Accurate dynamic power estimation for CMOS combinational logic circuits with real gate delay model

Omnia S. Fadl a,*, Mohamed F. Abu-Elyazeed a, Mohamed B. Abdelhalim b, Hassanein H. Amer c, Ahmed H. Madian d

ARTICLE INFO
Article history:
Received 5 January 2015
Received in revised form 23 February 2015
Accepted 23 February 2015
Available online 2 March 2015

Keywords:
CMOS combinational logic circuits
Logic pictures
Real-delay model
Switching activity
Dynamic power estimation
Toggle rate

ABSTRACT
Dynamic power estimation is essential in designing VLSI circuits where many parameters are involved but the only circuit parameter that is related to the circuit operation is the nodes’ toggle rate. This paper discusses a deterministic and fast method to estimate the dynamic power consumption for CMOS combinational logic circuits using gate-level descriptions based on the Logic Pictures concept to obtain the circuit nodes’ toggle rate. The delay model for the logic gates is the real-delay model. To validate the results, the method is applied to several circuits and compared against exhaustive, as well as Monte Carlo, simulations. The proposed technique was shown to save up to 96% processing time compared to exhaustive simulation.

Introduction
Power consumption is an essential factor in digital VLSI circuits. High power consumption affects circuit reliability and can cause runtime errors or shorten circuit lifetime [1]. Many power estimation methods have been developed; they can mainly be divided into two main categories. First, there are...
simulation-based methods which rely on simulating the circuit with an appropriate set of inputs to obtain the power consumption [2–7]. Second, there are non-simulation based methods which depend on probabilistic measures for the inputs and the switching activities to find the power consumption [8–14].

The simulation-based methods are input pattern-dependent and suffer from the enormous time and memory needed for the simulation of large circuits. Spice, as an example, is a circuit-level simulator that is used to accurately estimate the power; its main disadvantage is the large time consumed which makes it inapplicable for large circuits [2]. Another level of simulation is the switch-level simulator, IRSIM, which is a switch-level simulator for MOS transistors [3]. The switch-level simulator is not as accurate as the circuit-level simulator but it is faster and requires a smaller memory space. In gate-level simulation, the main unit is the logic gates and the power consumption at the circuit nodes can be estimated by calculating toggle rates and capacitances at these nodes. The gate-level simulation is up to four orders of magnitude faster than circuit-level simulation [4]. The average power consumption of the circuit is the sum of the average power values at the circuit nodes [5]. A Monte Carlo approach can be used to statistically estimate the power by applying a randomly-generated input pattern to the circuit and re-simulating it until the estimate of the consumed power converged [6]. A Curve fitting approach can also be used to estimate the power [7].

The non simulation-based methods are fast, but the calculated power is not accurate and only an estimate. They can be divided into behavioral-based approaches and gate-level approaches. For the behavioral-based approaches, not much information is available about the gate level and the technique depends on the description of the Boolean function [8,9]. For the gate-level based approaches [10–14], the signal probability and the transition probability are the key functions in estimating the power. The average fraction of clock cycles during which the steady state of a certain node is high is defined as the signal probability of this node. The transition probability is the average fraction of clock cycles where the node value is different from its initial value.

In this paper, a deterministic method to calculate the average power consumption for digital CMOS circuits is discussed at the gate level. The method uses a real-delay model, where different gate types have different delays [1]; this may lead to inevitable glitches and in turn increases the power consumption [10].

The paper is organized as follows. The Methodology Section discusses the Logic Pictures concept and the proposed power estimation method. The Results Section shows the experimental results compared to exhaustive and Monte Carlo simulations and the Conclusion Section summarizes this work.

Methodology

A Logic Picture (LP) consists of the steady state values of all gate outputs (nodes) for a certain input combination. The maximum possible number of LPs for an n-input circuit is $2^n$ LPs, but experimentally their actual number is much smaller than $2^n$ [15]. Finding the dynamic power using exhaustive testing requires $2^N$ simulation cycles to cover all the circuit transitions, but obtaining LPs for n-input circuits require only $2^n$ simulation cycles. This means that acquiring the dynamic power using LPs has lower complexity than that of exhaustive testing. Moreover it needs less memory space. Since the LP count is small, LPs were used in calculating the switching activity; the nodes’ toggle rate and hence average dynamic power were obtained. As the regularity of the circuit increases, the ratio of the number of logic pictures compared to the size the truth table entries ($2^n$) decreases.

Though the methods were deterministic, the delay model for the gates was assumed to be the zero-delay model where no gate delays were considered at all [15].

To enhance the accuracy of the calculation of the nodes’ transition rates, another deterministic method that used LPs was introduced. The method assumed that the gates propagation delays followed the unit-delay model where all gates have the same propagation delay [16]. However, the real-delay model is closer to reality and is expected to result in higher accuracy at the expense of higher memory requirements due to the extra LPs generated by the glitches (as will be shown later in the paper).

In CMOS digital circuits, the main sources of power consumption are: short-circuit power, leakage power and dynamic power [17]. Both the leakage power and the short circuit power depend on the device fabrication and can be minimized by careful circuit design. The main source of dynamic power dissipation is the charging and the discharging of the load capacitances [18], hence it depends on the circuit operation as well as the input patterns and this is the main focus of this paper.

The average dynamic power can be expressed as a function of the circuit parameters as follows [10]:

$$P_{\text{dynamic}} = \frac{V_{dd}^2 \sum_{i=0}^{N} x_i c_i}{2T}$$

where $V_{dd}$ is the source supply voltage, $N$ is the total number of nodes, $T$ is the clock cycle period, $C_i$ is the load capacitance of node $i$ and $x_i$ is the toggle rate of node $i$, i.e., the rate of the transitions between zeros and ones occurring at node $i$. $V_{dd}$ and $C_i$ depend on the device fabrication and $T$ depends on the circuit application specifications while $x_i$ is the only parameter in the equation that depends on the circuit operation; therefore, its value is a sufficient indicator of the average dynamic power of the circuit [1].

To explain the proposed method, consider for example the circuit in Fig. 1. The inputs are assumed to have equal probabilities of being 0 or 1. As shown in Table 1, the circuit has 3 inputs $a$, $b$ and $c$ as well as 3 gates’ outputs $d$, $e$ and $f$. The number of truth table entries is 8. The circuit has three different LPs (000, 011 and 110) that represent the status of circuit nodes for all input vectors. Logic Groups (LGs) can be formed to combine the inputs that lead to the same LP resulting into three LGs. The first group, LG$_1$, contains the inputs that lead
As shown in Table 2, starting from a certain initial state, the value of a circuit node changes at different time instants depending on the gate delays due to the applied input. The initial states are the available different LPs of the circuit. For the above mentioned delays, the transient intermediate states occur at time instants $d_1$, $d_2$, $d_1 + d_2$, $d_1 + d_3$ and $d_2 + d_3$ while the final state is at $d_1 + d_2 + d_3$. Those time instants cover all the changes that could appear because of the different delays.

In the transient states, the node values result into new intermediate LPs due to the glitches. The notations of the LPs in Table 2 define different LPs at different states. For example, the second column in Table 2 contains the initial 3 LPs acquired from the truth table at the initial state (time = 0). The final column in Table 2 is the merged LP of all intermediate LPs through time and it represents the node status through time except the initial state. The number of transitions can be calculated as in Table 3. For example, the transition between initial LP1,0 and merged LP1 is obtained as follows: the number of inputs that lead to LP1,0 is 3 ($||LG_1|| = 3$) and from Table 2 LP1 appeared after LP1,0 for 3 different inputs; then, the different input combinations that can lead from LP1,0 to LP1 is $3 \times 3 = 9$. LP2 and LP3 appear once after LP1,0; hence, the number of different combinations that lead to

---

**Table 1** Truth table and LPs.

| Inputs | Nodes | Logic pictures |
|--------|-------|----------------|
| abc    | def   | LPs            |
| 000    | 000   | LP1 = '000'    |
| 001    | 011   | LP2 = '011'    |
| 010    | 000   | LP1, LP2      |
| 011    | 011   | LP2           |
| 100    | 000   | LP1           |
| 101    | 011   | LP3           |
| 110    | 110   | LP3 = '110'   |
| 111    | 110   | LP3           |

---

**Table 2** Circuit LPs at different time instants.

| Inputs | T = 0 | T = $d_1$ | T = $d_2$ | T = $d_1 + d_2$ | T = $d_1 + d_2 + d_3$ | Merged LP |
|--------|-------|-----------|-----------|-----------------|---------------------|-----------|
| abc    | def   | LP1,0     | LP1,1     | LP1,1 + d_3    | LP1,0 + d_3 + d_3   |           |
| 001    | '000' | '000'     | '010'     | '010'          | '011'               | LP1       |
| 011    | '000' | '010'     | '010'     | '011'          | '011'               | LP1       |
| 101    | '000' | '010'     | '101'     | '101'          | '111'               | LP1       |
| 110    | '100' | '100'     | '100'     | '111'          | '111'               | LP1       |
| 111    | '100' | '110'     | '111'     | '111'          | '111'               | LP1       |
| 000    | '001' | '001'     | '001'     | '001'          | '001'               | LP2       |
| 010    | '001' | '011'     | '011'     | '011'          | '011'               | LP2       |
| 100    | '011' | '001'     | '101'     | '100'          | '110'               | LP2       |
| 110    | '111' | '111'     | '110'     | '110'          | '110'               | LP2       |
| 111    | '111' | '111'     | '110'     | '110'          | '110'               | LP2       |
| 000    | '110' | '111'     | '111'     | '111'          | '111'               | LP3       |
| 010    | '110' | '111'     | '110'     | '110'          | '110'               | LP3       |
| 100    | '110' | '111'     | '110'     | '110'          | '110'               | LP3       |
| 001    | '010' | '011'     | '011'     | '011'          | '011'               | LP4       |
| 011    | '010' | '011'     | '011'     | '011'          | '011'               | LP4       |
| 101    | '010' | '011'     | '011'     | '011'          | '011'               | LP4       |
| 110    | '010' | '011'     | '011'     | '011'          | '011'               | LP4       |
| 111    | '010' | '011'     | '011'     | '011'          | '011'               | LP4       |
| 000    | '010' | '011'     | '011'     | '011'          | '011'               | LP5       |
| 010    | '010' | '011'     | '011'     | '011'          | '011'               | LP5       |
| 100    | '010' | '011'     | '011'     | '011'          | '011'               | LP5       |
| 001    | '010' | '011'     | '011'     | '011'          | '011'               | LP5       |
| 011    | '010' | '011'     | '011'     | '011'          | '011'               | LP5       |
| 101    | '010' | '011'     | '011'     | '011'          | '011'               | LP5       |

---
LP2 and LP3 is 3 * 1 = 3 combinations. For LP4, LP5, LP6, LP7 and LP8, they never appeared in the table after LP1,0 which means that there is no transition between them.

To obtain the node transition, a toggle occurs if the node changes its value from 0 to 1 or 1 to 0. The toggle is multiplied by the number of all the possible inputs that lead to this toggle. For example, LP1,0 is '000'; due to the applied input, the nodes value changes to LP2,81 which is '100'. This means that node 'd' toggles its value from 0 to 1. All the possible transitions that may occur from LP1,0 to LP2,81 can be obtained from Table 2; since LP2,81 is a part of the merged LP2, then the number of transitions is 6. Toggles can also be found in this node for the transition between LP2,0 to LP4, 81 and LP3,0 to LP3,81. This results into 18 additional transitions. The total number of transitions that leads to a toggle in node 'd' is 18 + 6 = 24 transitions. The same analysis can be carried out at node 'e' resulting into 36 transitions and node 'f' resulting into 60 transitions. The toggle rate is then calculated by dividing the total number of toggles for a certain node by the maximum number of toggles which is 2^2n.

To conclude, the following equation is used to get the toggle rate \( z_i \) of node \( i \):

\[
z_i = \frac{\sum_{t=k}^{\infty} \sum_{t=1}^{\infty} R_{i,m} tr_j(P_t, P_m)}{2^{2n}} \tag{2}
\]

where \( s \) is the number of time instants; it is equal to number of gates in the critical path which represents the maximum number of time instants to be accounted for, \( k(T) \) is the number of LPs within time instant \( T \), \( k(T_{j+1}) \) is the number of LPs in the next time instant \( T_{j+1} \), \( R_{j,m} \) is the repetition of the LPs \( P_t \) and \( P_m \) within time instant \( j \), and \( tr_j(p_t, p_m) = 1 \) if there is a node transition between \( P_t \) and \( P_m \) and equals 0 otherwise at time instant \( j \). It is worthy to mention that LPs \( P_t \) and \( P_m \) must be in two consecutive time instants \( T_j, T_{j+1} \). The equation contains three summations, the outer most summation accounts for different time instances where a transition may occur. These time instances are calculated using the gates real delay model where the whole circuit is scanned from its inputs to its outputs. The two innermost summations account for subsequent time instances where each two consecutive time instances are considered for calculating possible transitions by using the transition function \( tr(...) \).

If the unit-delay model for gates is used to calculate the transitions of the circuit shown in Fig. 1, they will be 24, 36 and 48 respectively. This results in a lower accuracy (20%) for node \( f \). Experiments show that, as the number of stages for the circuit increases, the internal nodes’ transition will be less accurate. A compromise between the accuracy and the complexity of the algorithm is needed in choosing the appropriate delay model.

In addition to obtaining the toggle rate for a specific circuit, the above method can be used to get appropriate gate delay combinations for obtaining the minimum toggle rate and hence, the minimum switching power. The circuit in Fig. 2 has three different gates which means three different values of propagation delay; \( \delta_1 \) for the NAND gate, \( \delta_2 \) for the OR gate and \( \delta_3 \) for the AND gate. The circuit has 1 output \( e \) and 2 internal nodes \( c, d \). There are six different combinations for the delays \( \delta_1 < \delta_2 < \delta_3, \delta_1 < \delta_3 < \delta_2, \delta_2 < \delta_1 < \delta_3, \delta_2 < \delta_3 < \delta_1, \delta_3 < \delta_2 < \delta_1 \) and \( \delta_3 < \delta_1 < \delta_2 \); each combination may produce a different toggle rate.

The circuit is simulated for the different delay combinations and the results show that if \( \delta_1 < \delta_3 \) (regardless of \( \delta_2 \)), the transitions at nodes \( d, e \) and \( f \) will be 6. While if \( \delta_1 > \delta_3 \), nodes \( (c, d \) and \( e \) transitions will be 6, 6 and 0 respectively, which means less power consumption. The delay of the gates can be modified by adjusting the layout to control the total gate delay and get the minimum power.

### Results and discussion

To validate the proposed method, it was applied to gate-level implementations of well-known commercial and academic benchmark digital circuits and the efficiency was measured in terms of the amount of memory saved as well as the processing time compared to exhaustive and Monte Carlo simulations. Since the basic requirement of the simulation tool is the construction of Table 2 which is used to get all possible transitions between the LPs through time, its size is used to determine the memory saving ratio that the tool provides. The memory saving here is defined as the ratio of the amount of memory required to store the LPs in Table 2 to the memory space needed for exhaustive simulations.

For exhaustive simulations, the number of vectors that must be stored is \((2^n \times (2^n - 1))/2\). In the proposed method, the vectors required to be saved are the main truth table vectors used to calculate LPs (2^n vectors) in addition to processing vectors. Processing vectors are those stored for each processing cycle where a vector is applied to other LGs that do not contain the vector (all LGs except the LG that the vector belongs to), which means that the maximum number of vectors to be saved is \((2^n - \min ||LG||)\) in the worst case. This worst case is when an input is selected from the smallest LG “\( \min ||LG|| \)”.

The maximum size of the required memory is the summation of the truth table vectors \((2^n)\) and the value of the processing vectors which corresponds to a total of \((2^n - \min ||LG||)\) vectors. The method produces memory saving only if \(\min ||LG || > 2^n - 1 (5 - 2^n)\). As \(\min ||LG||\) is a positive integer value, so if \(n\) is greater than 2 (which is common in non-trivial circuits), the given formula is always true; hence, memory saving.
saving is always greater than 1. Note that, for the circuit in Fig. 1, the memory saving ratio is 7.

The method is applied to a Quad 2-Input multiplexer, a Look-Ahead Carry Generator, a 4-bit binary full adder as well as 74 CMOS ICs (ICs 74HC157, 74HC182 and 74HC83 respectively) and a sample circuit from ISCAS’85 benchmark circuits (C17). The circuits’ characteristics as well as the memory saving ratio are included in Table 4. The resulting power is compared to that obtained using the exhaustive simulations and it is found that the results are identical. Furthermore, it is found that the difference between the results obtained using the proposed method and the Monte Carlo approach [6] is negligible. In addition, the memory size used by the proposed approach is always less than the memory needed by exhaustive simulations.

Table 5 shows the processing time of the exhaustive approach compared to the proposed method as well as the Monte Carlo method. The time is measured for the worst case scenario where the building of the truth table is included in the calculation. If the truth table is already available (having been built during the design phase), the time required for the proposed method will be reduced. The used machine is a 2.0 GHz Intel Core i3 processor with 3 GByte memory running MATLAB 2009a tool, where the designer is only responsible of entering the circuit design and the rest of the process is fully automated. It is observed that the processing time of the technique presented in this research is lower compared to the processing time of other techniques except when the number of logic pictures is relatively high. Column 5 in Table 4 shows the ratio between the size of the truth table (2^n) and the number of LPs. This ratio is an indicator of the circuit regularity; larger values of this ratio indicate more regular circuits and vice versa. As shown in Table 4, column 5, for the circuits where the ratio is lower than 3.2, the time needed by the proposed method is greater than the exhaustive method due to circuit irregularity. Furthermore, this ratio has no relation to the size of the circuit (represented by the number of nodes in Table 4, column 2). Hence, the proposed technique is appropriate for small as well as large circuits as long as these circuits are regular.

Conclusions

The paper introduced a deterministic and fast method to calculate the nodes’ toggle rates and hence, estimate the circuit dynamic power. The gates delay model is assumed to be the real-delay model which assumes different delay values for different gates. The method is based on modifying the Logic Pictures (LPs) concept that illustrates the circuit nodes’ status to include the intermediate states arising from the glitches that appear due to different gates’ propagation delays. The method can also be used to redesign the circuit to obtain the minimum possible power by re-modifying the gate delays to obtain the minimum nodes’ toggle rate. To validate the results, the method was applied to some commercial combinational ICs as well as some academic benchmark circuits and compared to the exhaustive as well as Monte Carlo simulations. The results were found to be identical but with much lower complexity and much lower memory space requirements. The memory space saving is up to 30.1% and the time saving is up to 96% for highly regular circuits.

Conflict of interest

The authors have declared no conflict of interest.
Compliance with Ethics Requirements

This article does not contain any studies with human or animal subjects.

References

[1] Theodoridis G, Theoharis S, Soudris D, Goutis C. A probabilistic power estimation method for combinational circuits under real gate delay model. VLSI Des 2001;12(09):79.
[2] Kang SM. Accurate simulation of power dissipation in VLSI circuits. IEEE J Solid-State Circuits 1986;5(21, 88):891.
[3] Salz A, Horowitz MA. IRSIM: An incremental MOS switch-level simulator. In: Proceedings of the 26th Design Automation conference, Las Vegas, NV, USA; 1989. p. 173–78.
[4] Prime Power analyzer. <http://www.synopsys.com>.
[5] Ghosh A, Devadas S, Keutzer K, White J. Estimation of average switching activity in combinational and sequential circuits. In: Proceedings of the Conference on Design Automation (DAC), Anaheim, USA; 1992. p. 253–59.
[6] Burch R, Najm FN, Yang P, Trich TN. A Monte Carlo approach for power estimation. IEEE Trans VLSI Syst 1993;1(63):71.
[7] Aldeen AS, Saad H. A new method for power estimation and optimization of combinational circuits. In: Proceedings of the International Conference on Microelectronics, Cairo, Egypt, 2007. p. 395–98.
[8] Nemani M, Najm F. Towards a high-level power estimation capability. IEEE Trans Comput-Aided Des Integrated Circuits Syst 1996;5(15, 58);598.
[9] Nemani M, Najm F. High-level area and power estimation for VLSI circuits. IEEE Trans Comput-Aided Des Integrated Circuits Syst 1999;18(6):697–713.
[10] Ding CS, Tsui CY, Pedram M. Gate level power estimation using tagged probabilistic simulation. IEEE Trans Comput-Aided Des Integrated Circuits Syst 1998;11(17):1099–107.
[11] Schneider PH, Schlichtmann U, Wurth B. Fast power estimation of large circuits. IEEE J Des Test Comput 1996;1(13, 7):78.
[12] Bhanja S, Ranganathan N. Switching activity estimation of VLSI circuits using Bayesian networks. IEEE Trans VLSI Syst 2003;4(11):558–67.
[13] Li H, Antonio JK, Dhall SK. Fast and precise power prediction for combinational circuits. In: Proceedings of the IEEE Computer Society Annual Symposium on VLSI, Tampa, FL, USA; 2003. p. 254–59.
[14] Machado F, Riesgo T, Torroja Y. Disjoint region partitioning for probabilistic switching activity estimation at register transfer level. In: Proceedings of the 18th International Workshop on Power and Timing Modelling Optimization and Simulation, Lisbon, Portugal; 2008. p. 399–408.
[15] Fouda MF, Abdelhalim MB, Amer HH. Average and maximum power consumption of digital CMOS circuits using Logic Pictures. In: Proceedings of the International Conference on Computer Engineering and Systems ICCES, Cairo, Egypt; 2009. p. 225–30.
[16] Ahmed OS, Abu-Elyazeed MF, Abdelhalim MB, Amer HH, Madian AH. Logic picture-based dynamic power estimation for unit gate-delay model CMOS circuits. Circuits Syst 2013;4(276):279.
[17] Yip G. Expanding the Synopsys prime time solution with power analysis, Synopsys; 2006. www. Synopsys.com.
[18] Jevtic R, Carreras C, Caffarena G. Fast and accurate power estimation of FPGA DSP components based on high-level switching activity models. Int J Electron 2008;95(7):653–68.