Technology Mapping for Low Power

Vivek Tiwari
Dept. of EE, Princeton Univ

Pranav Ashar
C&CRL, NEC USA

Sharad Malik
Dept. of EE, Princeton Univ. *

Abstract

The last couple of years have seen the addition of a new 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. 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. Technology mapping for low power provides circuits with up to 24% lower power requirements than those obtained by technology mapping for area.

1 Introduction

There is a growing demand for low power circuits from two distinct sources: (1) circuits that have significantly high power requirements which necessitate expensive packaging and cooling (*e.g.* the 21064 (Alpha) processor from DEC consuming 25 Watts), (2) portable applications requiring long battery life.

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. [3, 2]). 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) [6]. Several interesting ideas are presented in this piece of research. However, one shortcoming of that work is that it is 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 the technology independent level will hold up after a final technology mapping step. Similar problems of meaningful modeling 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 recent research [5, 9, 8, 4], is one of accurately and efficiently measuring the power dissipated in a circuit. In our work, we use a power measuring technique akin to that reported by Ghosh *et al.* [5]. The cost function for technology mapping for low power is derived from this power measurement technique. The gate power models and power measurement techniques used in our work are reviewed in Section 2.

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 circuits with 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 3. 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 4 on a large set of benchmark circuits indicate a potential for reduction in the power requirements by optimal technology mapping. Typical reductions are more than 10% with cases where the improvement is even 24%.

2 Power Dissipation Model

In a CMOS gate, the most significant part of the power consumption occurs only during transitions at the output when the output is charging/discharging. In effect, the average power consumed by a CMOS gate is given by $\frac{1}{2} \times C_{output} \times V_{dd}^2 \times N$, where C_{output} is the output capacitance of the gate, V_{dd} is the supply voltage, and N is the expected number of transitions at the output per clock cycle. Besides V_{dd}, which we assume is fixed, there are two different parameters in the above expression, C_{output} and N. Let us consider models for each.

2.1 Transition Probabilities

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, the probabilities of transitions at the primary inputs, and the delay model being used.

The *signal probability* p_g at the output of a gate g is the probability of a logical 1 at the output of this gate. As an example of computing p_g, suppose the minterms for which the function at gate g evaluates to a 1 are $a_1 a_2' a_3$, $a_1' a_2 a_3$ and $a_1 a_2 a_3$. 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: $p_g = p_{a_1}(1-p_{a_2})p_{a_3} + (1-p_{a_1})p_{a_2}p_{a_3} + p_{a_1}p_{a_2}p_{a_3}$

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.

Thus, p_t is 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*. Note that under the zero-delay approximation, the signal transition probability at a node in the network is dependent only on the Boolean function being computed at the node. Glitches must be considered if we assume non-zero delays at gates. Thus the total power consumed in a circuit

30th ACM/IEEE Design Automation Conference®

is the sum of the logical switching power and the glitching power. The logical switching power is the same as the total power in a zero delay model. In this paper, we consider the zero delay power to be a valid approximation of the total power for the following reasons:

- The glitches are caused by unequal path lengths. This effect can be mitigated to some extent by balancing path lengths by buffer insertion as a post-processing step.
- In practice, glitching power forms a relatively small component of the total power dissipation in most circuits [6]. 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.

2.2 The Output Capacitance

The capacitance at the output of a CMOS gate is the sum of three components C_{INT}, C_{WIRE}, and C_{LOAD} [14]. C_{INT} represents the internal capacitance of the gate. This consists largely 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.

Given the capacitances, the power consumed by a gate g is $P_g = k \times (C_{INT_g} + C_{LOAD}) \times p_g$. 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, by traversing the circuit and tallying up the power consumed at each gate.

3 Mapping for Low Power

The overall flow of our algorithm for technology mapping for low power follows along the lines of technology mapping for area and delay as described in [10]. 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.

3.1 Relationship to Area/Delay Mapping

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. *Thus, 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 1(a). 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.[1] Assume that the gates available in the

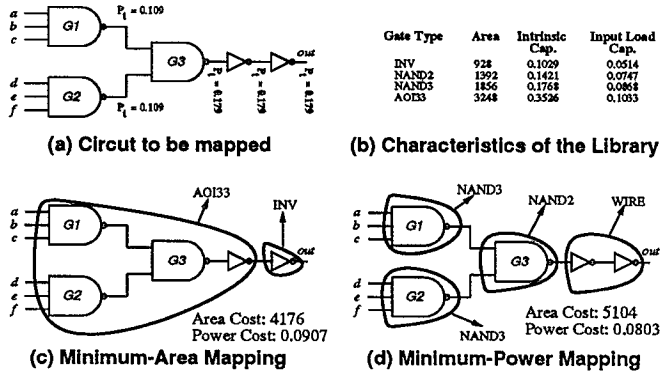

Figure 1: Example Illustrating the Difference Between Technology Mapping for Power and Area

library for technology mapping are inverters, two-input and three-input NAND gates and AOI33 gates. The areas and capacitances in standard units associated with these gates are shown in Figure 1(b).[2] (Since the point being made by this example is independent of the load driven by the *out* signal, assume that that load is zero.) The minimum-area mapping for this circuit is shown in Figure 1(c), and minimum-power mapping in Figure 1(d). The area and power cost of each mapping is shown in the corresponding figure. The power cost of a mapping is the sum of the power costs of each match in the mapping. Using the power models, the total power cost of the minimum-area mapping is 0.0907 while that for the minimum power mapping is 0.0803. Note that since the signal transition probability at the output of gate G_3 is much higher than the signal transition probabilities at the outputs of gates G_1 and G_2, it is less expensive in terms of power dissipation to have large gates feeding the outputs of gates G_1 and G_2 and small gates feeding the output of gate G_3, than have large gates feed the output of gate G_3. It turns out that the minimum-area mapping has a large gate and an inverter feeding the wires with high signal transition probability, and is therefore expensive in terms of power dissipation. As we can see from the figure, the power cost of the minimum-power mapping is found to be more than 10% lower than the power cost of the minimum-area mapping.

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,[3] 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 specialized 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

[1] The signal transition probabilities are computed by assuming that each primary input takes on a value of 1 or 0 with a probability of 0.5, independent of the values on the other primary inputs.

[2] Derived from the `lib2.genlib` library in the SIS [11] distribution.

[3] Equivalent to considering power dissipation due to glitching (*cf* Section 2).

into the model.

3.2 Computing 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 [5]. The probabilities are computed under the assumption that each primary input takes on a value of 1 or 0 with a probability of 0.5, independent of the values on the other primary inputs. This is typically true for datapath circuits, but may need refinement for control circuits. 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 counting the number of minterms in the off-set (on-set) of the Boolean function at the wire in terms of the primary inputs using Binary Decision Diagram [1] based algorithms. 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.

3.3 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. We assume familiarity with tree covering algorithms for area and delay mapping.

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 traverses 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 = ∞ ;
    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 ;
}
```

3.4 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 required time 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 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 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. (This part is currently being implemented.)

3.5 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.

A good approximation to DAG covering is to pose it as a composition of tree covering problems. The DAG is broken into trees at each multiple- fanout point and each tree is optimally mapped, separately. This heuristic has proven effective for area and delay optimization in the past and can also be used for power. However this approach has some problems. First, no matches across fanout points are allowed and thus tree overlap cannot occur (Figures 2 (a) and (b)). Usually, overlapping of trees can give better results for the three metrics, area, delay and power. Second, if tree boundaries are enforced while covering the subject DAG, the phase of the root and the leaves of each tree is fixed, reducing the flexibilty available to the matching algorithm. One way to solve this problem is to perform global phase optimization after an initial technology mapping pass. As an alternative, a cross-tree-phase heuristic is described in [10] to solve this problem and it has been implemented in the technology mapping package in SIS [11].

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[4] manner. All patterns (including ones which span multiple-fanout nodes) which 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

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

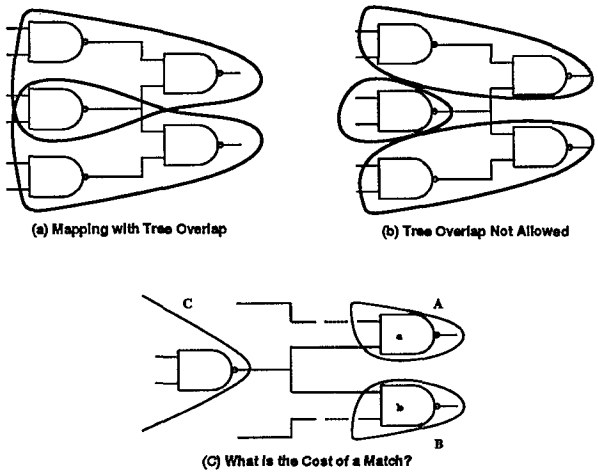

Figure 2: Ways of Handling Fanout

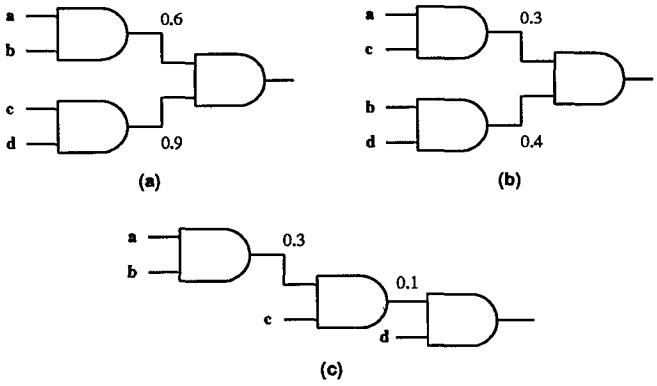

Figure 3: Effect of Decomposition on Transition Probabilities

the propagation of costs beyond such points. The latter problem occurs for the case of area and is illustrated by Figure 2(c). 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 both A and B. For the case of delay, evaluation of the cost of matches that are fed by multiple-fanout points, is a problem, as illustrated by Figure 2(c). The cost (arrival time) of matches at neither node A nor node B can be determined before the other. Thus neither can be mapped before the other. Touati [13] 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 change arbitrarily during the mapping process. Touati [13] 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. Like area, it is additive and like delay it depends upon the load value of the fanouts. The experience with generalized area and delay mapping, however, suggests that looking across tree boundaries can be beneficial. 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. The effectiveness of the DAG covering approach will be evaluated in Section 4.2.

3.6 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 3 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 differences in pin assignment and topology can lead to different transition probabilites at the internal nodes. 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 do not have a good solution to this problem yet. We are exploring this issue further.

4 Experimental Results

The algorithms and heuristics described in the previous sections have been implemented and integrated with the technology mapping package in SIS [11]. 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.

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`.[5] If explicit capacitance values were supplied with the libraries, they could be used directly.

4.1 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 experiment to be useful, it 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, i.e., the circuit was partitioned into a forest of trees and optimal tree-mapping used for each tree.

The results for maximum and minimum power dissipation show an average difference of around 46%, with individual differences going as high as 67%. The results indicate that power variations during technology mapping are significant and thus this provides a motivation for our work.

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 again used for area minimization. The results show that the minimum power dissipation is lower than the power dissipation of a circuit mapped for area by an average of about 12%.

For detailed results of these experiments, please refer to [12].

[5] 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 since 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 tables are meant for relative comparison rather than as absolute measures of power.

Circuit	Power Comparison			Area Comparison		
	Min Power	Area Power	% Lower	Power Area	Area Area	% Penalty
5xp1	8.559	9.697	11.74	145696	135488	7.53
5xp1-hdl	4.771	5.317	10.27	73776	71456	3.25
9sym	14.068	15.416	8.74	234320	199520	17.44
9symml	11.848	13.278	10.77	210656	187456	12.38
9sym-hdl	5.937	6.180	3.93	123424	115072	7.26
alu2	17.348	19.891	12.78	452864	376768	20.20
alu4	31.326	38.134	17.85	824528	722912	14.06
alupla	7.118	7.841	9.22	134096	122496	9.47
apex6	35.949	45.553	21.08	908512	717344	26.65
apex7	13.060	15.284	14.55	299744	253808	18.10
b9	7.371	8.379	12.03	139200	131312	6.01
bw	15.037	16.145	6.86	264944	226896	16.77
c8	10.675	11.817	9.66	206480	176784	16.80
con1	1.135	1.500	24.33	22272	20416	9.09
duke2	30.764	35.380	13.05	703888	600880	17.14
f2	1.527	1.767	13.58	24128	24128	0.00
f51m	8.429	8.956	5.88	141984	129920	9.29
f51m-hdl	4.663	5.081	8.23	71920	67744	6.16
frg2	63.402	75.914	16.48	1623536	1361376	19.26
misex1	4.708	5.321	11.52	83520	72848	14.65
misex2	5.587	6.784	17.64	162400	138736	17.06
misex3	41.510	46.368	10.48	972544	846800	14.85
misex3c	33.645	37.662	10.67	697856	589280	18.43
pair	86.235	96.600	10.73	1932096	1629568	18.56
sao2	8.334	9.387	11.22	161936	153120	5.76
sao2-hdl	12.052	12.886	6.47	257520	238960	7.77
rd53	3.130	3.346	6.46	50112	44544	12.50
rd53-hdl	2.446	2.450	0.16	44080	40368	9.20
rd73	9.486	10.417	8.94	178176	154976	14.97
rd73-hdl	4.098	4.119	0.51	78416	68208	14.97
rd84	19.696	21.218	7.17	381872	313664	21.75
rd84-hdl	4.917	4.886	-0.63	103472	90480	14.36
rot	37.667	42.479	11.33	762816	665840	14.56
vda	28.614	31.862	10.19	1325184	1147008	15.53
vg2	10.483	12.218	14.20	261696	193952	34.93
x1	19.790	23.744	16.65	411568	362848	13.43
x3	55.053	61.449	10.41	1071376	922432	16.15
x4	27.412	32.711	16.20	548448	541952	1.20
z4ml	4.108	4.570	10.11	64496	67280	-4.14
z4ml-hdl	2.901	3.095	6.27	50112	45936	9.09
C1355	27.498	28.260	2.70	735904	713632	3.12
C17	0.466	0.466	0.00	8352	8352	0.00
C1908	29.806	31.435	5.18	840304	610160	37.72
C432	11.210	12.687	11.64	239888	234784	2.17
C499	23.336	26.365	11.49	403680	458432	-11.94
Average			10.19			12.30

Table 1: Results of Technology Mapping Using DAG Covering Heuristics for Non-Tree Circuits

4.2 DAG Covering Results

The results of technology mapping using DAG covering are shown in Table 1. The second column shows the minimum power obtained using a variety of heuristics, some of which are mentioned in Section 3.5. The third column shows the power for the circuits that have been mapped for the minimum area using the SIS technology mapper. The power cost of the circuits optimized for power is, on the average, lower than the power of the area optimized circuits by more than 10%, with individual cases showing a difference of up to 24%. 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 DAG-covering 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 1. The last column shows the area penalty paid to achieve the reduction in power. The results show an average area increase of about 12%. 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. On the average, DAG covering results in a 12% improvement over tree-covering.

5 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 optimized circuit; showing that the least area 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. Minimization of glitching power needs to be considered explicitly, possibly as a post-mapping step by selectively balancing path lengths. Finally, it would be interesting to examine the impact of library sizes on the variation in power requirements. 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 until a certain point after which the additional gates provide little advantage [10, 7]. We would like to examine this aspect in the context of mapping for low power. Based on preliminary experiments, we have reason to believe that larger libraries will result in a more marked difference between the power consumption of a circuit mapped for area and that of a circuit mapped for low power.

References

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

[2] 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.

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

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

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

[6] 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.

[7] 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.

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

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

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

[11] 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.

[12] V. Tiwari, P. Ashar, and S. Malik. Technology Mapping for Low Power. *NEC CCRL Technical Report*, (92-C019-4-5509-2), October 1992.

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

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