A Structure-Oriented Power Modeling Technique for Macrocells

Jiing-Yuan Lin, *Student Member, IEEE,* Wen-Zen Shen, *Member, IEEE,* and Jing-Yang Jou, *Member, IEEE*

***Abstract*—To characterize the power consumption of a macrocell, a general method involves recording the power consumption of all possible input transition events in the look-up tables. However, though this approach is accurate, the size of the table becomes very large. In this paper, we propose a new power modeling technique that takes advantage of the structural information of a macrocell. In this approach, a subset of primary inputs and internal nodes in the macrocell are selected as the state variables to build a state transition graph (STG). These state variables can model the steady-state transitions completely. Moreover, by selecting the characterization patterns properly, the STG can also model the glitch power in the macrocell accurately. To further simplify the complexity of the STG, an incomplete power modeling technique is presented. Without losing much accuracy, the property of compatible patterns is exploited for a macrocell to further reduce the number of edges in the corresponding STG. Experimental results show that our modeling techniques can provide SPICE-like accuracy, while the size of the look-up table is significantly reduced.**

***Index Terms*— Power characterization, power modeling for macrocells, simulation-based RTL power estimation, state transition graph.**

I. Introduction

RECENTLY, with the progress in deep submicrometer technology, it has become easy to produce a high-density chip capable of running at high frequencies. Therefore, in addition to area and performance, power has become another critical concern in very large scale integration (VLSI) design. In general, power reduction can be achieved at various abstraction levels ranging from system and architecture to circuit and layout [1]. Regardless of the level, a fast and accurate power analysis tool is necessary to quickly identify the problem spots in the design [2].

In general, SPICE is often used to simulate the performance and power of application-specific integrated circuit (ASIC) designs. Although SPICE simulation is very accurate, it suffers from severe memory and execution time constraints for VLSI circuits. PowerMill [3] and IRSIM-CAP [4] are much faster than SPICE, however, their reduction in central processing unit (CPU) time are still not effective enough to make them capable in managing chips with hundreds of thousands of transistors. Recently, several cell-based power estimation algorithms [5]–[9] have been proposed to solve this problem. However, almost all of them focus on the circuits that only consist of basic gates. Macrocells such as adder, multiplexer, and bit-sliced arithmetic and logic unit (ALU) are seldom discussed, even though they are widely used in ASIC designs.

Recently, several behavior level and register–transfer (RT) level power estimators [10]–[15] have been proposed which consider the modeling and characterization of datapaths based on the input statistics. However, the input statistics are highly dependent upon the application domain of the circuit [16]. Therefore, the energy models of the macrocells derived from specific input statistics are not general. To overcome this drawback, a simple modeling technique, which characterizes the power consumption caused by all possible transition events at the primary inputs (PI's) was proposed in [17]. This technique is referred to as the **PI-oriented power modeling technique**. Any transition event occurring at the PI's may result in many signal transitions in the macrocell. Therefore, the estimated power of that event would automatically take into account all of the power consumption in the macrocell. This technique is accurate. However, for a macrocell with n inputs, a look-up table is required with a size of 2^{2n} to store the characterization data. Moreover, the complexity of the entire characterization process is too great. To reduce the size of the look-up table, a clustering algorithm was proposed by Mehta *et al.* [16]. They first simulated a circuit with m random vectors $(m \ll 2^{2n})$ using a switch-level simulator. Then, according to the simulated switching capacitance, energy patterns that were closely related were grouped into clusters. Since $m \ll 2^{2n}$, e.g., $m = 500$ for $n = 11$ in their experiments, the number of clusters was much less than 2^{2n}. In this regard, the size of the look-up table was reduced significantly. However, accuracy was obviously sacrificed because a lot of input transition vectors, which were not selected for characterization, were disregarded during clustering.

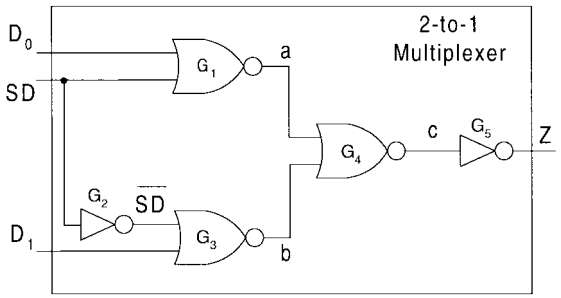
In a cell library, there are macrocells with different circuit structures. In this paper, we will focus on the power modeling of datapath macrocells. A macrocell without a feedback loop in the circuit is called a **combinational cell**. In this paper, the proposed power modeling techniques are illustrated based on a zero-delay model. Although the partial glitch power can be handled by these modeling techniques, we will not discuss this until Sections IV and V. Thus, unless otherwise specified, a macrocell will be referred to as a combinational cell with zero-delay model.

Manuscript received August 26, 1997; revised January 9, 1998 and May 22, 1998. This work was supported in part by the National Science Council under Contract NSC86-2221-E009-009.

J.-Y. Lin was with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. He is now with Global UniChip Corporation, Hsinchu 300, Taiwan, R.O.C.

W.-Z. Shen and J.-Y. Jou are with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: wzshen@cc.nctu.edu.tw).

Publisher Item Identifier S 1063-8210(99)02606-2.



(a)

D_0	D_1	SD	a	b	c	Z	$\overline{SD}$
0	0	0	1	0	0	1	1
0	0	1	0	1	0	1	0
0	1	0	1	0	0	1	1
0	1	1	0	0	1	0	0
1	0	0	0	0	1	0	1
1	0	1	0	1	0	1	0
1	1	0	0	0	1	0	1
1	1	1	0	0	1	0	0

(b)

Fig. 1. A 2-to-1 multiplexer and its extended truth table.

For a macrocell, there could be two or more input patterns, which generate the same states for all internal nodes in the macrocell. We refer to these patterns as **compatible patterns**. A **compatible pattern set** is the collection of all compatible patterns. As in the example of Fig. 1, (a) shows a schematic diagram of a 2-to-1 multiplexer and (b) shows its extended truth table in terms of the internal nodes and output node. In the extended truth table, we find that the input patterns 000 and 010 have the same logic values as the internal nodes as well as the output node and are, thus, compatible patterns. Likewise, patterns 001 and 101, 011 and 111, and 100 and 110 are also compatible patterns. There are four compatible-pattern sets in this example. Although the sequences formed by these compatible patterns do not make signal transitions in the macrocell, they are used for the characterization in the PI-oriented power modeling technique. The power characterization process would waste CPU time and memory storage space.

In this paper, we will propose a structure-oriented power modeling technique for macrocells. In our approach, all internal nodes in a macrocell as well as the PI's are all considered as potential state variables for building a state transition graph (STG) where the state variables can be used to completely model the signal transitions of all internal nodes and primary outputs (PO's). Based on the model, we will consider the compatible patterns for reducing the characterization vectors while retaining the same accuracy as that provided by a PI-oriented power modeling technique. For some macrocells, the constructed STG's may still be very complicated. To further reduce the complexity of the STG's, we will introduce an incomplete power modeling technique that can effectively simplify the STG's without losing much accuracy. Since the STG's are built based on a zero-delay model, they cannot model glitches precisely in the realistic circuit. However, those glitch powers are implicitly included in the edges. By selecting the characterization patterns properly, the STG's can model glitch powers in the circuits accurately. We will present a heuristic method to select those characterization patterns.

The remainder of this paper is organized as follows. In Section II, we will present a power modeling technique, which can completely model the steady-state transitions of all internal nodes and PO's. In Section III, we will present an incomplete power modeling technique to simplify the size of the STG. We will discuss the impact of glitch power on our STG model in Section IV. The methods for characterizing the macrocells and finding the edge activity number for STG's are also presented in this section. In Section V, some experimental results are presented and discussed. Finally, we present our conclusions in Section VI.

II. A Complete Power Modeling Technique

A. The STG

For a CMOS digital circuit, the majority of the power is dissipated when signals make transitions. Thus, if we can model all of the possible transition events at the internal nodes of a macrocell, we can estimate its power consumption accurately. In this paper, if a power model can model all of the signal transitions of the internal nodes, it is called a **complete power modeling technique**. As mentioned before, the PI-oriented power modeling technique belongs to this category. For a circuit with n inputs, the power characterization is performed with $2^n \times 2^n$ sequences. We can model these sequences using an STG which has 2^n states to represent the 2^n possible input combinations at time $k-1$. Each state has 2^n outgoing edges to represent the 2^n possible input combinations at time k. We refer to the STG built for the complete power modeling technique as a **completely modeled STG**. Though the PI-oriented power modeling technique can provide accurate results, a size of 2^{2n} would be easily exploded.

In this section, we try to develop a complete power modeling technique on which the STG can be built with a smaller number of edges as compared to that of the PI-oriented power modeling technique. Fortunately, the property of compatible patterns of a circuit can be used to achieve this goal. For a macrocell, after running an exhaustive logic simulation, we can build an extended truth table in terms of all of the internal nodes and PO's, as shown in Fig. 1(b). From the truth table, we can find all of the compatible pattern sets. Since the patterns in a compatible pattern set possess the same logic values of each internal node, the number of states in the STG would be equal to the number of compatible pattern sets. For the example in Fig. 1, there are four compatible pattern sets and, thus, the corresponding STG shown in Fig. 2 has only four states. In this STG, a, b, c, Z, $\overline{SD}$ are the state variables and D_0, D_1, SD are the input variables.

This graph can model all of the steady-state transitions of the internal nodes; therefore, it is a completely modeled STG. In

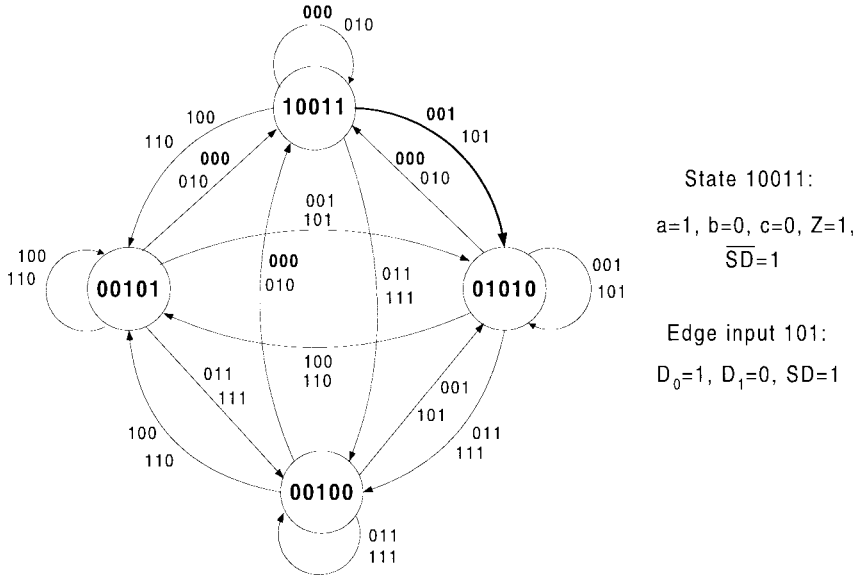


Fig. 2. The completely modeled STG of a 2-to-1 multiplexer.

this power modeling technique, the number of states is always less than or equal to that of the PI-oriented power modeling technique. The worst case is when all of the internal nodes have different logic values for each input pattern. In other words, there are no compatible patterns.

For a macrocell, the power consumed by its input capacitance due to the input transitions can be counted in the preceding stage. Therefore, the difference in power consumed by two compatible patterns is due to the following factors:

1) capacitive feedthrough current [9];
2) charging and discharging currents of parasitic capacitance inside the gates that are driven directly by PI's;
3) glitch power.

In general, the power from the first two factors is much smaller than that of the entire macrocell. The impact of the glitch power on the complete power modeling system is also minor, which will be discussed in Section IV. As this power is neutralized, the outgoing edges corresponding to the compatible patterns can be merged into a single edge for each state. In the simplified STG, the number of outgoing edges (or incoming edges) of each state is equal to the number of states. Thus, if the number of states is N, the total number of edges would be $N \times N$, which is independent of the number of PI's.

B. The State-Variable Selection Strategy

In Fig. 2, we selected all of the internal nodes as state variables for modeling the signal transitions in the circuit. In fact, if we utilize the logic relationship among the internal nodes, it is possible to use only part of those nodes as state variables to model the entire circuit. Finding a set of nodes which can completely model all of the transitions serves two purposes. First, we only need to check those nodes in the extended truth table when we evaluate the circuit for finding compatible patterns. Second, we can possibly remove part of those nodes from the modeled STG to form a simpler STG while losing little accuracy. We will discuss this in the following section. In this subsection, we will propose an algorithm for selecting the state variables of the completely modeled STG.

Let's define some notations first. For a circuit, if a node m incurs signal transitions whenever another node n makes a transition, and vice-versa, nodes m and n are referred to as **transition compatible nodes**. In Fig. 1(a), we find that SD and $\overline{SD}$ are transition compatible nodes, as are nodes c and Z. We only need to select one transition compatible node as the state variable for constructing the STG. If the signal transitions of a node m can be modeled by the transitions of a set of nodes, node m is referred to as the **transition dominated node** of those nodes. On the contrary, those nodes are called the **transition dominating nodes** of node m. In Fig. 1(a), D_0 and SD, $\overline{SD}$ and D_1, a and b are the transition dominating nodes of nodes a, b, and c, respectively. Certainly, nodes a and b are also the transition dominating nodes of node Z. In fact, if the STG already uses all of the transition dominating nodes of a logic gate as state variables, the output of the gate does not need to be selected as a state variable. Therefore, our goal is to find a set of transition dominating nodes which can completely model all of the transitions in a macrocell and use those nodes as state variables for constructing an STG with a limited size.

A combinational cell has the important property that the logic value of a node is only dependent upon the logic values of its transitive fan-in nodes. We can thus level the netlist starting from the PI's with level 0 and label the level of each node i with l_i. If a logic gate has m inputs with levels $l_1, l_2, \cdots$, and l_m, we define the level of its output as

$$\min(l_1, l_2, \cdots, l_m) + 1. \tag{1}$$

Based on this definition, the node would be in level 1 if at least one PI is among its inputs. Any node with a level greater than one is driven directly or indirectly by the nodes in level 1. Thus, if we choose all of the nodes in level 1 as state variables, the corresponding STG can model the transition behavior of all

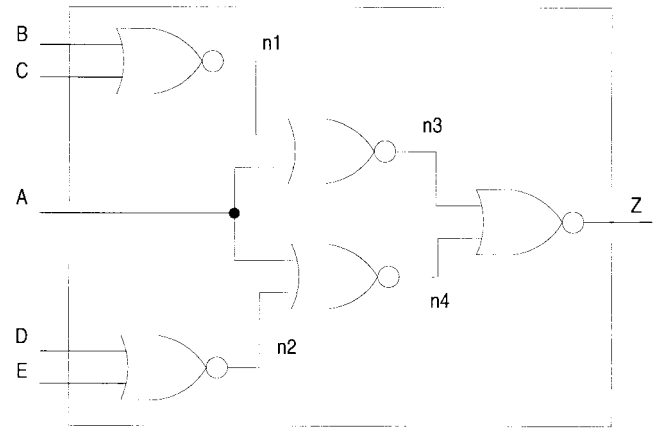


Fig. 3. An example for selecting state variables.

internal nodes in that circuit. For convenience, we collect those nodes in a **state variable set** (SVS). As shown in Fig. 1, nodes a, b, and $\overline{SD}$ are in level 1. Therefore, SVS $= \{a, b, \overline{SD}\}$.

In a circuit, it is possible that some nodes in level 1 are driven by a common PI and the side inputs of those nodes are already in the SVS. In this case, we can select the common PI as a new state variable and remove those nodes from the SVS without losing the accuracy of the model. This can further reduce the size of the STG. As shown in Fig. 3, the nodes $n1$, $n2$, $n3$, and $n4$ initially constitute the SVS for a complete power modeling technique. Thus, SVS $= \{n1, n2, n3, n4\}$. After removing invalid states such as 1010 ($n1 = 1$, $n2 = 0$, $n3 = 1$, $n4 = 0$), the corresponding STG would have nine states and 81 edges. However, if we select input A as a new state variable, node $n3$ ($n4$) can be removed from the SVS because nodes A and $n1$ ($n2$) can model node $n3$ ($n4$). Therefore, SVS $= \{n1, n2, A\}$ and the size of the corresponding STG would be reduced to eight states and 64 edges. This procedure guarantees that the number of states in the final STG would be reduced; however, the number of state variables is not always reduced. Thus, the basic idea of our algorithm is to find a set of the transition dominating nodes in levels 0 or 1, such that the corresponding completely modeled STG has the fewest states possible. We have implemented an algorithm whose pseudo-code is shown in Fig. 4.

In this algorithm, first we level the netlist of a macrocell, then, we initially select those nodes in level 1 as state variables and put them into SVS_{orig}. For each state variable combination C_i in SVS_{orig}, we evaluate the possible gain by replacing it with a PI to obtain a simpler STG. If we want to remove a state variable sv_i from the SVS_{orig}, the state variables in the SVS_{new} must be able to model the removed state variable sv_i. In other words, the transition dominating nodes of sv_i should all be included in the SVS_{new}. In our approach, we backtrack the circuit from SV_i to PI and add its transition dominating nodes (PI's) to the SVS_{new}. The added state variables may be able to model other nodes in the SVS_{new}. Therefore, we perform a forward traverse from PI to level 1 to find those nodes and delete them from the SVS_{new}. With the updated SVS_{new}, we evaluate whether the number of states is fewer than that of $\text{SVS}_{\text{final}}$. If the new

```
Input: the netlist of a macrocell
Output: state variable set
State_Variable_Selection_for_Complete_Power_Modeling()
{
    level the netlist;
    select the nodes in level 1 as state variables and put them into SVS_orig;
    SVS_final ← SVS_orig;
    for each state variable combination C_i in SVS_orig {
      backtrack the circuit from the nodes in C_i to PI and collect
their transition dominating nodes in a set TGN;
      forward traverse the circuit from the primary inputs in TGN
to the nodes in level 1 and find the state variable SV which can be
modeled by those inputs;
        SVS_new ← SVS_orig + TGN - SV;
        if STG(SVS_new) is simpler than STG(SVS_final)
            SVS_final ← SVS_new;
            /* STG(SVS_new) is the STG built by SVS_new. */
    }
    return (SVS_final);
}
```

Fig. 4. An algorithm for finding the SVS for complete power modeling.

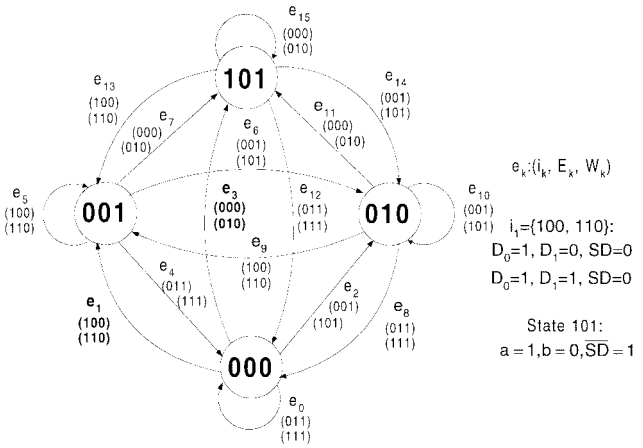


Fig. 5. The completely modeled STG of a 2-to-1 multiplexer.

STG is simpler than the original one, we replace $\text{SVS}_{\text{final}}$ with SVS_{new}. After running exhaustive enumeration on the possible state variable combinations in SVS_{orig}, we can find the best SVS which is the $\text{SVS}_{\text{final}}$. If the CPU time is limited, we can perform heuristic enumeration instead of exhaustive enumeration for the state variable combinations to obtain a near-optimal solution.

After applying the above algorithm to the 2-to-1 multiplexer, a, b, and $\overline{SD}$ are selected as the state variables. Fig. 5 shows the simplified STG that is similar to those graphs proposed in [6] and [7]. Each edge e_k:(i_k, E_k, W_k) models the power consumption of a state transition from one state to the other. i_k is the input patterns (compatible patterns) which trigger the state transition. To clearly demonstrate the dependency between the input pattern and state transition, compatible patterns with parenthesis are placed under the label of each edge. E_k is the edge activity number, which denotes the number of traverse times of the edge when a set of sequential

patterns are applied. W_k is the total energy consumed when the edge is traversed each time. Thus, if all E_k's and W_k's of each edge e_k are known, the total energy consumption can be obtained by summing up the $E_k \times W_k$ of each edge.

III. An Incomplete Power Modeling Technique

Although the number of edges of the completely modeled STG's is less than that of the PI-oriented power modeling technique, if we can remove some state variables and edges from the STG's, the power characterization process would become more efficient. In this section, an **incomplete power modeling technique** is presented. The corresponding STG is referred to as an incompletely modeled STG. In the following, we will present a heuristic method for selecting the state variables of the **incompletely modeled STG**.

As mentioned before, only the nodes in levels 0 and 1 are the possible state variables in the completely modeled STG. Any node n_i with level $l_i \geqq 2$ can be completely modeled by these state variables. To reduce the size of the STG further, we may remove some state variables from the SVS. However, all transition dominated nodes of the removed state variables may not be modeled completely in the simplified STG. Therefore, to model these transition dominated nodes completely, part or all of them need to be selected as new state variables in the simplified STG. The challenge of our approach is to find the potential nodes being deleted from the SVS without losing much accuracy.

Switching power is recognized as the dominating factor of power consumption in a complimentary metal–oxide–semiconductor (CMOS) logic gate. Basically, it is proportional to its capacitive loading times its transition activity. The capacitive loading of a node can be extracted directly from the circuit layout. However, during characterization phase, it is difficult to predict the transition activity of the node because the macrocell can be used in different applications with different input characteristics. In the following, we will roughly estimate the switching power of the nodes in a circuit. Then, according to the switching power, we try to remove some nodes from the SVS such that the accuracy loss is within a user-specified value and the reduction of the table size can be as large as possible. In our approach, we first run zero-delay Verilog simulation by applying a set of random patterns and then estimate the switching power of all internal nodes from their transition activities weighted by the corresponding capacitive loadings. Here, we introduce a parameter called **accuracy loss ratio** (ALR) of a node which is defined as the ratio of the switching power of a node to the switching power of the entire circuit. Clearly, the ALR of a node can be used to represent the possible accuracy loss in percentage if the node cannot be modeled completely by the resulting STG.

In conjunction with the accuracy loss, the removal of a node from the SVS would possibly reduce the size of the lookup table. The difference in table sizes before and after the removal of a node is referred to as the **table size reduction** (TSR) of the node. Obviously, if a node has low ALR and high TSR, then it is preferable to discard this node from the SVS for

Input: the netlist of a macrocell, the state variable set (*SVS*) of complete power modeling, a user-specified percentage error (*US_Error*)
Output: the SVS of incomplete power modeling

```
State_Variable_Selection_for_Incomplete_Power_Modeling()
{
    calculate_ALR_of_each_node();
    calculate_TSR_of_each_node();
    cu_ALR = 0;
    /* cu_ALR is the cumulative ALR of the nodes being removed
from SVS */
    while (cu_ALR < US_Error) {
        for each state variable SV in SVS {
            if (TSR(SV) is zero) {
            /* TSR(SV) is the TSR of state variable SV */
                collect the fanout gates of SV in FOG;
                ALR(FOG) += ALR(SV);
                /* ALR(SV) is the ALR of state variable SV */
                SVS = SVS + FOG - SV;
            }
            else if (ALR(SV)+cu_ALR < US_Error)
                SV_the_largest_TSR = find_the_largest_TSR(SV,
SV_the_largest_TSR);
                /* SV_the_largest_TSR is the state variable that
so far has the largest TSR */
        }
        cu_ALR += ALR(SV_the_largest_TSR);
        collect the fanout gates of SV_the_largest_TSR in FOG;
        SVS = SVS + FOG - SV_the_largest_TSR;
        /* the nodes in FOG are appended in the list of SVS */
    }
    return(SVS);
}
```

Fig. 6. An algorithm for finding the SVS for incomplete power modeling.

constructing the incompletely modeled STG. In the following, we will present a simple method to evaluate the TSR's of all internal nodes in a circuit. Given a macrocell, we first ran an exhaustive logic simulation and built an extended truth table. The TSR's of the internal nodes will be calculated sequentially from the nodes in level 1 to the nodes in the maximal level. The TSR of a node can be obtained easily by calculating the compatible pattern sets for two different conditions where the node is included and excluded in the extended truth table. After evaluating a node, the node is removed from the extended truth table forever. Basically, to examine a node, part or all of its transition dominating nodes must have been examined before. If all of its transition dominating nodes are not examined yet, then there is no gain by removing the node because it can still be modeled by its transition dominating nodes. Therefore, for the nodes in the same level, the one that has the largest number of removed transition dominating nodes will be given the highest priority for the next examination.

According to the ALR and TSR information, the objective of our approach is to find an SVS to satisfy the constraint of a user-specified percentage error such that the table size is as small as possible. We have implemented an algorithm for finding the SVS for incomplete power modeling. The pseudo-code is shown in Fig. 6.

In the algorithm, only one user-specified parameter is needed. The initial SVS is obtained from the complete power

modeling technique. The user-specified percentage error can be regarded as the accuracy budget that we must work within. In each evaluation, a state variable that satisfies the constraint of the remaining budget and has the greatest impact on the reduction of the table size will be selected as a potential candidate for removal. In a circuit, if we remove a node from the SVS and add its transition compatible node to the SVS at the same time, the size of the resulting STG will not be changed and the accuracy is certainly retained. For such a node, we will remove it from the SVS without deducting its ALR from the budget right away. Its ALR will be added to the transition compatible node. Therefore, if its transition compatible node is removed from the SVS later, the total accuracy loss of both nodes will be deducted from the budget. Before deleting a state variable, the output nodes of its fanout gates should all be selected as new state variables. The purpose is to guarantee that the simplified STG can still model all of the transition dominated nodes of the removed state variables. This evaluation process is repeated until the accuracy budget is used up.

IV. Power Characterization and Power Estimation for Macrocells

A. The Impact of Glitch Power on Complete and Incomplete Power Modelings

As mentioned before, the complete and incomplete power models are derived under a zero-delay model. Under steady state, the STG's can model the selected internal nodes as well as their transition dominated nodes exactly. However, even though the sequences formed by the compatible patterns do not make steady-state transitions for these nodes, they could result in glitches in the circuits. In the following, we will discuss the impact of this kind of glitch power on complete and incomplete power models respectively.

In the PI-oriented power modeling technique, the power consumption of an input transition event would contain all of the power consumption of the triggered signal transitions in the circuit. Those triggered signal transitions contain not only the steady-state transitions, but also glitches. However, our modeling techniques assume that the logic gates located between the PI's and the selected state variables possess zero-delay, while the remaining logic gates possess general delay. Therefore, our STG modeling technique can model not only the steady-state transitions of the state variables and their transition dominated nodes, but also the glitches caused by the steady-state transitions. The glitch power resulting from the transitions of compatible patterns depends on the levels of the selected state variables. The greater the level of the selected state variables, the greater the glitch power the STG cannot model. Fortunately, in the complete power modeling technique, only the nodes in levels 0 or 1 are the candidates for the state variables. Furthermore, because the selected state variables are very close to PI's, those input patterns in the same compatible pattern set, in general, differ by only a few bits. This could also reduce the probability of generating glitches. However, in the incomplete power modeling, the levels of the selected state variables could be distributed over a wide range. Therefore, the glitch power not being modeled by the STG could be large.

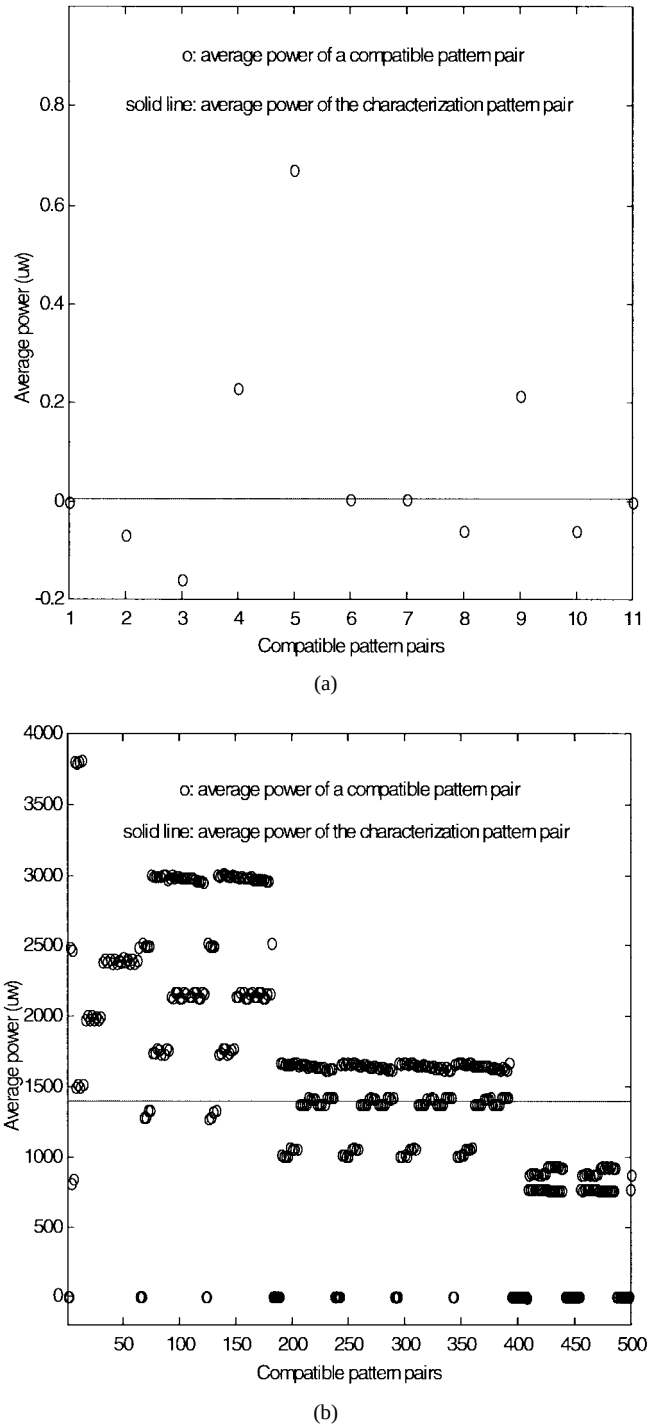


Fig. 7. The average power of compatible pattern pairs of (a) 1-b full adder using complete power modeling and (b) 4-b fast adder using incomplete power modeling.

To evaluate the impact of the glitch power on our STG models, some experiments involving 1-b full adder and 4-b fast adder were conducted. In these circuits, we selected a compatible pattern set randomly and ran compatible patterns exhaustively in pairs. The circles in Fig. 7(a) and (b) show the average power of compatible pattern pairs of a 1-b full adder

using complete power modeling and a 4-b fast adder using incomplete power modeling. In Fig. 7(a), although the circles were distributed over a wide range, their values are generally two orders of magnitude less than the power of a normal input transition. However, in Fig. 7(b), the average amount of power consumed by those compatible pattern pairs on average were close to the power of a normal input transition.

B. Power Characterization

In the two preceding sections, we proposed two algorithms to build completely and incompletely modeled STG's of macrocells. For each STG, we generated a file that records the compatible patterns of each state. We refer to this file as a **compatible pattern file** (CPF). During the power characterization and estimation phases, we can easily rebuild the STG based on the CPF. For an STG with n inputs, the size of the CPF would be 2^n.

Once the STG is built, we run SPICE and PowerMill simulations to characterize the energy consumption (W_k) associated with each edge. It is well known that the power consumption of a logic gate is strongly dependent on the input slope and output loading. Therefore, to characterize its power, several runs of circuit/transistor level simulation are needed for different input slopes and output loadings. However, for macrocells, the dependency of power on the input slope and output loading is not as strong as that for basic gates. In general, the input slope and output loading primarily affect the short-circuit power of the logic gates in level 1 and the dynamic power of the PO's, respectively. If a macrocell consists of many transistors, the impact of the input slope and output loading on the power would be minor. We can thus reduce the number of simulation runs without losing much accuracy. Since most of the macrocells have multiple inputs and outputs, it is too time consuming to characterize different slew rates to each PI and different loadings to each PO. In our approach, we assigned the same input slew rate and output loading for all PI's and all PO's, respectively, in each characterization run.

To characterize the W_k of each edge in STG, we required an input sequence which could traverse the corresponding edges. If the STG has M states, we traverse the states of the STG in the following sequence and generate the corresponding input patterns

$$S_1 \to S_1 \to S_2 \to S_1 \to S_3 \to S_1 \to \cdots \to S_1 \to S_M \to S_1 \to S_2 \to S_2 \to S_3 \to S_2 \to S_4 \to \cdots \to S_{M-2} \to S_{M-1} \to S_{M-1} \to S_M \to S_{M-1} \to S_M \to S_M$$

where S_i is the ith state and $S_i \to S_j$ represents an edge making a state transition from S_i to S_j. Based on this pattern generation procedure, the length of the sequence is n^2+n-1, where n is the number of states.

For each $S_i \to S_j$, we needed to choose a proper pattern among the corresponding compatible patterns as the characterization pattern. As mentioned before, the incoming edges of a state have the same compatible pattern set. In the following, for the sake of easy explanation, the compatible pattern set of a state represents the compatible pattern set of its incoming edges. If an edge e_k makes a state transition from S_i to S_j, and S_i and S_j have m and n compatible patterns respectively, there are $m \times n$ compatible pattern pairs, which can be used to characterize the edge e_k. As mentioned before, different compatible pattern pairs may cause different glitch power, especially an incomplete power model. To reduce the impact of glitch power on the STG model, we chose the characterization patterns according to the following strategy. For an edge $S_i \to S_j$, first we calculated the average hamming distance between the compatible pattern sets of S_i and S_j using

$$HD^{ij}_{\text{avg}} = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} H(I_i, I_j) \tag{2}$$

where I_i (I_j) denotes a compatible pattern belonging to the compatible pattern set of S_i $(S_j) \cdot H(I_i, I_j)$ denotes the total number of bits in which patterns I_i and I_j differ. Since we generate the pattern according to the sequences mentioned above, the starting input pattern for edge $S_i \to S_j$ has been determined before generating the succeeding pattern for edge $S_i \to S_j$. Assume that the pattern is I_{k-1}, which certainly belongs to the compatible pattern set of S_i. Next we try to find a pattern I_k from the compatible pattern set of S_j such that $H(I_{k-1}, I_k)$ is the closest to HD^{ij}_{avg}.

To evaluate the effectiveness of this strategy, we conducted an experiment as follows. We selected two compatible pattern sets S_i and S_j randomly from the incompletely modeled STG of a 4-b fast adder, where there are 17 and 27 compatible patterns of S_i and S_j, respectively. In the experiment, first we found the average power of exhaustive pattern pairs (17×27 pairs) using SPICE simulation. The result is shown in Fig. 8 by a solid line. Then, the average hamming distance between the two compatible pattern sets were calculated according to

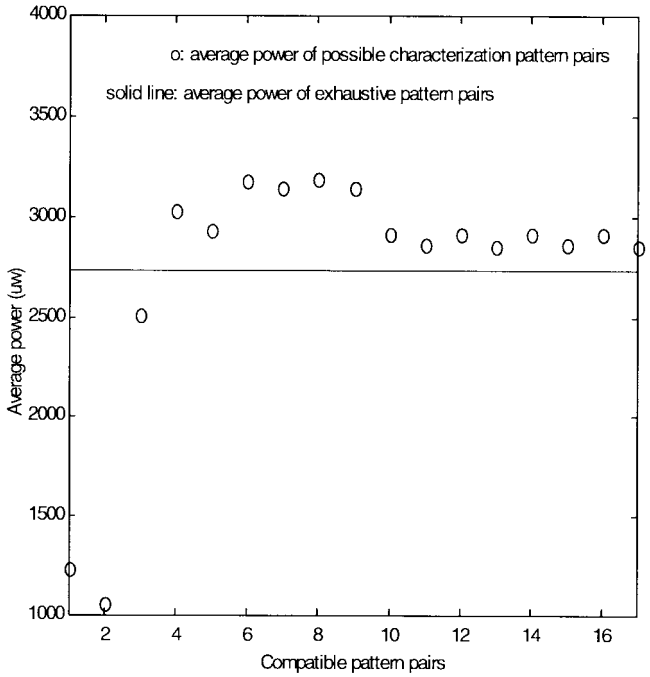


Fig. 8. The average power of exhaustive pattern pairs and the possible characterization pattern pairs of two compatible pattern sets.

(2). Following that, we found the possible pattern pairs for characterizing the edge starting from S_i to S_j. For each pattern I_i in the compatible pattern set of S_i, we searched for a suitable pattern I_j among the compatible patterns of S_j such that $H(I_i, I_j)$ is as close to the average hamming distance as possible. In Fig. 8, there are 17 circles that show the average power of those characterization pattern pairs. We found that most of circles were close to the solid line. Thus, it seems to be reasonable to use a pattern pair whose hamming distance is close to the average hamming distance to characterize the average power of the edge. In [21], the authors reached the similar conclusion that the average hamming distance between two consecutive vectors seemed to be a reliable indicator of the average power consumption of that circuit.

For an STG with n inputs, the complexity of computing the average hamming distance is $2^n \times 2^n$, which grows exponentially with the number of inputs. For some macrocells, if the number of inputs is large enough, we can use another method to find the characterization pattern pairs. Assume that an input I_i has been determined to characterize an edge $S_k \rightarrow S_i$ and we want to characterize the following edge $S_i \rightarrow S_j$. First, we calculate the average hamming distance between I_i and the compatible pattern set of S_j using

$$HD_{\text{avg}}^{I_i j} = \frac{1}{n} \sum_{j=1}^{n} H(I_i, I_j). \tag{3}$$

From the compatible pattern set of S_j, we find a pattern I_j such that $H(I_i, I_j)$ is the closest to $HD_{\text{avg}}^{I_i I_j}$. This approach can reduce the complexity of computing the average hamming distance to 2^n. By conducting the above experiment with this approach, we found that the distribution of the average power of all possible characterization pattern pairs is similar to that shown in Fig. 8.

During characterization, the characterized data of each edge are stored sequentially as the order of generating its characterization patterns. These data will be retrieved with the same sequence when the power estimation is performed. For an STG with n inputs and M states, if we characterized the macrocell with N different average loadings and different average slopes, the table size needed would be $N \times (M \times M) + 2^n$ where 2^n is the size of the CPF.

We have built an automatic procedure to generate the input sequences for characterization and to run SPICE and PowerMill simulations for calculating the W_k's. The power consumption of each characterized edge includes not only the dynamic power due to steady-state transitions and glitches, but also the short-circuit power and leakage power.

C. Power Estimation

Our power estimator is embedded in Verilog-XL, which is used as the simulation platform to obtain the signal transitions at the inputs of each macrocell in the circuit. As mentioned before, the power consumption of a macrocell can be obtained by summing up the product of E_k and W_k for each edge in the corresponding STG. In the following, we will present the way to calculate E_k and W_k for each edge.

When we perform power estimation for a macrocell, we build the corresponding STG according to its CPF. Before estimating the power, we need to calculate the average input slew rate and average output loading for finding the W_k of each edge. The loading of a PO can be obtained by summing up the input capacitance of its driven cells and the capacitance of the interconnection lines, while the input slew rate of a PI is calculated according to the precharacterized input rising/falling delays. For each macrocell, we use the average values of its input slew rates and output loading for looking up the W_k in the table. When the W_k is not available for a specific input slew rate and output loading, interpolation and extrapolation are used to find the value.

When we perform logic simulation, we monitor the signal transitions at the inputs of each macrocell. We will enumerate the STG according to the signal transitions at PI's and count the activities of each edge. As the logic simulation is completed, we can obtain the activity number of each edge. This approach is very simple and can accurately capture the correlation of the signals from the inputs during the logic simulation.

When we simulate a circuit composed of several macrocells, glitches may be generated and propagated from one macrocell to another [18]–[20]. Since we use Verilog-XL simulation to capture the signal transitions at the inputs of each macrocell, if all glitches are considered as complete transitions no matter how small the duration of each glitch, we will overestimate the power of the circuit. Najm [18] proposed a low-pass filter technique to filter out short pulses. However, those glitches not being filtered out may not be in full swing. Therefore, Rabe [19] and Tsai [20] proposed similar power models to resolve these glitches. In our approach, we adopted the method of [20]. During the characterization phase, we applied some input patterns with full swing from V_{dd} to ground or vice versa to characterize the power of the macrocells. The glitch power inside the macrocells, which may have full or partial swing, would be taken into account in the characterized energy weightings of the edges. Therefore, during the power estimation phase, we did not consider the glitches inside the macrocells and were concerned only with whether the transitions at the macrocell inputs were full swing or not. If the transitions were full swing, the characterized energy weightings could be used directly for power estimation. If not, those energy weightings required modification for better estimation. In our approach, we monitored the signal transitions at the inputs of the macrocells and filtered out those glitches whose duration were smaller than a delay factor that was empirically determined from a comparison of the SPICE and the Verilog-XL simulations. For a glitch not being filtered out, we calculate its peak voltage V_{peak} (actual swing voltage) according to the rising and falling slopes and the pulsewidth, where the pulsewidth is assumed to be the time interval at $V_{\text{dd}}/2$ between two consecutive transitions. After the V_{peak} is obtained, the partial glitch energy is estimated as follows:

$$E_{\text{partial}} = \frac{V_{\text{peak}}}{V_{\text{dd}}} (E_{\text{er}} + E_{\text{ef}}) \tag{4}$$

TABLE I
LOOK-UP TABLE SIZES NEEDED FOR SOME MACROCELLS (n: NUMBER OF INPUTS, m: NUMBER OF INTERNAL AND OUTPUT NODES, k: NUMBER OF TRANSISTORS)

Macrocell	n	m	k	Look-up table size			Characterization time			
							Structure-oriented			
				PI-oriented	Structure-oriented		Using SPICE		Using PowerMill	
					complete	Incomplete	complete	Incomplete	complete	Incomplete
1-bit half Adder	2	4	16	16	9	9	4.6s	4.6s	0.6s	0.6s
1-bit full Adder	3	4	28	64	16	16	11.5s	11.5s	1s	1s
Mux21	3	5	16	64	16	16	8.8s	8.8s	0.8s	0.8s
Mux41(TG)	6	8	22	4096	256	256	210s	210s	2.4s	2.4s
Mux81	11	10	62	4194304	16384	2304	10.9hr	1.18hr	363s	51s
4-bit Parity Checker	4	7	32	256	81	16	51.3s	10.8s	1.7s	0.8s
XOR	2	2	10	16	9	9	4.8s	4.8s	0.8s	0.8s
1-bit ALU	8	18	72	65536	484	121	967.8s	248s	10s	3s
2-bit ALU	10	33	214	1048576	15376	625	29.6hr	1.16hr	542.5s	25s
4-bit fast adder	9	31	212	262144	26244	1024	29.3hr	1.18hr	1029s	41.2s

where E_{er} and E_{ef} are the energy consumption of the edges traversed by the rising and falling transitions of the glitch, respectively.

Basically, to estimate the power of a macrocell, we are only concerned with which input patterns are applied to the macrocell because the characterized power of those input patterns have included all of the power consumed by the triggered signal transitions inside the macrocell. The output signals of a macrocell are treated as the input signals of the succeeding macrocells. Therefore, the power due to these signal transitions will be taken into account when we estimate the power of the succeeding macrocells. Although two compatible patterns may incur different glitches, those glitches generated inside the macrocell will be counted in this stage and those glitches propagated to the macrocell outputs will be counted in the succeeding stage.

V. EXPERIMENTAL RESULTS

The power characterization and estimation algorithms were implemented in C language on a SUN SPARCstation 20 with 256 Mbytes of memory. To evaluate the quality of our approach, we conducted experiments on several macrocells using complete and incomplete power modeling techniques. We also tested several circuits that were composed of bit-sliced cells. The transistor model used was a level-3 model of 0.8-μm SPDM CMOS technology provided by CIC (Chip Implementation Center, Taiwan, R.O.C.). The signal probabilities and transition densities of the PI's were set to 0.5 for all macrocells and bit-sliced circuits. Based on the input characteristics, a random pattern generator generates 1000 patterns with a 10-ns clock cycle time for running SPICE, PowerMill, and Verilog-XL simulations.

In Table I, the second, third, and fourth columns show the number of PI's, the number of internal and output nodes, and the number of transistors of some macrocells, respectively. The size of the look-up table required for the PI-oriented power modeling technique is shown in the fifth column. The subsequent columns show the table size and the characterization times for the complete power modeling technique and the incomplete power modeling technique, respectively. The complete and incompletely modeled STG's of these macrocells were constructed using the algorithms mentioned before. In the incomplete power model, the user-specified percentage error was 10%. The CPU time for constructing the complete and the incompletely modeled STG's were within 8 s for most macrocells, except the circuits of Mux81, 2-b ALU, and 4-b fast adder in which less than 100 s were used to get the near-optimal STG's.

The characterization time shown is the total CPU time for characterizing all edges of the STG with one fixed loading for each output and one fixed slew rate for each input. The columns labeled with "Using SPICE" and "Using PowerMill" show the CPU time that SPICE and PowerMill were used as characterization engines, respectively. The time unit s represents seconds and hr represents hours. In general, the characterization time using PowerMill is two orders of magnitude less than that using SPICE.

In Table II, the second column and the third column show the average power consumption of the exact SPICE and PowerMill simulations, respectively. The fourth and fifth columns are the estimation results according to the characterization data using SPICE. The sixth and seventh columns are for the characterization data using PowerMill. The percentages shown in parenthesis under the estimated data denote the estimation error as compared to the SPICE estimation. The experimental results show that most of the power estimations based on the two proposed power modeling techniques are very close to those estimated by the exact SPICE simulation and PowerMill simulation. For some macrocells, our estimations are also more accurate than the PowerMill simulations.

Table III shows the test results of some large macrocells using the incomplete power modeling technique with different specified percentage errors. The second and third columns show the number of internal and output nodes and the average power consumption of the SPICE simulation. The subsequent columns show m_ic (which will be discussed later), table size needed, and average power estimated by PowerMill for specified percentage errors 5%, 10%, and 20%, respectively. In the table, we find that as the specified percentage errors are increased, the sizes of the resulting STG's decrease dramatically, but the accuracy losses increase slightly. It is worthwhile

TABLE II
ESTIMATION RESULTS FOR SOME MACROCELLS

Macrocell	Average power consumption (uw)						CPU time (sec.)			
	SPICE	Power-Mill	Structure-oriented				SPICE	Power-Mill	Structure-oriented	
			SPICE charac.		PowerMill charac.					
			complete	Incomplete	complete	Incomplete			complete	Incomplete
1-bit half adder	267.11	266.59 (-0.19%)	253.39 (-5.14%)	253.39 (-5.14%)	262.13 (-1.9%)	262.13 (-1.9%)	312	6	1.12	1.12
1-bit full adder	465.58	467.77 (0.47%)	483.97 (3.95%)	483.97 (3.95%)	484.55 (4.08%)	484.55 (4.08%)	536	8	1.39	1.39
Mux21	264.22	267.53 (1.25%)	260.62 (-1.36%)	260.22 (-1.36%)	274.26 (3.8%)	274.26 (3.8%)	365	6	1.45	1.45
Mux41 (TG)	355.81	395.12 (11.1%)	348.12 (-2.16%)	348.12 (-2.16%)	387.14 (8.81%)	387.14 (8.81%)	607	11	2.99	2.99
Mux81	1033	1075.15 (4.08%)	993.95 (-3.78%)	1009.03 (-2.32%)	1047.17 (1.37%)	1054.64 (2.09%)	2043	23	11.18	12.92
4-bit Parity Checker	514.83	547.62 (6.37%)	512.7 (-0.42%)	538.2 (4.54%)	541.54 (5.19%)	567.05 (10.14%)	839	9	1.73	1.7
XOR	139.36	130.72 (-6.2%)	130.9 (-6.07%)	130.9 (-6.07%)	123.51 (-11.3%)	123.51 (-11.3%)	264	4	1.06	1.12
1-bit ALU	1157.9	1221.65 (5.51%)	1144.1 (-1.19%)	1177 (1.65%)	1186.07 (2.43%)	1215.27 (4.95%)	2410	19	2.85	3.14
2-bit ALU	2711.1	2798.31 (3.22%)	2663.39 (-1.76%)	2726.18 (0.56%)	2755.38 (1.63%)	2843.57 (4.89%)	6275	37	7.89	7.8
4-bit fast adder	3060.4	3138.48 (2.55%)	3009.71 (-1.66%)	3019.43 (-1.34%)	3096.62 (1.18%)	3123.17 (2.05%)	4932	39	12.1	5.2

TABLE III
EXPERIMENTAL RESULTS FOR LARGE MACROCELLS USING INCOMPLETE POWER MODELING TECHNIQUE (m: NUMBER OF INTERNAL AND OUTPUT NODES, m_ic: NUMBER OF INTERNAL NODE AND OUTPUT NODES THAT ARE MODELED EXPLICITLY)

Macrocell	m	SPICE (uw)	User-specified percentage error								
			5%			10%			20%		
			m_ic	Table size	Avg. Power	*m_ic*	Table Size	Avg. Power	*m_ic*	Table size	Avg. Power
MUX81	10	1033	9	9216	1038.88 (0.57%)	7	2304	1054.6 (2.1%)	7	1296	912.2 (-11.69%)
1-bit ALU	18	1157.9	17	256	1186 (2.42%)	16	121	1215.3 (4.96%)	15	49	1213.4 (4.79%)
2-bit ALU	33	2711.1	30	3136	2805.1 (3.47%)	28	625	2843.6 (4.89%)	29	484	2849.5 (5.1%)
4-bit fast adder	31	3060.4	31	26244	3138.5 (2.55%)	5	1024	3123.2 (2.05%)	5	1024	3123.2 (2.05%)

to note that the accuracy losses are far less than the specified percentage error for most evaluations. This is because those nodes that are not modeled explicitly by the STG can be modeled implicitly, and, therefore, their power consumption is not totally lost in the final estimations. In reality, they may be distributed in some edges of the STG. In the table, the columns labeled m_ic represent the number of internal and output nodes that are modeled explicitly by the STG's, which can be an indicator to show how many nodes whose spatial correlation are taken into account in the STG's. In general, the spatial correlation of the selected state variables and their transition dominated nodes are considered in the incompletely modeled STG. Although the correlation of other nodes would be ignored, the impact is minor in these experiments. The following is the main reason. For all circuits, except the 4-b fast adder, those nodes that are not modeled explicitly are all in level 1. However, their transition dominated nodes are totally considered in the incompletely modeled STG. Thus, the impact of ignoring these nodes is minor. The circuit 4-b fast adder is a special case. No matter how we remove the nodes in levels 1 and 2, the table size is identical to that of the completely modeled STG. When the user-specified percentage errors are set to 10% and 20%, the selected state variables are all PO's. However, due to the high correlation of the nodes in levels 1 and 2, the impact of ignoring these nodes is also minor.

In Table IV, we compare our approach with the clustering method proposed by Mehta *et al.* [16]. The third and sixth columns are the number of clusters and the number of edges for an STG obtained using the clustering method and our incomplete power modeling technique, respectively. The fourth and seventh columns are the CPU time needed for running the clustering algorithm and our algorithm, respectively. In terms of the look-up table size and characterization speed, our approach might not perform well for macrocells with large inputs. However, for smaller circuits, the difference is limited. The fifth and eighth columns are the average errors of their results, as compared to IRSIM-CAP, and our results, as compared to SPICE, respectively. The average error of the

TABLE IV
COMPARISON OF OUR APPROACH WITH THE CLUSTERING METHOD [16]

Circuit	Inputs (n)	Clustering method (cluster II)			Incomplete power modeling		
		Clusters	CPU time (secs.)	Avg. error	Edges of STG	CPU time (secs.)	Avg. error
1-bit full adder	3	14	2	10.44%	16	1.3	3.95%
Mux21	3	15	1	13.33%	16	1.4	-1.36%
Mux81	11	47	202	11.36%	324	100	-8.91%

TABLE V
THE EXPERIMENTAL RESULTS OF BIT-SLICED CIRCUITS (n: NUMBER OF INPUTS, m: NUMBER OF INTERNAL AND OUTPUT NODES, k: NUMBER OF TRANSISTORS)

Circuit	n	m	k	Average power (uw)				CPU time (secs.)		
				SPICE	Power-Mill	Structure-oriented		SPICE	Power-Mill	Structure-oriented
						SPICE charac.	PowerMill charac.			
8-bit ripple carry adder	17	32	224	5136	5240.1 (2.03%)	5418.9 (5.51%)	5472.1 (6.54%)	9694	66	11.7
8-bit ALU	22	144	576	10759	11130.87 (3.46%)	10031 (-6.77%)	9982.4 (-7.22%)	37760	137	28.9
8-bit by 8-bit Multiplier	16	352	1856	39270	39204.66 (0.17%)	35471 (-9.67%)	35283 (-10.15%)	176273	519	44.2

TABLE VI
LOCAL POWER OF EACH MODULE OF AN 8-b RIPPLE–CARRY ADDER

cell #	Average power (uw)		Error	
	SPICE	Structure-oriented	(uw)	(%)
1	475.77	478.39	2.62	0.55
2	570.27	637.96	67.69	11.87
3	647.73	712.51	64.78	10
4	666.49	718.63	52.14	7.82
5	683.71	721.27	37.56	5.49
6	688.3	728.74	40.44	5.87
7	710.79	717.52	6.73	0.95
8	692.96	703.97	11.01	1.59
total	5136.02	5418.99	282.97	5.51

clustering method is over 10% for all circuits; however, our method incurs less than 9% error. In general, a switch-level power simulator may incur another 15% error, as compared to the SPICE simulation.

Table V shows the estimation results for three bit-sliced circuits. The fifth and sixth columns show the power estimated using SPICE and PowerMill simulations, respectively. The following two columns show the power estimated using our approach according to the completely modeled STG's characterized by SPICE and PowerMill, respectively. For each circuit, we estimated the power of each bit-sliced cell individually and then obtained the total power by summing up the estimated power. The glitch power was estimated with the method discussed in the previous section. Experimental results demonstrate that our modeling technique produces results within an 8% error margin of SPICE simulation on average while the CPU time consumed is more than three orders of magnitude less. Compared to the PowerMill simulation, the CPU time was more than eight times less.

For a circuit 8-b ripple–carry adder, the power consumed by each macrocell is reported in Table VI. The results show that the average power consumed by each module is within 12% error as compared to the exact SPICE simulation.

VI. CONCLUSION

For a macrocell with numerous inputs, the PI-oriented power modeling technique is impractical. Based on the structural information in a macrocell, we proposed an STG to completely model all of the steady-state transitions of its internal nodes. Moreover, we presented a heuristic method for selecting the proper patterns for characterization, which can effectively resolve the glitch power due to the transitions between compatible patterns. In addition, we exploited the compatible patterns to further reduce the number of edges of the corresponding STG. To further simplify the STG, we also presented an incomplete power modeling technique that can reduce the size of the STG effectively without losing much accuracy. Experimental results show that our modeling techniques can provide SPICE-like accuracy and significantly reduce the size of the look-up table.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments, which improved the final version of this paper.

REFERENCES

[1] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," in *Proc. 32nd ACM/IEEE Design Automation Conf.*, June 1995, pp. 242–247.

[2] F. N. Najm, "Power estimation techniques for integrated circuits," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1995, pp. 492–499.

[3] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski, "The design and implementation of PowerMill," in *Proc. Int. Symp. Low Power Design*, Apr. 1995, pp. 105–110.

[4] A. Salz and M. A. Horowitz, "IRSIM: An incremental MOS switch-level simulator," in *Proc. 26th Design Automation Conf.*, June 1989, pp. 173–178.

[5] B. George, G. Yeap, M. Wloka, S. Tyler, and D. Gossain, "Power analysis for semi-custom design," in *Proc. Custom Integrated Circuits Conf.*, 1994, pp. 249–252.

[6] J.-Y. Lin, T.-C. Liu, and W.-Z. Shen, "A cell-based power estimation in CMOS combinational circuits," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 304–309.

[7] J. H. Satyanarayana and K. K. Parhi, "HEAT: Hierarchical energy analysis tool," in *Proc. 33rd ACM/IEEE Design Automation Conf.*, June 1996, pp. 9–14.

[8] A. Bogliolo, L. Benini, and B. Ricco, "Power estimation of cell-based CMOS circuits," in *Proc. 33rd ACM/IEEE Design Automation Conf.*, June 1996, pp. 433–438.

[9] J.-Y. Lin, W.-Z. Shen, and J.-Y. Jou, "A power modeling and characterization method for the CMOS standard cell library," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1996, pp. 400–404.

[10] P. E. Landman and J. M. Rabaey, "Architecture power analysis: The dual bit type method," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 173–187, June 1995.

[11] P. E. Landman and J. M. Rabaey, "Activity-sensitive architecture power analysis," *IEEE Trans. Computer-Aided Design*, pp. 571–587, vol. 15, June 1996.

[12] A. Raghunathan, S. Dey, and N. K. Jha, "Register-transfer level estimation techniques for switching activity and power consumption," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1996, pp. 158–165.

[13] C.-T. Hsieh, Q. Wu, C.-S. Ding, and M. Pedram, "Statistical sampling and regression analysis for RT-level power evaluation," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1996, pp. 583–588.

[14] S. Gupta and F. N. Najm, "Power macromodeling for high level power estimation," in *Proc. 34th ACM/IEEE Design Automation Conf.*, June 1997, pp. 365–370.

[15] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "Analytical estimation of transition activity from word-level signal statistics," in *Proc. 34th ACM/IEEE Design Automation Conf.*, June 1997, pp. 582–587.

[16] H. Mehta, R. M. Owens, and M. J. Irwin, "Energy characterization based on clustering," in *Proc. 33rd ACM/IEEE Design Automation Conf.*, June 1996, pp. 702–707.

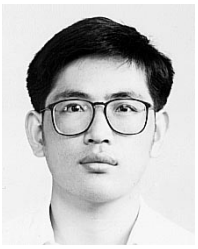
[17] H. Mehta, R. M. Owens, and M. J. Irwin, "Instruction-level power profiling," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, May 1996.

[18] F. N. Najm, "Low-pass filter for computing the transition density in digital circuits," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 1123–1131, Sept. 1994.

[19] D. Rabe and W. Nebel, "New approach in gate-level glitch modeling," in *Proc. European Design Automation Conf. with Euro-VHDL*, 1996, pp. 66–71.

[20] W.-C. Tsai, C. B. Shung, and D. C. Wang, "Accurate logic-level power simulation using glitch filtering and estimation," in *Proc. IEEE Asia–Pacific Conf. Circuits Syst.*, Nov. 1996, pp. 314–317.

[21] R. Marculescu, D. Marculescu, and M. Pedram, "Hierarchical sequence compaction for power estimation," in *Proc. 34th ACM/IEEE Design Automation Conf.*, June 1997, pp. 570–575.



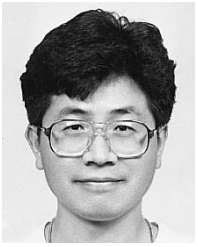
Jiing-Yuan Lin (S'91) received the B.S. degree in electrical engineering from National Taiwan University of Science and Technology, Taiwan, R.O.C., in 1991, and M.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University, Taiwan, R.O.C., in 1993 and 1998, respectively.

In 1998, he joined Global UniChip Corporation, Hsinchu, Taiwan, R.O.C., where he works on the development of a low-power cell library. His research interests are in the areas of the low-power design methodologies and power estimation with emphases on gate- and RT-levels.

Wen-Zen Shen (S'78–M'90) received the Ph.D. degree in electronics engineering from National Chiao Tung University, Taiwan, R.O.C., in 1982.

He is currently a Professor and Dean of the College of Electrical Engineering and Computer Science, National Chiao Tung University. His current research interests include VLSI design, low-power circuit design, computer-aided design (CAD), and VLSI for signal processing.

Dr. Shen is a member of Phi Tau Phi.



Jing-Yang Jou (S'82–M'83) received the B.S. degree in electrical engineering from National Taiwan University, Taiwan, R.O.C., and M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign.

He is currently a Professor in the Department of Electronics Engineering, National Chiao Tung University, Taiwan, R.O.C. His professional experience also includes having been with GTE Laboratories and Bell Laboratories. He has published 60 journal and conference papers. His research interests include behavioral and logic synthesis, VLSI designs and CAD for low-power design verification, synthesis and design for testability, and hardware/software co-design.

Dr. Jou is a member of Tau Beta Pi. He was the recipient of the 1990 Distinguished Paper Award of the IEEE International Conference on Computer-Aided Design. He has served as the technical programmer chair of the IEEE Asia–Pacific Conference on Hardware Description Languages (APCHDL'97).