I733T001028 . 733
T001028d.733

Multi. Val. Logic., 2000, Vol. 00, pp. 1 – 17
Reprints available directly from the publisher
Photocopying permitted by license only

Power Estimation in Binary CMOS Circuits Based on Multiple-valued Logic

XUNWEI WU[a,*], BANGYUAN CHEN[b] and MASSOUD PEDRAM[c]

[a]*Institute of CAS, Ningbo University, Ningbo 315211, China;* [b]*Dept. of I.S. and E.E., Zhejiang University, Hangzhou 310027, China;* [c]*Dept. of E.E.-Systems, Univ. of Southern California, CA 90007, USA*

Recommended for Publication by Jiu-Lun Fan

(Received 1 December 1998; In final form 21 February 1999)

This paper proposes the use of quaternary and ternary descriptions of signal behavior for power estimation of binary CMOS circuits. Taking into account the effect of race hazards due to signal delay, we find that the operations of three simple gates conform to the definitions of three basic operations in ternary logic. Thus, we propose a multiple-valued-logic simulation algorithm that can be used at the circuit level, where the MOS device is replaced by a simply modeled device, as well as at the gate level. This is a new technique for power estimation in binary CMOS circuits and can be easily extended to probabilistic estimation of power dissipation.

Keywords: Power estimation; Multiple-valued logic; Signal behavior; CMOS

AMS Classifications: 03B50, 94C05

I. INTRODUCTION

The continuing increase in chip density and operating frequency have made power consumption a major concern in VLSI design. For example, the PC chip from Motorola consumes 8.5 W, the Pentium chip from Intel consumes 16 W, and DEC's Alpha chip consumes 30 W. The excessive power dissipation in integrated circuits not only

* Corresponding author. e-mail: xunweiwu@mail.hz.zj.cn

discourages their use in a portable environment, but also causes overheating, which degrades performance and reduces chip life-time [1]. All of these factors drive designers to devote significant resources to reduce the circuit power dissipation. Indeed, the Semiconductor Industry Association has identified low-power design techniques as a critical technological need [2].

The design for low power cannot be achieved without accurate power prediction tools. Therefore, there is a critical need for a technique to estimate power dissipation during the design process to meet the power budget without having to go through a costly redesign effort [3]. In power estimation, we have to know the system clock frequency f_{CLK} and the working input sequence in addition to details of circuit construction. Assume the sequence length is L; thus the input time is T, $T = L/f_{\mathrm{CLK}}$. Theoretically, as long as we can measure the current I_{DD} of the power supply V_{DD}, the average power can be calculated as:

$$P = \frac{1}{T}\int_0^T V_{\mathrm{DD}} \cdot I_{\mathrm{DD}}(t) \cdot dt. \tag{1}$$

Available circuit simulation techniques, such as SPICE3 [4], can simulate the circuit with a representative set of input vectors. They are accurate and capable of handling various device models with complicated parameters, different circuit design styles, single and multi-phase clocking methodologies, tri-state drives, *etc*. However, they suffer from memory and execution-time constraints, and it is almost impossible to simulate a large circuit. In order to make simulation possible the model of a MOS transistor should be significantly simplified, possibly for example, to a simpler switch with certain resistance and capacitance. However, in that case, Eq. (1) cannot be used directly any more, for the following reason.

The basis of power estimation by modeling the MOS transistor as switch is that the power dissipation of a CMOS circuit is related to the switching activity of devices in the circuit. The dominant energy term, $(1/2)C_{\mathrm{load}}(i) \cdot V_{\mathrm{DD}}^2$, represents the energy required to charge or discharge the load capacitance of each device's output node i [5–8]. Thus, we can use the following formula to calculate the average power dissipation:

$$P = \frac{1}{T} \sum_{\text{cycle}=1}^{L} \sum_{\text{all node } i} \frac{1}{2} [f_{\text{out}}(i) \cdot C_{\text{G}}] \cdot V_{\text{DD}}^2 \cdot E_{\text{SW}}(i), \tag{2}$$

where $f_{\text{out}}(i)$ is used to represent as the fan-out of the node i and C_{G} is the gate capacitance of a pair of minimum sized nMOS and pMOS transistors, and the switching activity of the node i is represented by $E_{\text{SW}}(i)$:

$$E_{\text{SW}}(i) = \begin{cases} 1 & \text{if the signal at node } i \text{ is switched in the cycle;} \\ 0 & \text{if the signal at node } i \text{ holds steady in the cycle.} \end{cases}$$

It should be pointed out that while Eq. (2) seems more complicated than Eq. (1), it requires less computation to evaluate.

By using $T = L/f_{\text{CLK}}$, Eq. (2) can be rewritten as

$$\begin{aligned} P &= \sum_{\text{all node } i} \frac{1}{2} [f_{\text{out}}(i) \cdot C_{\text{G}}] \cdot V_{\text{DD}}^2 \cdot f_{\text{CLK}} \cdot \left[\frac{1}{L} \sum_{\text{cycle}=1}^{L} E_{\text{SW}}(i) \right] \\ &= \sum_{\text{all node } i} \frac{1}{2} [f_{\text{out}}(i) \cdot C_{\text{G}}] \cdot V_{\text{DD}}^2 \cdot f_{\text{CLK}} \cdot \overline{E_{\text{SW}}(i)} \end{aligned} \tag{3}$$

where $\overline{E_{\text{SW}}(i)}$ is used to represent the average switching activity of the node i under the input sequence. Correspondingly, $\overline{E_{\text{SW}}(i)}$ could be considered as the transition probability of the node i, in which case, Eq. (3) becomes the basis of probability algorithm in power estimation.

Equation (2) also can be rewritten as

$$P = \frac{1}{T} \sum_{\text{cycle}=1}^{L} \left[\sum_{\substack{\text{node } i \\ E_{\text{SW}}(i)=1}} f_{\text{out}}(i) \right] \cdot \frac{1}{2} C_{\text{G}} \cdot V_{\text{DD}}^2, \tag{4}$$

The above formula can lead to an algorithm for power estimation, by which the fan-outs of all nodes which change in a cycle are summed over all cycles, and the average power dissipation is derived directly.

In a sense, the switching activity of a node is easy to calculate if we consider CMOS devices as simple switches. Assuming x and x' represent the signals at a node before and after a clock transition, the transition can be expressed by $x \oplus x' = 1$ and the holding case (no transition) can be expressed by $x \oplus x' = 0$. However, this is only an

ideal calculation in zero-delay mode. If the signal delay is taken into account, because of the race between input signals there may exist glitches at output, which will increase the transition number at the output node and introduce additional power dissipation. It has been observed that this additional power dissipation is typically 20% of the total power, but can be as high as 200% of the total power in some cases such as in a multiplier [2].

Instead of calculating changes in signal value, this paper uses quaternary and ternary signal descriptions to estimate the power, including the possibility of producing glitches. The algorithm developed is based on multiple-valued logic used in simulation at a circuit level, in which each MOS device is replaced by a simple model.

This paper is organized as follows: In the next section, we describe the signal behavior using a quaternary variable and investigate basic operation rules. Section III contains a ternary description of signal behavior and an algorithm based on ternary logic simulation to calculate the transition activity. Section IV shows how a binary CMOS circuit can be transformed into a ternary circuit for estimating its power dissipation using a common circuit description. Conclusions are given in Section V.

II. QUATERNARY DESCRIPTION OF SIGNAL BEHAVIOR AND RELATED OPERATION RULES

For any signal x in the circuit, we denote its logic values before and after a clock transition as $x(t)$ and $x(t')$ respectively. Correspondingly, four combinations can be used to express all behaviors of the signal, as shown in Table I, where a special quaternary variable $\vec{x}$ denotes the signal behavior. Its four values are $(0, \alpha, \beta, 1)$, where α, β represent the two kinds of transition behavior and 0, 1 represent the two kinds of

TABLE I Quaternary representation for behaviors of a signal

$\vec{x}$	$x(t) \rightarrow x'(t)$	*Behavior*
0	0 0	0-holding
α	0 1	α-transition
β	1 0	β-transition
1	1 1	1-holding

holding behavior. (Note that while the latter have the same forms as signals 0 and 1, their meanings are different.)

Let us discuss how the operations on behaviors for three basic gates: Note that in this discussion we continue to use the traditional operation symbols, but their operating objects are behaviors rather than signals.

If we consider the operation on signals before and after the clock, separately, the quaternary behavior of three basic gates can be written as depicted in Figure 1. Taking the Inverter as an example, when the input signal varies from 0 to 1 ($\vec{x} = \alpha$) its output will vary from 1 to 0 based on the logic operation of the inverter (*i.e.*, $\overline{\vec{x}} = \beta$), *etc.* For the AND operation $\alpha \cdot \alpha$ or $\beta \cdot \beta$, the output behavior is simple. However, for the operation $\alpha \cdot \beta$, it is found that two entries in Figure 1(b) are 0^* (as defined below). Obviously, if we don't take delay into account, signals x and y transit simultaneously and the output 0 will remain unchanged. However, if two transitions don't happen at the same time due to delay, then a peak-like glitch may be created. If we consider the two possible cases for the relative timing of α and β, then we have:

$$0^* = \begin{cases} 0 & (\text{if } \beta - \text{behavior is earlier}) \\ \wedge & (\text{if } \alpha - \text{behavior is ealier}) \end{cases} \tag{5}$$

Similarly, two entries in Figure 1(c) are noted 1^* as the result of the operation $\alpha + \beta$. If considering the two possible cases in which one α or β transition is earlier than the other, a valley-like glitch may appears and we have:

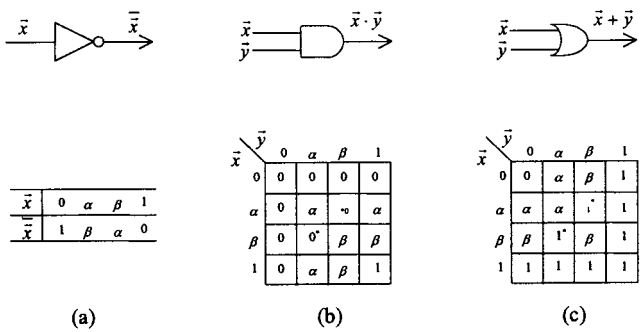

FIGURE 1 Operations of three basic gates to the quaternary behavior.

$$1^* = \begin{cases} 1 & (\text{if } \beta - \text{behavior is earlier}) \\ \bigvee & (\text{if } \alpha - \text{behavior is ealier}) \end{cases} \tag{6}$$

Thus, the physical operations $\alpha \cdot \beta$ and $\alpha + \beta$ may lead to the production of glitches. Since a glitch consists of two transitions, corresponding extra power will be used. If we consider the probabilities for α being earlier or later than β to be equal, we could count the average number of transitions for $\alpha \cdot \beta$ and $\alpha + \beta$ as 1. Here we have taken the signal delay into account, statistically. But, for mixed AND-OR operations involving more transition behaviors, how many transitions are generated at the output on average? As an example, for operations involving three transitions, there are 16 different cases if the last operation is AND:

$$\begin{aligned} &(\alpha \cdot \alpha) \cdot \alpha, (\alpha \cdot \alpha) \cdot \beta, (\alpha \cdot \beta) \cdot \alpha, (\alpha \cdot \beta) \cdot \beta, (\alpha + \alpha) \cdot \alpha, \\ &\quad (\alpha + \alpha) \cdot \beta, (\alpha + \beta) \cdot \alpha, (\alpha + \beta) \cdot \beta, \\ &(\beta \cdot \alpha) \cdot \alpha, (\beta \cdot \alpha) \cdot \beta, (\beta \cdot \beta) \cdot \alpha, (\beta \cdot \beta) \cdot \beta, (\beta + \alpha) \cdot \alpha, \\ &\quad (\beta + \alpha) \cdot \beta, (\beta + \beta) \cdot \alpha, (\beta + \beta) \cdot \beta. \end{aligned}$$

It is easy to understand that the minimum number of transitions is zero, such as when the β behavior is earlier in $(\alpha \cdot \alpha) \cdot \beta$; and the maximum number of transitions is three, such as when the OR operation $(\alpha + \beta)$ generates a glitch, which then is ANDed with the earlier α transition: $(\alpha + \beta) \cdot \alpha$. Now the question is how many transitions will be counted at the output involving three AND-OR operations on average? This is a question we will answer next.

Question Assume that an AND-OR operation has m transition behaviors (at its inputs or internal to the AND-OR structure). As the output behavior, obviously, the minimum number of transitions is zero (if $m > 1$), and the maximum number of transitions at the output is m. We can use $q_{m,0}, q_{m,1}, \ldots, q_{m,i}, \ldots, q_{m,m}$, where $i = 0, 1, \ldots, m$ is the transition number, to represent their relative event numbers for various possible outputs. The total event number is

$$Q_m = \sum_{i=0}^{m} q_{m,i}. \tag{7}$$

Correspondingly, the average transition number for AND-OR operations involving m transition behaviors is

$$L_m = \frac{\sum_{i=0}^{m} i \cdot q_{m,i}}{Q_m}. \tag{8}$$

THEOREM *For mixed AND-OR operations involving m transition behaviors, the average number of transitions at the output is* 1, *that is* $L_m = 1$.

Proof For $m = 1$, *i.e.*, only one transition behavior is involved, obviously we have $q_{1,0} = 0$, $q_{1,1} = 1$, $Q_1 = 1$, and $L_1 = 1$;

For $m = 2$, from the middle four entries in Figure 1(b), we have $q_{2,0} = 1$, $q_{2,1} = 2$, $q_{2,2} = 1$, $Q_2 = 4$, and $L_2 = 1$.

Subsequently, we can use induction to prove the theorem as follows.

Assume that the result holds for $m = 1$, that is, $L_m - 1 = 1$. Consider the case where the output is ANDed with the mth transition behavior as shown in Figure 2(a). We consider two situations: the mth transition behavior is earlier or later than the output. Figure 2(b) shows various

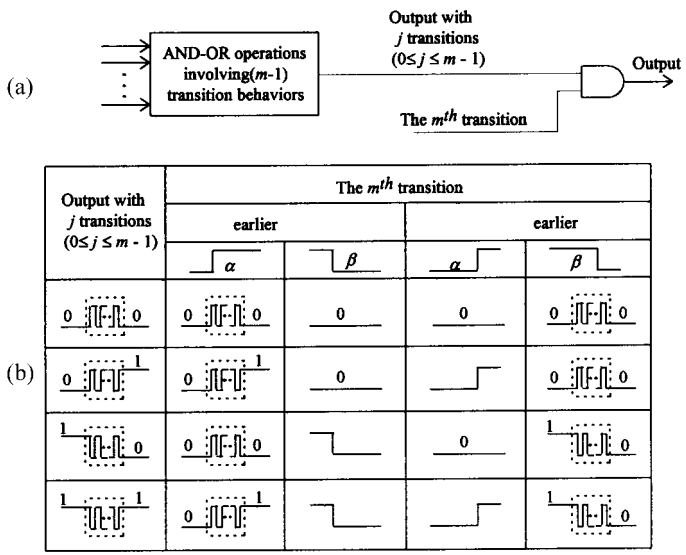

FIGURE 2 AND operation between the mth transition behavior and an earlier output.

possible results for the AND operation. After we take various possible shapes of the former output into account, we only find the following four kinds of result regardless of whether the mth transition behavior is earlier or later: (i) reduced to zero; (ii) reduced to one; (iii) unchanged; (iv) increased to $j+1$. For the OR operation between the mth transition behavior (α or β) and the output, which has j transition instead, we can derive the same conclusion as studied above. Now, we can deduce the relationship between $q_{m,i}$ and $q_{m-1,j}$:

$$\begin{aligned}
&\text{for } i \neq 0, 1, m: \quad q_{m,i} = q_{m-1,i-1} + q_{m-1,i};\\
&\text{for } i = m: \quad q_{m,m} = q_{m-1,m-1};\\
&\text{for } i = 0: \quad q_{m\cdot 0} = \sum_{j=0}^{m-1} q_{m-1,j} + q_{m-1,0} = Q_{m-1} + q_{m-1,0};\\
&\text{for } i = 1: \quad q_{m\cdot 1} = \sum_{j=0}^{m-1} q_{m-1,j} + q_{m-1,0} + q_{m-1,1}\\
&\qquad\qquad = Q_{m-1} + q_{m-1,0} + q_{m-1,1}.
\end{aligned}$$

In the last two equations, $q_{m-1,0}$ is due to case (iii) above, and $q_{m-1,1}$ is due to case (iv) above. Based on these recurrence formulas and the inductive method, we can prove:

$$Q_m = \sum_{i=0}^{m} q_{m,i} = 2^{2(m-1)}; \tag{9}$$

$$\sum_{i=0}^{m} i \cdot q_{m,i} = 2^{2(m-1)}. \tag{10}$$

Since the numerator and the denominator in Eq. (8) are equal, the average number of transitions of AND-OR operations among m transition behaviors always is 1. Q.E.D.

As examples, we list the results for $m = 1,2,3,4,5$ as follows:

$m=1: q_{10}=0\ q_{11}=1$	$\sum q_{1i} = \sum i \cdot q_{1i} = 1$
$m=2: q_{20}=1\ q_{21}=2\ q_{22}=1$	$\sum q_{2i} = \sum i \cdot q_{2i} = 4$
$m=3: q_{30}=5\ q_{31}=7\ q_{32}=3\ q_{33}=1$	$\sum q_{3i} = \sum i \cdot q_{3i} = 16$
$m=4: q_{40}=21\ q_{41}=28\ q_{42}=10\ q_{43}=4\ q_{44}=1$	$\sum q_{4i} = \sum i \cdot q_{4i} = 64$
$m=5: q_{50}=85\ q_{51}=113\ q_{52}=38\ q_{53}=14\ q_{54}=5\ q_{55}=1$	$\sum q_{5i} = \sum i \cdot q_{5i} = 256$

III. TERNARY DESCRIPTION OF BEHAVIOR AND CALCULATION OF TRANSITION ACTIVITY

As mentioned in the above section, the entries 0^* and 1^* in Figures 1(b) and (c) can be treated as one transition, however, it cannot be identified as α (rising transition) or β (falling transition). In fact, if we consider only power dissipation, we don't have to distinguish between an α transition and a β transition since the power dissipation for rising and falling transitions are the same [8]. Therefore we use 1/2 to express both α and β, and represent the behavior $\vec{x}$ by using a ternary variable which takes three values (0, 1/2, 1), as shown in Table II.

According to the above discussion, when the signal delay is taken into account the operations of the three basic gates according to the ternary behavior model are as shown in Figure 3. We find that the relationship between inputs and outputs in Figure 3 happen to conform to the definitions of NOT, AND, and OR operations for ternary signals in ternary logic [9]. Therefore, we can unite the representations of both signals and behaviors by a simple variable. That is, we no longer use an arrow to mark the behavior. Correspondingly, the operations on signal and behavior will have unified definitions in form as follows.

NOT operation

$$\bar{x} = 1 - x \tag{11}$$

AND operation

$$x \cdot y = \min(x, y) \tag{12}$$

OR operation

$$x + y = \max(x, y) \tag{13}$$

TABLE II Ternary representation for behavior of a signal

$\vec{x}$	$x(t) \to x'(t)$	*Behavior*
0	0 0	0-holding
1/2	$x(t) \neq x'(t)$	transition
1	1 1	1-holding

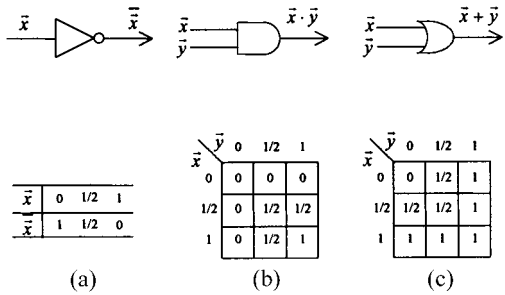

FIGURE 3 Operations of three basic gates for ternary behaviors.

In the equations above, if x, y are binary signals, *i.e.*, $x, y \in \{0, 1\}$, the operations are only for signals; if x, y are behaviors, *i.e.*, $x, y \in \{0, 1/2, 1\}$, the operations are only for behaviors. Thus, for a logic circuit, the output signal of each gate in the circuit can be obtained by the traditional logic operation rules if the input signals are given. Meanwhile, if the input behaviors are given, output behaviors of each gate in the circuit can be derived by using the corresponding ternary operation rule.

In addition, we also can define the Literal operation as follows:

Literal operation

$$x^i = \begin{cases} 1 & \text{if } x = i \\ 0 & \text{if } x \neq i \end{cases} \tag{14}$$

where $x, i \in \{0, 1/2, 1\}$. Thus, the total transition number in a circuit can be expressed as $\sum_{\text{all node } i} x_i^{1/2}$, where x_i is the signal at node i. Obviously, $x_i^{1/2}$ is the $E_{\text{SW}}(i)$ in Eq. (2). In fact, if the input signal sequence is "clean", *i.e.*, without any glitches, its corresponding input behavior sequence will be transformed easily. For example, for the following signal sequence:

$$x = 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ \ldots$$

we obtain the corresponding behavior sequence by averaging the adjacent two entries:

$$\vec{x} = 0\ 1/2\ 1/2\ 1/2\ 1\ 1/2\ 0\ 1/2\ 1\ 1\ 1/2\ 0\ 0\ 1/2\ 1\ldots$$

Notice that in the above sequence 0 and 1 symbols represent the 0-holding and 1-holding behaviors rather than signal values, and 1/2 symbol represents the transition behavior.

As long as the input behaviors are given, it will be easy to derive the output behaviors for each output node from the circuit; the race hazards resulting from delay have been taken into account. As an example, all input behaviors of the circuit shown in Figure 4 are given and the signal values before and after the clock have been given in the brackets. If we don't consider the possible glitches resulting from delay, the output behaviors for each gate in Figure 4 will be A:$(0 \rightarrow 1)$, B:$(1 \rightarrow 0)$, C:$(0 \rightarrow 0)$, D:$(1 \rightarrow 0)$, E:$(1 \rightarrow 1)$, F:$(1 \rightarrow 0)$, G:$(1 \rightarrow 0)$. The total number of transitions is 5. However, if delay is taken into account, the operation rules in Eqs. (5) and (6) can be used to get $A = 1/2$, $B = 1/2$, $C = 0$, $D = 1/2$, $E = 1/2$, $F = 1/2$, $G = 1/2$. The consideration of glitches associated with operation $\alpha + \beta$ results in an additional transition: $E = 1/2$.

Since simulation of the minimum and maximum operations in Eqs. (5) and (6) are computationally expensive, we can use binary coding to express behaviors instead. Based on the quaternary behavior in Table I, we can encode its four values (0, α, β, 1) by using a pair of bits: (00), (01), (10), (11). For the ternary behavior, among its three value (0, 1/2, 1) 0 and 1 are encoded with (00) and (11), and 1/2 can be doubly encoded using both (01) and (10). Correspondingly, NOT of 1/2 still is 1/2 by inverting two digits in the operation. However, if two or more 1/2 are taken as inputs of a gate they should have the same encoding, such as (01), to avoid (00) from (01) ANDing with (10), *etc*. Besides, the minimum and maximum operations become binary AND and OR

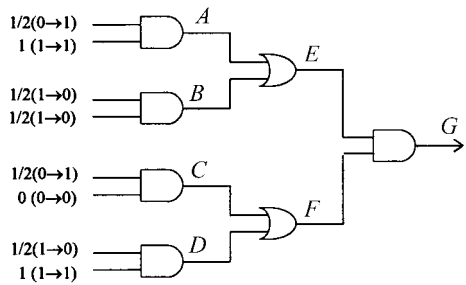

FIGURE 4 Transition calculation of logic circuit.

for each digit respectively. Therefore, all ternary operations are transformed into the corresponding simple two-bit logic operations.

IV. POWER ESTIMATION BY TRANSFORMING BINARY CIRCUITS INTO TERNARY ONES

With the algorithm proposed in the previous section, the original logic diagram of the circuit remains the same except for conceptually transforming all binary gates into corresponding ternary ones. Thus, we can suppose that the original circuit description program, such as SPICE, can be used to realize the ternary algorithm and to estimate the power dissipation. If we use three levels (0 V, 2.5 V, 5 V) to represent three behaviors (0, 1/2, 1) as the input signals for the circuit, the various binary gates have to be transformed to the corresponding ternary gates, which can respond to ternary signals. Reference [10] proposed the transformation for inverter, NAND gate and OR gate, as shown in Figure 5, where the absolute value of thresholds for all MOS transistors are raised, *e.g.*, $V_{Tp} = -3.75\,V$ and $V_{Tn} = 3.75\,V$. When input level is 2.5 V, both *p*MOS and *n*MOS transistors are shut off and the output nodes will be 2.5 V due to the connected auxiliary supply. Obviously, on the resistor there will be a current of $2.5\,V/R$ drawing out of the auxiliary supply if the node value is 0 V, and a current of $2.5\,V/R$ injecting into the auxiliary supply if the node value is 5 V. It is easy to confirm that all three ternary gates in Figure 5 have exact logic functions given by Eqs. (11)–(13).

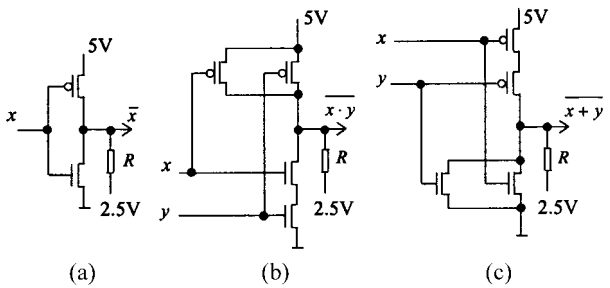

FIGURE 5 Ternary gates (a) inverter, (b) NAND gate, (c) NOR gate.

In the original circuit description program, we can modify the device model to achieve the new thresholds, and to add one resistor and an auxiliary supply, as shown in Figure 6. The common 4-terminal *p*MOS transistor is replaced by a 4-terminal sub-circuit, which contains a *p*MOS transistor with a bias diode-resistor branch, as shown in Figure 6(a). The ideal *p*MOS transistor is just an ideal voltage switch with a threshold of -3.75V, and the diode is also ideal, having no capacitance, no voltage drop, no reverse-biased leakage *etc*. The case for an *n*MOS transistor is similar, and is shown in Figure 6(b).

Figure 7 shows an example of such a transformation. In Figure 7(b), two diode-resistor branches play the role of the resistor R in Figure 5. We can characterize the node P_i as follows. If P_i is low (0V), there will be a current $(V_{PP}-V_{SS})/(R_P)$ being drawn out of the auxiliary supply V_{PP}; If P_i is high (5V), there will be a current $(V_{DD}-V_{NN})/(R_N)$ flowing into the auxiliary supply V_{NN}. However, If P_i assumes the intermediate value (2.5V), there will be no current flow through either R_P or R_N as long as $V_{PP}=V_{NN}=2.5$V. In Figure 7(b) we also mark the possible fan-outs at node P_i and find that the higher this fan-out is, the greater the current is in the auxiliary supplies. Therefore, the current I_{PP} of the auxiliary supply V_{PP} can be used to measure the summation of fan-outs of nodes, which exhibit the 0-holding behavior: $I_{PP}=\sum_{x^0_{\text{node } i}=1} f_{\text{out}}(i)\cdot(V_{PP}-V_{SS})/(R_P)$, where, x_i is the signal at the node P_i. On the other hand, the current I_{NN} of the auxiliary supply

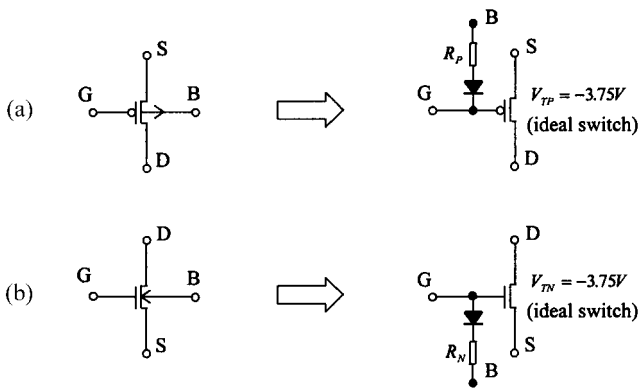

FIGURE 6 Transformations for device modeling.

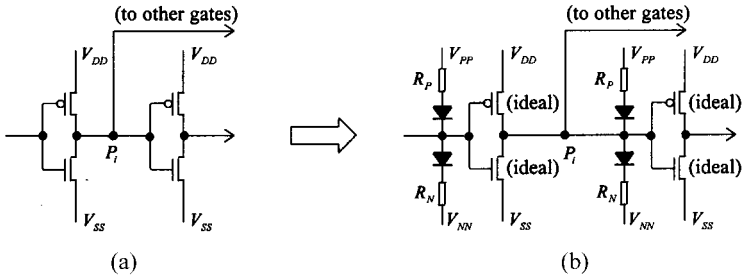

FIGURE 7 Transformation of a conventional binary CMOS circuit.

V_{NN} can be used to measure the summation of fan-out of nodes, which exhibit the 1-holding behavior:

$$I_{NN} = \sum_{\substack{\text{node } i \\ x_i^1 = 1}} f_{out}(i) \cdot \left(\frac{V_{DD} - V_{NN}}{R_N} \right)$$

If we take $V_{DD} = 5\,V$, $V_{SS} = 0\,V$, $V_{PP} = V_{NN} = 2.5\,V$ and $R_P = R_N = 25\,k\Omega$, we have

$$I_{PP} = \left[\sum_{\substack{\text{node } i \\ x_i^0 = 1}} f_{out}(i) \right] \cdot 0.1\,mA,$$

$$I_{NN} = \left[\sum_{\substack{\text{node } i \\ x_i^1 = 1}} f_{out}(i) \right] \cdot 0.1\,mA.$$

Thus, as long as we measure I_{PP} and I_{NN} when simulating this fictitious ternary circuit, we will obtain the summation of fan-outs of nodes, which exhibit 0-holding behavior and 1-holding behavior, respectively. Furthermore, if we know the summation of fan-out of all nodes in the circuit: $\sum_{\text{all node } i} f_{out}(i)$, the summation of fan-out of nodes, which exhibit a transition behavior can be derived as:

$$\sum_{\substack{\text{node } i \\ x_i^{1/2} = 1}} f_{out}(i) = \sum_{\text{all node } i} f_{out}(i) - \sum_{\substack{\text{node } i \\ x_i^0 = 1}} f_{out}(i) - \sum_{\substack{\text{node } i \\ x_i^1 = 1}} f_{out}(i).$$

Then, the average power dissipation in Figure (4) can be obtained.

In fact, $\sum_{\text{all node } i} f_{\text{out}}(i)$ also can be measured by simulation. For example, we make $V_{DD} = V_{SS} = 0\,\text{V}$, $V_{PP} = 2.5\,\text{V}$ and $V_{NN} = -2.5\,\text{V}$. Since $V_{DD} = 0\,\text{V}$, all MOS transistors in the circuit are cut off and there will be current flowing through all diode-resistor branches between V_{PP} and V_{NN} in the circuit. The total current is

$$I_{\text{all}} = \left[\sum_{\text{all node } i} f_{\text{out}}(i)\right] \cdot \frac{V_{PP} - V_{NN}}{R_P + R_N} = \left[\sum_{\text{all node } i} f_{\text{out}}(i)\right] \cdot 0.1\,\text{mA}$$

Then we have

$$\sum_{\substack{\text{node } i \\ x_i^{1/2}=1}} f_{\text{out}}(i) = \frac{I_{\text{all}} - I_{PP} - I_{NN}}{0.1\,\text{mA}}$$

Finally, we can notice that the algorithm using average current is formally similar to Eq. (1). However, while the modified circuit and the measured current are fictitious, the difficulty in circuit simulation with the true model is avoided.

As an example, we have simulated a 4-bit full adder. The simulation shows that the power dissipation (curve *a*) measured by SPICE

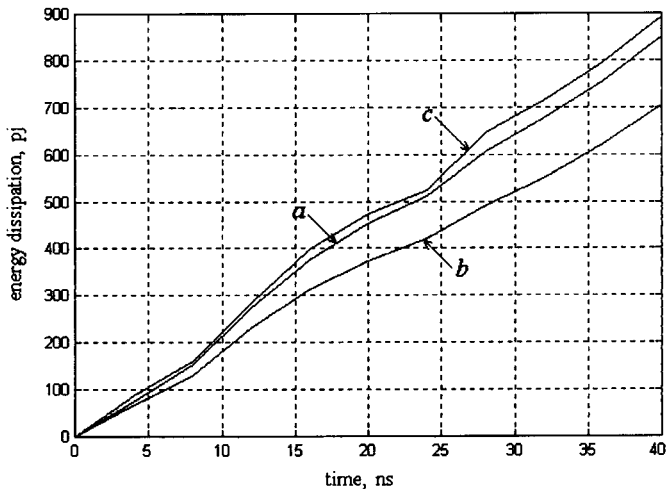

FIGURE 8 Power dissipation curves for a CMOS 4-bit full adder: a – actual energy dissipation; b – energy estimation ignoring glitches; c – energy estimation based on multiple-valued logic.

simulation for the actual circuit is close to the power estimation (curve *c*) based on multiple-valued logic, but is evidently different from the power estimation (curve *b*), which ignores glitches, as shown in Figure 8. Besides, we also find that the output of the gates receiving input signals do not exhibit glitches on the whole. Thus, the power estimation based on multiple-valued logic will differ from that found by calculating the output transition behaviors of these gates receiving input signals. The reason is that the input signals change simultaneously, so the AND, OR operations for their reverse transitions will not generate a glitch. The power dissipation curve *c* shown in Figure 8 is obtained by considering the above modification.

V. CONCLUSION

This paper proposes quaternary and ternary descriptions for signal behavior in power estimation of binary CMOS circuits, whereby the transition activity of signals is investigated. Taking the influence of race hazards into account, we find that the operations of three basic gates on the signal behavior conform to the definitions of three basic operations in ternary logic. Thus, using ternary operations can derive circuit responses to the behavior of input signals. Note that it should be clear that the discussion can be extended to the NAND gate and NOR gate although we have only mentioned the three basic gates (NOT gate, AND gate and OR gate) in this paper. The developed algorithm based on multiple-valued logic can be used in circuit-level simulation, in which the MOS device is replaced by a simply modeled device, as well as in gate-level simulation. The algorithm provides a new method for the power estimation of binary CMOS circuits. The power-estimation procedure based on multiple-valued logic could be developed further for sequential circuits.

Acknowledgments

This project was supported by the NNSF of China (#69773034) and DARPA under contract #F33615-95-C-1627.

References

[1] Najm, F. N. (1995). A survey of power estimation techniques in VLSI circuits, *IEEE Transactions on VLSI Systems*, **2**, 446 – 455.
[2] Semiconductor Industry Association (1992). *Workshop Working Group Reports* (Irving, TX) pp. 22 – 23.
[3] Pedram, M. (1996). Power minimization in IC Design: Principles and applications, *ACM Trans. on Design Automation of Electronic Systems*, **1**, 3 – 56.
[4] Quarles, T. (1989). The SPICE 3 Implementation Guide, *Technical Report No. M89-44* (Electronics Research Laboratory, University of California, Berkeley, California).
[5] Weste, N. H. E. and Eshraghian, K. (1993). *Principles of CMOS VLSI Design: A Systems Perspectives*, 2nd edition (Addison-Wesley Publishing Company, New York).
[6] Devadas, S. and Malik, S. (1995). A survey of optimization techniques targeting low power VLSI circuits, *Proc. 32nd DAC*, pp. 242 – 247.
[7] Shen, A. *et al.* (1992). On average power dissipation and random pattern testability of CMOS combinational logic networks, *Proc. IEEE ICCAD*, pp. 402 – 407.
[8] Dickinson, A. G. and Denker, J. S. (1995). Adiabatic dynamic logic, *IEEE Journal of Solid-State Circuits*, **30**, 311 – 315.
[9] Rine, D. C. Ed. (1985). *Computer Science and Multiple Valued Logic* (North-Holland, Amsterdam and N.Y.).
[10] Mouftah, H. T. and Smith, K. C. (1982). Injected voltage low-power CMOS for three-valued logic, *IEE Proceedings*, **129-G**, 270 – 272.