Technology Decomposition for Low-Power Synthesis†

Rajendran Panda and Farid N. Najm
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

Technology decomposition and technology mapping are two potential stages for minimizing circuit power during logic synthesis. Since power in CMOS circuits is directly dependent on the extent of circuit switching activity, we present a novel procedure to construct a low-activity circuit structure in the technology decomposition stage. This would result in low-power circuits when mapped. The algorithm uses the transition density as a measure of switching activity and is applicable to both synchronous and asynchronous static circuits. Our results show power reductions of up to 48% (on average ≈ 10%), with little area or delay penalty.

1. Introduction

The high device counts and clock frequencies of modern ICs has made power dissipation of VLSI chips a major consideration during chip design. Hence the need for low-power logic synthesis CAD tools to aid in the design process.

In the popular CMOS and BiCMOS technologies, logic gates draw power only during logic transitions, so that the power-dissipation depends on the *switching activity* inside the circuit. This complicates the low-power synthesis problem, since the exact input signals are generally unknown during the design phase. Furthermore, it is practically impossible to estimate the power by simulating the circuit for all possible inputs.

This input pattern-dependence problem can be solved by using probabilities to describe the set of *all possible logic signals*. Specifically, we will use a measure of switching activity, called the *transition density* [1], that can be efficiently evaluated without requiring *exact* information about the primary input signals. The transition density at a node x in the circuit is the *average number of logic transitions per second* at that node, denoted by $D(x)$. As part of this formulation, it is found [1] that the density $D(x)$ is directly related, through a simple linear expression, to the Boolean function at node x, as follows:

$$D(y) = \sum_{i=1}^{n} P\left(\frac{\partial y}{\partial x_i}\right) D(x_i) \qquad (1)$$

† This work was supported by the National Science Foundation (NSF) under grant MIP–9308426.

where $P(\frac{\partial y}{\partial x_i})$ is the probability of the Boolean difference of y with respect to x_i. Therefore, more judicious choices for internal functional blocks based on (1) will, *by-design*, give less active and lower-power circuits. This idea is the basis of our work on low-power synthesis. While it is important to attack the synthesis problem at all levels, in this paper we focus on the technology decomposition step and propose a new algorithm that achieves lower power circuits on a variety of test cases, with little area or delay penalty.

2. Background

Consider a synchronous sequential circuit. Assuming edge-triggering, the latches draw power in synchrony with the clock, updating the inputs to the combinational blocks. Despite the synchronous switching of the inputs to the combinational blocks, the internal gates may make several transitions before settling to their steady state values for that clock period.

These additional transitions, called hazards or glitches, can cause as high as 70% of the total power [2]. The extent of glitching depends on (1) the mode of switching of inputs (synchronous or asynchronous) and (2) the racing of the signals arriving at a gate. Hence, for a low-power technology decomposition procedure to yield best results, it should address these challenging aspects of design, *viz.*, handling glitches and allowing for both synchronous and asynchronous operations.

Previous work in this area makes use of *signal probability* (average fraction of clock cycles in which the steady state value of the node is a logic high) and *transition probability* (average fraction of clock cycles in which the steady state value of the node is different from its initial value). Both these measures are unaffected by the circuit internal delays and make the implicit assumption that the switching is synchronous for *all* the gates, so that glitching is not accounted for.

Another aspect of this problem that usually requires approximations is the signal independence issue. In practice, the signals inside a circuit may be correlated so that, for instance, two nodes may never be simultaneously high. It is computationally too expensive to compute these correlations. So the circuit nodes are usually assumed to be *independent*. We refer to this as a *spatial independence* assumption. Another independence issue is whether the values of the same signal in two consecutive clock cycles are independent or not. If assumed independent, then the transition probability can be easily obtained from the signal probability according to $P_t(x) = 2P_s(x)P_s(\overline{x})$, where $P_s(x)$ denotes signal probability. We refer to this as a *temporal independence* assumption.

The technology decomposition technique that we will propose makes only a *spatial independence* assumption. Otherwise, it accounts for toggle power and allows for synchronous or asynchronous operation. We do not make a temporal independence assumption.

A low-power technology decomposition algorithm was proposed in [3]. They use transition probabilities to measure switching activity and present algorithms for decomposing both dynamic and static circuits. However, they make the restrictive assumptions of *zero-delay* and assume both *temporal and spatial independence*. References [4] and [5] present extensions of technology mapping algorithms for area and delay optimization to power optimization. Like [3], these algorithms also use transition probabilities for power estimation, and are subject to the *zero-delay* and both the *temporal and spatial independence* assumptions.

3. Technology Decomposition for Power

In logic synthesis systems like MIS/SIS [6], a network is first optimized in a technology-independent manner (we refer to as *logic-optimized*) and is then transformed into a feasible circuit for optimum area and/or speed, by selecting gates from a given gate library. This transformation is a two step process consisting of technology decomposition and technology mapping. The role of technology decomposition is to represent the logic-optimized network in terms of a simple set of base functions, usually consisting of a 2-input NAND-gate and an inverter. In the technology mapping step, the decomposed network is mapped to appropriate physical gates from the gate library, in a way that optimizes some parameter of the network, like area or delay. Although the role of technology decomposition is merely to provide the intermediate representation, nevertheless it can contribute to the optimality of the design by giving an orientation to the network such that the mapper produces best results. Traditionally, a node is decomposed into a balanced tree, with a hope that the delay of the mapped circuit is better. But, when power is an objective, it turns out that this is not the best decomposition.

The idea behind low-power technology decomposition is to transform the optimized circuit into a structure, wherein the average switching activity at the nodes is minimal. Through experimental results in Section 4, we will show that a circuit restructured thus serves as a good starting point for the mapper to realize implementations that have less power dissipation. This is true even when the mapper explores the alternatives with a different objective than power, such as area or delay.

To motivate the decomposition procedure, suppose an n-input AND gate is to be decomposed into a tree built of 2-input ANDs, called a binary tree. The input signals are leaf nodes of the tree, and each 2-input AND becomes a tree internal node. A binary tree with n leaves always has $(n-1)$ internal nodes. Let $x_i, i = 1, \ldots, n$ denote the leaves of the tree and $a_j, j = 1, \ldots, (n-1)$ denote the internal nodes. The average power dissipated in the tree is given by:

$$P_{av} = \frac{1}{2} V_{dd}^2 \sum_{j=1}^{(n-1)} C_j D(a_j) \qquad (2)$$

where $D(a_j)$ is the transition density at the output of the gate corresponding to node a_j and C_j is its output capacitance. The gate output capacitances are not known exactly at this point. They can be determined exactly only after the technology mapping step. However, at the decomposition stage, all the nodes of a tree (except the root node) have identical loads. Hence it is reasonable, at this stage, to assume that the capacitances are equal. Thus, one can write:

$$P_{av} = \frac{1}{2} V_{dd}^2 C \sum_{j=1}^{(n-1)} D(a_j) \qquad (3)$$

If the transition density of the output of a tree node were a linear function of *only* its input densities, then an exact optimal tree would result if the decomposition tree is constructed in the same manner as for a Huffman-coding tree (a linear time algorithm). However, the presence of the Boolean difference probability term in (1) violates this requirement. Since determination of an exact optimal decomposition involves computation that is exponential in the number of tree leaves, we resort to a heuristic greedy approach. Our algorithm attempts to construct the tree recursively from subtrees in which the total node density is minimum. This is done by recursively choosing pairs of nodes with the least transition density as inputs to a node at the next higher level.

Before presenting the decomposition algorithm, we summarize below the overall synthesis procedure that we adopt in conjunction with our algorithm. Our implementation uses SIS to do the initial optimization and also the final technology mapping.

1. Optimize the circuit with the conventional objective of minimizing the number of literals.

2. Represent the optimized circuit using large modules implementing AND/OR functions.
3. Compute the transition density values at all circuit nodes using (1).
4. Decompose the modules using the decomposition algorithm given below.
5. Perform mapping of the decomposed circuit for optimum area/delay.

Algorithm. *The algorithm is as follows:*

for each module (**Y**) in the logic-optimized network
    set **F** := type (AND/OR) of Y
    store inputs of **Y** to **InputsList**
    sort **InputsList** by *transition density*
    while **InputsList** is not a singleton set
        create a 2-input node of type **F**
        get a pair of least *density* nodes
            from **InputsList** and
            bind them to inputs of new node
        calculate the *density* of new node
            output using (1)
        add the new node output to **InputsList**,
            maintaining the list order
    end
    replace **Y** in the network with the singleton
            in **InputsList**
end

Since calculation of the node density values is linear time, the time complexity of the algorithm is $\mathcal{O}(n \log n)$ per module (due to sorting) where n is the number of input nodes to the module.

Our synthesis procedure has two important features:

(1) As noted before, an exact delay model is unavailable at this early stage of synthesis. However, any step towards reducing the power due to glitching requires some estimate of the glitches. One simple, but less effective, way is to exclude the consideration of glitches at this stage, as others [3–5] have done. We overcome this problem through our selection of *transition density* as the guiding measure. The stochastic formulation of *transition density* breaks this dead-lock by summarily capturing both the zero-delay activity and the glitching activity.

(2) As the effectiveness of the decomposition algorithm depends on the accuracy of the estimated values of transition density, we ensure this by excluding glitches that are too short in comparison to the inertial delay of the gates [7]. We carry out this filtering based on the inertial delay of a 2-input gate, as the inertial delays of the larger gates in the mapped circuit would be at least of this size.

4. Experimental Results

We evaluated the performance of our algorithm by estimating the power consumed in a large number of MCNC combinational benchmark circuits that were decomposed using our algorithm and mapped using the SIS package. To compare the results, we also let the SIS package decompose these circuits in the usual manner. The SIS *lib2.genlib* library of gates was used for mapping. Two sets of experiments were conducted, one with minimum area mapping and the other with minimum delay mapping.

The power consumption of the final mapped circuits was estimated using a statistical power estimator, called MED [8]. Using user specified input probability and density values, MED runs a number of simulations of the circuit, driven by randomly generated inputs. The simulation is event-driven, based on a timing model that scales delay with fanout and handles inertial delay. The desired accuracy of the results and a confidence factor can be specified by the user. The results presented below were obtained at 2% error and at 95% confidence level. We used the timing model of the *lib2.genlib* library gates for the simulations.

Due to lack of space, we present only the results for the circuits mapped for minimum delay. Table 1 compares the power, delay and area results of our decomposition with balanced tree decomposition. The third column in the table compares the power values for the decomposed but yet *unmapped* circuit. This comparison helps to see clearly how the low-power decomposition is reducing the total switching activity. The values shown in this column are the ratio of power of low-power decomposed unmapped circuit to that of balanced decomposed unmapped circuit. The fourth column gives the power consumption (in μW/MHz) of the circuit obtained through balanced tree decomposition. Column 5 gives the power consumption of the circuit obtained through our decomposition algorithm. The values in column 5 are normalized with respect to the values in column 4. Columns 6 and 7 similarly give the delay and area results normalized by the delay and area of the mapped circuit based on the SIS balanced decomposition.

The results show that our low-power decomposition algorithm leads to an average of 10% reduction in power at no area/delay penalty (on average), when mapped for minimum delay. Similar results were obtained in the case of SIS mapping for minimum area. In that case, the power reduction was 6.5% on average, with 0.7% area penalty, at no speed penalty. For a comparison, [3] reported that their decomposition algorithm resulted in a reduction of 3.6% in power.

TABLE 1
EVALUATION OF LOW-POWER DECOMPOSITION
(MIN. DELAY MAPPING)

CIRCUIT	SIZE	POWER			DELAY	AREA
Name	#gates	unmap.	Bal.	LP	LP	LP
cm42a	25	1.000	2.686	0.521	1.000	0.747
cm85a	54	0.885	2.196	1.052	0.809	0.900
cc	55	0.993	3.731	0.962	0.933	0.969
pm1	58	0.977	2.554	0.838	0.659	1.064
cu	65	0.943	4.355	0.893	0.821	0.970
cm162a	71	1.009	3.678	0.848	0.685	1.000
unreg	102	1.000	6.178	0.992	1.010	1.010
cht	125	0.997	5.271	0.850	0.796	0.988
comp	139	0.990	7.945	0.819	0.865	1.010
my_adder	222	0.987	14.320	0.750	1.000	0.990
apex7	249	1.005	12.994	0.915	0.926	1.008
example2	296	1.004	14.167	1.093	0.518	0.981
x4	352	0.995	17.147	1.021	1.047	0.978
x1	368	0.953	23.136	0.923	1.036	0.993
alu2	383	0.934	25.743	0.967	0.965	1.036
vda	500	0.872	29.876	0.897	0.977	0.972
i9	539	0.931	42.602	0.902	1.221	1.025
alu4	739	0.940	53.215	0.825	1.003	0.991
rot	838	0.979	57.159	0.967	0.984	0.987
x3	923	0.968	52.319	1.019	0.842	0.979
t481	978	0.929	27.603	0.982	1.084	0.978
i8	1100	0.884	76.115	0.713	0.856	0.929
Average		0.963		0.898	0.910	0.977

As the table shows, for 4 of the 22 circuits that we tested, the power increased slightly. This is due to the non-optimality of the algorithm. Otherwise, all the other circuits show reduced power. Significantly, for the largest circuit in the table, i8 (with 1100 gates), the power reduction was 29%, and the algorithm performed better on the larger circuits.

It may be observed that the unmapped circuits almost always show improvement when decomposed for low-power. This is not so with the mapped circuits, as the mapping algorithm changes the structure of the circuit to some extent. For this reason, we believe that a mapping algorithm with power minimization objective can produce even better results, when set to work on a circuit decomposed with our algorithm.

Finally, the algorithm is very fast, as would be expected from the $\mathcal{O}(n \log n)$ complexity. For any of the circuits in Table 1, the execution time is only a few seconds on a SUN Sparc ELC workstation.

5. Summary and Conclusions

We have presented an algorithm for low-power technology decomposition that heuristically attempts to minimize the average number of signal transitions at the circuit nodes. It has been shown that such a decomposition yields attractive power reduction in the final mapped circuits. The algorithm runs in $\mathcal{O}(n \log n)$ time and is applicable to both synchronous and asynchronous circuits.

The idea of using the *transition density* measure to minimize the node activity can be extended to other phases of the logic synthesis also. We are currently working on using this measure in the logic minimization and technology mapping phases of our synthesis system.

References

[1] F. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 310–323, Feb. 1993.

[2] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," IEEE/ACM International Conference on Computer-Aided Design, pp. 402–407, 1992.

[3] C-Y Tsui, M. Pedram, and A. Despain, "Technology Decomposition and Mapping Targeting Low Power Dissipation," *30th ACM/IEEE Design Automation Conference*, pp. 68–73, June 1993.

[4] V. Tiwari, P. Ashar, and S. Malik, "Technology Mapping for Low Power," *Proc. 30th ACM/IEEE Design Automation Conference*, pp. 74–79, Dallas, TX, June 14–18, 1993.

[5] B. Lin and H. de Man, "Low-Power Driven Technology mapping under Timing Constraints," *International Workshop on Logic Synthesis*, pp. 9a-1–9a-16, 1993.

[6] R. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, "Multilevel Logic Synthesis," *Proc. of the IEEE*, vol. 78, no. 2, pp. 264–300, February 1990.

[7] F. Najm, "Low-pass filter for computing the transition density in digital circuits," *IEEE Transactions on Computer-Aided Design*, vol. 13, no. 9, pp. 1123–1131, September 1994.

[8] M. Xakellis and F. Najm, "Statistical Estimation of the Switching Activity in Digital Circuits," *31st ACM/IEEE Design Automation Conference*, 1994.