	28.61	56.74	98.31	46.59	25.63	44.99
vg2	10.48	12.99	23.91	20.13	9.56	52.51
x1	19.79	26.42	33.52	18.96	10.43	44.99
x3	55.05	75.15	36.51	29.25	15.10	48.38
x4	27.41	44.46	62.18	28.45	15.61	45.13
z4ml	4.11	4.72	14.85	10.10	7.22	28.51
z4ml-hdl	2.90	3.25	11.96	11.83	9.71	17.92
C1355	27.50	67.46	145.33	29.70	28.13	5.29
C17	0.47	0.85	83.26	3.20	3.12	2.50
C1908	29.81	69.68	133.78	46.67	44.91	3.77
C432	11.21	28.38	153.20	51.44	29.26	43.12
C499	23.34	57.02	144.33	29.29	20.98	28.37
Average			58.22			29.79

Table 4: Results of Comparison Between Technology Mapping for Power and for Delay

Circuit	**Power Comparison**			**Delay Comparison**		
	Constrained Power	**Delay Power**	**% Higher**	**Power Delay**	**Delay Delay**	**% Lower**
5xp1-hdl	4.99	6.88	37.90	13.64	13.53	0.81
9sym-hdl	6.54	7.20	10.24	18.74	18.32	2.24
alu2	20.08	24.45	21.77	45.58	40.06	12.11
apex6	38.61	49.40	27.96	17.59	15.44	12.22
apex7	13.25	17.28	30.40	17.11	14.54	15.02
b9	7.43	9.13	22.81	9.80	8.19	16.43
f51m-hdl	4.99	6.636	33.12	14.21	11.71	17.59
frg2	63.40	115.00	81.39	50.43	26.84	46.78
pair	89.96	150.63	67.44	27.53	26.04	5.41
rd53-hdl	2.31	4.70	103.38	11.11	10.97	1.26
rd73-hdl	5.33	9.10	70.85	13.46	13.63	-1.26
rd84	20.43	24.50	19.93	21.83	20.77	4.86
rd84-hdl	4.53	11.35	150.41	16.91	16.14	4.55
sao2	8.78	11.04	25.75	14.88	12.29	17.41
sao2-hdl	22.71	50.16	120.90	43.19	31.31	27.51
vda	30.17	56.74	88.09	40.42	25.63	36.59
x3	58.63	75.15	28.19	20.54	15.10	26.48
x4	30.98	44.46	43.49	21.33	15.61	26.82
C1355	27.50	67.46	145.33	29.70	28.13	5.29
C17	0.47	0.85	83.26	3.20	3.12	2.50
C1908	40.06	69.68	73.92	40.14	44.91	-11.88
C499	33.68	57.02	69.27	26.35	20.98	20.38
Average			61.63			13.14
Originally			83.43			24.90

Table 5: Results of Delay Constrained Power Mapping

Library	**# Gates**	**Benchmark Circuit**							
		alu2	**apex7**	**b9**	**c8**	**f51m**	**misex1**	**sao2**	**z4ml**
22-1	3	470.4	347.7	200.2	289.5	216.5	122.5	240.0	112.4
33-1	5	388.2	295.8	168.4	255.4	200.3	101.3	186.5	102.3
44-1	7	367.5	284.0	154.3	242.4	199.3	96.0	172.6	102.3
22-2	7	339.9	278.5	162.2	215.5	140.8	89.2	178.0	72.9
33-2	33	279.2	240.5	132.2	181.7	129.7	69.2	124.3	67.1
33-4	87	254.9	222.1	121.6	130.7	105.6	64.6	115.6	59.0
44-2	131	259.5	227.9	118.9	168.4	128.6	64.8	112.5	67.1
43-5	396	233.3	207.6	113.0	116.2	100.7	54.9	101.4	54.1

Table 6: Effect of Library Size on Technology Mapping for Power

more importantly, the difference is delay now comes down to about 13%. This shows that it is possible to bring down the performance penalty to acceptable levels while still achieving significant power savings.

Note that for most of the circuits, delay constrained mapping did not lead to satisfaction of the delay constraint. This is because of the lack of optimality in the discretization approach. A more aggressive discretization of required times and load values, which would require more computational resources, can lead to better results. The results in Table 4 nevertheless still serve an illustrative purpose. Some more methods that can lead to the delay improvement of power mapped circuits are discussed in Section 6.

5.4 Effect of Library Size

The impact of the size of the library, i.e., the number of gates in the library on the power consumption of power mapped circuit was the subject of the last experiment. Prior experience with varying libraries in the case of area mapping has shown that increasing the library size does result in a significant improvement in the quality of the results [21, 14]. Since we did not have access to any industrial libraries we used the sample libraries that come with the SIS distribution. The results for a subset of circuits are shown in Table 6. Column 1 gives the name while Column 2 gives the number of gates in each library. The other columns give the power costs of power mapped circuits for increasing library size. Even though the libraries used are of an experimental nature, the results provide evidence that the increased choices provided by larger libraries can lead to a marked improvement in the quality of the final mapped circuit.

6 Summary and Future Directions

There are several contributions made by this paper. The first is the development of power models for library gates. The second is the use of these models in a technology mapping algorithm that optimizes for power. Using tree circuits, for which we know that the technology mapping algorithms are optimal, we have demonstrated that there is significant difference in the minimum and maximum possible values of the power consumptions. In addition, there is a difference between the power consumption of the power optimized circuit and the area and delay optimized circuit; showing that the least area or least delay circuit is not necessarily the least power circuit. These two results show that there is potential to reduce the power consumption during the technology mapping stage.

While the potential for power reduction has been demonstrated; not all of it has been tapped yet. There are several issues that are currently under exploration that we believe hold the key to even greater reductions in power requirements. The first of these deals with the generation of the subject DAG, which is the starting point for the mapping process. More work needs to be done in improving this starting point since the final result can be very sensitive to this. Trying to take care of internal fanout points in non-tree circuits has been a problem for both area and delay mapping, it continues to be a bother in power mapping. Two different directions are being pursued here. The first is doing DAG covering using binate covering. This is likely to be slow. We do not have much hope for this for general use, but would like to use it to see what the exact results are for at least some small circuits. The second direction is to explore more heuristics for handling the fanout points in an attempt to find one that works well. Besides delay constrained power mapping other methods can be applied to explore the power-delay tradeoff space. One such method is post-mapping gate resizing. Standard cell libraries typically provide a number of cells which perform the same function but which vary in their size. Replacing a given cell with another of a different size provides an avenue for tuning the power-delay cost of the mapped circuit. Another approach can be to start with the delay minimized circuit. This circuit might have some non-delay-critical paths, which can be remapped for low power using the methods presented here. Minimization of glitching power needs to be considered explicitly, possibly as a post-mapping step in which path lengths are selectively balanced. Finally, it would be interesting to repeat the experiment of Section 5.4 using a set of industrial libraries. Larger libraries could result in a greater difference between the costs of area, delay and power mapped circuits.

7 Acknowledgments

We would like to thank Rick Rudell for interesting discussions and valuable suggestions on technology mapping. We would also like to thank Abhijit Ghosh and Srinivas Devadas for the power estimation tool.

References

[1] A. Aho and S. Johnson. Optimal code generation for expression trees. *Journal of the Association for Computing Machinery*, pages 488–501, July 1976.

[2] R. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35:677–691, Aug. 1986.

[3] A. Chandrakasan, M. Potkonjak, J. Rabaey, and R. Broderson. HYPER-LP: A System for Power Minimization Using Architectural Transformations. In *Proceedings of the International Conference on Computer-Aided Design*, pages 300–303, November 1992.

[4] A. Chandrakasan, S. Sheng, and R. Broderson. Low-Power CMOS Digital Design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, April 1992.

[5] A. Chatterjee and R.K. Roy. Synthesis of Low Power DSP Circuits Using Activity Metrics . In *Proceedings of the VLSI Design Conference*, pages 265–270, January 1994.

[6] K. Chaudhary and M. Pedram. A near-optimal algorithm for technology mapping minimizing area under delay constraints. In *Proceedings of the Design Automation Conference*, pages 492–498, June 1992.

[7] M. Cirit. Estimating Dynamic Power Consumption of CMOS Circuits. In *Proceedings of the International Conference on Computer-Aided Design*, pages 534–537, November 1987.

[8] M. Pedram C.Y. Tsui and A. Despain. Technology Decomposition and Mapping Targeting Low Power Dissipation. In *Proceedings of the Design Automation Conference*, pages 68–73, June 1993.

[9] M. Pedram C.Y. Tsui and A. Despain. Power efficient Technology Decomposition and Mapping under an extended power consumption model. *IEEE Transactions on Computer-aided design*, 13(9):1110–1122, September 1994.

[10] S. Ercolani, M. Favalli, M. Damiani, P. Olivio, and B. Ricco. Estimate of signal probability in combinational logic networks. In *Proceedings of the European Test Conference*, pages 132–138, April 1989.

[11] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of average switching activity in combinational and sequential circuits. In *Proceedings of the Design Automation Conference*, pages 253–259, June 1992.

[12] A. Ghosh, A. Shen, S. Devadas, and K. Keutzer. On Average Power Dissipation and Random Pattern Testability. In *Proceedings of the International Conference on Computer-Aided Design*, November 1992. To appear.

[13] K. Keutzer. DAGON: Technology Mapping and Local Optimization. In *Proceedings of the Design Automation Conference*, pages 341–347, June 1987.

[14] K. Keutzer, K. Kolwicz, and M. Lega. Impact of Library Size on the Quality of Automated Synthesis. In *Proceedings of the International Conference on Computer-Aided Design*, pages 120–123, November 1987.

[15] H. Kriplani, F. Najm, and I. Hajj. Maximum Current Estimation in CMOS Circuits. In *Proceedings of the Design Automation Conference*, pages 2–7, June 1992.

[16] P. Landman and J. Rabaey. Power Estimation for High Level Synthesis. In *Proceedings of the European Design Automation Conference*, pages 361–366, February 1993.

[17] D. Liu and C. Svensson. Trading Speed for Low Power by Choice of Supply and Threshold Voltages. *IEEE Journal of Solid-State Circuits*, 28(1), January 1993.

[18] F. Najm. Transition Density, A Stochastic Measure of Activity in Digital Circuits. In *Proceedings of the Design Automation Conference*, pages 644–649, June 1991.

[19] S. Prasad and K. Roy. Circuit Activity Driven Multilevel Logic Optimization for Reliable Low Power Operation. In *Proceedings of the European Design Automation Conference*, pages 368–372, February 1993.

[20] K. Roy and S. Prasad. SYCLOP: Synthesis of CMOS Logic for Low Power Applications. In *Proceedings of the International Conference on Computer Design*, pages 464–467, October 1992.

[21] R. Rudell. *Logic Synthesis for VLSI Design.* PhD thesis, University of California, Berkeley, 1989.

[22] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. Sangiovanni-Vincentelli. Sequential circuit design using synthesis and optimization. In *Proceedings of the International Conference on Computer Design*, pages 328–333, October 1992.

[23] V. Tiwari, P. Ashar, and S. Malik. Technology mapping for low power. In *Proceedings of the Design Automation Conference*, pages 74–79, June 1993.

[24] H. Touati. *Performance Oriented Technology Mapping.* PhD thesis, University of California, Berkeley, 1990.

[25] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design, A Systems Perspective.* Addison-Wesley Publishing Company, 1985.