On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks

Amelia Shen* Abhijit Ghosh† Srinivas Devadas* Kurt Keutzer‡

Abstract

We explore the implications of the observation that the probability of the occurrence of a transition on a wire of a circuit affects both the average power dissipation and the random pattern testability of a circuit. We show that restructuring a logic circuit can significantly affect its average power dissipation. We present various methods for the synthesis of combinational logic networks, and show the effect of different algorithms on the power dissipation of the circuit. We also focus on the dual problem of improving the random pattern testability of logic circuits. We show that modifying the signal probabilities can significantly affect the random pattern testability of a circuit.

1 Introduction

We explore the application of logic optimization techniques to modify signal probabilities on internal wires of a circuit and demonstrate the impact of these techniques on random pattern testability and average power dissipation. For many consumer electronic applications low average power dissipation is desirable and for certain special applications low power dissipation is of critical importance. The average power dissipation of a circuit, like its area or speed, may be significantly improved by changing the architecture or the technology of the circuit [5]. But once these architectural or technological improvements have been made, it is the switching of the logic that will ultimately determine its power dissipation. Traditionally, logic synthesis has been applied to improve the area or speed of a circuit [3, 4, 8] and logic synthesis techniques for improving the testability of circuits are more recent [2, 7]. In this paper we explore a number of methods for improving the average power dissipation of a circuit.

We also explore the implications of modifying average signal probabilities to improve the random pattern testability of a circuit. Pseudo-random patterns have been used for the testing of VLSI circuits for many years. However, for this approach to be effective, the circuit under test must be inherently random pattern testable. Unfortunately, not all circuits are random pattern testable, and there exists a class of circuits for which even weighted random pattern test generation is fairly ineffective. Techniques to improve the random pattern testability of circuits have been presented in [1, 11]. Most of these methods rely on the addition of control and observation points and are relatively *ad hoc*. Random pattern testability and power dissipation are functions of *signal probability*. The signal probability of a wire in a circuit is defined as the probability that a randomly generated input vector will produce a 1 value at the wire [1]. For random pattern testability, we want signals to have probabilities of being 1 or 0 approaching 0.5, so it is possible to excite both stuck-at faults on the wire. On the other hand, for the sake of power dissipation, we want a gate output to switch as rarely as possible, implying highly skewed signal probabilities for 1 or 0. Therefore, the goals of synthesis for low power and random pattern testability are conflicting. However, modifications of general synthesis techniques that impact signal probabilities on wires in the circuit can be used to target both cost functions. We explore this relationship in the paper.

The rest of this paper is organized as follows. In Section 2 we review the physical model for power estimation. Tradeoffs between area, power, and delay are the subjects of Section 3. General methods for synthesis for low power as well as synthesis for better random pattern testability are presented in Section 4. The use of logic optimization and don't-care sets in the synthesis of circuits and their effect on power dissipation and random pattern testability is the subject of Section 5. We describe the effects of some logic transformations on power dissipation in Section 6. Experimental results are presented in Section 7.

2 A Power Dissipation Model

The amount of energy dissipated by a CMOS logic gate each time its output changes is roughly equal to the change in energy stored in the gate's output capacitance. If the gate is part of a synchronous digital system controlled by a global clock, it follows that the average power dissipated by the gate is given by:

$$P_{avg} = 0.5 \times C_{load} \times (V_{dd}^2/T_{cyc}) \times E(transitions) \quad (1)$$

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

*Massachusetts Institute of Technology, Cambridge, MA
†Mitsubishi Electric Research Laboratories, Sunnyvale, CA
‡Synopsys, Mountain View, CA

0-8186-3010-8/92 $03.00 © 1992 IEEE

3 Design Tradeoffs

There are complicated tradeoffs between area, delay, and power of a CMOS circuit, and methods for minimizing power may increase circuit area and/or compromise circuit delay. Methods for minimizing power can be applied at the algorithmic, architectural, logic, or circuit levels [5]. A simple analysis of the effect of an architectural transformation that duplicates the logic in a circuit, increasing parallelism and throughput by a factor of 2, on power dissipation follows. The average power dissipated is proportional to:

$$V_{dd}{}^2 \times f \times \sum_i C_i N_i$$

where f is the clock frequency, C_i is the output load capacitance of gate i, and N_i is the average number of transitions made by the gate over all input vector sequences. Assume that the quantity $\sum_i C_i N_i$ is proportional to the circuit area; this is a good approximation since the total capacitance is proportional to circuit area and the average number of transitions per gate stays constant if the same library modules are duplicated during the architectural transformation. Therefore the power dissipated by the circuit is proportional to:

$$V_{dd}{}^2 \times f \times A$$

where A is the circuit area. Since the parallelism in the circuit and the circuit area increased by a factor of 2 due to the architectural transformation, we can decrease f by a factor of 2 while maintaining the original throughput.

The frequency f is inversely proportional to circuit delay. The delay of a logic gate is given by:

$$\frac{C_i \times \Delta V}{I_{av}}$$

where C_i is the load capacitance at the output of the gate and is unaffected by the architectural transformation for most gates. ΔV is the voltage change at the output of the gate and I_{av} is the average charging current available during the transition. We can lower V_{dd} and increase circuit delay, since we only need the combinational logic to be clocked at $f/2$. Assume we scale V_{dd} by a factor of $k < 1$. ΔV is scaled by a factor of k. I_{av} is scaled by a factor of k^2 for a CMOS gate [10], implying that the circuit delay is scaled by a factor of $1/k$. We can thus scale V_{dd} by $k = 0.5$ to obtain combinational logic that can be clocked at $f/2$.

The increase in A and the decrease in f more or less cancel out. However, the decrease in V_{dd} by a factor of 2 results in a *four-fold* decrease in power dissipation. Note that lowering V_{dd} reduces noise margins, increases subthreshold conduction and short circuit current [10]. The first affects circuit reliability and the latter two effects decrease the obtainable gains in power by V_{dd} reduction. Further, increasing circuit area by a factor of two typically impacts power dissipation by a larger factor. However, large reductions in power dissipation can certainly be obtained from algorithmic and architectural transformations for any given circuit design style.

In this paper we are concerned with the minimization of power at the logic level. We assume that system-level decisions regarding the power supply voltage and the clock frequency have already been made. This implies that we have a constraint on the maximum delays of the combinational logic blocks that are to be implemented in the circuit. Once these quantities are decided, there exists further room for power minimization at the logic level. The combinational logic will be restructured to minimize the power dissipation while maintaining circuit performance. Circuit area may increase and we will have to compensate for this by decreased circuit activity.

4 The General Strategy

Our general strategy for synthesizing circuits for low power dissipation or high random pattern testability is to restructure the circuit to obtain "appropriate" signal probability values at each internal gate. The signal probability values have to be minimized or maximized based on whether we are targeting low power or testability and also based on the chosen circuit design style, *e.g.*, Domino CMOS, Static CMOS, *etc.* In the following section, we will present our general methodology for logic synthesis targeting power and testability cost functions. Due to lack of space, we will discuss static circuits only.

4.1 Static CMOS with Zero Delay

We denote the logical function implemented by a gate g_i in the circuit as h_i and the probability of the function h_i being a 1 as p_i^{one}; the probability that h_i is a 0 as $1 - p_i^{one}$.

In the case of static CMOS circuits, power dissipation is due to the application of a vector pair $< V0,\ Vt >$ that causes transitions to occur at gate outputs. Assuming a zero-delay model, each gate in a CMOS circuit can make at most one transition upon the application of a vector pair, from low to high or from high to low. If the vectors in any vector pair applied are uncorrelated, then at any gate g_i in the circuit, the probability of there being a low to high or a high to low transition is $(1 - p_i^{one})p_i^{one}$. Therefore, to minimize power dissipation we minimize:

$$\sum_i p_i^{one}(1 - p_i^{one})C_i$$

The above quantity can be minimized by making p_i^{one} close to 0 or close to 1. One has to first evaluate the current value of p_i^{one} and then change it in the right direction to make it close to 0 or 1.

4.2 Static CMOS with General Delays

In the case of static CMOS circuits with arbitrary gate delays, a gate in the circuit may make multiple transitions, termed *glitching*, on the application of a vector

pair. Computing average power dissipation exactly requires a process of symbolic simulation [9]. The computations are significantly more expensive than in the zero delay case. In our experiments we found that glitching makes up approximately 20% of the power dissipation over a range of circuits. However, we have also noticed that in circuits such as a combinational adder, power dissipation due to glitches can be as high as 70%. Further, there is a correlation between the estimated power for circuits under zero delay and general delays. Hence, using the zero-delay approximation during synthesis is reasonable.

4.3 Random Pattern Testability

The stuck-at fault random pattern testability of a circuit is also dependent on the signal probabilities at each gate in the circuit. However, it is not dependent on the gate delays or gate capacitances.

It is intuitive that single stuck-at fault random pattern testability is related to values of both p_i^{one} and $1 - p_i^{one}$. Obviously, if $p_i^{one} = 0$, the stuck-at-0 fault on gate g_i is redundant. If $1 - p_i^{one} = 0$, the stuck-at-1 fault on gate g_i is redundant. If $p_i^{one} = 1 - p_i^{one} = 0.5$, then we have a very good chance of controlling the gate g_i to a 1 and a 0 by randomly applied input vectors, assuming a uniform distribution for the random patterns.

Stuck-at fault testability for a stuck-at-1 fault on a gate g_i requires that we control a gate g_i to a 0 value, and further propagate the error to the primary outputs of the circuit. We approximate the synthesis for random pattern testability problem as maximizing controllability over all gates in the circuit. That is, we maximize:

$$p_i^{one}(1 - p_i^{one})$$

implying that we wish to push p_i^{one} toward 0.5 for all gates. A similar observation has also been made in [1], where extra inputs have been used to change the signal probabilities. However, uniformly changing signal probabilities to 0.5 might not always increase random pattern testability[1, 12], but it is a good objective to use during synthesis.

While the above formulation has not specifically targeted the observability requirements for a stuck-at fault detection at the primary outputs, since the requirement is itself a controllability requirement on side-inputs to a path along which the fault is propagated, the formulation of maximizing $p_i^{one}(1 - p_i^{one})$ for each gate g_i captures the essence of random pattern testability for stuck-at faults. However, it does not capture the fact that we require several signals along the propagation path to be at 0 or 1 values simultaneously, nor does it take into account signal reconvergence.

5 Don't-Care Minimization

Observability, satisfiability, and external don't-care sets [4] can be used for the minimization of the functions of each node (complex gate) in the network. In [15], an algorithm was presented to compute local don't-cares for each node in the network (in terms of the immediate input variables of each node). These local don't-cares are derived from the total don't-care set using image computation techniques [6]. The local don't-cares can then be used to minimize each node. In [14], procedures for calculating the full observability don't-care set for each node are presented.

We use a synthesis environment similar to `misII` [4] with similar techniques for logic optimization. Requirements for the synthesis for lower power or better random pattern testability can be built into `misII` by suitably changing the cost functions that are used during network optimization. Don't-care minimization affects circuit size, delay, power consumption, and random pattern testability. After a circuit is synthesized without don't-cares, taking into account the area and performance constraints, further minimization with don't-cares has the potential of altering the above parameters. We focus on don't-care minimization in this section.

5.1 Exploiting Don't-Cares

Assume that we want to synthesize a circuit for lower power consumption. One can change the cost function of the two-level logic minimizer (which is used to minimize each node in the network) to produce a gate with low signal probability (close to 0) or high signal probability (close to 1). For random pattern testability, we require the opposite, *i.e.*, we want to obtain signal probabilities close to 0.5. This can be done by choosing prime implicants of the node that have appropriate signal probabilities.

It is not computationally feasible to estimate signal probability exactly after each step of logic optimization. Therefore approximations have to be used. In our method, the signal probability of each node is calculated globally (in terms of the primary inputs) whenever possible, otherwise it is calculated locally, *i.e.*, in terms of its local input variables, assuming that the variables are uncorrelated. Global signal probability is calculated before and after node minimization, while during node minimization, local probability is used.

Assuming that a gate has a certain signal probability, we can change this signal probability by altering the ON-set or the OFF-set by adding points from the don't-care set. This method of changing signal probability is only approximate because the don't-care set is in terms of local variables and during node minimization the local signal probability is used. Changing the local signal probability does not guarantee a change in global signal probability. However we can state the following (proof excluded):

Theorem 5.1 : *If the local probability of a function is decreased (increased) by altering the ON or OFF sets of the function, the global probability of the function can either decrease (increase) or stay constant.*

Our procedure for changing signal probability is as follows. At first, an optimized multi-level implementation of the circuit is obtained using standard techniques [4]. Then, we proceed in breadth-first order

from the primary inputs of the network to the primary outputs. For the primary inputs, the exact signal probabilities are assumed to be known. For each gate, the complete set of don't-cares is derived and converted to local don't-cares using the procedure of [15]. The global probability of the function is calculated. If the global probability of the function is at the desired value (0.5 for random pattern testability or away from 0.5 for low power), we minimize the function with the don't-care set using a two-level logic minimizer like ESPRESSO [3]. The minimized function is accepted if it firstly decreases the literal count, and secondly if it does not alter the global signal probability significantly.

If the global signal probability is not at the desired value, then the node minimization technique described in the next section is used to minimize the node.

5.2 Node Minimization

The final result of two-level minimization can be changed by altering the cost function used. While targeting low power dissipation, the number of product terms is minimized and the number of inputs is minimized, but the prime implicants that implement the two-level cover of the node are chosen so that they have the appropriate local signal probability. The cost function used in node minimization is:

$$n_{fo} \times k \times \sum_j p_j^{one}(1 - p_j^{one}) + \sum_i p_i^{one}(1 - p_i^{one})$$

where n_{fo} is the fanout load, k is the number of product terms in the cover, p_j^{one} is the local signal probability associated with the j^{th} cube in the cover and p_i^{one} is the global signal probability associated with the i^{th} input.

Note that $\sum_j p_j^{one}(1-p_j^{one})$ approximates the local switching probability of the node for the given two-level representation. The factor k is used to approximate the penalty in power dissipation due to a larger implementation of the function at a node.

The $\sum_i p_i^{one}(1-p_i^{one})$ term approximates the power contributed by the extra load capacitance of the fanin gates. The load capacitances of the nodes that drive a given node increase with the size of the driven node and the number of nodes driven. The factor k can also represent the number of literals in the node.

The method described above can be used for both low-power-driven or random-testability-driven synthesis. For the latter, the cost function is:

$$k + R/(\sum_j p_j^{one}(1 - p_j^{one}))$$

where R is a heuristically selected constant. Note that using the above cost function, if there are different covers with k primes in them, then the one with the maximum value of $\sum_j p_j^{one}(1-p_j^{one})$, *i.e.*, p_j^{one} closest to 0.5 will be selected.

6 Logic Transformations

In this section, we present some general circuit transformations that result in circuits with low power dissipation. We make some observations on what circuit transformations are likely to lower power dissipation.

6.1 Disjoint Covers

A method to improve the power consumption of a Domino CMOS circuit is to implement each node in the network as a disjoint cover. Thereby, only a small subset of **AND** gates and **OR** gates will switch each time a new vector is applied.

It can be shown, under a simplified assumption regarding the loading on gates, that a disjoint cover results in the minimum power dissipation of all two-level circuits implementing the same function. The following theorem is stated without proof.

Theorem 6.1 : *If the load capacitances for all* **AND** *gates in any two-level circuit are equal, then a two-level representation of a Domino CMOS circuit, where each cube is disjoint from every other cube, will have minimum power dissipation.*

Note however, that a disjoint two-level network will typically have **AND** gates with larger fanin and more gates than a minimum product-term network, invalidating the loading assumption. This may imply a larger power dissipation for the disjoint network.

It can also be shown that for Domino CMOS, extracting a single cube or kernel from a two-level circuit will increase power dissipation. Extracting a kernel increases power dissipation for Domino CMOS more than the increase due to extracting a cube. Thus, a heuristic method to synthesize Domino circuits for low power is to keep the **AND** gates closer to the primary inputs.

Another heuristic to obtain Domino circuits with low power dissipation is to break up the given two-level cover into disjoint subsets. For example, a two-level cover C is broken up into C_1 and C_2 such that $C_1 \cap C_2 = \phi$. C_1 and C_2 can be implemented separately and then **OR**'ed together. This may increase circuit area, but typically will not decrease circuit performance. This helps lower power dissipation for Domino circuits because it produces a partially disjoint cover.

The observations of the previous section are applicable, albeit to a lesser extent for static CMOS. Lowering the probabilities as long as they are below 0.5 helps lower power dissipation for static CMOS as well.

6.2 Partial Collapsing

A useful technique to reduce power dissipation is to collapse nodes that are not on the critical path into large nodes that can each be implemented as a two-level circuit. A two-level circuit typically dissipates less power than a multilevel network, but performance may degrade. Since nodes that are not on the critical path are collapsed, the resultant increase in their delay often does not affect the overall performance of the circuit. The procedure for performing this selective collapsing is simple. Nodes not on the critical path

are identified and their slack [4] is calculated. Nodes are selectively eliminated until the slack disappears.

6.3 Timing Optimization and Power

There are complex relationships between the delay of a circuit and its power dissipation. As described in Section 3, if a circuit has to be clocked at T_{cyc}, we can design the circuit to have a delay of kT_{cyc} where $k < 1$ and lower V_{dd} to reduce the power dissipation. One method of designing low power circuits is thus to make the circuit as fast as possible, regardless of area and power, and then lower V_{dd} to lower power dissipation. While scaling V_{dd} sounds like a significant circuit modification, common circuit design techniques may be applied on the chip to scale V_{dd} with 90% efficiency, keeping the power loss below 10%. Furthermore, scaling V_{dd} from $5V$ to $2.5V$ typically does not have bad side-effects. However, this does not always produce the best results (*cf.* Section 7) and further changing V_{dd} to less than $2V$ may have unpleasant side-effects.

Glitching in static CMOS circuits can range from 20% to 70% of the power dissipated. Glitching is reduced if paths in the circuit that converge at gates all have roughly equal lengths (delays). This implies that multiple transitions converge at the inputs to the gate more or less at the same time, and if the transitions are of opposite polarity, the inertial delay of the gate will dampen the transitions and result in less switching activity at the gate output. Thus, equalizing path lengths in the circuit can lower power dissipation.

Combinational logic timing optimizers (*e.g.* [16]) use transformations that shorten the length of the critical path at the possible expense of increasing the length of non-critical paths. Typically, after applying timing optimization the difference between path delays reduces (*cf.* [17]). We report on the empirical analysis of unoptimized and speed-optimized circuits for power dissipation in Section 7.

7 Experimental Results

As mentioned in Sections 4, 5 and 6, various methods can be used to synthesize a circuit for lower power consumption or better random pattern testability. For this section, the examples chosen are a subset of the ISCAS–89 benchmark set. Power estimation was performed using the techniques outlined in [9] assuming a clock frequency of 20MHz. All power estimates are in micro-Watts. All power, delay, and area estimates were obtained for the final technology mapped circuit. The delays for circuits were obtained using the library delay model of `misII`.

The effect of selective collapsing on power consumption is shown in Table 1. In method I, the rugged script of `misII` has been used for logic optimization. For method II, the starting point is the circuit obtained after method I. Nodes on non-critical paths are collapsed to decrease power consumption. For both methods, the number of literals, the delay, and the power consumption are shown. Power dissipation has been improved by as much as 60%, but in some cases increasing area by 8%. The delay of the circuit remains unaltered, as nodes that are not on the critical path are collapsed. The nodes that are collapsed are initially implemented as two-level networks, and the transformations described in Section 6 are applied to these nodes to check if power consumption can be further reduced.

CKT	Method I			Method II		
	Lit	Delay	Pwr	Lit	Delay	Pwr
s208	159	10.87	217	136	9.28	107
s344	281	14.62	394	285	13.69	284
s382	296	12.26	416	319	12.20	285
s420	318	18.43	391	273	17.03	155
s444	329	14.44	487	350	13.75	342
s510	586	15.92	728	485	15.84	444
s526	423	10.90	575	413	10.65	268
s641	365	18.23	486	306	17.75	264
s713	366	18.50	482	348	17.38	281
s820	621	11.64	680	610	11.57	400

Table 1: Effect of selective collapsing on power

CKT	Unoptimized			Optimized			Pwr
	Lit	Dly	Pwr	Lit	Dly	Pwr	v_{dd}
s208	111	13	108	128	11	104	70
s344	218	33	324	264	28	372	254
s382	208	32	251	313	28	334	256
s420	231	23	211	293	11	240	58
s444	207	31	278	374	27	411	299
s510	333	50	514	364	35	526	259
s526	242	36	307	452	29	533	333
s641	336	37	441	466	23	514	197
s713	335	39	434	532	21	577	161
s820	349	22	329	388	20	357	274

Table 2: Effect of delay optimization on power

The effect of delay optimization on power dissipation is shown in Table 2. Delay optimization was performed using the method of [16]. As can be seen, the larger delay optimized circuits dissipate more power than their unoptimized counterparts. However, the V_{dd} of a delay optimized circuit can be scaled to obtain the same delay as the unoptimized circuit. The effect of scaling V_{dd} on power dissipation has been discussed in Section 3. The power dissipation of the circuits after scaling V_{dd} is shown in the last column of the table. Power dissipation decreases in most cases, sometimes as much as 73%. However, examples(s444) exist where scaling V_{dd} in a delay optimized circuit does not improve power dissipation over the unoptimized circuit.

The effect of node minimization on power dissipation is shown in Table 3. For method I, the circuit was obtained using the rugged script of `misII` and standard node minimization. In method II, node min-

CKT	Method I			Method II		
	Lit	Delay	Pwr	Lit	Delay	Pwr
s208	94	10.9	86	91	10.9	81
s344	169	15.1	188	169	15.1	188
s382	180	12.0	177	180	12.3	176
s420	196	17.9	155	194	17.9	151
s444	170	14.5	167	171	14.3	166
s510	293	15.6	250	300	15.9	250
s526	229	10.7	210	232	10.9	209
s641	221	18.5	227	221	18.5	227
s713	221	18.5	228	221	18.5	228
s820	367	11.5	200	367	11.5	200

Table 3: Effect of power node simplify on power

imization targeted the power cost function. Power dissipation improved in some examples, but in other examples the power consumption did not change.

Finally, each node in the network was simplified with don't-care sets to determine the effects of using don't-cares during logic optimization on random pattern testability. In half the examples, the testability coverage increased by 0.1% to 0.6%. In some examples, there is a slight decrease in random pattern testability due to the fact that making signal probabilities close to 0.5 is only positively correlated to random stuck-at fault testability.

8 Conclusions

We have shown that both random pattern testability and average power dissipation are linked to signal probability, and we have explored the application of logic optimization techniques to modify signal probabilities on internal wires of a circuit. Our probability modification procedures have been implemented in circuit restructuring algorithms that improve power dissipation and/or random testability.

We assume that technology related improvements such as transistor sizing and load buffering will be applied to diminish the power dissipation of a circuit and that architectural tradeoffs will also be exploited. After that, additional improvements in power dissipation will require modifications to the combinational logic of the circuit. There is a very complex interaction between the area, delay, and power of combinational circuits. Our best results thus far in lowering power dissipation came from focusing on non-critical nodes in the network and reducing their power dissipation; however, the full potential for logic restructuring on power dissipation is still under evaluation.

References

[1] P. H. Bardell, W. H. McAnney, and J. Savir. *Built-In Test for VLSI: Pseudorandom Techniques.* John Wiley and Sons, New York, New York, 1987.

[2] K. Bartlett, R. Brayton, G. Hachtel, R. Jacoby, C. Morrison, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang. Multi-level Logic Minimization Using Implicit Don't Cares. In *IEEE Transactions on Computer-Aided Design*, pages 723–740, June 1988.

[3] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, 1984.

[4] R. K. Brayton, R. L. Rudell, A. Sangiovanni-Vincentelli, and A. Wang. MIS: A Multiple-Level Logic Optimization System. In *IEEE Transactions on Computer-Aided Design*, pages 1062–1081, Nov. 1987.

[5] A. Chandrakasan, T. Sheng, and R. W. Brodersen. Low Power CMOS Digital Design. In *Journal of Solid State Circuits*, pages 473–484, Apr. 1992.

[6] O. Coudert, C. Berthet, and J. C. Madre. Verification of Sequential Machines Using Boolean Functional Vectors. In *IMEC-IFIP Int'l Workshop on Applied Formal Methods for Correct VLSI Design*, pages 111–128, Nov. 1989.

[7] S. Devadas and K. Keutzer. A Unified Approach to the Synthesis of Fully Testable Sequential Machines. In *IEEE Transactions on Computer-Aided Design*, pages 39–50, Jan. 1991.

[8] D. Bostick et. al. The Boulder Optimal Logic Design System. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 62–65, Nov. 1987.

[9] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of Average Switching Activity in Combinational and Sequential Circuits. In *Proceedings of the* 29^{th} *Design Automation Conference*, June 1992.

[10] L. Glasser and D. Dobberpuhl. *The Design and Analysis of VLSI Circuits.* Addison-Wesley, 1985.

[11] J. Hayes and A. Friedman. Test point placement to simplify fault detection. In *Proceedings of the Fault Tolerant Symposium*, pages 73–78, 1974.

[12] V. S. Iyengar and D. Brand. Synthesis of Pseudo-Random Pattern Testable Designs. In *Proceedings of the International Test Conference*, pages 501–508, Sept. 1989.

[13] F. Najm. Transition Density, A Stochastic Measure of Activity in Digital Circuits. In *Proceedings of the* 28^{th} *Design Automation Conference*, pages 644–649, June 1991. Extended version submitted to IEEE Transactions on CAD.

[14] H. Savoj and R. Brayton. Observability relations and observability don't-cares. In *Proceedings of the International Conference on Computer-Aided Design*, pages 518–521, Nov. 1991.

[15] H. Savoj, R. Brayton, and H. Touati. Extracting local don't-cares for network optimization. In *Proceedings of the International Conference on Computer-Aided Design*, pages 514–517, Nov. 1991.

[16] K. J. Singh, A. R. Wang, R. K. Brayton, and A. Sangiovanni-Vincentelli. Timing Optimization of Combinational Logic. In *Proceedings of the International Conference on Computer-Aided Design*, pages 282–285, Nov. 1988.

[17] T. W. Williams, W. Underwood, and M. R. Mercer. The Interdependence Between Delay-Optimization of Synthesized Networks and Testing. In *Proceedings of* 28^{th} *Design Automation Conference*, pages 87–92, June 1991.