A Power Modeling and Characterization Method for Macrocells Using Structure Information

Jiing-Yuan Lin, Wen-Zen Shen, Jing-Yang Jou

Dept. of Electronics Engineering and Institute of Electronics
National Chiao Tung University, Hsinchu 30050, Taiwan R.O.C.

Abstract

To characterize a macrocell, a general method is to store the power consumption of all possible transition events at primary inputs in the lookup tables. Though this approach is very accurate, the lookup tables could be huge for the macrocells with many inputs. In this paper, we present a new power modeling method which takes advantage of the structure information of macrocells and selects minimum number of primary inputs or internal nodes in a macrocell as state variables to build a state transition graph (STG). Those state variables can completely model the transitions of all internal nodes and the primary outputs. By carefully deleting some state variables, we further introduce an incomplete power modeling technique which can simplify the STG without losing much accuracy. In addition, we exploit the property of the compatible patterns of 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 and can reduce the size of the lookup table significantly compared to the general approach.

1. Introduction

Recently, several behavior level power estimators [1-4] considered the modeling and characterization of datapaths based on input statistics. However, as stated in [5], input statistics is highly dependent on the application the circuit is running on and the context of the usage of the circuit. Therefore, the energy models of macrocells which are derived from specific input statistics may not match that of the operating environment.

To overcome the drawbacks of the input statistics modeling, a simple modeling method is proposed in [6] to characterize the power consumption caused by all possible transition events at primary inputs. We refer to this technique as the primary input oriented (PI-oriented) power modeling technique. During the logic simulation, the power estimator monitors the power events occurring at the primary inputs of each macrocell and computes the total power consumption as the sum of the power consumption of each event. This technique can provide accurate estimation results; however, for a macrocell with $n$ inputs, it needs a lookup table with size of $2^{2n}$ to record the characterization data. Moreover, the complexity is too high for this exhaustive characterization. To reduce the size of lookup table, Mehta et. al. [5] presented a clustering algorithm to characterize the energy dissipation of macrocells. They first simulate the characterized circuit with $m$ random vectors ($m \ll 2^{2n}$) by a switch-level simulator. Then, according to the simulated switching capacitances, they group closely related energy patterns to obtain clusters which will be used to characterize the circuit. Because $m \ll 2^n$, e.g., $m=500$ for $n=11$ in their experiments, the number of clusters obtained would be far less than $2^{2n}$. So, this approach can achieve a significant reduction on the size of lookup table. However, the accuracy would be sacrificed because the energy value of each input transition vector which is not simulated is undefined and is used as don’t-care during clustering.

For a macrocell, there could be two or more input patterns which generate the same states of all internal nodes in the macrocell. In this paper, we refer to these patterns as compatible patterns. A compatible pattern set is the collection of all compatible patterns. Input sequence formed by compatible patterns will not cause any signal transitions at any internal nodes. As an example, Fig. 1(a) shows a schematic diagram of a 2-to-1 multiplexer and Fig. 1(b) shows its extended truth table in terms of the internal nodes and the output node. In the truth table, we find that input patterns 000 and 010 have the same values of internal nodes and the output node. Thus, the truth table is compatible patterns. Similarly, patterns 001 and 101, 011 and 111, 100 and 110 are all compatible patterns. Thus, 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 characterization in the PI-oriented power modeling technique. This would waste memory space for storing data and CPU time of power characterization process.

In this paper, we propose a structure-oriented power modeling technique for macrocells. In our approach, in addition to PIs, all internal nodes in a cell are also 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 POs. Based on the model, we will consider the compatible patterns for reducing the characterization vectors while retaining the accuracy as that provided by the PI-oriented technique. For some macrocells, the constructed STG may still be very complicated. To further reduce the complexity of STG, we introduce an incomplete power modeling technique which can effectively simplify the STG without losing much accuracy.
2. A Complete Power Modeling Technique

For a macrocell, if we can model all the possible transition events at the internal nodes, we can get an accurate estimation of its power consumption. In this paper, if a power model can model all the signal transitions of internal nodes in a macrocell, then it is called a complete power modeling technique. As mentioned before, PI-oriented power modeling is a complete power modeling technique. For a circuit with \( n \) inputs, it performs power characterization with \( 2^n \times 2^n \) sequences. We can model these sequences by using a state transition graph (STG) where the graph has \( 2^n \) states to represent the \( 2^n \) possible input combinations at time \( n-1 \) and each state has \( 2^n \) outgoing edges to represent the \( 2^n \) possible input combinations at time \( n \). We refer to such STG which is built for the complete power modeling as a complete modeling STG. Though PI-oriented power modeling technique can provide accurate results, the sizes \( 2^n \) would be easily exploded.

In this section, we try to find a complete power modeling technique using STG with smaller number of edges as compared to that of the PI-oriented power modeling technique. Fortunately, the property of compatible patterns in a circuit can be used to achieve this goal. For a circuit, after running exhaustive logic simulation, we can build an extended truth table in terms of all the property of compatible patterns in a circuit can be used to achieve this goal. For a circuit, after running exhaustive logic simulation, we can build an extended truth table in terms of all the signal transitions of internal nodes in a macrocell, then it is called a complete power modeling technique. As mentioned before, PI-oriented power modeling is a complete power modeling technique. For a circuit with \( n \) inputs, it performs power characterization with \( 2^n \times 2^n \) sequences. We can model these sequences by using a state transition graph (STG) where the graph has \( 2^n \) states to represent the \( 2^n \) possible input combinations at time \( n-1 \) and each state has \( 2^n \) outgoing edges to represent the \( 2^n \) possible input combinations at time \( n \). We refer to such STG which is built for the complete power modeling as a complete modeling STG. Though PI-oriented power modeling technique can provide accurate results, the sizes \( 2^n \) would be easily exploded.

In this section, we try to find a complete power modeling technique using STG with smaller number of edges as compared to that of the PI-oriented power modeling technique. Fortunately, the property of compatible patterns in a circuit can be used to achieve this goal. For a circuit, after running exhaustive logic simulation, we can build an extended truth table in terms of all the signal transitions of internal nodes in a macrocell, then it is called a complete power modeling technique. As mentioned before, PI-oriented power modeling is a complete power modeling technique. For a circuit with \( n \) inputs, it performs power characterization with \( 2^n \times 2^n \) sequences. We can model these sequences by using a state transition graph (STG) where the graph has \( 2^n \) states to represent the \( 2^n \) possible input combinations at time \( n-1 \) and each state has \( 2^n \) outgoing edges to represent the \( 2^n \) possible input combinations at time \( n \). We refer to such STG which is built for the complete power modeling as a complete modeling STG. Though PI-oriented power modeling technique can provide accurate results, the sizes \( 2^n \) would be easily exploded.

In this section, we try to find a complete power modeling technique using STG with smaller number of edges as compared to that of the PI-oriented power modeling technique. Fortunately, the property of compatible patterns in a circuit can be used to achieve this goal. For a circuit, after running exhaustive logic simulation, we can build an extended truth table in terms of all the signal transitions of internal nodes in a macrocell, then it is called a complete power modeling technique. As mentioned before, PI-oriented power modeling is a complete power modeling technique. For a circuit with \( n \) inputs, it performs power characterization with \( 2^n \times 2^n \) sequences. We can model these sequences by using a state transition graph (STG) where the graph has \( 2^n \) states to represent the \( 2^n \) possible input combinations at time \( n-1 \) and each state has \( 2^n \) outgoing edges to represent the \( 2^n \) possible input combinations at time \( n \). We refer to such STG which is built for the complete power modeling as a complete modeling STG. Though PI-oriented power modeling technique can provide accurate results, the sizes \( 2^n \) would be easily exploded.

we neglect this power, then the outgoing edges corresponding to the compatible patterns can be merged into a single edge for each state. Fig. 3 shows the STG after merging the compatible patterns. In this simplified STG, the total number of edges is \( N \times N \) where \( N \) is the number of states. The number of edges is independent of the number of primary inputs. This STG is referred to as the complete modeling STG with merged edges.

3. An Incomplete Power Modeling Technique

In Fig. 2, we use all internal nodes as state variables for modeling the signal transitions in the circuit. In fact, if we utilize the logic relationship among internal nodes, it is possible to use only part of those nodes as state variables to model the entire circuit. Finding a minimum set of nodes which can completely model all the transitions serves two purposes. First, we only need to check those nodes in the truth table when we evaluate the circuit for finding the compatible patterns. Second, we can possibly remove part of those nodes from the complete modeling STG to form a simpler STG while losing little accuracy. In this paper, if a power modeling technique can not completely model all the nodes in a macrocell, we call it an incomplete power modeling technique. The STG constructed for an incomplete power modeling technique is referred to as an incomplete modeling STG.

In this section, we first propose an algorithm for selecting a minimum set of internal nodes and primary inputs as state variables for the complete modeling STG. Then, we present an algorithm to remove some state variables of the complete modeling STG to form an incomplete modeling STG.

Let’s define some notations first. In Fig. 1, node \( Z \) incurs signal transitions whenever node \( c \) makes transition, and vice-versa. Node \( Z \) and node \( c \) are referred to as transition compatible nodes in this paper. For transition compatible nodes, we only need to select one of them as state variable when we construct the STG. Because the output value of a logic gate is determined by its input values, if the state transition graph already uses all the inputs of the gate as state variables, the output of the gate does not need to be selected as a state variable. In Fig. 1, the signal transitions of node \( c \) can be modeled by the transitions of nodes \( a \) and \( b \). Nodes \( a \) and \( b \) are referred to as the transition dominating nodes of node \( c \). On the contrary, node \( c \) is the transition dominated node of nodes \( a \) and \( b \). Therefore, our goal is to find a minimum set of transition dominating nodes which can completely model all the
transitions in the macrocell and to use those nodes as state variables for constructing STG.

For a macrocell, if the inside of it has no feedback loop, we call this cell combinational cell. A combinational cell has an important property that the logic value of a node is only dependent on the logic states of its transitive fan-in nodes. Given a circuit, we can levelize the netlist starting from primary inputs with level 0 and label the level of each node \( i \) with \( l_i \). If a logic gate has \( m \) inputs with levels \( l_1, l_2, ..., l_m \), we define the level of its output as

\[
\min(l_1, l_2, ..., l_m) + 1.
\]

Based on this definition, the nodes with at least one primary input as its inputs would be in level 1. Any node with level higher than 1 is driven directly or indirectly by the nodes in level 1. Thus, if we choose all the nodes in level 1 as state variables and collect them in a state variable set (SVS), the STG can completely model the transition behavior of all nodes in the circuit. In addition to the nodes in level 1, some additional primary inputs can also be selected as state variables to form a simpler complete modeling STG where some state variables in the original SVS can possibly be replaced.

As an example shown in Fig. 4, the nodes \( n1, n2, n3, \) and \( n4 \) constitute the state variable set for a complete power modeling technique first. Thus, \( \text{SVS} = \{n1, n2, n3, n4\} \). After removing the invalid states such as 1010 (\( n1=1, n2=0, n3=1, n4=0 \)), the corresponding STG would have 9 states and 81 edges. However, if we choose nodes \( n1, n2, \) and input A as state variables, \( \text{SVS} = \{n1, n2, A\} \), we can reduce the number of states to 8 (\( 2^3 \)) and the number of edges to 64. Thus, the basic idea of our algorithm is to find a minimum set of the nodes with level 0 or level 1 such that the corresponding complete modeling STG has the fewest states. Algorithm 1 shows the pseudo-code of our algorithm.

**Algorithm 1**

**Input:** the netlist of a macrocell

**Output:** state variable set

State Variable Selection for Complete Modeling()

\[
\begin{align*}
&\text{levelize the netlist;} \\
&\text{select the nodes in level 1 as state variables and put them into } \text{SVS}_\text{orig}; \quad \text{SVS}_\text{final} \leftarrow \text{SVS}_\text{orig}; \\
&\text{for each state variable combination } C_i \text{ in } \text{SVS}_\text{orig} \\
&\quad \text{backtrace the circuit from the nodes in } C_i \text{ to PI and collect their transition dominating nodes in a set } TGN; \\
&\quad \text{forward traverse the circuit from the primary inputs in } TGN \text{ to the nodes in level 1 and find the state variables } SV \text{ which can be modeled by those inputs}.
\end{align*}
\]

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 above algorithm on the 2-to-1 multiplexer, \( 5, a, \) and \( b \) are selected as the state variables. Fig. 5(a) and (b) show the truth table for the state variables and the corresponding state transition graph, respectively.

![Fig. 4 An example for selecting state variables.](image)

![Fig. 5 (a) The truth table of the state variables, (b) the complete modeling STG of a 2-to-1 multiplexer.](image)
In addition to the above consideration, we should also consider the amount of power consumed by node \( n_i \) before deleting it from SVS. If node \( n_i \) contributes most power of the whole macrocell, deleting it from SVS would result in significant loss in accuracy. In general, the node that has larger capacitive loading than others would potentially dissipate more power. Therefore, in our approach, if the ratio of the capacitive loading of node \( n_i \) over the total capacitive loading of the whole macrocell is larger than a specified threshold value, we will retain the state variable in SVS. Otherwise, the state variable would be deleted. In this paper, the ratio is referred to as loading ratio.

![Fig. 6 The incomplete modeling STG of a 2-to-1 multiplexer.](image)

As the example of 2-to-1 multiplexer, both node \( SD \) and its fanout node \( b \) are in the SVS of the complete modeling STG. If we delete \( SD \) from the SVS, the signal transitions at nodes \( a, b, c \), and \( Z \) all can be modeled by the state variables \( a \) and \( b \). Fig. 6 shows the new STG after removing state variable \( SD \).

The algorithm of selecting the state variables of incomplete modeling STG is shown as follows.

Algorithm 2
Input: the netlist of a macrocell, the state variable set (SVS) of complete power modeling, the threshold value (THV)
Output: the SVS of incomplete power modeling
State_Variable_Selection_for_Incomplete_Modeling()

\[
\text{Algorithm 2: State_Variable_Selection_for_Incomplete_Modeling()}
\]

\[
\text{Input: netlist of a macrocell, state variable set (SVS) of complete power modeling, threshold value (THV)}
\]

\[
\text{Output: SVS of incomplete power modeling}
\]

\[
\text{State_Variable_Selection_for_Incomplete_Modeling()}
\]

\[
\text{for each state variable } SV \text{ in SVS }
\]

\[
\text{collect the fanout nodes of } SV \text{ in FON;}
\]

\[
\text{if (SVS ∩ FON = FON and } LR(SV) < THV) \}
\]

\[
\text{SVS = SVS } \setminus SV; \}
\]

\[
/* LR(SV) is the loading ratio of SV */
\]

\[
\text{return(SVS)};
\]

\[
\text{After applying the above algorithm on a complete modeling STG with N states, the resulting incomplete modeling STG may have M states and } M \times M \text{ edges where } M \leq N.
\]

4. Power Characterization and Calculation of Edge Activity Number
4.1. Power Characterization

Once the STG is built, we need to characterize the energy consumption \( W_k \) associated with each edge. In our approach, we use SPICE simulation to do this work. We have built an automatic procedure for power characterization which accounts for the effect of input slope and output loading. To characterize the \( W_k \) of each edge in STG, we need a set of input sequences which can traverse each edge. The sequences can be derived from the behavior of STG. As the example shown in Fig. 5(b), to characterize the state transition from state 000 to state 001, we need to set the state to 000 at time \( n-1 \), apply input 100 or 110 at time \( n \), and calculate the difference of the energies measured at time \( n \) and \( n-1 \). Because we use SPICE simulation for power characterization, the power consumption of each characterized edge includes not only the dynamic power due to state transitions but also the short-circuit power and the leakage power.

4.2. Calculation of Edge Activity Number

To estimate the energy consumption of a macrocell, we also need to calculate the edge activities \( E_k \) of each edge in the corresponding STG. Here we use a state enumeration method which is simple and accurate. The transition waveforms of the inputs of macrocells are monitored when we perform logic simulation. We then enumerate the corresponding STG using those transition waveforms. As the logic simulation is over, we can obtain the activity numbers of each edge of the STG. This approach is very simple but it can accurately capture the signal correlation of inputs during the logic simulation.

5. Experimental Results

The power characterization and estimation algorithms have been implemented in C language on a SUN SPARCstation 20 with 256 Mbytes of memory. To evaluate the quality of our approach, several macrocells and several circuits constructed by bit-slice cells are tested. The transistor models used are the level 3 model of 0.8um SPDM CMOS technology provided by CIC (Chip Implementation Center in Taiwan). The signal probabilities and transition densities of the primary inputs are assigned to be 0.5 for all macrocells and the bit-slice circuits. Based on the input characteristics, a random pattern generator generates 1000 patterns with 10ns clock cycle time for running both SPICE and Verilog simulations.

In Table 1, the size of lookup table needed for the PI-oriented power modeling technique is shown in the fifth column. The subsequent columns show the table sizes and the characterization time for the complete and incomplete power modeling techniques respectively. In these macrocells, the threshold value of the loading ratio is specified as 0.15 for constructing the incomplete modeling STGs. The characterization time shown is the total CPU run-time for characterizing all edges of the STG with one fixed output loading and one fixed input slope. The time units labeled \( s \) represents seconds and \( hr \) represents hours. In general, a macrocell would be characterized several times with different output loadings and input slopes. Therefore, the whole characterization time would be several times of that shown in the table. The characterization is done only once even though the characterization time is long for some macrocells. In this table we find that the lookup table size of 4-bit Fast Adder can not be further reduced after applying the incomplete power modeling technique. This is because the nodes in level 1 are all symmetrical and thus no any state variables in the complete modeling STG can
be removed. The CPU time of constructing the complete and incomplete modeling STGs are within 8 seconds for most macrocells except the circuits of 2-bit ALU and 4-bit Fast Adder which, however, are also within 100 seconds.

In Table 2, the second to seventh columns show the average power consumption and CPU time of the exact SPICE simulation and the estimations using the complete and incomplete power modeling techniques respectively. The percentage shown in the parenthesis under the estimated data denotes 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 except the incomplete modeling of the circuit Mux81. This is because the internal nodes of Mux81 are all in level 1 and most of them are deleted from the SVS based on the specified threshold value of loading ratio. If we set a lower threshold value, the estimation error would be reduced at the expense of increasing the size of lookup table.

Table 3 shows the estimation results of an 8-bit ripple carry adder, an 8-bit arithmetic logic unit, and an 8-bit by 8-bit array multiplier. These circuits are constituted with the bit-slice cell. For each circuit, we estimate the power of each bit-slice cell individually by using the complete modeling STG with merged edges and then obtain the total power by summing up the estimated powers of them. The experimental results show that our modeling technique provides within 7% error of SPICE simulation on average while the CPU consumes more than 3 orders of magnitude less CPU time.

In the incomplete power modeling technique, all the state variables in the final SVS are in level 0 or level 1. If we are allowed to loose more accuracy to obtain a simpler STG, we can consider to include the nodes whose levels are higher than 1 in the final SVS. This possibly can further reduce the lookup table size of the circuit 4-bit Fast Adder. We will investigate this possibility in the future. Moreover, the characterization complexity is too high to characterize some macrocells using the SPICE simulation. We can use switch-level power estimator or PowerMill instead of SPICE to characterize the macrocells when the requirements on the accuracy are not strict as in high-level or RT-level power estimation. This could reduce the characterization time dramatically.

### Table 1: Lookup table sizes needed for some macrocells.

<table>
<thead>
<tr>
<th>Macrocircuit</th>
<th>n</th>
<th>m</th>
<th>k</th>
<th>PI-oriented</th>
<th>Characterization time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-bit Half Adder</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td>1-bit Full Adder</td>
<td>3</td>
<td>4</td>
<td>28</td>
<td>28</td>
<td>16</td>
</tr>
<tr>
<td>Mux21</td>
<td>3</td>
<td>5</td>
<td>16</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>Mux81</td>
<td>6</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>210s</td>
</tr>
<tr>
<td>Mux81</td>
<td>11</td>
<td>10</td>
<td>62</td>
<td>4096</td>
<td>576</td>
</tr>
</tbody>
</table>

### Table 2: Estimation results for some macrocells.

<table>
<thead>
<tr>
<th>Macrocircuit</th>
<th>Average Power (uw)</th>
<th>CPU Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-bit Half Adder</td>
<td>SPICE (N)</td>
<td>Mux100</td>
</tr>
<tr>
<td>1-bit Full Adder</td>
<td>SPICE (N)</td>
<td>Mux100</td>
</tr>
<tr>
<td>Mux21</td>
<td>SPICE (N)</td>
<td>Mux100</td>
</tr>
<tr>
<td>Mux81</td>
<td>SPICE (N)</td>
<td>Mux100</td>
</tr>
</tbody>
</table>

### 6. Conclusion and Future Works

For a macrocell with many inputs, PI-oriented power modeling technique is no longer applicable. In this paper, we present a structure-oriented power modeling technique to solve this problem. Based on the structure information of macrocells, we propose a STG to completely model all the signal transitions of the internal nodes of the macrocells. To further simplify the STG, we also present an incomplete power modeling technique. In addition, we exploit the compatible patterns of a macrocell to further reduce the number of edges of the corresponding STG. Experimental results show that our modeling techniques can provide SPICE-like accuracy and reduce the size of the lookup table significantly.

In the incomplete power modeling technique, all the state variables in the final SVS are in level 0 or level 1. If we are allowed to loose more accuracy to obtain a simpler STG, we can consider to include the nodes whose levels are higher than 1 in the final SVS. This possibly can further reduce the lookup table size of the circuit 4-bit Fast Adder. We will investigate this possibility in the future. Moreover, the characterization complexity is too high to characterize some macrocells using the SPICE simulation. We can use switch-level power estimator or PowerMill instead of SPICE to characterize the macrocells when the requirements on the accuracy are not strict as in high-level or RT-level power estimation. This could reduce the characterization time dramatically.

### Reference


