in Integration, the VLSI Journal, vol. 25 (1998)
pp.37-52

Accurate and Efficient Power Simulation Strategy by Compacting the Input Vector Set

Chi-ying Tsui
Department of Elect. and Elect. Engineering
Hong Kong University of Science and Technology
Clear Water Bay Hong Kong

Massoud Pedram
Department of Electrical Engineering - System
University of Southern California
Los Angeles, CA 90089

Abstract

Accurate power estimation of digital CMOS circuit can be obtained by explicit simulation. However, power dissipation is input dependent. To obtain an accurate power estimate, a large input vector set has to be used resulting in large simulation time. One solution is to generate a representative vector set of the original input vector set that contains only a few thousand vectors which can then be simulated in a reasonable time. This paper addresses the problem of vector compaction for power simulation. We compact the input vector set such that the statistical properties which affect the power consumption are preserved. Experimental results show that compaction ratio of 100X is achieved with less than 2% average error in the power estimates.

Keyword: power estimation, simulation, input vector compaction, spatial temporal correlation

1. Introduction

The increase in integration and higher clock frequency have made power consumption an important (and in some cases dominant) concern in today's IC design. The popularity of portable applications and the increase in the packaging and cooling cost are also driving forces behind the push for low power design. Accurate power estimation tools are required to assist the designers in designing low power circuits.

For digital CMOS circuits, the total power consumption is given by the following formula:

$$P_{total} = P_{dyn} + P_{sc} + P_{leakage} \tag{1}$$

The first term P_{dyn} represents the dynamic power required to charge and discharge the load capacitances during the logic transitions and is given by:

$$P_{dyn} = 0.5 \times \frac{V_{dd}^{2} C_L E(switching)}{T_{cycle}} \tag{2}$$

where C_{load} is the load capacitance, V_{dd} is the supply voltage, T_{cycle} is the clock cycle time, and *E(switching)* is the expected number of transitions per clock cycle.

The second term P_{sc} is the power consumption due to the direct (short circuit) current I_{sc} which flows when both the *pMOS* and the *nMOS* transistors are simultaneously on for a time duration ΔT during the output transition [1] and is given by:

$$P_{sc} = V_{dd} I_{sc} \frac{\Delta T}{T_{cycle}} \tag{3}$$

The third term $P_{leakage}$ is the power consumption due to the subthreshold leakage current ($I_{leakage}$) which is governed by a bipolar-like mechanism and is carried by minority carriers and diffusion currents.

Circuit level simulation based techniques[2][3] have been developed which can capture the fine details of the transistor model. These methods are accurate and capable of handling various device models and design styles. Because of the fine details of the transistor models that these techniques employ, they suffer from high computational cost and memory overhead. A fast transistor-level power simulator was developed in [4] which applies an event-driven timing simulation algorithm to increase the speed by two to three orders of magnitude over conventional circuit level simulators. It gives detailed power information (instantaneous, average and rms current values) as well as the total power consumption which includes short circuit power, leakage power and capacitive switching power. However, although orders of magnitude of speed improvement are achieved, for large designs, still only thousands or hundreds of vectors can be input to the simulator. 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 resulted. 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 is obtained from higher level simulations (such as instruction level, or behavioral level simulation) and has a size of millions of vector. 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. So although in each simulation step the power estimation is very accurate, because of the limited number of input vectors simulated, circuit level simulation may not give an accurate estimation of the overall power consumption. One method to solve this problem is to compact the millions of input vectors into a characteristic stimulus vector set which has a size of thousands, yet is statistically equivalent to the larger vector set.

If the design is good [1] and the threshold voltage is not too low, the short circuit power and leakage power is negligible compared to the capacitive power consumption. Under these assumptions, the power estimation problem can be reduced to that of calculating the transition count of each circuit node under a given delay model. Techniques have been developed which offer orders of magnitude speed-up compared to the conventional circuit simulation. The techniques can be classified into two categories, dynamic and static.

Dynamic techniques explicitly simulate the circuit under a gate-level logic model with a typical input vector sequence (or input stream). This has the same problem as the circuit simulation in which the accuracy strongly depends on the input vector sequence. However, since gate level simulation runs much faster than circuit level simulation, more vectors can be run. The gate-level simulation still takes a lot of time to run tens of thousands of vectors (or more). Statistical techniques such as Monte Carlo simulation approach [5] alleviate the above mentioned problem of pattern dependence. These methods are based on the technique of sampling. Power consumption is estimated by simulating a small fraction of the input vector space. The average power consumption for each sample is regarded as a random variable. From the central limit theorem, the mean value of the random variable will converge to the average power consumption of the circuit. This method assumes the signal and transition probabilities of the primary inputs are independent and cannot handle signal correlations at the inputs. For many circuits such as control path of a microprocessor or a finite state machine, the data inputs are indeed both spatially and temporally correlated (we use the term spatiotemporal correlation to represent spatial and temporal correlations at the same time). The Monte-Carlo methods may thus give inaccurate estimates.

Static techniques do not explicitly simulate the circuit. Instead, they rely on statistical information (such as the mean activity and correlations) about the input stream and then calculate (directly or incrementally) these statistical information for the internal nodes of the circuit in order to obtain the average power consumption of the circuit [6][7][8]. The problem of input dependence is alleviated by using appropriate statistical information that captures the characteristics of the input vector. Static techniques usually employ probabilistic methods which calculate the signal and transition probabilities of the internal nodes based on the probabilities of the primary inputs. These methods are fast and alleviate the input dependence problem. Under a real delay model, it becomes difficult to estimate the power due to glitches and thus the estimation accuracy varies as a function of the gate inertia model and the glitch filtering scheme employed. The accuracy is in general not as high as that obtained from explicit simulation.

In conclusion, explicit simulation can give an accurate power estimation but requires from thousands to millions of simulation vectors to obtain an accurate overall power estimate. This is clearly very costly (if not impractical). Probabilistic methods are fast and efficient since the input dependence is implicitly captured, but they usually employ crude circuit models and hence cannot provide very accurate estimates. To achieve very accurate power estimates (e.g. estimating the power consumption of the chip before taping out), explicit simulation is a better choice but to reduce the complexity of power estimation, 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 the 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.[1] In particular, the spatiotemporal correlations of the inputs has a direct and sizable impact on the power consumption. 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.

The rest of this paper is organized as follows. Section 2. discuss the factors and input vector properties that affect the power consumption of the circuit. Section 3. presents the algorithm of vector compaction. Section 4. describes an accurate and efficient multi-level power estimation framework based on vector compaction. Experimental results are presented and conclusion remarks are given in Section 5. and Section 6. respectively.

2. Factors Affecting the Dynamic Power Consumption

Since power consumption is input pattern dependent, the following signal properties have significant impact on the power consumption of the circuit.

2.1. Transition Probability and Signal Correlation

The switching of a node can be viewed as a probabilistic event and hence the expected number of logic 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 of an internal node *n* is given by:

$$TP_n = P_n(1 - P_n) \tag{4}$$

where TP_n and P_n are the transition and signal probabilities of *n,* respectively. When calculating the signal probability of the internal nodes, signal correlation among the internal nodes due to reconvergent fanout has to be taken into account. An exact method for calculating the signal probability under the uncorrelated input assumption was developed using Ordered Binary

1. We use the term compaction as the process is a lossy compression.

Decision Diagrams (OBDDs) in [9].

The assumption of spatial and temporal independence at the primary inputs is however not always true. Indeed, in many 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 for typical applications.

Spatial correlation among the primary inputs is determined by their conditional probabilities. Two signals, *x* and *y*, are spatially correlated if:

$$P(xy) \neq P(x)P(y) \tag{5}$$

When calculating the signal probability of the internal nodes, two types of correlations have to be considered. The first is the structural dependency due to reconvergent fanout. The second is the stochastic dependency due to the primary input correlations. Nodes that are structurally independent may become correlated because of the input dependency [10].

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 [8], transition probability of a signal is modeled by the state transition probabilities of a lag-one Markov Chain consisting of two states, 0 and 1. In [11], it is shown that by reshuffling the input vector sequence to achieve a different temporal correlation with the same input signal probabilities, a power consumption difference as high as 30% can be observed.

In [10], it is shown that the spatiotemporal correlation at the primary inputs has significant impact on the power consumption of the circuit. For an input sequence having high correlation (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 correlation (generated by a random number generator). To exactly account for the correlations is practically impossible even for small circuits. (The number of correlation coefficient that must be considered among *n* inputs is 2^n.) In [10], the correlations are approximated by considering only pairwise signal correlations. These pairwise correlations are captured among the sixteen possible transitions of a pair of signals (*x*,*y*). The mathematical foundation of the model is a lag-one Markov Chain with four states (0, 1, 2, 3 which correspond to 00, 01, 10, 11 encodings of (*x*,*y*)). In [8], 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. These properties are related to each other. Indeed if we know the pairwise transition probabilities, then the signal and transition probabilities can be derived according to the following equations (which are related to the spatiotemporal correlations in a straight-forward manner):

$$P(x) = \sum_{i=0}^{1}\sum_{j=0}^{1} P_{x_{(0\to 1)}y_{(i\to j)}} + \sum_{i=0}^{1}\sum_{j=0}^{1} P_{x_{(1\to 1)}y_{(i\to j)}} \tag{6}$$

$$P^{x}_{a\to b} = \sum_{i=0}^{1}\sum_{j=0}^{1} P_{x_{(a\to b)}y_{(i\to j)}} \tag{7}$$

where *P(x)* and $P^{x}_{a->b}$ are the signal and transition probabilities of *x* and $P_{x(a->b),y(i->j)}$ is the pairwise transition probabilities of *x* switching from *a* to *b* and *y* switching from *i* to *j* simultaneously.

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 the power estimation application, the relevant properties are the joint transition probabilities of the input signals. As it is difficult to account for the exact joint transition probabilities, we use the pairwise transition probabilities to approximate it.

From the discussion in Section 2., if we closely resemble the pairwise transition probabilities, the compacted vector set should be able to capture the power-determining factor 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 all pairwise transition probabilities among all possible combinations of inputs and is given by the following equation:

S_1			S_2		
x_1	x_2	x_3	x_1	x_2	x_3
0	0	0	0	0	0
1	1	1	1	0	0
0	1	0	0	0	1
1	1	0	1	1	1
0	1	1	0	1	0
0	1	1	0	1	1
0	0	1	0	0	1
1	0	1	1	0	1
0	0	1	0	0	1

$$P^1_{x1(a\text{->}b),x2(i\text{->}j)} \qquad \begin{array}{c|cccc} x_{2(i\text{->}j)} \backslash x_{1(a\text{->}b)} & 0\text{->}0 & 0\text{->}1 & 1\text{->}0 & 1\text{->}1 \\ \hline 0\text{->}0 & 0 & 1/8 & 1/8 & 0 \\ 0\text{->}1 & 0 & 1/8 & 0 & 0 \\ 1\text{->}0 & 1/8 & 0 & 0 & 0 \\ 1\text{->}1 & 1/8 & 1/8 & 1/4 & 0 \end{array}$$

$$P^2_{x1(a\text{->}b),x2(i\text{->}j)} \qquad \begin{array}{c|cccc} x_{2(i\text{->}j)} \backslash x_{1(a\text{->}b)} & 0\text{->}0 & 0\text{->}1 & 1\text{->}0 & 1\text{->}1 \\ \hline 0\text{->}0 & 0 & 1/4 & 1/4 & 0 \\ 0\text{->}1 & 0 & 1/8 & 0 & 0 \\ 1\text{->}0 & 1/8 & 0 & 0 & 0 \\ 1\text{->}1 & 1/8 & 0 & 1/8 & 0 \end{array}$$

Diff(x_1,x_2) = 0.5

Similarly *Diff*(x_1,x_3) = 0.5 and *Diff*(x_1,x_2) = 0.75

C_1 = 0.5 + 0.5 + 0.75 = 1.75

Figure 1 Example showing how to calculate C_1

$$C_1 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Diff(x_i, x_j) \tag{8}$$

where

$$Diff(x_i, x_j) = \sum_{a=0}^{1} \sum_{b=0}^{1} \sum_{c=0}^{1} \sum_{d=0}^{1} \left| P^1_{x_{i(a \to b)} x_{j(c \to d)}} - P^2_{x_{i(a \to b)} x_{j(c \to d)}} \right| \tag{9}$$

x_i is the i^{th} input signal, $P^1_{x(a\text{->}b),y(i\text{->}j)}$ and $P^2_{x(a\text{->}b),y(i\text{->}j)}$ are the pairwise transition probabilities of *x* and *y* for S_1 and S_2, respectively. Figure 1 shows an example of calculating C_1.

.

So the problem of the vector compaction for power estimation is re-formulated as:

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 (8) and (9), is minimized.

❐

The algorithm for generating the compacted vector is described next.

We reduce 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 probabilities of two signals *x* and *y* are then translated to the pairwise signal probabilities of these two signals

$$\begin{array}{c} S_1 \\ \begin{array}{ccc} 0&0&0\\1&1&1\\0&1&0\\1&1&0\\0&1&1\\0&1&1\\0&0&1\\1&0&1\\0&0&1 \end{array}\end{array} \longrightarrow \begin{array}{ccc} b&b&b\\c&d&c\\b&d&a\\c&d&b\\a&d&d\\a&c&d\\b&a&d\\c&a&d \end{array}$$

Figure 2 Binary vector vs. symbolic vector

with the new signals taking on values (*a,b,c,d*) instead of (0,1). For example,

$$P_{x_{(0\to 0)}y_{(1\to 0)}} = P((x = a) \wedge (y = c)) \tag{10}$$

and

$$P_{x_{(1\to 1)}y_{(0\to 0)}} = P((x = d) \wedge (y = a)) \tag{11}$$

The vector sequence S_1 of Figure 1 can then be cast as a vector sequence of symbols as shown in Figure 2.

Generating a vector sequence that satisfies the pairwise transition probabilities is thus reduced to generating a vector set of symbols which satisfies the pairwise symbolic probabilities. (See Section 3..1) After the generation of the symbolic vectors, we have to convert it back to a sequence of vectors of signal values 0 and 1 for the simulation purpose. For a sequence of n binary vectors, *n-1* symbolic vectors can be obtained. Conversely, if we have *n-1* symbolic vectors, we should be able to reconstruct *n* binary vectors accordingly. However, the consecutive symbol vector may not be "temporally compatible" and thus we may fail to generate any 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 either *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 way to solve this problem is that when we generate the symbolic vectors, we make sure that the consecutive vectors are temporally compatible. So the problem of vector compaction is formulated as follows:

Constrained Symbolic Vector Compaction Problem: Given a symbolic input vector set S_1 of length L_1, compact or reconstruct another symbolic vector set S_2 of length L_2 such that the cost metric C_1 is minimized subject to the constraint that S_2 satisfies the temporal compatibility *constraint.*

❐

Another method is to neglect the temporal incompatibility between pairs of consecutive symbolic vectors. Instead of connecting the pairs of binary vectors generated from the two consecutive symbol vectors, two separate pairs of binary vectors are generated (we call it the unconstrained symbolic vector compaction problem). 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.

3.1. The Compaction Algorithm

The overall flow of the compaction process is shown in Figure 3.

The first phase is the statistical data analysis stage. Here the information on the pairwise transition probabilities is collected. Note that this is the same as the pairwise symbolic probabilities $P(x=\alpha \wedge y=\beta)$ (where α and β are one of the four symbols: *a,b,c,d*).

The main phase of the process is the vector generation stage. It takes in 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 algorithm 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. $P(x_0=\alpha)$ can be obtained from the pairwise transition probability as follows:

$$P((x_0 = \alpha)) = \sum_{p=0}^{1} \sum_{q=0}^{1} P(x_{0_{k \to l}} x_{j_{p \to q}}) \qquad (12)$$

where α is symbolic code for the *k->l* transition and x_j is any input bit which is different from x_0.

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

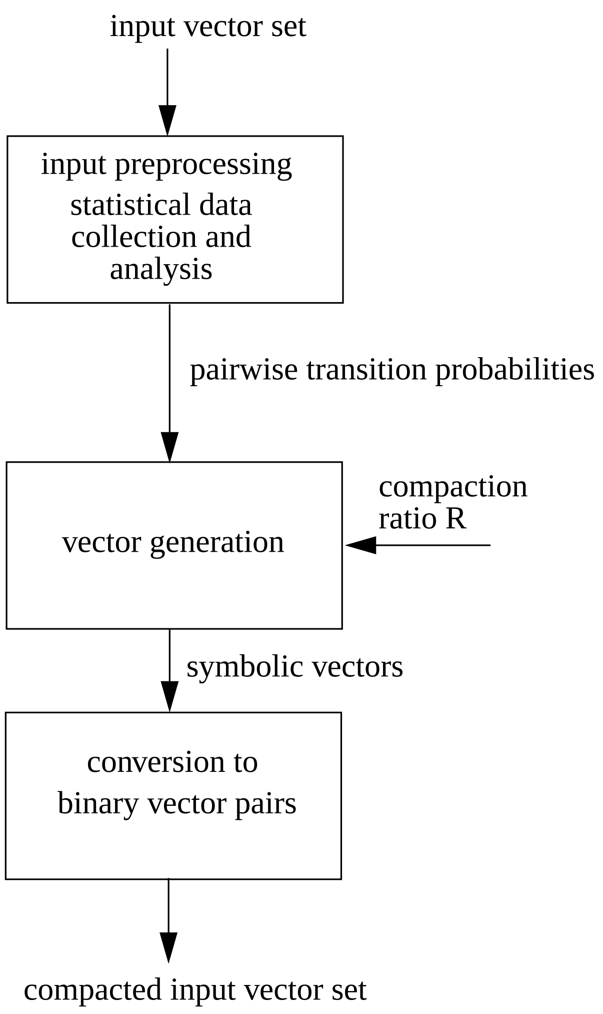

input vector set
input preprocessing
statistical data collection and analysis
pairwise transition probabilities
vector generation
compaction ratio R
symbolic vectors
conversion to binary vector pairs
compacted input vector set

Figure 3 Flow of compaction algorithm

selected. The following objective function F_1 is used to measure the pairwise transition probabilities (0 through j-1)

$$F_1 = \sum_{i=0}^{j-1} P((x_j = \alpha) \wedge (x_i = \beta_i)) \tag{13}$$

where α is the symbol being considered for x_j and β_i is the symbol already chosen for bit x_i. The symbol α^* that maximizes F_1 is chosen for x_j. The vector is generated after a symbol is chosen for each bit.

Let L_1 be the number of vectors to be generated. After symbols are chosen for all the bits for the first vector V_1, the pairwise symbolic probabilities are different for the remaining L_1-1 vectors since some symbolic pairs have already occurred in $V_{1.}$ The pairwise symbolic probabilities have to be updated as follows. Let 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 expected number of occurrence 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 expected number of occurrence 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 follows:

$$P((x_i = \alpha_i) \wedge (x_j = \beta_j)) = \frac{P((x_i = \alpha_i) \wedge (x_j = \beta_j))L_1 - 1}{L_1 - 1} \tag{14}$$

For the other pairwise symbolic probabilities of $x_i = \alpha$ and $x_j = \beta$ where $\alpha \neq \alpha_i$ or $\beta \neq \beta_j$, they have to be updated as follows:

$$P((x_i = \alpha) \wedge (x_j = \beta)) = \frac{P((x_i = \alpha) \wedge (x_j = \beta))L_1}{L_1 - 1} \tag{15}$$

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

For the constrained symbolic compaction problem, we have to 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 minimum F_1 given by equation (13). Because of the temporal compatibility restriction, the symbolic vectors generated do not necessarily result in minimum C_1. It is a very difficult problem to optimize C_1 for the constrained symbolic compaction. Here we propose a greedy mechanism. After we obtain the first set of symbolic vectors from the procedure described above, 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 flipping and is calculated as follows. Let *S** be the new binary vector sequence if x_i is flipping, G_i is then given by:

$$G_i = \sum_{j \neq i} Diff(x_i, x_j) - Diff^*(x_i, x_j) \tag{16}$$

where $Diff(x_i,x_j)$ and $Diff^*(x_i,x_j)$ are the sum of the absolute difference in pairwise transition probabilities between x_i and x_j for *S* and *S**, respectively. The calculation of $Diff(x_i,x_j)$ is given by equation (9). 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 reaches a user-defined number.

3.2 Complexity Analysis

For unconstrained symbolic vector compaction, we have to choose a symbol for bit x_k such that F_1 given in equation (13) is maximum. Therefore the complexity to select symbols for a row is $O(n^2)$ where n is the number of input bits. The total complexity is $O(ln^2)$ where l is the number of vector in the new vector set.

For constrained symbolic vector compaction, the complexity of calculating the gain during the greedy algorithm is $O(n)$ and the complexity of processing each vector is hence $O(n^2)$. The total complexity of each iteration is thus also equal to $O(ln^2)$ where l is the number of vectors in the new vector set.

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 one or two thousands, it is still very time consuming to simulate the whole chip using the compacted vector set at the circuit (or even the gate) level. The fastest transistor level simulators have a performance of 1000 transistor-vectors per 1 CPU second (on Sun Sparc 20). So simulating a 100K transistor circuit for 1000 vectors takes 100,000 seconds! An efficient but accurate chip level power estimation paradigm which can solve the problem is proposed as follows. The chip is first divided into several building blocks. Each building block has a detailed structural model at the gate level or the circuit level. A behavioral model of the chip is built using the building blocks as components. The behavioral model can be described in VHDL, Verilog or other high level hardware modeling languages. 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 input of each building block. After the statistical data are collected, a vector generation 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.

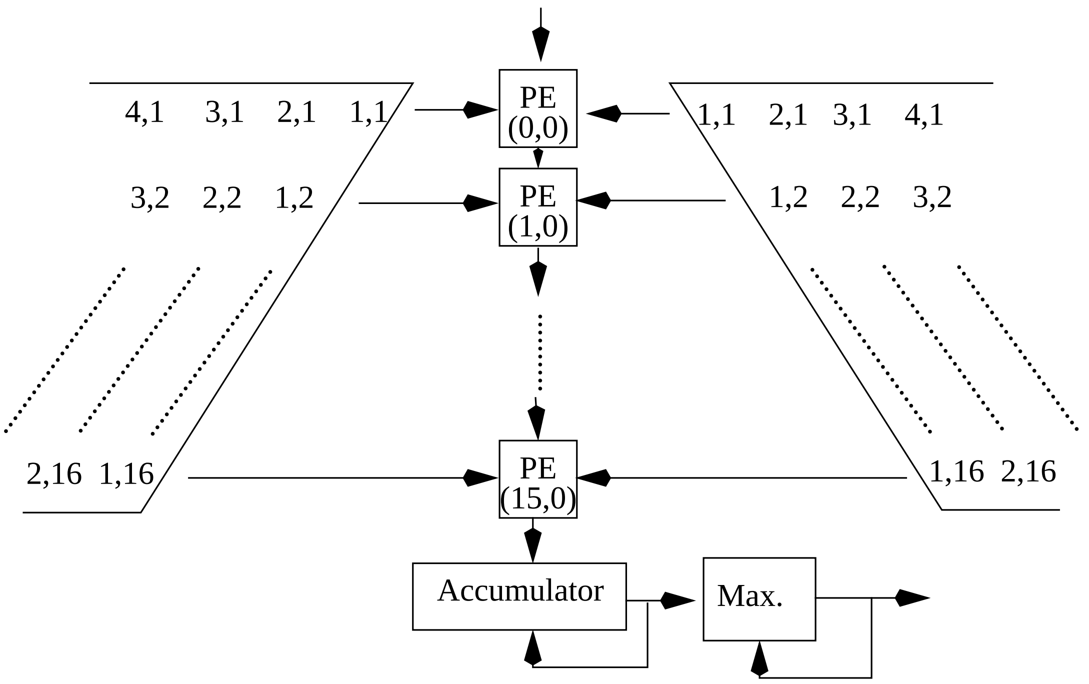
5. Experimental Results

To demonstrate the effectiveness of the vector compaction technique, we carried experiments on MCNC-91 benchmark circuits and some datapath circuits such as adders and multipliers. Two sets of vectors, each having 100,000, are used as input vector sets. The first vector set consists of a highly correlated vector set used for testing purpose. The second one is a vector sequence generated by applying a set of operators on the previous generated vector. The operators include random vector generation, flipping all bits, shift right or left by a fixed number of bits, flip alternate bits, etc. The operators are selected randomly and applied multiple times. The simulator used is a gate-level logic simulator which can measure real delay dynamic power consumption. A clock frequency of 20MHz is assumed and all power measures are in µW. The run time is reported on a SUN SparcStation 20.

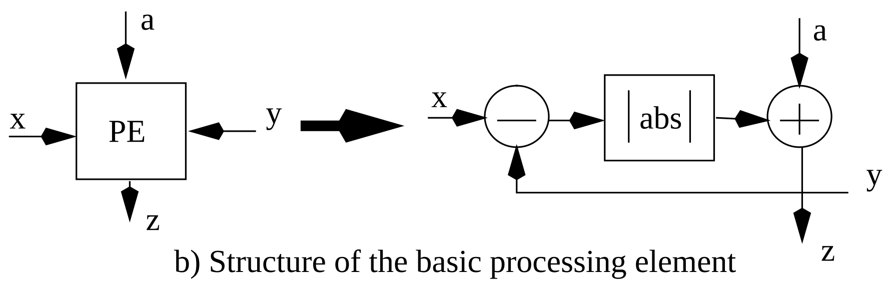
The first experiment is to show the significance of preserving the spatiotemporal correlations during the vector compaction process. We compact the vector set with a compaction ratio of 100X. Gate level power simulator with real delay model is used for the power estimation. Table 1 summarize the results for both unconstrained and constrained symbolic vector compaction, respectively. For each vector set, we provide the compaction results corresponding to when we account for both spatial and temporal correlations and when we only account for temporal correlations. 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 gate-level simulator. The average errors compared to the original vector set are less than 2% for both unconstrained and constrained symbolic vector compaction. The maximum errors are 5.07% and 4.45%, respectively. It is more accurate compared to the case when we only consider temporal correlations which has an average error of 15% and a maximum error of 75%. The reason is that the latter approach does not fully capture the input signal correlation which has significant impact on the total power consumption.The above trend is also observed when we measured the power consumption using zero delay model. Table 2 summarizes the CPU time to compact the input vectors by 100X for the unconstrained symbolic vector generation. It can be seen that the runtime is very fast.

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 using unconstrained symbolic vector generation. Table 3 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 vector compaction program in preserving the power-determining behavior of the original sequence. % error is below 5% in all cases and the average % error is less than 2%.

We also used the vector compaction methodology to implement a multi-level power estimation framework for a video compression application. The power estimation framework is built to estimate the power consumption for different motion estimation architectures. An architectural level simulator was written using C. Streams of video data was input during the architectural level simulation and bit statistics at the inputs of the processing elements in the architecture were collected. After the architectural level simulation, the vector compaction program is called to generate a compacted set of input vector for the detailed gate-level power simulation. We used the framework to simulate the power consumption of a 1-dimensional systolic mesh-connected array architecture for full search motion estimation algorithm [12]. Figure 4 shows the block diagrams of the architecture and the basic processing element. 1,000,000 input vectors derived from real video sequences were used in the architectural simulation and 2,000 vectors were generated using unconstrained symbolic vector compaction. Real delay gate-level power estimations were carried out for both the original vector set and the compacted vector set. Table 4 summarizes the power estimation results. It can be seen that high accuracy can be maintained (about 1.3% error) while the simulation time is reduced by 500 times if the compacted vector set is used.



a) 16 X 1 systolic mesh-connected array architecture



b) Structure of the basic processing element

Figure 4 16 X 1 systolic mesh-connected array architecture and basic processing element

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 cycle required for obtaining power estimation is dramatically reduced. We showed that by keeping the pairwise transition probabilities which is used to mimic the spatiotemporal correlation during the vector compaction, the average error in power estimation using the compacted vector is within 2% of that using the original vector. Also the vector compaction methodology was used to implement an accurate and efficient multi-level power estimation framework.

Bibliography

[1] H.J.M. Veendrick, “Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits”, IEEE Journal of Solid State Circuits, vol. 19, pp. 468-472, August, 1984

[2] S. M. Kang, “Accurate simulation of power dissipation in VLSI circuits”, IEEE Journal of Solid State Circuits, vol. 21, no. 5, pp. 889-891, Oct., 1986

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

[6] A.A. Ghosh, S. Devadas, K. Keutzer and J. White, “Estimation of Average Switching Activity in Combinational and Sequential Circuits”, in Proceedings of IEEE Design Automation Conference, pp. 253-259, June, 1992

[7] 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

[8] 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

[9] S. Chakravarty, “On the complexity of using BDDs for the synthesis and analysis of boolean circuits.”, in Proceedings of the 27th Annual Allerton Conference on Communication, Control and Computing, pp 730-739, 1989

[10] 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

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

[12] P. Pirsch and T. Komarek, “Array architectures for block matching algorithms”, IEEE Transactions on Circuits and Systems, vol. 36, no. 36, pp. 1301-8, October, 1989

Circuits	Vector Sequence S1				Vector Sequence S2			
	Power Est. of orig. vector set	% Error			Power Est. of orig. vector set	%Error		
		Comp. using te only	Unconst. comp. using sp_te	Const. comp. using sp_te		Comp. using te only	Uncons. comp. using sp_te	Const. comp. using sp_te
9symml	2142.1	9.24	0.34	0.35	2210.0	36.67	0.96	0.00
C432	856.8	24.94	1.72	1.11	701	9.60	0.43	1.39
C880	1403.2	3.00	0.98	0.07	1282.2	3.74	0.44	0.12
C1908	1173.7	7.85	0.38	0.17	1110.9	8.60	0.40	1.38
C3540	11985	13.36	0.58	0.04	13516.6	3.65	2.02	1.04
C6288	98573	0.25	3.25	1.17	122789	44.21	0.57	2.32
add16	1368.7	3.98	0.77	0.57	1286.2	16.39	0.49	1.71
mul2	153.1	75.31	0.85	0.69	140.9	2.63	0.92	0.55
mul4	770.2	14.31	0.52	0.61	879.8	15.51	1.07	0.54
mul8	5206.1	8.45	1.14	0.80	6193.6	11.05	0.28	1.52
mul16	27815	18.67	5.07	2.10	37552	40.70	0.36	2.10
cordic	502.0	11.67	0.44	0.51	483.8	7.28	0.12	0.05
f51m	798.0	15.48	0.15	0.94	970.8	30.28	3.09	3.10
comp	4355.2	12.62	0.99	1.97	3782.1	22.32	0.40	3.11
count	3314.5	6.66	3.43	2.16	2500.1	14.94	0.64	0.21
k2	4301.6	0.24	3.08	4.45	3738.7	3.58	2.57	3.74
x1	2379.1	2.53	1.65	0.39	2147.1	9.27	0.15	0.95
apex7	1725.6	8.19	0.31	0.11	1653.4	6.35	1.14	0.57
Average % error		13.15	1.43	1.01		15.93	0.89	1.35

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

Circuit	unconstrained symbolic vector compaction (sec)
9symml	0.5
C432	7.8
C880	26.1
C1908	7.65
C3540	17.2
C6288	8.43
adder16	6.63
mul2	0.93
mul4	1.6
mul8	7
mul16	25.3
cordic	4.34
f51m	0.4
comp	5.73
count	6.48
k2	13.25
x1	20.7
apex7	13.6

Table 2: CPU time (in sec) for unconstrained symbolic vector compaction

	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
C1908	1.282	1.254	1.923	1.936
C3540	3.397	3.488	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		1.87		1.41

Table 3. Current Estimation by PowerMill

Circuit	Power estimation using 1,000,000 vectors	Power estimation using 2,000 compacted vectors
16X1 motion estimation architecture	150.61	152.58

Table 4: Comparison of the power estimation for the motion estimation architecture using the compacted vector set