Maximum Leakage Power Estimation for CMOS Circuits*

S. Bobba and I. N. Hajj
Coordinated Science Lab & ECE Dept.
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
E-mail: {bobba, i-hajj}@uiuc.edu

Abstract

Low supply voltage requires the device threshold to be reduced in order to maintain performance. As the device threshold voltage is reduced, it results in an exponential increase of leakage current in the subthreshold region. The leakage power is no longer negligible in such low voltage circuits. Estimates of maximum leakage power can be used in the design of the circuit to minimize the leakage power. The leakage power is dependent on the input vector. This input pattern dependence of the leakage power makes the problem of estimating the maximum leakage power a hard problem. In this paper, we present graph based algorithms for estimating the maximum leakage power. These algorithms are pattern-independent and do not require simulation of the circuit. Instead the circuit structure and the logic functionality of the components in the circuit are used to create a constraint graph. The problem of estimating the maximum leakage power is then transformed to an optimization problem on the constraint graph. Efficient algorithms on the graph are used to estimate the maximum leakage power dissipated by a circuit. We also present comparisons with exhaustive/long simulations for MCNC/ ISCAS-85 benchmark circuits to verify the accuracy of the method.

1: Introduction

Power dissipation is a critical issue in present day VLSI circuit design. Integrated circuits with large power dissipation require expensive packaging to ensure proper heat dissipation. Excessive power dissipation results in over-heating of the components in the circuit and makes them susceptible to failure. Also, the increasing use of portable computing makes power minimization an important objective in circuit design. Integrated circuits used in portable applications need to consume less power because it extends the battery life. Since the power is proportional to the square of the supply voltage, the supply voltage has been scaled to design low power circuits. To reduce the effect of reduced supply voltage on the performance, the threshold voltages have also been lowered [1]. This however increases the leakage power due to the increased subthreshold currents. Hence, CAD tools to estimate the leakage power are required for the design of low-voltage, low-power circuits.

Event driven systems triggered by an external event with long period of inactivity may have significant leakage power dissipation. For instance, embedded microcontrollers spend most of their time in an inactive state (standby mode). Although the magnitude of the leakage current in the standby mode is significantly smaller than the current in normal operation, the standby energy consumption can be a dominant component due to the large portion of time the circuit spends in standby mode. The leakage power dissipated by a circuit in the standby mode is dependent on the input vector to the circuit [2, 3, 4, 5]. The total number of input vectors for a circuit with N inputs is 2^N. An estimate of the leakage current (power) in the standby mode for a particular input vector can be obtained by circuit simulation.

* This work was supported by SRC under contract 98-DJ-109

A circuit can enter and exit the standby mode a large number of times, each time having a different input vector. The average leakage power in the standby mode can be computed using the following parameters: the probability of each input vector in the standby mode and the leakage power dissipated by the circuit for that input vector. In general, the number of possible input vectors can be large and the probability values for the input vectors in the standby mode can be application dependent. Hence, it may be difficult to accurately estimate the average leakage power dissipated by a circuit in the standby mode and the circuit designer may not be able to ensure that the system leakage power specifications are met. In such a case, an estimate of the maximum leakage power (current) can be used in the design of the circuit to guarantee that the standby power dissipation constraints for a circuit are met. The maximum leakage power is the maximum value of the leakage power dissipated by a circuit over all input vectors and is an upper-bound on the average leakage power. The maximum leakage power (current) can also be used to find the leakage hot-spots and estimate the worst-case battery life. Estimates of the maximum leakage power can be used to minimize the leakage power by design techniques such as the dual threshold optimization technique [6] and the sizing of transistors. Another approach to minimize leakage power is to apply in the standby mode, the input vector that causes minimum leakage power dissipation [3].

The input pattern dependence makes the problem of determining the maximum leakage power dissipated by the circuit a hard problem. The exact solution to the problem requires an exhaustive search of the exponential input vector search space. In this work, we transform the circuit into the constraint graph and perform optimization on the constraint graph to obtain estimates of the maximum leakage power. In the next section, we describe the method used to obtain the leakage power values for each input assignment to a logic block. These leakage power values are used as weights on the vertices of the constraint graph. Logic and structural constraints are represented as edges in the constraint graph. In section 3, we describe the technique to construct the graph using the circuit information. In section 4, we present the algorithms for optimization on the constraint graph to estimate the maximum leakage power. In section 5, we present the experimental results. Finally, in section 6 we present the conclusions.

2: Leakage Current: Models and Characterization

The two main sources of leakage current are: reverse-biased diode leakage current and the subthreshold leakage through the channel of an *OFF* transistor. The diode leakage occurs from the source or drain to the substrate through the reverse-biased diode when a transistor is turned off. For instance, in case of an inverter with low input voltage, the NMOS is turned *OFF* and the PMOS is turned *ON*. The output voltage will be high because the PMOS is *ON*. Hence, the drain-to-substrate voltage of the *OFF* NMOS transistor is equal to the supply voltage. This results in a current leakage from the drain to the substrate through the reverse biased diode. The magnitude of the diode leakage current is dependent on the area of the drain diffusion and the leakage current density, which is set by the technology. The subthreshold current is the drain-source current of an *OFF* transistor. This is due to the diffusion current of the minority carriers in the channel for a MOS device operating in the weak inversion mode (subthreshold region). For instance, in case of an inverter with low input voltage, the NMOS is turned *OFF* and the output voltage is high. Even if the V_{GS} is $0V$, there is still a current flowing in the channel of the *OFF* NMOS transistor due to the V_{DS} potential of V_{dd}. The magnitude of the subthreshold current is a function of temperature, supply voltage, device size and the process parameters. The process parameter that has a dominant effect on the subthreshold current values is the threshold voltage (V_T). Reducing V_T results in an exponential increase in the subthreshold current.

In CMOS circuits, the logic blocks (gates) consist of series and/or parallel connected transistors. The leakage power dissipated by a circuit can be estimated as the sum of the leakage power dissipated by each of the logic blocks in the circuit. In this work, we define the logic block as the set of DC or channel connected transistors. This is the most general representation of the logic block and it can be used to estimate the leakage power dissipation of a static or dynamic or pass-transistor

Table 1. Leakage power using HSPICE for a two input NAND gate

Inputs A	Inputs B	Output O	Leakage Power (in pW)
0	0	1	19.4844
0	1	1	27.3161
1	0	1	40.3637
1	1	0	68.9426

logic block in the inactive mode. The current drawn by the logic block is dependent on the the configuration of the *ON* and *OFF* transistors. The *OFF* transistors draw the leakage current and the *ON* transistors provide the conducting paths to the power supply nodes. The magnitude of the leakage current for a logic module is determined by the configuration of the transistors in the logic module, dimensions of the transistors, the input logic value to the logic module, temperature, supply voltage and other process parameters. The leakage power (P^i_{leak}) for a logic module i in an inactive or stand-by state can be denoted as,

$$P^i_{leak} = V_{dd} \;\; I^i_{leak}, \tag{1}$$

where V_{dd} is the nominal supply voltage and I^i_{leak} is the module i leakage current. In this work, we consider static CMOS gates only. This method can also be applied to dynamic or pass-transistor circuits. The estimates of leakage power for each input assignment to a logic gate are obtained using circuit level simulator HSPICE. This is a one-time leakage power characterization of the logic cells in a technology library. This characterization can also be performed along with the delay or power characterization of logic cells in a technology library. The BSIM3 transistor model parameters of MOSIS 0.35μ process are used with the nominal supply voltage of 3.3V to estimate the leakage power. The nominal temperature was 25°C. For simplicity all transistors are assumed to have the same channel length of 0.4μ, while the channel width for PMOS and NMOS transistors are assumed to be 4.0μ and 2.0μ respectively. Table 1 shows the leakage power values for all the input assignments of a two-input NAND gate. It can be seen that the maximum leakage power dissipation occurs when the two parallel PMOS transistors are *OFF* and the minimum leakage power dissipation occurs when the two series NMOS transistors are *OFF*.

3: Graph Formulation

Using the characterization technique described in the previous section, the leakage power for each input assignment to a logic block can be obtained. A trivial upper bound estimate of the maximum leakage power can be obtained as the sum of the maximum leakage power for each of the logic blocks in the circuit. This assumes that all the logic blocks can simultaneously draw their respective maximum leakage current from the power supply. Due to spatial correlations of the logic lines, all the blocks may not draw their respective maximum current simultaneously. In this section we describe the structural and logic constraints that account for some of these spatial correlations.

Every logic block with k inputs has 2^k different input assignments. Each logic block with k inputs is represented using 2^k nodes which correspond to the different input assignments to the logic block. For each input assignment the leakage power dissipated by the logic block can be different. The leakage power dissipated by the circuit for an input assignment to the logic block is used as the weight on the vertex that corresponds to that input assignment to the logic block. The vertices in the constraint graph for a circuit consists of all the nodes due to each of the logic block in the circuit.

Logic blocks cannot take input assignments independent of other block in the circuit. A logic assignment to a logic block can imply a logic value at the input of some other block. This results in

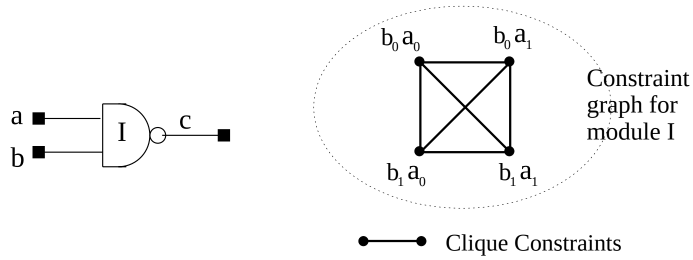

Figure 1. Clique constraints

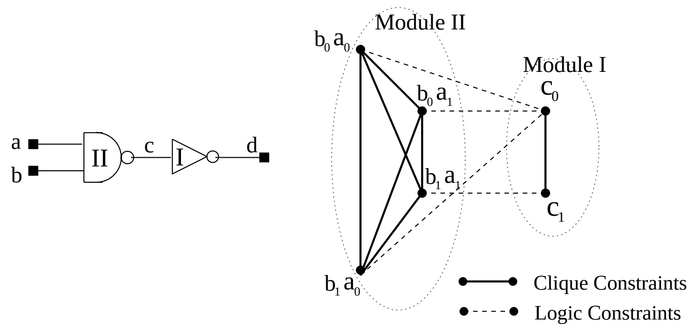

Figure 2. Logic constraints

constraints between the vertices that correspond to the logic assignments to the two logic blocks. The structural relationship due to the connectivity of logic blocks can also result in constraints between vertices in the constraint graph. An edge is used to denote a constraint. An edge between two vertices implies that the logic assignments corresponding to the two vertices are incompatible. Under any input vector the two input assignments can never occur simultaneously. Hence, the modules cannot simultaneously dissipate the leakage power corresponding to the input assignments for any input vector. Observe that the constraints tighten the trivial upper bound by enforcing the conflicts due to the spatial correlations. The logical and structural constraints described next account for these correlations. There are three types of constraints:

1. Clique constraints: This constraint enforces the rule that only one of the 2^k input assignments to a k input logic block can be present for any input vector. This is represented by a set of edges between every pair of vertices corresponding to the 2^k input assignments to a logic block. These constraints are called the clique constraints. Fig. 1 shows the clique constraints for a two input NAND gate. A two input NAND gate has four different input assignments. Each of these is represented as a vertex in the constraint graph.

2. Logic constraints: This constraint enforces the logic functionality of a logic block. This constraint results in edges between vertices of logic blocks that are input-output related. These constraints ensure that the output node to a logic block has a consistent logic value as implied by the input assignment to the logic block. Hence, this constraint can add edges between the vertices corresponding to all the fanout blocks of a logic block to the vertices of the logic block. These constraints are called the logic constraints. Fig. 2 shows the logic constraints for a two input NAND gate driving an INVERTER. A two input NAND gate has four different input assignments. An INVERTER has two different input assignments. Each of these is represented as a vertex in the constraint graph. If an input to the NAND gate is LO then the output is HI and if both the the inputs are HI then the output is LO. These are represented as logic constraint edges in the constraint graph.

3. Stem constraints: This constraint enforces the rule that a stem node can have only one logic value. This constraint results in edges between vertices of logic blocks that are driven by the same stem node. These constraints ensure that the input node to a logic blocks driven by

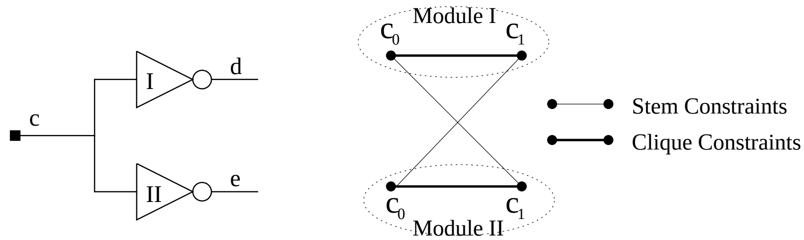

Figure 3. Stem constraints

the same stem has a unique logic value. Hence, this constraint can add edges between the vertices corresponding to all the fanout blocks driven by a stem. These constraints are called the stem constraints. Fig. 3 shows the stem constraints for a node driving two INVERTERs. An INVERTER has two different input assignments. Each of these is represented as a vertex in the constraint graph. The stem node can take only one logic value. This is represented as stem constraint edges in the constraint graph.

4: Optimization on the Constraint Graph

A vertex in the constraint graph denotes an input assignment to a logic module. The weight on the vertex denotes the leakage power dissipated by the logic module for the corresponding input assignment to the logic block. An edge between two vertices denotes that the input assignments corresponding to the vertices are incompatible. Hence, the problem of finding the maximum leakage power dissipation reduces the problem of finding a set of vertices in the constraint graph such that the sum of weights on the vertices is maximum and there are no edges between any pair of the vertices. This problem is exactly the problem of finding the maximum weight independent set in the constraint graph. An independent set in a graph is defined as a set of vertices with no edges between any pair of the vertices in the set. A maximal independent set is an independent set for which none of the vertices in the remaining graph can be appended to increase the size of the independent set. A maximum weight independent set in a graph is a maximal independent set for which the sum of weights on the vertices is maximum over all maximal independent sets.

An exact solution to the maximum weight independent set gives an *upper*-bound on the maximum leakage power dissipated by a circuit. The rules enforced by the clique, logic and stem constraints always hold for any input vector. The clique constraint enforces the rule that for any input vector there is only one input logic assignment for each logic block. The logic constraint enforces the logic functionality of a logic block. The stem constraint enforces the rule that a stem node can take only one logic value. These constraints always hold for any input vector. But, the rules enforced by the clique, logic and stem constraints do not account for all the spatial correlations. Due to reconvergent fan-out it may be possible that a particular input assignment to a logic block can never occur for *any* input vector. In the construction of the constraint graph, we assume that *all* the input assignments to a logic block are possible and denote each of them with a vertex. Solving for the maximum weight independent set results in a search of a larger solution space. The correlations due to reconvergent fan-out can reduce the solution space by eliminating a few vertices corresponding to the input assignments that can never be excited for any input vector. Hence, the exact solution to the maximum weight independent set on the constraint graph yields an *upper*-bound on the maximum leakage power dissipation of the circuit.

The problem of determining the maximum weight independent set in an arbitrary graph is NP-Complete [7]. There exists some classes of graphs like perfect graphs, claw-free graphs for which the problem can be solved in polynomial time. Since the constraint graph does not belong to any of these special classes of graphs, it is not known if the maximum weight independent set problem can be solved in polynomial time for the constraint graph. In this work, we use fast linear time greedy algorithms to estimate the maximum weight independent set in the constraint graph. The greedy algorithms generate a *lower*-bound on the maximum weight independent set. Since the exact solution to the maximum weight independent set is an *upper*-bound on the maximum leakage power dissipation, the greedy algorithms for the maximum weight independent set can generate good estimates for the maximum leakage power dissipation. In the next sub-section, we describe three greedy algorithms for estimating the maximum weight independent set in the constraint graph. Each of the greedy algorithms is applied on the constraint graph and the maximum (best) solution of the three algorithms is taken as the estimate of the maximum leakage power dissipation of the circuit for which the constraint graph was constructed.

4.1: Greedy Algorithms for Maximum Weight Independent Set

The greedy algorithms pick a vertex in the graph using some gain function and the selected vertex and the vertices adjacent to it are then deleted to obtain the new subgraph. The greedy algorithm is iterated on the subgraph till all the vertices in the graph are deleted. Different gain functions may give different solutions for the same constraint graph. The different gain functions place different emphasis on the following parameters: weight on a vertex, weights on the neighbors of a vertex, number of neighbors of a vertex. The idea behind a gain function is to combine different quantities associated with a vertex into a single number so that one can pick a *good* locally optimal vertex. For the maximum weight independent set, a vertex with a large weight and a small number of neighbors or small value for the sum of weights on neighbors can be a *good* choice. One can obtain a large number of different greedy algorithms for the maximum weight independent set by changing the gain function. We have experimented with different heuristics and we present the experimental results for three gain functions that we observed perform well consistently. The three algorithms are applied to the constraint graph and the maximum (best) solution of the three algorithms is picked as the estimate of the maximum weight independent set in the constraint graph. These linear time greedy heuristics are fast and give a lower-bound estimate on the maximum weight independent set in the constraint graph.

Let v_i denote the vertex i and $N(v_i)$ denote the list of neighbors of vertex v_i in the graph. Let $w(v_i)$ denote the weight on the vertex v_i. The three gain functions are defined below:

- G1: This gain function uses the weight of a vertex and the weights of the neighbors of the vertex in the graph to compute the gain for the vertex. The gain for a vertex i is given by,

  $$g(v_i) = w(v_i) - \sum_{j \in N(v_i)} w(v_j). \tag{2}$$

  The vertex i with the maximum gain $g(v_i)$ in the graph is picked.

- G2: This gain function uses the weight of a vertex and the number of neighbors of the vertex to compute the gain for the vertex. The gain for a vertex i is given by,

  $$g(v_i) = \frac{w(v_i)}{1 + \ |N(v_i)|} \tag{3}$$

  The vertex i with the maximum gain $g(v_i)$ in the graph is picked.

- G3: This gain function uses the weight of a vertex and the number of neighbors of the vertex (in case of conflict) to compute the gain for the vertex. The gain for a vertex i is given by,

  $$g(v_i) = w(v_i) \tag{4}$$

  The vertex i with the maximum gain $g(v_i)$ in the graph is picked. If there is more than one vertex with the same weight, then the vertex with the minimum number of neighbors is picked.

In the next section, we present the experimental results for ISCAS-85/ MCNC benchmark circuits and compare the maximum leakage power dissipation obtained using the greedy heuristics on the constraint graph with exhaustive or long random input vector simulations.

Table 2. Comparison of graph based methods with exhaustive simulation for estimating the maximum leakage power

Ckt	Num. PI	P_{g1}	P_{g2}	P_{g3}	Run Time	Trivial UB	P^{g}_{leak} graph	P_{leak} exhaust.	% error
9symml	9	7.451	7.508	7.511	11.88	11.545	7.511	8.207	-8.48
C17	5	0.327	0.327	0.327	0.21	0.413	0.327	0.327	0.00
alu2	10	13.338	13.525	13.475	50.34	19.662	13.525	14.467	-6.51
alu4	14	27.369	26.244	25.970	244.99	38.819	27.369	28.373	-3.53
b1	3	0.448	0.394	0.373	0.27	0.563	0.448	0.480	-6.66
cm151a	12	1.240	1.041	1.041	0.36	1.689	1.240	1.254	-1.11
cm152a	11	1.130	1.053	1.053	0.34	1.551	1.130	1.143	-1.13
cm162a	14	2.037	2.032	1.859	0.22	2.715	2.037	2.185	-6.77
cm163a	16	1.830	1.859	1.855	0.49	2.284	1.859	1.925	-3.42
cm42a	4	1.099	1.125	1.285	0.34	1.718	1.285	1.171	9.73
cm82a	5	1.076	0.990	0.943	0.29	1.557	1.076	1.151	-6.51
cm85a	11	2.177	2.116	1.719	0.67	2.787	2.177	2.241	-2.85
cmb	16	2.541	2.137	2.456	0.77	3.315	2.541	2.502	1.55
cu	14	2.526	2.540	2.601	0.89	3.456	2.601	2.680	-2.94
f51m	8	3.327	3.320	2.988	1.20	4.723	3.327	3.552	-6.33
majority	5	0.474	0.465	0.527	0.21	0.695	0.527	0.519	1.54
parity	16	3.179	2.561	2.368	0.87	4.136	3.179	3.179	0.00
pm1	16	2.051	2.003	2.078	0.63	2.752	2.078	2.147	-3.21
t481	16	12.847	13.595	13.537	36.69	18.829	13.595	14.578	-6.74
x2	10	2.138	2.105	2.048	0.66	2.704	2.138	2.180	-1.92
z4ml	7	1.721	1.761	1.517	0.57	2.582	1.761	1.868	-5.72

5: Experimental Results

The experimental results were obtained on the MCNC/ISCAS-85 benchmark circuits [8]. Each of these combinational multilevel circuits were optimized by *script.rugged* and mapped to a technology library consisting of NAND, NOR, INVERTER and BUFFER using SIS [9]. In this paper, we present a comparison of the accuracy of the maximum leakage power dissipation generated using the greedy heuristics on the constraint graph and the exhaustive/random input vector simulations. Exhaustive simulation is performed for circuits with a small number of primary inputs. Logic simulations for 100000 randomly generated input vectors are performed for larger circuits. The logic simulator used in the simulations is a zero-delay simulator and the pre-characterized leakage power values for each logic module were used to compute the leakage power dissipated by the circuit for a particular input vector. The run time values are in CPU seconds on UltraSparc2 workstation. The leakage power values are in nano-Watts.

Table 3 shows the comparison of the trivial upper bound on the maximum leakage power dissipation, and the maximum leakage power dissipation obtained by exhaustive simulation with the maximum leakage power obtained using the graph based methods. The second column (Num. PI) denotes the number of primary inputs for the specified circuit. P_{g1}, P_{g2} and P_{g3} denote the estimates of the maximum leakage power obtained using the greedy heuristics on the constraint graph for gain functions G1, G2 and G3 respectively. The run time values correspond to the CPU time used for computing the maximum leakage power using the three greedy algorithms. The trivial upper bound on the leakage power dissipation is obtained as the sum of the maximum leakage power for each logic block in the circuit. P^{g}_{leak} denotes the maximum leakage power obtained using the graph based methods computed as the maximum value of P_{g1}, P_{g2} and P_{g3}. P_{leak} denotes the maximum leakage power dissipation obtained by exhaustive simulation. The % error is computed as $100 * (P^g_{leak} - P_{leak})/P_{leak}$. It can be seen that the graph based method generates a significantly tighter bound than the trivial upper bound. Also the graph based method generates maximum leakage power values very close to the actual values. The graph based algorithms are fast and the run time requirements are nominal.

Table 4 shows the comparison of the trivial upper bound on the maximum leakage power dissipation, and the maximum leakage power dissipation obtained by random input vector simulation for 100000 input vectors with the maximum leakage power obtained using the graph based methods. The notations for the terms in the table is the same as described before. Since the random input vector simulation for 100000 covers only a small fraction of the entire search space, the maximum leakage power dissipation value (P^l_{leak}) obtained using the random input vector simulation is only a lower bound on the maximum leakage power dissipation. Hence, we do not present the % error computation for this case. From the results it can be seen that the graph based method generates good estimates of the maximum leakage power for a circuit. The CPU time requirements of these algorithms are nominal.

6: Conclusions

We have presented an input pattern independent algorithm for computing the maximum leakage power dissipation of a circuit. We use the circuit structure and functional information to transform the problem to a graph problem and use graph-theoretic algorithms to find the solution. The exact solution to the maximum weight independent set on the constraint graph gives an *upper*-bound estimate of the maximum leakage power. The greedy heuristics for the maximum weight independent set give a *lower*-bound solution. Hence, the linear time greedy heuristics for the graph problem generate good estimates of the maximum leakage power for a circuit. The method we presented is quite general and it can be applied to dynamic and pass-transistor circuits. We have presented comparisons with results obtained by exhaustive/long random input vector simulations to show that the constraint graph based method generates tight results. The graph based algorithms for computing the maximum leakage power dissipation are fast and they require only a few minutes of CPU time for the largest circuit.

References

[1] A. Chandrakasan, I. Yang, C. Vieri, and D. Antoniadis, "Design considerations and tools for low-voltage digital system design," in *Proc. of DAC*, pp. 113-118, June 1996.

[2] R. X. Gu and M. I. Elmasry, "Power dissipation analysis and optimization of deep submicron CMOS digital circuits," *IEEE Journal of solid-state circuits*, vol. 31, no. 5, pp. 707–713, May 1996.

[3] J. P. Halter and F. N. Najm, "A gate-level leakage power reduction method for ultra-low-power CMOS circuits," in *Proc. of CICC*, pp. 475–478, May 1997.

[4] A. Ferre and J. Figueras, "On estimating leakage power consumption for submicron CMOS digital circuits," in *Proc. of PATMOS*, pp. 269–279, Oct. 1997.

[5] D. T. Blaauw, A. Dharchoudhury, R. Panda, S. Sirichotiyakul, C. Oh, and T. Edwards, "Emerging power management tools for processor design," in *Proc. of ISLPED*, pp. 143–148, Aug. 1998.

[6] L. Wei, Z. Chen, M. Johnson, and K. Roy, "Design and optimization of low voltage high performance dual threshold CMOS circuits," in *Proc. of DAC*, June 1998.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability*. New York, NY: W. H. Freeman and Company, 1979.

[8] S. Yang, *Logic synthesis and optimization benchmarks user guide, Version 3.0*. MCNC, Research Triangle Park, NC, 1991.

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

Table 3. Comparison of graph based methods with random input vector simulation for estimating the maximum leakage power

Ckt	Num. PI	P_{g1}	P_{g2}	P_{g3}	Run Time	Trivial UB	P^{g}_{leak} graph	P^{l}_{leak} random
C1355	41	20.333	20.417	20.050	96.72	29.987	20.417	20.428
C1908	33	20.897	19.575	19.077	91.40	29.418	20.897	20.709
C2670	233	38.252	38.007	36.656	390.13	49.906	38.252	38.699
C432	36	8.349	8.106	8.147	9.29	11.376	8.349	8.831
C880	60	18.416	18.203	17.620	63.52	24.499	18.416	18.564
C3540	50	50.154	50.593	48.945	789.18	73.570	50.593	51.814
C5315	178	72.774	71.562	70.675	1296.66	101.731	72.774	72.290
C6288	32	121.149	121.147	108.916	2854.74	163.793	121.149	114.287
C7552	207	94.850	95.629	94.763	3209.24	139.180	95.629	98.065
apex6	135	31.148	30.693	29.953	258.73	42.814	31.148	30.843
apex7	49	9.539	9.600	9.295	17.02	13.493	9.600	10.335
b9	41	5.314	5.200	5.437	3.05	7.154	5.437	5.489
c8	28	5.117	5.741	5.082	3.71	7.752	5.741	5.752
cc	21	2.499	2.268	2.340	0.74	3.237	2.499	2.545
cht	47	6.173	6.191	5.752	5.38	8.655	6.191	6.128
cm150a	21	2.178	2.151	2.191	0.57	3.134	2.191	2.286
comp	32	5.329	4.755	4.479	1.93	6.270	5.329	5.137
cordic	23	2.947	2.883	2.634	0.90	4.088	2.947	3.022
count	35	5.819	5.625	5.919	6.40	7.550	5.919	5.897
dalu	75	31.672	31.567	30.892	209.88	43.819	31.672	31.598
example2	85	13.000	12.438	12.018	34.94	17.642	13.000	12.957
frg2	143	31.038	30.163	30.971	260.71	40.461	31.038	30.141
i1	25	2.131	2.152	2.252	0.53	2.646	2.252	2.250
i2	201	12.598	10.406	12.152	8.69	13.074	12.598	8.769
i3	132	6.527	5.294	6.812	2.57	7.376	6.812	5.911
i4	187	14.558	14.835	15.311	19.73	17.895	15.311	13.026
i5	133	9.017	7.623	8.013	5.36	11.375	9.017	8.160
i6	138	23.515	23.548	19.585	182.32	26.227	23.548	21.426
i7	199	30.627	30.771	27.808	348.45	35.042	30.771	28.064
i8	133	43.145	42.821	42.042	704.25	56.482	43.145	43.678
k2	45	46.768	46.509	48.148	635.54	60.320	48.148	46.734
pair	173	66.709	66.450	65.675	1059.26	93.464	66.709	66.725
unreg	36	5.129	4.828	4.694	3.62	6.267	5.129	4.991
vda	17	24.906	24.393	25.298	147.01	32.413	25.298	25.006
x3	135	31.715	31.350	30.045	221.31	43.893	31.715	31.419
x4	94	16.055	15.018	15.809	59.21	20.824	16.055	15.981