Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modeling of Transistor Stacks *

Zhanping Chen, Mark Johnson, Liqiong Wei, and Kaushik Roy
School of Electrical and Computer Engineering
Purdue Univ., W. Lafayette, IN 47907-1285
{zhanping,mcjohnso,liqiong,kaushik}@ecn.purdue.edu

Abstract

Low supply voltage requires the device threshold to be reduced in order to maintain performance. Due to the exponential relationship between leakage current and threshold voltage in the weak inversion region, leakage power can no longer be ignored. In this paper we present a technique to accurately estimate leakage power by accurately modeling the leakage current in transistor stacks. The standby leakage current model has been verified by HSPICE. We demonstrate that the dependence of leakage power on primary input combinations can be accounted for by this model. Based on our analysis we can determine good bounds for leakage power in the standby mode. As a by-product of this analysis, we can also determine the set of input vectors which can put the circuits in the low-power standby mode. Results on a large number of benchmarks indicate that proper input selection can reduce the standby leakage power by more than 50% for some circuits.

1 Introduction

The increasing use of portable computing and wireless communication systems makes power dissipation a major concern in low power circuit designs [2, 21]. Therefore, accurate power estimation tools are of critical importance.

In CMOS digital circuits, power dissipation consists of dynamic and static components. In circuits with a high supply voltage, a relatively high transistor threshold voltage can be used, making subthreshold current negligible. That is the common assumption for the existing techniques for average power estimation [1] [3] [4] [5] [6] [7] [8] [15] [16] [17] [18] [19] [23] [24]. However, low power applications have been driving the supply voltage to become lower and lower, which requires the device threshold to be reduced so as to satisfy performance requirements. Unfortunately, due to the exponential relationship between leakage current and threshold voltage, such scaling leads to a dramatic increase in leakage current. Consequently, leakage power is no longer negligible in low voltage circuits. This necessitates the development of accurate estimation tools for leakage power.

The accuracy of leakage power estimation is critically dependent on the standby leakage current model. In this paper, we have developed an accurate standby leakage current model which has been verified by HSPICE. Considering reverse biasing between gate and source in transistor stacks, DIBL (Drain Induced Barrier Lowering), and the body effect, the leakage power of a circuit depends on primary input combinations. Hence, the primary input combination(s) corresponding to the minimum leakage power can be applied to the circuit during standby mode to minimize the leakage power, and thereby leading to a reduction in total power consumption. Minimizing leakage can be especially important in battery powered applications where leakage drains the battery when a circuit is idle for a long time.

The most straight-forward way to find the minimum leakage power is to enumerate all combinations of primary inputs. For a circuit with n primary inputs, there are 2^n combinations. Due to the exponential complexity with respect to the number of primary inputs, such an exhaustive method is limited to circuits with a small number of primary inputs. The method proposed in [10] uses a random search to determine low-leakage states of the circuit without regard to the circuit structure or the underlying mechanism of leakage power reduction. The bounds obtained using the above technique are also not tight. In this paper, we developed an accurate leakage model considering transistor stacks and used that model in a genetic algorithm framework to estimate the minimum and maximum values for leakage power dissipation.

This rest of this paper is organized as follows. Section 2 presents an accurate standby leakage current model in transistor stacks with the consideration of body effect and DIBL. Section 3 describes the genetic algorithm to estimate the minimum and maximum values for leakage power. Implementation details and experimental results for MCNC and ISCAS benchmark circuits are shown in Section 4. Finally, conclusions are given in Section 5.

* This research was supported in part by Intel, DARPA (F33615-95-C-1625), NSF CAREER award (9501869-MIP), and AT&T

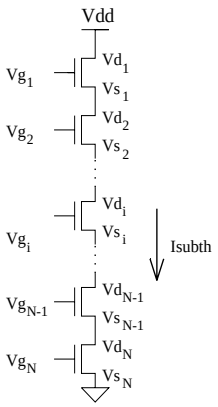

Figure 1: Schematic and notation for stacking effect analysis

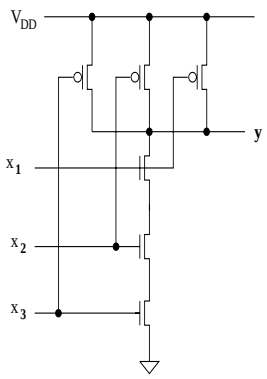

Figure 2: Schematic for a 3-input NAND

2 An Accurate Leakage Current Model

An accurate estimate of standby leakage power must consider circuit topology as well as signal levels when the circuit is idle. Kawahara [13] demonstrated this in the design of a low power decoded-drivers for a DRAM. An extra transistor was placed between the supply line and the pull-up transistor for the driver. This causes a slight reverse bias between the gate and source of the pull-up transistor when both transistors are turned off. Because subthreshold current is exponentially dependent on gate bias, a substantial current reduction was obtained. This phenomenon is referred to as the "stacking effect".

We have developed a more general model of the stacking effect [12]. This model considers the general case of transistor stacks of arbitrary height. It takes into account both body effect and DIBL. DIBL (reduction of threshold voltage as V_{DS} increases) is especially significant for sub-micron devices. The leakage of a transistor stack is shown to directly depend on the magnitude of the DIBL effect.

Let Figure 1 depict a transistor stack to be analyzed. Steady state leakage values can be estimated as a function of the number of transistors that are turned off. A more detailed derivation can be found in [12]. The general approach is to equate the subthreshold current through each transistor and then solve for the voltage (V_{DS_i}) across each transistor. These voltages can then be used to estimate the magnitude of the leakage current. The following analysis is done for an NMOS pull down stack, but is equally applicable to a PMOS stack.

From the BSIM2 MOS transistor model [11, 22, 25], the subthreshold current of a MOSFET can be modeled as

$$I_{subth} = A\, e^{\frac{q}{nkT}(V_G - V_S - V_{TH_0} - \gamma' V_S + \eta V_{DS})}(1 - e^{\frac{-qV_{DS}}{kT}}) \quad (1)$$

where $A = \mu_0 C'_{ox} \frac{W}{L_{eff}} (\frac{kT}{q})^2 e^{1.8}$. V_G, V_D and V_S are the gate voltage, drain voltage, and source voltage of the transistor, respectively. The bulk is connected to ground. V_{TH_0} is the zero bias threshold voltage. The body effect for small values of V_S is very nearly linear. It is represented by the term $\gamma' V_S$, where γ' is the linearized body effect coefficient. η is the DIBL coefficient, representing the effect of V_{DS} ($V_{DS} = V_D - V_S$) on threshold voltage. C_{ox} is the gate oxide capacitance. μ_0 is the zero bias mobility. n is the subthreshold swing coefficient of the transistor. For the conditions illustrated in Figure 1, the gate voltage is zero.

First we equate the current of the first and second transistor in the stack. We obtain equation (2) by solving for V_{DS_2} in terms of V_{dd} and BSIM model parameters.

$$V_{DS_2} = \frac{nkT}{q(1 + 2\eta + \gamma')} ln(\frac{A_1}{A_2} e^{\frac{q\eta V_{dd}}{nkT}} + 1) \quad (2)$$

One can similarly equate the current through the $(i-1)^{th}$ and i^{th} transistors, solving for V_{DS_i} in terms of $V_{DS_{i-1}}$. This results in equation (3). Equation (3) can be used iteratively to find V_{DS_i} for each transistor, starting with the third in the stack. Finally, V_{DS_1} can be determined (if desired) by subtracting the sum of these V_{DS_i} values from V_{dd}.

$$V_{DS_i} = \frac{nkT}{q(1 + \gamma')} ln(1 + \frac{A_{i-1}}{A_i}(1 - e^{-\frac{q}{kT} V_{DS_{i-1}}})) \quad (3)$$

The voltage offset at the source of each transistor is given by

$$V_{S_i} = \sum_{j=i+1}^{N} V_{DS_j} \quad (4)$$

Since we are now only interested in the magnitude of the leakage current, we can use V_{DS_N} in equation (1) to compute the leakage through the bottom transistor. To verify this computation, one could compute the leakage of other transistors in the stack.

Please note that this analysis only considers transistors that are turned off. Transistors that are turned on can be treated as a short circuit. Because of the very small currents involved, the voltage drop across transistors that are turned on will be orders of magnitude smaller than the voltage drop across transistors in the subthreshold region.

In [10], it is noticed that leakage power depends on primary input combinations with no explanation for the mechanism. The model proposed in this paper, on the other hand, can offer a clear explanation. Consider the 3-input CMOS NAND gate illustrated in Figure 2. For the 111 input combination, the three NMOS transistors are turned on and treated as a short circuit, the leakage current of the gate is the sum of the leakage current through the three PMOS transistors. For the 011, 101, and 110 combinations, the leakage current is computed for the NMOS transistor which is turned off. For the 001, 010, and 100 combinations, the V_{DS} of the second off transistor is first obtained by equation (2), substituting into equation (1) yields the leakage current. For the 000 combination, V_{DS_2} is first obtained by equation (2), then V_{DS_3} is computed by equation (3), and $V_{DS_1} = V_{dd} - V_{S_1} = V_{dd} - V_{DS_2} - V_{DS_3}$ by equation (4). Substituting V_{DS_1} and V_{S_1} into equation (1) yields the leakage current. Due to stacking effect and differences in the leakage of PMOS and NMOS, the leakage current of a gate can vary widely with input combinations.

Leakage current is exponentially dependent on the threshold voltage. Consequently, the values of V_{TH_0}, subthreshold swing coefficient, body effect, and DIBL coefficient are critical to the accuracy of this model. Leakage is also very sensitive to temperature, doubling for every 8^o to 10^oK increase.

This leakage model is only meaningful when the circuit has been idle for some time. Anywhere from a few microseconds up to several milliseconds may be required for the circuit to settle to quiescent levels [12].

3 Genetic Algorithm to Obtain Bounds for Leakage Power

Given a primary input combination, the logic value of each internal node can be obtained simply by starting from primary inputs and simulating the circuit level by level (level of a node is equal to the maximum of the level of its fan-in nodes plus 1; level of all primary inputs being 0). After applying the standby leakage current model presented in the previous section, the leakage current of each gate can be evaluated. The leakage power $Plkg$ of the simulated circuit for the given primary input combination is the sum of the leakage power consumed by all transistors, i.e.

$$Plkg = \sum I_{DS_i} V_{DS_i} \tag{5}$$

Note V_{DS_i} and I_{DS_i} depend on primary input combinations. This makes the total leakage power of a circuit also depend on primary inputs.

Enumerative method is the most straightforward method to obtained the minimum and maximum values for standby leakage power dissipation. However, the exponential complexity with respective to the number of primary inputs makes it prohibitive for circuits with large number of primary inputs. The random search method involves generating a large number of primary inputs, evaluating the leakage of each input, and keeping track of the minimum and maximum. One advantage of the random search method is that average leakage power $Plkg_{avg}$ can be obtained by

$$Plkg_{avg} = (\sum_{i=1}^{m} P_{lkg_i})/m \tag{6}$$

as long as the sample length m is long enough, where P_{lkg_i} is the leakage power for primary input combination i. However, the primary input combination is randomly generated with no knowledge of previous information.

Genetic algorithm [9] has the advantage of exploiting historical information to speculate on new search points with expected improved performance. It derives its behavior from a metaphor of natural selection and natural genetics by the creation of a *population* of *chromosomes* or *individuals* represented by artificial strings called *chromosomes*. A general genetic algorithm works as follows. First an initial population is randomly generated. Then the following steps are repeated. An objective function is applied to evaluate the *fitness* of each chromosome in the population. A new population will be produced by operations such as *selection*, *crossover*, and *mutation* in that order. Each such iteration is referred to as a *generation*. A near optimal solution is obtained when stopping criterion is satisfied.

The genetic algorithm used in this paper is similar to the one proposed by Goldberg [9]. For each simulated circuit, 100 initial chromosomes (primary input combinations of zeros and ones) are randomly generated. The objective function is to obtain the fitness (leakage power) for the applied primary input combination based on the leakage current model presented in the previous section. After the 100 chromosomes are evaluated, they are sorted by non-decreasing fitness. Therefore, for the present generation, the first chromosome represents the best case primary input combination which gives the minimum leakage power up to this point. Then a fitness technique called *linear normalization* [14] is adopted to calculate fitnesses that begin with 100 and linearly decrease to 1. It should be emphasized that the selection operation is based on the normalized fitnesses instead of the actual values.

The reason for a fitness technique is quite simple. A commonly used selection technique is the *roulette wheel* technique in which the parent chromosomes are selected with probabilities proportional to their fitnesses. In this way, more highly fit chromosomes have a higher number of children in the succeeding generation. But when fitnesses become closer, things can be different. Consider a generation with the respective fitnesses of the best and the worst chromosomes being 555.15 and 555.75 and the average fitness of the population being 555.45. The best and the worst chromosomes will produce almost identical number of children in the next generation, and hence, the effect of natural selection is almost lost. By using linear normalization fitness technique, we greatly increase the selection pressure in favor of the best chromosome.

After being selected as parents, the chromosomes are entered into a mating pool for further operations such as crossover and mutation. The crossover operation uses parent chromosomes to produce children chromosomes by exchanging substrings of the two parent chromosomes. In this paper, a one-point crossover technique with crossover rate of 0.8 is used. The mutation operation takes each chromosome of a generation and randomly changes the bit value with a given probability called *bit mutation rate*. In this paper, the mutation rate is chosen to be 0.01.

The genetic algorithm will stop after 50 generations and output the minimal leakage power as well as the best case primary input combination. The pseudo code of the genetic algorithm is given as follows.

```
Minimum Leakage Power() {
  GENERATIONS=1; MAX_GEN=50;
  POPULATION_SIZE=100;
  Chromosome_length = the number of primary inputs
  Initialize a population of chromosomes
  do {
    Evaluate each chromosome in the population
    Sort chromosomes by non-decreasing fitnesses
    Apply linear normalization fitness technique
    Roulette wheel selection to select parent chromosomes
    Crossover & mutation to create children chromosomes
    Produce a new generation with children chromosomes
  } While(++GENERATIONS < MAX_GEN)
}
```

It should be pointed out that the above genetic algorithm will give maximum leakage power as well as the worst case

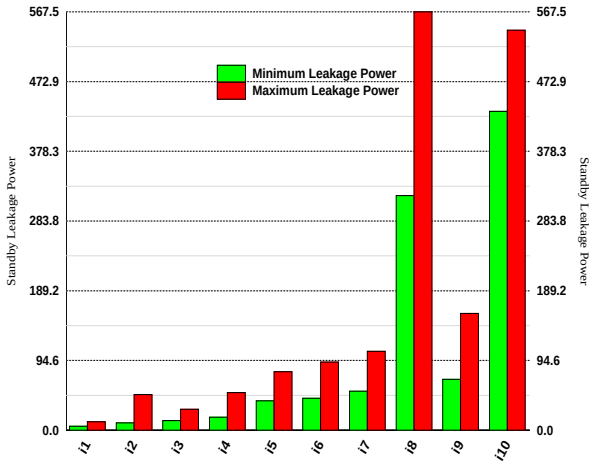

Figure 3: Leakage power in μW for MCNC benchmark circuits

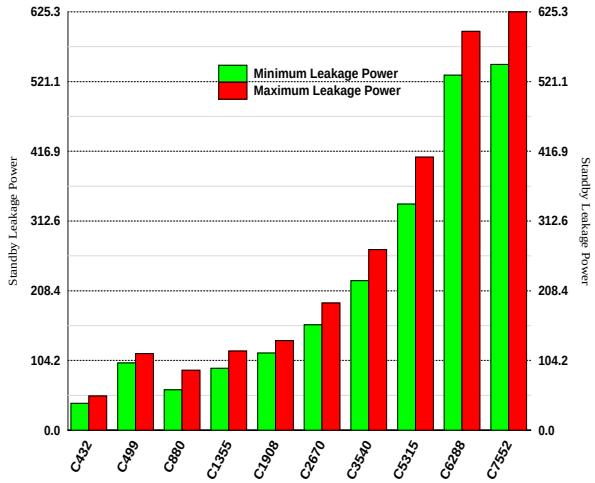

Figure 4: Leakage power in μW for ISCAS benchmark circuits

primary input combination if the chromosomes are sorted by non-increasing fitnesses.

4 Implementation and Experimental Results

The genetic algorithm (GA) to estimate the standby leakage power in CMOS circuits has been implemented in C under the Berkeley SIS environment. In order to simplify the analysis, gate-mapping was used to map the circuits to a library which contains NAND, NOR, INVERTER, and BUFFER. The supply voltage V_{dd} and the zero bias threshold voltage V_{TH0} used in this experiment are 1.0V and 0.2V, respectively. The subthreshold swing coefficient is 1.5, which corresponds to the subthreshold slope of $89.8mv/decade$. η and γ' are 0.05 and 0.24 for NMOS and 0.047 and 0.11 for PMOS, respectively. For simplicity, all transistors are assumed to have the same channel length of 0.3μ while the channel widths for PMOSFETs and NMOSFETs are assumed to be $3.6\mu m$ and $1.8\mu m$, respectively. However, the method is not limited to such assumptions.

4.1 Results for Minimum and Maximum Leakage Power

The validation of the standby leakage current model has been verified in [12]. Figures 3 and 4 illustrate the minimum and maximum leakage power obtained by GA (where the standby leakage current model has been incorporated) for MCNC and ISCAS benchmark circuits, respectively. The exact values are given in columns 7 and 9 of Table 1. Results indicate that for some circuits leakage power varies widely with primary input combinations and the maximum leakage power can be more than twice as large as the minimum. Consider circuit $i2$. The maximum leakage power is 4.89 times larger than the minimum value. Therefore, for some circuits, applying the input combination which gives the minimum leakage power does reduce leakage power significantly.

Column 11 of Table 1 shows the ratios of the maximum leakage power over the minimum obtained by the genetic algorithm. These ratios represent the sensitivity of leakage power with respect to primary input combinations and the possible leakage power saving by applying different primary input combinations. For the ten MCNC benchmarks except $i10$, such ratios are over 1.7, for circuit $i2$, the ratio is even close to 5, which indicates that the leakage power is very sensitive to primary input combinations. For the ten ISCAS benchmarks, on the other hand, most of the ratios are less than 1.3. This indicates that the leakage of ISCAS benchmarks are not sensitive to primary inputs, leaving little freedom to reduce leakage power by applying different primary inputs.

The reason why ISCAS benchmarks are not as sensitive to primary input combinations as MCNC benchmarks is probably due to the fact that ISCAS benchmarks have considerably more levels of logic than MCNC benchmarks. Except for circuit $i10$, all MCNC benchmarks have less than 10 levels (the level number of each circuit is given in column 3). ISCAS benchmarks, on the other hand, have more than 20 levels except for circuits $C432$ and $C499$. The most complicated ISCAS circuit $C6288$ has over 120 levels. As the number of logic levels increases, it becomes more and more difficult to control the state of gates that are several levels away from the primary inputs. On average, the greater the level number, the more insensitive the leakage power to primary input combinations.

4.2 Comparison of GA and Random Search Technique

In order to show the efficiency and effectiveness of the genetic algorithm, the random search (RAND) method is performed for comparison. Sample numbers for GA and RAND used in this experiment are $5,000$ and $50,000$, respectively.

Table 1 reports the results for MCNC and ISCAS benchmark circuits obtained by the random search method and

Table 1: Minimum & maximum leakage power ($Plkg_{min}$,$Plkg_{max}$) for MCNC & ISCAS benchmarks

Circuit	PI's	level	CPU time (s)		$Plkg_{min}(\mu W)$		$Plkg_{max}(\mu W)$		$Plkg_{max}/Plkg_{min}$			$Plkg_{avg}$
Chosen	#	#	Rand	GA	Rand	GA	Rand	GA	Rand	GA	(%)	(μW)
i1	25	5	48.3	4.5	5.24	5.19	11.09	11.37	2.12	2.19	3.3	7.05
i2	201	4	194.8	17.1	11.63	9.89	37.21	48.41	3.20	4.89	52.8	15.89
i3	132	2	134.9	12.2	15.43	13.16	27.89	28.25	1.81	2.15	18.8	19.22
i4	192	4	228.8	20.8	21.32	17.79	46.78	51.10	2.19	2.87	31.1	27.12
i5	133	6	371.5	33.1	46.80	39.95	76.70	79.30	1.64	1.98	20.7	54.92
i6	138	3	478.3	45.5	44.81	43.49	86.91	92.42	1.94	2.13	9.8	70.45
i7	199	3	598.3	58.6	54.26	52.91	100.03	106.94	1.84	2.02	9.8	83.80
i8	133	8	2570.3	254.9	328.55	318.37	526.07	567.50	1.60	1.78	11.3	447.16
i9	88	7	683.8	67.2	72.56	69.08	152.02	158.35	2.10	2.29	9.0	133.81
i10	257	54	2881.8	282.5	446.18	432.28	509.82	542.45	1.14	1.25	9.6	471.59
C432	36	17	222.4	21.5	40.24	40.07	48.77	51.02	1.21	1.27	5.0	44.60
C499	41	11	554.12	53.4	101.75	100.54	111.83	114.56	1.10	1.14	3.6	106.39
C880	60	23	382.5	37.1	63.69	60.44	86.56	89.62	1.36	1.48	8.8	71.83
C1355	41	23	563.2	55.0	92.55	92.55	116.58	118.31	1.26	1.28	1.6	109.14
C1908	33	40	676.0	65.6	116.02	115.35	133.63	133.79	1.15	1.16	0.9	123.72
C2670	233	32	991.4	94.4	159.98	157.65	182.40	189.88	1.14	1.20	5.3	167.90
C3540	50	47	1296.4	127.0	226.51	223.55	265.27	270.07	1.17	1.21	3.4	242.00
C5315	178	49	2201.2	214.2	344.85	338.13	390.70	408.27	1.13	1.21	7.1	367.09
C6288	32	124	2921.8	276.0	530.52	530.52	590.05	595.87	1.11	1.12	0.9	561.76
C7552	207	43	3367.0	327.9	554.47	546.67	605.90	625.29	1.09	1.14	4.6	575.70

by the genetic algorithm. Columns 6 and 7 show the minimum leakage power obtained by RAND and GA, respectively while columns 8 and 9 report the maximum values. The last column of the table gives the average leakage power obtained by equation (6).

Results for minimum leakage power indicate that for 90% of the simulated circuits the genetic algorithm can give a tighter bound than the random search. For the other 10% of the simulated circuits, both GA and RAND give the same minimum values. For the maximum leakage power, the GA runs better for all simulated circuits. Consider circuit $i2$. The maximum value obtained by GA is $48.41\mu W$ which is over 30% higher than $37.21\mu W$, the maximum obtained by RAND.

The tightness of the bounds obtained by RAND and GA can be further illustrated by the percentage difference between the maximum/minimum leakage ratios (next to last column in Table 1). For the MCNC benchmarks, the average percentage difference is nearly 20%. For circuit $i2$, the percentage difference is over 50%.

Column 4 and column 5 in table 1 show the CPU time for an ULTRA SPARC ONE with 256MB of memory. Since the genetic algorithm only searches $5,000$ combinations of primary inputs, it is much faster than the random search method which searches $50,000$ combinations.

Note that the CPU time shown in column 5 is the time for obtaining the minimum leakage power. In order to obtain the minimum and maximum values, the GA has to run twice because the chromosomes in a generation are sorted differently.

5 Summary & Conclusions

With the scaling of supply voltage and transistor threshold, the leakage current is of critical importance. We proposed a novel technique to accurately estimate leakage current in CMOS circuits during the standby mode of operation. Leakage current model uses the effect of transistor stacks in both N-MOS and P-MOS trees and the results indicate that the model is accurate. The leakage current model offers a clear explanation of why the leakage power depends on primary input combinations. A genetic algorithm based technique was used to determine the bounds for leakage power in CMOS circuits due to different input combinations. Results for minimum and maximum leakage power indicate that for some circuits leakage power can vary widely with different primary input combinations. Applying the best primary input combination to a circuit during standby mode will significantly reduce the leakage power.

References

[1] R.Burch, F.Najm, P.Yang and T.Trick, "A Monte Carlo Approach for Power Estimation", *IEEE Trans. VLSI Systems*, Vol.1, No. 1, pp. 63-71, March, 1993.

[2] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-Power CMOS Digital Design," *Journal of Solid-State Circuits*, Vol.27, No.4, pp. 473-483, April,1992.

[3] Z. Chen, K. Roy and T.L. Chou,"Power Sensitivity - A New Method to Estimate Power Considering Uncertain

Specifications of Primary Inputs," *IEEE Intl. Conf. on Computer-Aided-Designed*, pp. 40-44, 1997.

[4] Z. Chen and K. Roy, "A Power Macromodeling Technique Based on Power Sensitivity," *ACM Design Automation Conference*, 1998.

[5] T.L. Chou and K. Roy, "Accurate Power Estimation of CMOS Sequential Circuits," *IEEE Trans. on VLSI*, pp. 369-380, 1996.

[6] T.L. Chou, K. Roy, and S. Prasad, "Estimation of Circuit Activity Considering Signal Correlations and Simultaneous Switching," *IEEE Intl. Conf. on Computer-Aided-Designed*, pp.300-303, 1994.

[7] S. Devadas, K. Keutzer and J. White, "Estimation of Power Dissipation in CMOS Combinational Circuits", *IEEE Custom Integrated Circuits Conf.*, pp. 19.7.1-19.7.6, 1990.

[8] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," *ACM/ IEEE Design Automation Conf.*, pp. 253-259, 1992.

[9] D.E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Mass.: Addison-Wesley, 1989.

[10] J.P. Halter and F. Najm,"A gate-level leakage power reduction method for ultra-low-power CMOS circuits," *Proceedings, IEEE Custom Integrated Circuits Conference*, pp. 475-478, 1997.

[11] M.C. Jeng, "Design and Modeling of Deep-Submicrometer MOSFETS", Electronics Research Laboratory, Rep. No. ERL-M90/90, University of California, Berkeley, 1990.

[12] M.C. Johnson, D. Somasekhar, and K. Roy,"A model for leakage control by MOS transistor stacking," Technical Report TR-ECE 97-12, Purdue University, School of Electrical and Computer Engineering, 1997.

[13] T. Kawahara, et al., "Subthreshold current reduction for decoded-driver by self-reverse biasing," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 11, pp. 1136-1143, Nov. 1993.

[14] D. Lawrence, "Handbook of Genetic Algorithms," New York: Van Nostrand Reinhold, 1991.

[15] R. Marculescu, D. Marculescu, and M. Pedram, "Efficient Power Estimation for Highly Correlated Input Streams," *ACM/IEEE Design Automation Conf.*, 1995, pp. 628-634.

[16] R. Marculescu, D. Marculescu, and M. Pedram, "Switching Activity Analysis Considering Spatiotemporal Correlations," *IEEE Intl. Conf. on Computer-Aided-Design*, pp. 294-299, 1994.

[17] J. Monteiro, S. Devadas, and B. Lin, "A Methodology for Efficient Estimation of Switching Activity in Sequential Logic Circuits," *Intl. Workshop on Low Power Design*, pp. 12-17, 1994.

[18] F.N.Najm, "Transition Density, A Stochastic Measure of Activity in Digital Circuits" *28th ACM/ IEEE Design Automation Conf.*, pp. 644-649, 1991.

[19] F.N.Najm, S. Goel, and I.N. Hajj, "Power Estimation in Sequential Circuits," *ACM/IEEE Design Automation Conf.*, pp. 635-640, 1995.

[20] J.M. Rabaey, "Digital Integrated Circuits," New Jersey: Prentice-Hall, 1996.

[21] K. Roy and S. Prasad, "Circuit Activity Based Logic Synthesis for Low Power Reliable Operations," *IEEE Trans. on VLSI Systems*, pp. 503-513, Dec. 1993.

[22] J. Sheu, D.L. Scharfetter, P.K. Ko, and M.C. Jeng, "BSIM: Berkeley Short-Channel IGFET Model for MOS Transistors", *IEEE J. Solid-State Circuits*, SC-22, 1987.

[23] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs," *Intl. Workshop on Low Power Design*, pp. 18-23, 1994.

[24] M. G. Xakellis and F. N. Najm, "Statistical Estimation of the Switching Activity in Digital Circuits," *ACM/ IEEE Design Automation Conf.*, pp. 728-733, 1994.

[25] "HSPICE User's Manual", Vol.II, 1996