Improving the efficiency of power simulators by input vector compaction*

Chi-ying Tsui

Department of Electrical and Electronic Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong

Radu Marculescu, Diana Marculescu, Massoud Pedram

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089

Abstract

Accurate power estimation is essential for low power digital CMOS circuit design. Power dissipation is input pattern dependent. To obtain an accurate power estimate, a large input vector set must be used which leads to very long simulation time. One solution is to generate a compact vector set that is representative of the original input vector set and can be simulated in a reasonable time. In this paper, we propose an input vector compaction technique that preserves the statistical properties of the original sequence. Experimental results show that a compaction ratio of 100X is achieved with less than 2% average error in the power estimates.

1. Introduction

The major source of power dissipation in digital CMOS circuits is the charging and discharging of capacitances during logic transitions and is calculated for each gate in the circuit as:

$$P_{cap} = 0.5 \times V_{dd}^2 C_L E(switching) f \quad (1)$$

where C_L is the load capacitance seen by the gate, V_{dd} is the supply voltage, f is the clock frequency, and *E(switching)* is the expected number of output transitions per clock cycle.

Circuit level simulation based techniques [3][4] have been developed which can capture the fine details of the transistor model. These methods are accurate but suffer from high computational cost and memory overhead, which limits the size of the input vector set to hundreds or thousands of vectors. This results in inaccuracy in the power estimation process as described next. Power consumption in digital circuits is input pattern dependent, i.e. depending on the input vectors applied to the circuit input, very different power estimates may be obtained. To obtain an accurate average power consumption, a set of input vectors that resemble the characteristics of data for typical applications is required. Usually these characteristic input vector set has a size of millions of vectors. Input vector size of hundreds or thousands, if selected arbitrarily, may not be able to capture this typical behavior and may thus lead to underestimation or overestimation of the power consumption of the circuit. One method for solving this problem is to compact the millions of input vectors into a characteristic stimulus vector set which has a much smaller size, yet is statistically equivalent to the original larger vector set.

Gate-level power estimation uses either *dynamic* or *static* techniques. Dynamic techniques explicitly simulate the circuit under a gate-level logic model with a typical input vector sequence (or input stream). Statistical techniques such as Monte Carlo simulation approach [1] alleviate the above mentioned problem of pattern dependence. This method however assumes that the signal and transition probabilities of the primary inputs are independent and may therefore give inaccurate estimates.

Static techniques do not explicitly simulate the circuit. Instead, they rely on statistical information (such as the mean switching activity and correlations) about the input stream and then calculate the same statistical information for the internal nodes of the circuit in order to obtain the average power consumption of the circuit [2][5][6]. The problem of input dependence is alleviated by using appropriate statistical information that captures the characteristics of the input vector. These methods are fast, but the accuracy is in general not as high as that obtained from explicit simulation.

In conclusion, to achieve accurate power estimates (e.g. estimating the power consumption of the chip before taping out), explicit simulation is a better choice. To reduce the simulation time, a compact input vector set that can capture the power consumption behavior of the given input data has to be derived. In this paper, we investigate and identify factors and properties of the input vector set that influence the power consumption of the digital circuit and develop an algorithm to compact a set of input vectors such that these *power-determining properties* are preserved. In particular, the spatiotemporal correlations of the inputs have direct impact on the power consumption [6]. We describe a method of compacting a set of input vectors such that the spatiotemporal correlations are preserved. From the experimental results, a compaction ratio of 100X can be easily achieved with a less than 2% average error in the power estimates.

2. Factors Affecting the Dynamic Power Consumption

Logic transition at the output of a node can be viewed as a probabilistic event and hence the expected number of these transitions over a period of time can be estimated by the transition probability of the node. Under the assumption that the primary inputs are temporally uncorrelated, the transition probability at the output of an internal node (gate) *n* is given by:

. * This work was funded in part by ARPA under contract no. F33615-95-C1627, the SRC under contract no. 94-DJ-559 and by a grant from the Intel Corp.

33rd Design Automation Conference ®

DAC 96 - 06/96 Las Vegas, NV, USA

$$TP_n = P_n(1 - P_n) \quad (2)$$

where TP_n and P_n are the transition and signal probabilities of n, respectively. The signal probability depends on whether the inputs of n are correlated or not. The assumption of spatial and temporal independence is however not always true. Indeed, in most applications, only some input patterns out of all possible input patterns are feasible and the sequence of the input patterns is far from random. For example, in the microprocessor domain, input patterns are generated from architectural level traces which are driven by the instruction opcodes and the instruction mix of typical applications.

Signal correlations among inputs of a gate are generated from two sources. The first is the structural dependency due to reconvergent fanout. The second is the stochastic dependency due to the primary input correlations, that is, input lines that are structurally independent may become correlated because of the circuit input correlations [6].

If the circuit inputs are not temporally independent, the switching activity has to be captured by the transition probability which depends on the sequence of input patterns applied and hence the transition probability of the primary inputs. In [7], it is shown that reshuffling the input vector sequence to achieve a different temporal correlation while preserving the input signal probabilities leads to large variations in the circuit power consumption. Similarly, in [6], it is shown that spatiotemporal correlations at the primary inputs have a significant impact on the power consumption of the circuit. For an input sequence having high correlations (generated by a sequence counter), the power consumption can be as low as 5% of the power consumption for another input sequence which has low correlations (generated by a random number generator).

Accounting for the all possible correlations is practically impossible even for small circuits. In [6], correlations are approximated by considering only pairwise signal correlations. This gives rise to sixteen correlation coefficients corresponding to the sixteen possible transitions of a pair of signals (x,y). In [5], it is shown that the accuracy in estimating the switching activity of individual nodes in a circuit improves by an average factor of 6X compared to the approaches that do not account for any of the correlations.

To summarize, signal probabilities, transition probabilities and spatiotemporal correlations are the important properties of the primary inputs which affect the power consumption of the circuits. Note that given the spatiotemporal correlations between two input signals x and y, the signal and transition probabilities of x and y can be easily calculated [6].

3. A Vector Compaction Algorithm

The problem of vector compaction is stated as follows:

Vector Compaction Problem 1: Given an input vector sequence S_1 of length L_1 with some property P_1, compact or reconstruct another vector sequence S_2 of length L_2 with property P_2 such that P_1 and P_2 are the same (or nearly the same). ❒

The compaction ratio is equal to L_1/L_2. For power estimation, relevant properties are the joint transition probabilities of the signal lines (which are related to their spatiotemporal correlation in a straight-forward manner [6]). As it is difficult to account for the exact joint transition probabilities, we use the pairwise transition probabilities as an approximation.

From the discussion in Section 2, if we closely capture the pairwise transition probabilities, the compacted vector set should represent the power-determining behavior of the original vector set. To measure how closely the compacted vector set resembles the original vector set, we use a metric C_1 which measures the absolute error in pairwise transition probabilities among all possible combinations of inputs and is given by the following equation:

$$C_1 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Diff(x_i, x_j) \quad (3)$$

where $Diff(x_i,x_j)$ is equal to:

$$\sum_{a=0}^{1} \sum_{b=0}^{1} \sum_{c=0}^{1} \sum_{d=0}^{1} \left| P^1_{x_{i(a\to b)} x_{j(c\to d)}} - P^2_{x_{i(a\to b)} x_{j(c\to d)}} \right| \quad (4)$$

x_i is the i^{th} input signal, $P^1_{x(a->b),y(c->d)}$ and $P^2_{x(a->b),y(c->d)}$ are the pairwise transition probabilities of signals x and y for S_1 and S_2, respectively.

The vector compaction problem for power estimation is therefore formulated as follows.

Vector Compaction Problem 2:Given an input vector sequence S_1 of length L_1 and a compaction ratio R, generate an output vector sequence S_2 of length L_2 where $L_1/L_2 = R$ and such that C_1, as defined in equations (3) and (4), is minimized. ❒

We cast the problem of observing pairwise transition probabilities to that of observing pairwise signal probabilities as follows. The four types of signal transitions, *0->0, 0->1,1->0, 1->1* are encoded by 4 symbols, *a, b, c,d,* respectively. The pairwise transition probability of two signals x and y is then translated to the pairwise signal probabilities of this two signals with the new signals taking on values (a,b,c,d) instead of (0,1). For example,

$$P_{x_{(0\to 0)} y_{(1\to 0)}} = P((x = a) \wedge (y = c)) \quad (5)$$

Generating a vector sequence that satisfies the pairwise transition probabilities is thus reduced to generating a vector set of symbols that satisfies the pairwise signal probabilities. After the generation of the sequence of symbolic vectors, we have to convert it back to a sequence of binary vectors for the simulation purpose. If we have *n-1* symbolic vectors, we should be able to reconstruct n binary vectors accordingly. However, the consecutive symbolic vectors may not be "temporally compatible" and thus we may fail to generate a valid binary vector. The following example illustrates the incompatibility problem. Let α_i^{n-1} and α_i^n be two consecutive symbols for signal i. If $\alpha_i^{n-1} = a$, then the corresponding binary bit pair b_i^{n-1} and b_i^n is (0,0). α_i^n is then used to generate the binary bit pair (b_i^n, b_i^{n+1}). But b_i^n is already bound to 0 by α_i^{n-1}, so α_i^n can only be a or b since it corresponds to binary pairs (0,0) and (0,1). Therefore only the following pair of symbols are temporally compatible: *(a,a), (a,b), (b,c), (b,d),(c,a),(c,b), (d,c),(d,d).*

One method for solving this problem is to neglect the temporal incompatibility between pairs of consecutive symbolic vec-

tors. Instead of connecting the pairs of binary vectors generated from the two consecutive symbol vectors, two separate pairs of binary vectors are generated (we will refer to this technique as the *unconstrained symbolic vector compaction*). In this case we generate *2n* binary vectors. When the vectors are input to the simulator, only the power consumption due to alternative pairs of vectors will be included. This method has a drawback that *2n* binary vectors will be generated instead of *n*. Therefore the compaction ratio will be reduced by a factor of 2.

Another method for solving this problem is that when we generate the symbolic vectors, we make sure that the consecutive vectors are temporally compatible (we will refer to this technique as the *constrained symbolic vector compaction*).

3.1. The Compaction Algorithm

First information on the pairwise transition probabilities (or pairwise symbolic probabilities $P(x=\alpha \wedge y=\beta)$) is collected. The next phase of the process is the vector generation stage which takes the pairwise transition probabilities and a user-given compaction ratio R and generates a set of symbolic vectors.

For the unconstrained symbolic vector compaction, a row based construction is used in which one vector is built at a time until all required vectors are generated. The objective of the construction process is to maintain the pairwise transition probabilities. During the vector construction process, a symbol is selected for each input bit as follows. For the first bit x_0, the symbol α that has the maximum transition probability $(P(x_0=\alpha))$ is picked.

For an input bit x_j $(j>0)$ the symbol that has the largest sum of joint pairwise symbolic probabilities with the symbols that have already been picked for the previous input bits $(x_i, i<j)$ is selected. In other words, the following objective function F_1 is maximized:

$$F_1 = \sum_{i=0}^{j-1} P((x_j = \alpha) \wedge (x_i = \beta_i)) \qquad (6)$$

where α is the symbol being considered for x_j and β_i is the symbol already chosen for bit x_i.

After symbols are chosen for all the bits for the first vector V_1, the pairwise symbolic probabilities are different for the remaining vectors since some symbolic pairs have already occurred in V_1. The pairwise symbolic probabilities have to be updated as follows. Let L_1 be the number of vectors to be generated and α_i and β_j be the symbols chosen for bit x_i and x_j in V_1, respectively. Before generating V_1, the number of occurrences of $x_i = \alpha_i$ and $x_j = \beta_j$ in L_1 vectors is given by $P((x_i = \alpha_i) \wedge (x_j = \beta_j))L_1$. After they are chosen for V_1, the number of occurrences of $x_i = \alpha_i$ and $x_j = \beta_j$ in the remaining L_1-1 vectors is equal to $P((x_i = \alpha_i) \wedge (x_j = \beta_j))L_1 - 1$. Therefore the pairwise symbolic probability of $x_i = \alpha_i$ and $x_j = \beta_j$ has to be updated as $(P((x_i = \alpha_i) \wedge (x_j = \beta_j))L_1 - 1) / (L_1 - 1)$.

Other pairwise symbolic probabilities of $x_i = \alpha$ and $x_j = \beta$ where $\alpha \neq \alpha_i$ or $\beta \neq \beta_j$ are updated as $(P((x_i = \alpha_i) \wedge (x_j = \beta_j))L_1) / (L_1 - 1)$.

The final phase is translating the symbolic vectors into binary vector pairs which is a simple decoding mechanism.

For the constrained symbolic compaction, we ensure that the symbolic vector generated at step *t* is temporally compatible with the one generated at step *t-1* as described next. When selecting a symbol α_i^t for bit x_i for the t^{th} vector, we can only choose from the symbols that are temporally compatible with the symbol α_i^{t-1} for bit x_i where α_i^{t-1} is the symbol of x_i selected for the $t\text{-}1^{th}$ vector. We choose the symbol that is temporally compatible and has the maximum F_1 value as given in equation (6). Because of the temporal compatibility restriction, the symbolic vectors generated do not necessarily result in minimum C_1. It is a difficult problem to optimize C_1 for the constrained symbolic compaction. Here we propose a greedy mechanism. After we obtain the first symbolic vector sequence, we translate them to a binary vector sequence *S*. For each binary vector, we calculate the gain G_i of changing the value of bit x_i. The gain is the change in C_1 if x_i is flipped. The bit that has the largest positive gain *G* is chosen to change value. The gains of the rest of the bits are recalculated after the value of x_i is changed. The process is repeated for every bit that has a positive gain. Then we go to the next vector and repeat the bit changing mechanism. After we try on every vector, the whole process is iterated again until no reduction in C_1 is observed or the number of iterations exceeds a user-specified number.

4. A Paradigm for Efficient and Accurate Multi-level Power Estimation

Modern VLSI chips may contain millions of transistors. Even though we may compact the input vectors to a few hundreds, it is still very time consuming to simulate the whole chip at the circuit (or even the gate) level. An efficient but accurate chip level power estimation paradigm which addresses this problem is described next.

The chip is first divided into building blocks. Each building block has a detailed structural model at the gate or circuit level. A behavioral model of the chip is built using the building blocks as components. Simulation is then carried out at the behavioral level (as it will be significantly faster than gate or transistor level simulation). The input vectors to the simulator are derived from the set of typical benchmark applications that the chip is designed for. A statistical data collection agent is then used to collect the bit switching statistics for the buses or nets that are connected to the inputs of each building block. After the statistical data is collected, the vector compaction program is used to generate the compacted input vector set for each building block which can then be fed to the corresponding low level simulator to estimate the power consumption of each building block. The total power consumption of the whole chip can be obtained by adding the power consumption of all building blocks and the power consumed at the buses which connect the building blocks.

5. Experimental Results

To demonstrate the effectiveness of the vector compaction program (called **vcct**), we carried experiments on MCNC-91 benchmark circuits and some datapath circuits such as adders and multipliers.

The first experiment uses two sets of vectors, each having 100,000. The first one is a counter sequence with the sequence restarting at a random number after a fixed number of vectors are generated. The second vector set consists of a highly correlated vector set used for testing purpose. The vectors are fed to a gate-level logic simulator which can measure real delay dynamic power consumption. A clock frequency of 20MHz is assumed and all power values are in mW. The results presented here are for the unconstrained symbolic vector compaction which has a faster runtime and better accuracy than constrained symbolic vector compaction. This experiment shows the importance of preserving the spatiotemporal correlations during the vector compaction process. We compact the vector set with a compaction ratio of 100X. Table 1 summarizes the results. For each vector set, we provide the compaction results corresponding to when we account for both spatial and temporal correlations (*sp_te* column) and when we only account for temporal correlations (*te only* column). It can be shown that by considering spatiotemporal correlations using pairwise transition probabilities, the compacted vector gives a very accurate power estimation when the vectors are pumped through the logic simulator.

The second experiment uses **Powermill** [4], a circuit level power simulator which is the industrial standard for power estimation. Two sets of vector sequences are used. Each has 4000 vectors. The first set is a highly biased vector set and the second set is a randomly-generated vector set. Each vector set is compacted by 20X. Table 2 summarizes the results. Power is measured by the average current drawn from the power supplies and accounts for both capacitive and short-circuit currents. The result clearly shows the effectiveness of the *vcct* program in preserving the power-determining behavior of the original sequence. The maximum error is below 5% while the average error is less than 2%.

6. Conclusion

We presented a method of compacting a large vector set into a characteristic vector set with smaller number of vectors. By doing so, the number of simulation cycles required for obtaining accurate power estimation is dramatically reduced. We showed that by keeping the pairwise transition probabilities during the vector compaction, the average error in power estimation using the compacted vector is within 2% of that using the original vector.

Acknowledgement

The authors would like to thank Deo Singh of Intel Corp. for posing the problem to us and Qing Wu of USC for helping with the PowerMill runs.

Bibliography

[1] R. Burch, F. Najm, P. Yang and T. Trick, “A Monte Carlo approach for power estimation”, IEEE Transaction on VLSI Systems, vol. 1, no. 1, pp. 63-71, March,1993

[2] C-Y. Tsui, M. Pedram and A. M. Despain, “Efficient Estimation of Dynamic Power Dissipation under a Real Delay Model”, in Proceedings of IEEE International Conference on Computer-Aided Design, pp. 224-228, Nov., 1993

[3] F. Rouatbi, B. Haroun and A. J. Al-Khalili, “Power Estimation Tool for Sub-Micron CMOS VLSI Circuits”, in Proceedings of European Design Automation Conference, pp. 204-209, 1992

[4] C. Deng, “Power Analysis for CMOS/BiCMOS Circuits”, in Proceedings of International Workshop on Low Power Design, pp. 3-8, April, 1994

[5] R. Marculescu, D. Marculescu and M. Pedram, “Logic level power estimation considering spatiotemporal correlations”, in Proceedings of IEEE International Conference on Computer-Aided Design, pp. 294-299, Nov., 1994

[6] R. Marculescu, D. Marculescu and M. Pedram, “Efficient power estimation for highly correlated input streams”, in Proceedings of the 32nd IEEE Design Automation Conference, pp. 628-634, June, 1995

[7] S. Rajgopal and G. Mehta, “Experiences with Simulation-Based Schematic Level Current Estimation“, International Workshop on Low Power Design, pp. 9-14, April,1994

Table 1: %Error in power estimation using unconstrained symbolic vector compaction: Real Delay Gate Level Simulation (sp-te: spatiotemporal correlations; te: temporal correlation only)

	vector sequence S1			vector sequence S2		
		%Error			%Error f	
Cir.	**Power Est.**	**te only**	**sp-te**	**Power Est.**	**te only**	**sp-te**
C432	856.8	24.94	1.72	701	9.60	0.43
C880	1403.2	3.00	0.98	1282.2	3.74	0.44
C1355	4499.6	19.96	2.77	3538.4	4.76	6.90
C1908	1173.7	7.85	0.38	1110.9	8.60	0.40
C3540	11985	13.36	0.58	13517	3.65	2.02
C6288	98573	0.25	3.25	122789	44.21	0.57
add16	1368.7	3.98	0.77	1286.2	16.39	0.49
mul2	153.1	75.31	0.52	140.9	2.63	0.92
mul4	770.2	14.31	0.52	879.8	15.51	1.07
mul8	5206.1	8.45	1.14	6193.6	11.05	0.28
mul16	27815	18.67	5.07	37552	40.70	0.36
parity	292.5	1.88	1.20	256.1	61.69	12.3
x1	2379.1	2.53	1.65	2147.1	9.27	0.15
apex7	1725.6	8.19	0.31	1653.4	6.35	1.14
Aver.		14.48	1.51		17.01	1.96

TABLE 2. Current Estimation by PowerMill

	current(mA) for biased seq.		current(mA) for random seq.	
circuit	original	comp.	original	comp.
C432	0.407	0.411	0.775	0.748
C880	0.779	0.765	1.467	1.516
C1355	1.124	1.136	2.018	2.013
C1908	1.282	1.254	1.923	1.936
C3540	3.397	3.480	5.718	5.822
C6288	14.57	13.83	47.60	47.62
mul2	0.070	0.070	0.096	0.097
mul4	0.579	0.582	0.839	0.833
mul8	3.185	3.131	6.305	6.315
Avg. % Error		1.78		1.28