A Symbolic Method to Reduce Power Consumption of Circuits Containing False Paths*

R. Iris Bahar Gary D. Hachtel Enrico Macii † Fabio Somenzi

University of Colorado
Department of Electrical and Computer Engineering
Boulder, CO 80309

Abstract

Power dissipation in technology mapped circuits can be reduced by performing gate re-sizing. Recently we have proposed a symbolic procedure which exploits the compactness of the ADD data structure to accurately calculate the arrival times at each node of a circuit for any primary input vector. In this paper we extend our timing analysis tool to the symbolic calculation of required times and slacks, and we use this information to identify gates of the circuit that can be re-sized. The nice feature of our approach is that it takes into account the presence of false paths naturally. As shown by the experimental results, circuits re-synthesized with the technique we present in this paper are guaranteed to be at least as fast as the original implementations, but smaller and substantially less power-consuming.

1 Introduction

As the density, size, and complexity of VLSI chips continue to increase, the difficulty in providing adequate cooling might either add significant cost or limit the functionality of the computing systems which make use of these integrated circuits.

Power dissipation in CMOS devices can be reduced by means of precise architectural choices [1] and accurate selection of technology mapping algorithms [2, 3, 4]. However, even after these design decisions have been made, power consumption can be further reduced by *gate re-sizing*. This technique consists of replacing some gates of the circuit with devices in the gate library having smaller area and, therefore, smaller capacitive load. Given that the power dissipated by a gate is directly proportional to its load, reducing that load leads to a reduction of the power dissipated by the circuit as well as a reduction of the chip area. Smaller gates are also slower; therefore, in order to preserve the timing behavior of the circuit, not all gates can be re-sized; only the ones that do not belong to a critical path can be slowed down. Clearly, the applicability of the gate re-sizing method to reduce the power dissipated by a circuit depends heavily on the accuracy provided by the timing analysis tool in detecting the false paths and calculating the true delay of the circuit being re-synthesized for low-power.

Gate re-sizing is very effective when detailed timing information is available to identify non-critical gates of the circuit. In [5] we proposed a symbolic procedure based on Algebraic Decision Diagrams (ADDs) [6] to accurately calculate the arrival time at the output of each gate for any primary input vector. In this paper we extend the capability of the timing analysis tool to the symbolic calculation of required times and slacks, and we use this information to determine by how much gates can be re-sized without changing the original speed of the circuit. Our ADD-based re-synthesis tool handles circuits with false paths in the same way it treats circuits which are false path free.

*This work is supported in part by NSF/DARPA grant MIP-9115432 and SRC contract 92-DJ-206.

†Enrico Macii is also with Politecnico di Torino, Dipartimento di Automatica e Informatica, Torino, ITALY 10129.

2 Background

2.1 Terminology and Notation

A *combinational circuit* is a directed acyclic graph composed of gates and connections between gates. If the output of a gate, g_i, is connected to an input of a gate, g_j, then g_i is a *fanin* of g_j. Gate g_j is a *fanout* of gate g_i. A *controlling value* at a gate input determines the value at the output of the gate independent of the other inputs. We adopt the *floating mode* delay model, according to which the state of a node is unknown until it is set by the current vector. Each connection, c, has two delays, $d_r(c)$, rise delay, and $d_f(c)$, fall delay, associated with it. The *delay function* of c from gate h to gate g, $d(c,x)$, equals $d_r(c)$ if g carries a 1 when input vector x is applied to the primary inputs of the circuit. Otherwise, $d(c,x) = d_f(c)$. If all fanin connections of g have the same values of $d_r(c)$ and $d_f(c)$, we define the delay function of g as $d(g,x) = d(c,x)$, where c is any fanin connection of g. The *arrival time*, $AT(g,x)$, is the time at which the output of g settles to its final value if input vector x is applied at time 0; the *required time*, $RT(g,x)$, is the time at which the output of g is required to be stable when input vector x is applied; the *slack*, $ST(g,x)$, of a gate g is the difference between its required time and its arrival time, i.e., $ST(g,x) = RT(g,x) - AT(g,x)$. The *true delay* of a gate g is its maximum arrival time: $\max_x\{AT(g,x)\}$. The true delay of a circuit is the largest true delay of a primary output. A *path* in a combinational circuit is an alternating sequence of gates and connections, where connection c_i, $0 \le i < n$, connects the output of gate g_i to the input of gate g_{i+1}. The *length* of a path, $P = (g_0, c_0, \ldots, c_{n-1}, g_n)$ is defined as $d(P,x) = \sum_{i=0}^{n-1} d(c_i,x)$. The *topological delay* of a combinational circuit is the length of its longest path. Under a specified delay model, a path $P = (g_0, c_0, \ldots, c_{n-1}, g_n)$ is said to be *sensitizable* if a $0 \to 1$ or $1 \to 0$ transition at gate g_0 can propagate along the entire path P to gate g_n. The *critical path* of a circuit is the longest sensitizable path under a specified delay model; if a path is not sensitizable, then it is a *false path*. In the presence of false paths, the topological delay may exceed the true delay.

2.2 Power Estimation

Under a simplified model of energy dissipated by CMOS devices, the power consumption of a CMOS gate is directly related to its switching activity factor [8]. In constructing our power dissipation model we assume that the only capacitance in a CMOS gate is the output capacitance, that the current flows through some path either from the power supply to the output capacitor, or from the output capacitor to the ground, and that any change in the output voltage of a gate is either a change from V_{dd} to $0V$ or vice versa. The energy dissipated by a CMOS gate each time its output switches equals the change in the energy stored in the capacitor associated with its output. If the gate is part of a synchronous circuit driven by a global clock, the average power dissipated by the gate is $P_{\text{avg}} = \frac{1}{2} \cdot C_{\text{load}} \cdot \frac{V_{dd}^2}{T_C} \cdot E(transitions)$,

where C_{load} is the capacitance of the output load, V_{dd} is the supply voltage, T_C is the global clock period, and *E(transitions)* is the *switching activity factor* of the output of the gate. All the parameters of the equation above can be determined from technology or circuit layout information, except *E(transitions)*, which depends on both the function implemented by the gate and the probability distribution of the primary inputs. The method we use for power estimation is the one proposed in [9], which assumes the transition probabilities of the primary inputs to be given, and calculates the switching activity factor of each gate output using symbolic simulation.

2.3 ADD-Based Timing Analysis

In Section 2.1, we have defined the arrival time of a gate as a function of the primary inputs. Despite its obvious advantages, this definition has not been used so far, because it was impractical to manipulate the very large functions that occur for meaningful numbers of inputs. Instead, in static timing analysis, the arrival time would be described by the maximum arrival time for all inputs that cause the gate output to be 1, and by the maximum arrival time for all inputs that cause the gate output to be 0. Though the computation of these summary values is simple and inexpensive, ignoring the dependence on the inputs may provide the designer with inaccurate timing information. In fact, for circuits with false paths, static timing analysis may yield overly pessimistic results. Fortunately, Algebraic Decision Diagrams can be used to compactly represent real functions from $\{0,1\}^n$ for large values of n; this capability allows us to store more detailed timing information about a circuit than was previously feasible. It also allows us to take into account don't care conditions in a very natural way. It is sufficient to consider the restrictions of the various functions to the set of *care* input vectors. Given a gate g of the network and a primary input vector x, the arrival time at its output line, $AT(g,x)$, is evaluated in terms of the arrival times of its inputs, and the delays of its fanin connections, $d(c_j,x)$. Let c_j be the connection to pin j of gate g. If at least one fanin c_j of g has a controlling value for input $x \in X$, where X is the set of all possible care input vectors,

$$AT(g,x) = \min_j \{AT(c_j,x) + d(c_j,x) \mid c_j = \textit{controlling}\}.$$

If all fanins of g have non-controlling values,

$$AT(g,x) = \max_j \{AT(c_j,x) + d(c_j,x)\}.$$

Finally, if $x \notin X$, $AT(g,x) = -\infty$.
The ADD-based timing analysis algorithm of [5] computes, for each gate of the circuit, the arrival time at its output for all input vectors, and stores it as an ADD; it uses a recursive procedure, which explores simultaneously the functional BDD of each node of the network and the arrival time ADD of the fanin nodes computed at previous steps of the process.

3 Reducing Power by Gate Re-Sizing

3.1 Required Time Calculation

Given the required time for the output of a gate, computing the required times for its fanin connections is easy, if all input connections carry non-controlling values. Indeed, it is sufficient to subtract the appropriate connection delays from the output required time. However, this is no longer true when controlling input connections are present. First, the non-controlling inputs can arrive arbitrarily late. Second, controlling inputs can also arrive arbitrarily late, as long as at least one controlling value arrives early enough to guarantee the required time. Clearly, unlike the arrival times, the required times are not uniquely determined by the circuit.

One way to systematically make required times consistent is to assign a priority to the fanin connections of a gate. Whenever more than one connection carries a controlling value, the one with the lowest priority is chosen as the *designated connection*. The others are then given unlimited freedom, that is, infinite required time. If the designated connection is chosen carelessly, it is possible, in the presence of false paths, for the required time of the designated connection to be earlier than its arrival time. It is a simple matter, however, to prevent that from occurring, provided the required times of the primary outputs are greater than or equal to their respective arrival times. It can be proved, under these assumptions, that one designated connection that has a required time greater than its arrival time always exists. If the choice of the designated connection is made accordingly, the required times will be *compatible*, in the sense that every gate in the circuit can be slowed down until it meets its required times exactly, without changing the true delay of the circuit. If the required time of a signal is infinite for all $x \in X$, then the signal is redundant; the converse is not true.
Let $E = \{e_j\}$ be the set of fanin connections for gate g, and $C(x) \subseteq E$ be the subset of fanin connections to g that carry controlling values for input $x \in X$. If, for a given $x \in X$, all inputs to g are non-controlling, that is, $C(x) = \emptyset$, then

$$RT(e_j,x) = RT(g,x) - d(e_j,x), \quad \forall e_j \in E.$$

If some inputs to g are controlling, let $c_d \in C(x)$ be the designated connection, that is a connection such that $AT(c_d,x) \leq RT(g,x) - d(c_d,x)$. Then:

$$\begin{array}{lcll} RT(c_d,x) & = & RT(g,x) - d(c_d,x) & \text{and} \\ RT(e_j,x) & = & +\infty & \forall e_j \in E, e_j \neq c_d. \end{array}$$

If $x \notin X$, $RT(e_j,x) = +\infty$. Let $F = \{f_j\}$ be the set of fanout connections for gate g. Then:

$$RT(g,x) = \min\{RT(f_j,x)\}, \quad \forall f_j \in F.$$

The required times are computed in post-order, while performing a depth-first search of the circuit from the primary inputs. A node is not processed until all gates in its fanout have been processed.

3.2 Slack Calculation

Once the arrival and required time calculation is complete, we compute the slack for each gate, g, which is given by:

$$ST(c_j,x) = RT(c_j,x) - AT(g,x).$$

Let $F = \{f_j\}$ be the set of fanout connections for g. Then:

$$ST(g,x) = \min_j \{RT(f_j,x)\} - AT(g,x), \quad \forall f_j \in F.$$

Previous methods for slack computation take $\min_x\{RT(g,x)\} - \max_x\{AT(g,x)\}$ as the slack of g. Let $I_{AT}(g)$ be the set of input vectors that give an arrival time equal to $\max_x\{AT(g,x)\}$, and $I_{RT}(g)$ the set of input vectors that give a required time of $\min_x\{RT(g,x)\}$. If $I_{AT}(g) \cap I_{RT}(g) = \emptyset$ then this method will underestimate the slack of gate g. With ADDs we take the term-by-term difference of RT and AT ADDs to get a true calculation of the slack on the gate for every set of input values.

3.3 Gate Re-Sizing Algorithm

The gate re-sizing algorithm is combined with the required time and slack calculation. In fact, as it will be illustrated in Section 3.4, separately executing required time/slack calculation and gate re-sizing may lead to circuits which violate the original timing constraints when such circuits contain false paths.

The amount a gate may be slowed down for both rise and fall values is given by:

$$slack(g)_{rise} = \min_x\{ST(g,x)\}, \forall x \in X, f(g,x) = 1,$$
$$slack(g)_{fall} = \min_x\{ST(g,x)\}, \forall x \in X, f(g,x) = 0.$$

If a gate has a non-zero slack, we try to re-size the gate with a functionally equivalent gate from the library which has a larger gate delay. Once the gate is re-sized, the required time for this node will not change, as stated by the following theorem:

Theorem 3.1 *Given gates g_i and g_j, where g_i is the fanin of g_j via c_{g_i}, if the connection delay on g_i increases by no more than its slack, then the same selection of designated inputs remains possible and yields the same required times.*

If the gate is re-sized, the new gate delay information is updated and the required time and slack for its fanin gates are computed using this new delay information. If after re-sizing there still remains some residual slack on the gate, the computed slack for its fanin gates will "absorb" this extra slack. Since the delay of a gate is never increased by more than its available slack, our method guarantees we will never increase the delay of the critical path of the circuit. An example of a circuit with its ADD required time and slack construction is shown in Figure 1.

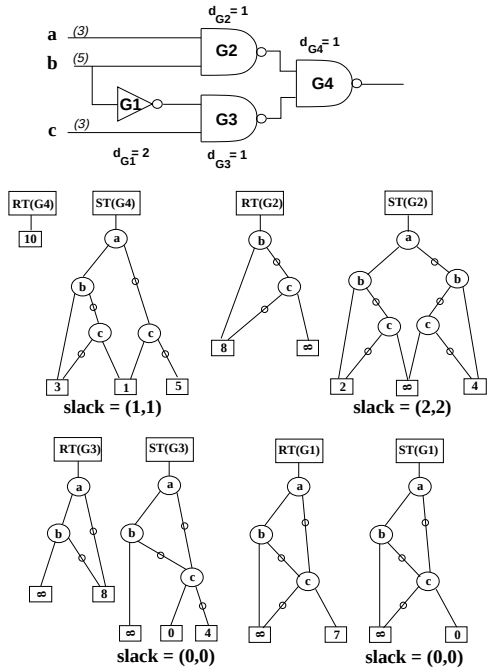

Figure 1: A Circuit and its RT and ST ADDs.

We start by computing the required time and slack for output gate $G4$, and obtain $slack(G4) = 1$. The gate is then re-sized so its delay, d_{G4}, is 2 nanoseconds. The required time and slack can now be computed for the fanin gates of gate $G4$ using the new delay information of $G4$. We proceed to compute the required time and slack of gate $G2$ and find that its delay may be increased by 2 ns. Notice that if we did not consume all the available slack on gate $G4$, then the slack available on gate $G2$ would have been greater than 2 ns. The gate is then re-sized so its delay, d_{G2}, is 3 nanoseconds, and we can continue computing the required times and slack for its inputs a and b. The required time and slack for gates $G3$ and $G1$ are computed in a similar fashion; however, in these two cases we see that there is no slack available; hence, the gates remain unaltered.

3.4 Why Gate Re-Sizing must be Interleaved with RT Calculation

In order for circuits with false paths to be re-sized correctly, it is essential that gate re-sizing be interleaved with the required time and slack calculation and not done as a post-processing step after all the required times and slacks have been computed for all gates. This post-processing step may be implemented by taking an *incremental slack* on each gate (extracted from the slack on the gate and the slack on its fanout gates), and using this value to re-size the gate. Let c_j be the fanout connection of gate g to gate g_j. For a given $x \in X$, the incremental slack on gate g with respect to c_j is given by:

$$IS(c_j, x) = ST(c_j, x) - ST(g_j, x).$$

Let $F = \{f_j\}$ be the set of fanout connection for g. Then:

$$IS(g, x) = \min\{IS(f_j, x)\}, \quad \forall f_j \in F.$$

The rise and fall incremental slack for a gate can be computed in the same way as the slack given in Section 3.3.
Now, consider the circuit of Figure 2, which contains a false path through $(G1, G3, G6, G7, G8, G9)$.

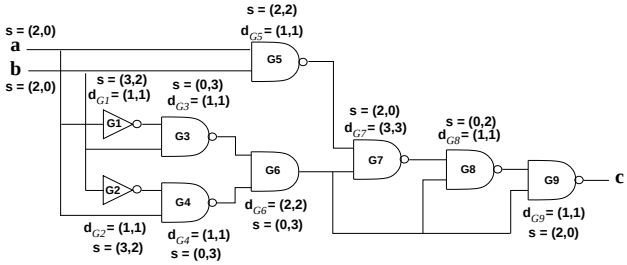

Figure 2: Sample Circuit Illustrating Error.

Rise and fall connection delays and computed slacks are shown next to each gate. Note that not all gates have unit delay. The true delay of the circuit is sensitized through the path $(G3, G6, G7, G8, G9)$ when $a = 0$, and $b = 0$ giving a delay of $\sum d_{g_i} = 1 + 2 + 3 + 1 + 1 = 8$. After computing incremental slacks, we end up with $IS(G1) = IS(G2) = (0, 2)$, $IS(G6) = (0, 1)$, $IS(G5) = IS(G9) = (2, 0)$. All other gates have zero slack. If the connection delays of these gates were increased by these incremental slack values, we would increase the critical delay of the circuit to 10 nanoseconds; in fact, for input vector $a = 1, b = 1$ the path $(G1, G3, G6, G8, G9)$ is sensitized, giving a delay of $\sum d_{g_i} = 3 + 1 + 2 + 1 + 3 = 10$. The reason we get this error is that a single gate along a path may have a slack that reflects a particular set of sensitizable paths; however, a fanin to this gate may not be part of this same set and may in fact reflect a different set of sensitizable paths. Since we are considering two different sets of sensitizable paths, we cannot simply take the difference of the two slacks to compute the amount by which the fanin gate may be re-sized. The slack on gate $G3$ is given as $ST(G3) = (0, 3)$. The rising slack is computed as zero because the gate is along the critical path of the circuit when inputs $a = 0$, and $b = 0$. However, since gate $G1$ is not sensitizable for this input vector, the falling slack on $G1$ has a value of 2, reflecting the slack on the non-critical path $(G1, G3, G6, G8, G9)$, which has a delay of 6. The incremental slack of $G1$ is $IS(G1) = (0, 2)$, but is computed from slacks representing two different path constraints and therefore cannot be guaranteed to give correct results. In fact, given the re-sizing of gate $G9$ by 2 nanoseconds on the rising delay, $G1$ cannot be re-sized at all without increasing the critical delay of the circuit.

Circuit	PI	PO	Gates	Init.Static Rise / Fall	Init.ADD Rise / Fall	Method	Final Static Delay Rise / Fall	Final ADD Delay Rise / Fall	Power Saved
5xp1	7	10	174	15.88 / 15.88	15.63 / 15.74	ADD	17.15 / 17.15	15.69 / 15.69	11.54%
						static	15.82 / 15.82	15.73 / 15.74	8.22%
bw	5	28	241	17.58 / 17.58	17.21 / 16.71	ADD	20.92 / 20.92	17.18 / 16.66	31.89%
						static	17.53 / 17.53	17.46 / 17.35	3.03%
clip	9	5	192	12.29 / 12.29	12.22 / 12.24	ADD	13.45 / 13.45	12.22 / 12.20	15.68%
						static	12.29 / 12.29	12.17 / 12.20	2.95%
rd73	7	3	96	15.10 / 15.10	14.85 / 15.05	ADD	15.50 / 15.50	14.94 / 14.97	10.89%
						static	15.10 / 15.10	15.00 / 15.05	3.75%
sao2	10	4	213	15.46 / 15.46	15.46 / 15.40	ADD	16.58 / 16.58	15.43 / 15.41	13.38%
						static	15.46 / 15.46	15.46 / 15.38	6.21%
sct	19	15	132	16.96 / 16.96	16.96 / 16.96	ADD	20.99 / 20.99	16.88 / 16.86	9.52%
						static	16.96 / 16.96	16.96 / 16.96	1.44%
squar5	5	8	101	13.75 / 13.75	13.46 / 12.61	ADD	16.68 / 16.68	13.26 / 13.39	38.23%
						static	14.01 / 14.01	13.72 / 13.02	22.03%
ttt2	24	21	306	11.36 / 11.36	11.36 / 11.16	ADD	12.36 / 12.36	11.29 / 11.16	38.95%
						static	11.28 / 11.28	11.28 / 11.28	37.61%
cbp8	16	9	186	19.30 / 19.30	15.28 / 15.28	ADD	21.58 / 21.58	15.17 / 15.17	14.81%
						static	19.25 / 19.25	15.61 / 15.61	0.51%
mult4	8	8	202	24.41 / 24.41	22.94 / 21.05	ADD	26.19 / 26.19	22.89 / 21.24	23.78%
						static	24.34 / 24.34	23.57 / 21.95	9.44%

Table 1: Comparison of Power Savings on Circuits with False Paths using ADD versus Static Timing Analysis.

4 Experimental Results

In Table 1 we report the results we have obtained on some of the MCNC'91 benchmarks [7], which have been optimized for area with the SIS `script.rugged`. We also present data for an 8-bit carry by-pass adder, and a 4-bit combinational multiplier. All the experiments were run within the SIS [10] environment on a DEC-Station 5000/200 with 80M of memory.

Columns *PI*, *PO*, and *Gates* show the number of primary inputs, primary outputs, and gates of each circuit. Columns *Init. Static* and *Init. ADD* give the initial rise and fall delays of the circuit computed by the static timing analysis procedure of SIS and by the ADD-based timing analyzer of [5], respectively. Column *Method* indicates whether ADD-based or static timing analysis was used within the re-sizing procedure. Columns *Final Static Delay* and *Final ADD Delay* report circuit delays, after gate re-sizing, using static and ADD-based timing analysis, respectively. The required times were set to the corresponding initial arrival times, that is, the initial true delay for the ADD-based re-sizing procedure, and the initial static delay for the static re-sizing method. Finally, column *Power Saved* reports the percentage of power reduction obtained on each circuit.

The library we used for the experiments had NANDs, NORs, and inverters; each type of gate had 5 different size/drive options and up to 4 inputs. Power dissipation was obtained by using the actual loads of the mapped network. The switching activity factors were computed with the symbolic simulation method of [9] together with the final delays of the mapped network.

Usually, after one pass of re-sizing through the circuit, additional re-sizing is still possible, due to the fact that when a gate is being processed, it does not have information on the new arrival and required times of any gate along its fanin paths that have yet to be resized. Therefore, repeated applications of the re-sizing procedure are necessary to get the best power results.

Circuits were mapped with the SIS `map -n 1 -AFG` command. In general, the optimized circuits were smaller, but slower, than the original ones. In most cases the circuits contained false paths which were not apparent in the initial mapping; by observing the frequency at which this phenomenon occurs, we can claim that false paths are often a by-product of boolean optimization. Clearly, the false paths may be removed by using the methods of [11]; however, this may cause the addition of extra gates, and therefore, it may increase power dissipation of the circuit.

The effectiveness of the ADD-based method is shown by the results of the table. In fact, in some cases, without using ADD timing analysis, power savings are quite small, while they are significantly higher if ADDs are used. Notice that false paths may not even be apparent from the initial mapped circuit, but with ADD analysis, we can still identify them and re-size gates along non-critical paths more effectively with this information.

5 Conclusions

We have presented a technique to reduce power consumption of combinational circuits which have already been mapped. The re-synthesis procedure we have proposed alternates timing analysis operations with gate re-sizing. Consequently, it is able to handle circuits with and without false paths exactly in the same way. We have compared the power saving results obtained on circuits containing false paths using both static timing analysis and ADD-based timing analysis. As expected, circuits re-sized using our ADD-based tool are much better in terms of power savings than the ones re-sized using static timing analysis.

Acknowledgments

We wish to thank Hyunwoo Cho for interesting discussions on symbolic methods for timing analysis and low power synthesis, Srinivas Devadas and Abhijit Ghosh for the power estimation code, and Mauro Pipponzi for his help in defining the libraries.

References

[1] A. P. Chandrakasan, S. Sheng, R. W. Brodersen, "Low-Power CMOS Digital Design," IEEE Journal of Solid-State Circuits, Vol. 27, No. 4, pp. 473-484, April 1992.

[2] C. Y. Tsui, M. Pedram, A. M. Despain, "Technology Decomposition and Mapping Targeting Low Power Dissipation," DAC-30: ACM/IEEE Design Automation Conference, pp. 68-73, Dallas, TX, June 1993.

[3] V. Tiwari, P. Ashar, S. Malik, "Technology Mapping for Low Power," DAC-30: ACM/IEEE Design Automation Conference, pp. 74-79, Dallas, TX, June 1993.

[4] B. Lin, H. de Man, "Low-Power Driven Technology Mapping under Timing Constraints," ICCD'93: IEEE International Conference on Circuits Design, pp. 421-427, Cambridge, MA, October 1993.

[5] R. I. Bahar, H. Cho, G. D. Hachtel, E. Macii, F. Somenzi, "Timing Analysis of Combinational Circuits using ADDs," EDAC-94: IEEE European Conference on Design Automation, Paris, France, February 1994.

[6] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, F. Somenzi, "Algebraic Decision Diagrams and their Applications," ICCAD-93: ACM/IEEE International Conference on Computer Aided Design, pp. 188-191, Santa Clara, CA, November 1993.

[7] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0," Technical Report, Microelectronics Center of North Carolina, Research Triangle Park, NC, January 1991.

[8] F. N. Najm, "Transition Density, a Stochastic Measure of Activity in Digital Circuits," DAC-28: ACM/IEEE Design Automation Conference, pp. 644-649, San Francisco, CA, June 1991.

[9] A. Ghosh, S. Devadas, K. Keutzer, J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," DAC-29: ACM/IEEE Design Automation Conference, pp. 253-259, Anaheim, CA, June 1992.

[10] E. M. Sentovich, K. J. Singh, C. W. Moon, H. Savoj, R. K. Brayton, A. Sangiovanni-Vincentelli, "Sequential Circuits Design Using Synthesis and Optimization," ICCD-92: IEEE International Conference on Computer Design, pp. 328-333, Cambridge, MA, October 1992.

[11] A. Saldanha, R. K. Brayton, A. L. Sangiovanni-Vincentelli, "Circuit Structure Relations to Redundancy and Delay: The KMS Algorithm Revisited," DAC-29: ACM/IEEE Design Automation Conference, pp. 245-248, Anaheim, CA, June 1992.