Gate-Level Power and Current Simulation of CMOS Integrated Circuits

Alessandro Bogliolo, Luca Benini, Giovanni De Micheli, *Fellow, IEEE,* and Bruno Riccò

***Abstract*—In this paper, we present a new gate-level approach to power and current simulation. We propose a symbolic model of complementary metal–oxide–semiconductor (CMOS) gates to capture the dependence of power consumption and current flows on input patterns and fan-in/fan-out conditions. Library elements are characterized once for all and their models are used during event-driven logic simulation to provide power information and construct time-domain current waveforms. We provide both global and local pattern-dependent estimates of power consumption and current peaks (with accuracy of 6 and 10% from SPICE, respectively), while keeping performance comparable with traditional gate-level simulation with unit delay. We use VERILOG-XL as simulation engine to grant compatibility with design tools based on Verilog HDL. A Web-based user interface allows our simulator (PPP) to be accessed through the Internet using a standard web browser.**

***Index Terms*—Current waveform, gate-level simulation, power consumption.**

I. Introduction

POWER consumption and current flows have recently become critical metrics for design evaluation. A large number of power estimation techniques has been proposed [1]–[4] based on models at different levels of abstraction, ranging from electrical level to architectural level [5]–[8]. Electrical-level simulators produce the most accurate results (providing also detailed information on time-domain current waveforms), but are often very demanding in terms of computational resources. Moreover, the large number of simulations needed to reach a significant estimate of average power dissipation further restricts the class of circuits that can be analyzed with electrical simulators in a reasonable time.

At a higher level of abstraction, logic-level simulators handle very large blocks, often enabling full-chip simulation. Furthermore, for digital applications the behavior of the system itself is often described at the logic level. This makes the interface between high-level behavioral specification and logic simulation of the implementation completely straightforward. Consequently, gate-level simulation is usually the core of the debugging and validation strategy for complementary metal–oxide–semiconductor (CMOS) digital designs.

Manuscript received September 8, 1996; revised May 15, 1997. This work was supported in part by the NSF under Contract MIP-9421129, by the AEI (under Grant De Castro), and by the Stanford Center for Integrated Systems.

A. Bogliolo and B. Riccò are with the Department of Electronics, Computer Science and Systems (DEIS), University of Bologna, Bologna, Italy.

L. Benini and G. De Micheli are with the Computer Systems Laboratory (CSL), Stanford University, Stanford, CA USA.

Publisher Item Identifier S 1063-8210(97)06544-X.

Gate-level power/current estimation is critical during two distinct phases of the design process: optimization and validation. When optimizing for power, several transformations are applied to a circuit to reduce its power dissipation. Speed is the main requirement for power estimation performed during optimization. Absolute accuracy is of secondary importance, and the focus is on relative accuracy. *Pattern independent* techniques are well-suited for this kind of power estimation. These techniques provide an estimate of the average switching activity without actually simulating the circuit with a large number of test patterns (see [2] for an overview). Pattern independent techniques for supply current estimation have also been proposed [9], [10].

In contrast, during the validation phase, the designer wants to know the absolute power dissipation of the final implementation with the maximum achievable accuracy. Pattern independent estimators have limited accuracy, mainly because they are based on a simplified model that does not consider phenomena such as noninstantaneous and spurious signal transitions, short-circuit currents and charge redistribution among internal capacitances of logic gates, that may have a sizable impact on the total power dissipation. Although speed is still important, accuracy is the main requirement for power/current estimation during this phase. This work focuses on the accurate estimation required during design validation.

In the recent past, advanced logic simulation techniques for power estimation have been proposed [11]–[13]. In these approaches, lookup tables are obtained by electrical simulation of the basic library elements, and the collected data are then used during gate-level simulation. Although these techniques reported promising results, they have three main limitations. First, they do not assume any model for the internal structure of the basic building blocks (*gates*). Second, they do not deal with multiple input transitions that are not perfectly aligned in time (with misalignments smaller than the propagation delay of the gate). Third, they do not provide any information about supply currents and instantaneous power consumption, that may impair circuit reliability because of voltage drops and electromigration.

Existing current simulators are all essentially based on the following observation: the current drawn by a complex CMOS gate for *any given input transition* has almost the same behavior of the current drawn by an elementary gate with the same driving capability and switching capacitance. Hence, only a small set of reference gates actually need to be characterized, while any other gate simply needs to be reduced into the equivalent elementary one whenever a

transition occurs at its inputs [14]–[20]. These techniques provide good approximations of the current behavior of gates with single input transitions, but they lose accuracy when dealing with internal charge redistributions, signal glitches and (misaligned) multiple transitions. Moreover, *gate-collapsing* techniques are usually not compatible with logic-level design tools. More accurate estimates of supply currents and voltage drops are provided by commercial tools for power analysis and simulation working at transistor level [21].

In this paper we propose a new approach to gate-level average and instantaneous power simulation that exploits a symbolic model of CMOS gates (based on the physical understanding of the main power dissipating phenomena) to overcome the above-mentioned limitations while keeping computational efficiency competitive with traditional gate-level simulators. We use a BDD-based approach to trace the charge status of internal and load capacitances during event-driven logic simulation. Mixed Boolean and regression models capture the dependence of power consumption on input patterns and fan-in/fan-out conditions. Similar models are used to compute state-dependent propagation delays. Triangular pulse approximations are used to represent the time-domain behavior of the current drawn by a CMOS gate corresponding to each input event. Vertices of the triangles are computed at runtime to make current waveforms consistent with the corresponding power and delay estimates. An algorithm is also proposed to model the effect of signal glitches and misaligned input transitions.

We restrict our scope to CMOS circuits mapped on a predefined cell library and we follow the two-step paradigm of library characterization and event-driven logic simulation. Library elements are characterized once for all, and their models are used during logic simulation to provide power information with small computational overhead. Time-domain current waveforms are also constructed by means of current pulse composition. Our model is flexible and can be used to accurately estimate power dissipation and current flows for gates in a large range of load and input conditions. As a result, our method is accurate also for single gate (*local*) estimate, allowing the individuation of critical gates (subcircuits).

We implemented our algorithms in C, using Verilog-XL as simulation platform, therefore maintaining full compatibility with design environments based on Verilog HDL. For our test library the accuracy on local power estimation is within 6% from Spice under a wide range of fan-in and fan-out conditions, while the accuracy on the average power dissipation for large benchmarks is even higher. Peak currents are provided with an average absolute error of 10%. The speed penalty with respect to unit-delay Verilog simulation is within a factor of eight, while the speedup with respect to fast Spice simulation ranges from two to three orders of magnitude.

Our simulator has been integrated in PPP [22], a web-based EDA environment for synthesis and simulation of low-power CMOS circuits. The graphical interface of PPP is a net of interactive HTML pages that can be accessed through the Internet using traditional Web browsers.

In the next section we discuss the main issues involved in gate-level power simulation. Our symbolic model of CMOS gates is presented in Sections III, IV and V. Starting from the physical understanding of the most relevant phenomena, we construct consistent pattern-dependent models for supply energy, current pulses and propagation delays. In Section VI we discuss how to deal with misaligned multiple transitions. In Section VII we present an efficient algorithm for the composition of current pulses in the context of event-driven logic simulation. Implementation details and experimental results are reported in Section VIII. Section IX concludes the work.

II. Gate-Level Power Simulation: Previous Work

Traditional gate-level power estimation is based on the simplifying assumption that the supply current required by a CMOS circuit is essentially spent in charging load capacitances at the outputs of the switching gates. Because of this assumption, the inner structure of the gates is neglected and the average power consumption is evaluated simply by looking at the switching activity (toggle count) and the capacitive load at the gate outputs.

In this way, however, the actual power consumption can be heavily underestimated since several second-order effects (such as short-circuit currents, charging and discharging of internal capacitances and charge sharing) that may have a sizable impact on the global power, cannot be captured.

Example 1: In this and in the following examples we consider a specific CMOS implementation of a three-input OR gate. A transistor-level representation is shown in Fig. 1 where, for the sake of simplicity, parasitics are represented as constant capacitors connected to a common *sink* node (more refined models will be introduced in the next section). Starting from input configuration $\mathbf{x} = 100$, consider a transition of input signal x_1 from 1 to 0 (boldfacing is hereafter used to denote Boolean vectors: $\mathbf{x} = [x_1 x_2 x_3]$). The only effect of this transition that can be captured at logic level is the discharging of C_L, that does not cause any current from power supply. However, a sizable power is actually drawn by the gate due to the charging of internal capacitances (C_1, C_2, and C_3) and to the presence of transient conductive paths from power-supply to ground. In particular, for an input transition time of 0.1 ns and a clock period of 20 ns, power consumption is of 0.22 mW (that is of the same order of power required by a rising transition of the output node, even with an additional load of 50 fF).

Moreover, spurious transitions (glitches) that may represent the 20% of the switching activity [23], cannot be accurately accounted for, due to the use of simplified cell delay models.

Example 2: Using a zero-delay logic model, changing the inputs of the OR gate of Fig. 1 from $\mathbf{x} = 100$ to $\mathbf{x} = 010$ does not cause any effect. However, a misalignment between the falling and rising edges of input signals x_1 and x_2 (i.e., x_2 rising 0.4 ns after x_1 has fallen), gives rise to a power consumption of 0.08 mW because of two phenomena (see Fig. 1):

i) a double, spurious transition (glitch) at the output node, causing the charging/discharging of both C_L and C_3 and
ii) short-circuit currents through both the CMOS stages.

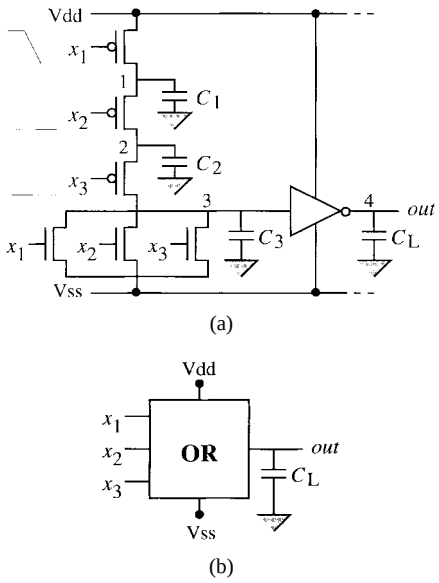


Fig. 1. CMOS realization of a three-input OR gate. Parasitics are modeled by means of four constant capacitors to ground: $C_1 = C_2 = 11$ fF, $C_3 = 157$ fF, and $C_L = 136$ fF.

The above mentioned limitations can be overcome by taking advantage, during gate-level simulation, of previously collected information about the basic building blocks of the circuit.

A. Cell-Based Approaches

Cell-based power estimation [11], [24], [25], [26], [27], [28] improves upon simple logic-level estimation and consists of cell characterization and logic simulation. The characterization phase entails a set of electrical simulations of each library-cell for all possible input transitions and for a wide range of fan-in and fan-out conditions. Timing and power information obtained in this way is used to construct lookup tables for the basic library elements.

Logic simulation is then performed by a back-annotated event-driven simulator, that can be either tightly or loosely coupled with the cell models. In the first case [11], [24], [25], whenever a transition occurs at the input of a gate the corresponding look-up table is addressed to provide power information. In the second case [26], [27], [28], toggle rates and signal switching activities are collected during simulation and used as input data for off-line power estimation. Notice that during simulation the gate is always seen as a black-box, no information about its internal structure and status is exploited.

In principle, as long as the lookup tables have entries corresponding to the actual fan-in/fan-out conditions of each gate of the circuit, back-annotated gate-level simulation reaches the accuracy of electrical simulation. However, power consumption and propagation delay of library cells cannot be precharacterized for all possible transitions and for all possible values of parameters they depend on (input slopes and skews, output loads). In practice, electrical simulations are performed only for single input transitions and for a given set of typical values of the I/O parameters, thus reducing both the number of electrical simulations and the size of the lookup tables. The subsequent discretization ultimately impairs the accuracy of the power estimate. On the other hand, it is difficult to find an algebraic formula that fits accurately the results of electrical simulation.

The power consumption of a CMOS cell depends also on the charge status of its internal capacitances, that is usually neglected in the context of gate-level simulation, giving rise to further approximations.

Example 3: With respect to Fig. 1, consider a transition from $\mathbf{x} = 101$ to $\mathbf{x} = 001$. The actual energy drawn by the OR gate corresponding to this input transition depends on the charge status of its internal capacitances. In particular, no supply energy is required if both C_1 and C_2 have already been charged at V_{dd} by a previously applied input vector $\mathbf{x} = 001$, while otherwise 0.44 pJ (corresponding to 22 μW with a 20 ns cycle-time) are dissipated. Hence, internal voltages should also be taken into account in order to obtain accurate power estimates.

Recently, more refined methods have been proposed that partially exploit the knowledge of the power consuming phenomena inside the cells. In [13], it is observed that the power dissipated by a cell is characterized by two radically different behaviors depending on the ratio between the slopes of input and output transitions. If the ratio is larger than one, short-circuit current becomes important, while if it is smaller this contribution is less relevant. Based on this observation, a model is proposed in which two different fitting formulae are used depending on the above mentioned ratio. The accuracy in this approach is limited by the simple analytic model and by the lack of information on the internal state of the cells.

In [12] a finite-state machine model for the cell is proposed, in which the internal charge status of the gate is modeled and the power dissipated during input transitions is represented by weights associated with the state transitions of the FSM. However, the power dissipated during a transition depends not only on the initial and final charge status, but also on the capacitive load and on the slope of input and output transitions. This dependency is not explicitly modeled, thus requiring the use of (large) lookup tables associated with each transition. Input misalignments and parasitic phenomena causing intermediate voltage levels (such as signal glitches and charge sharing) are not accurately modeled.

Independently of the accuracy, pattern-dependent power estimates do not provide any information about instantaneous power consumption and current flows, that are primary design concerns when dealing with reliability constraints.

Example 4: A comparison between power consumption and current peaks is reported in Fig. 2 for the benchmark circuit C7552. Points on the graph correspond to different input transitions. Notice that there is no linear relation between the two measures. Moreover, they take maximum values corresponding to completely different input transitions. Hence, current peaks cannot be obtained from pattern-dependent power estimate provided by traditional power simulators.

In the next sections, we address the above mentioned issues in order to find a better tradeoff between efficiency and accuracy. In particular, we construct an accurate symbolic model

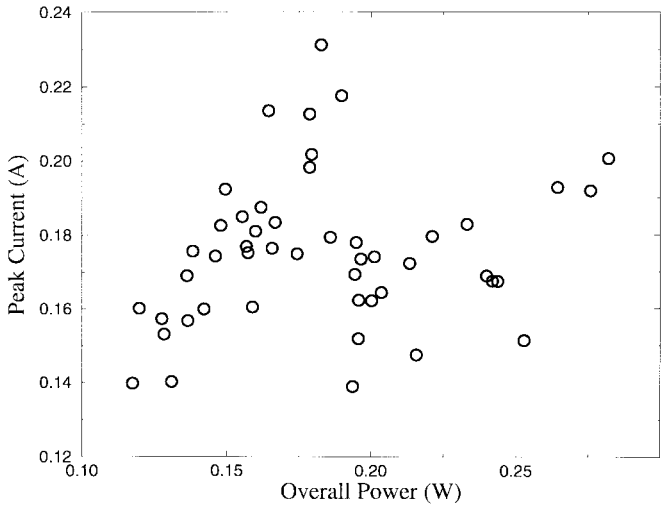


Fig. 2. Peak current versus overall power consumption of benchmark circuit C7552 for 50 different input transitions. Data are obtained by electrical-level simulation.

of CMOS cells that provides consistent pattern-dependent estimates of supply energies, current profiles and propagation delays.

III. Modeling the Supply Energy

A. The Gate Model

When evaluating the average power consumption of a CMOS gate, parasitic capacitors can be assumed to be connected to a common node with a constant voltage level (usually, the ground), that acts as a sink (see Fig. 1). The overall energy drawn from power supply in a whole charging/discharging cycle does not depend on which node the capacitors are lumped to. This affects, however, the time-domain current profile.

In order to find a consistent model for pattern-dependent power consumption and time-domain current waveforms, we represent parasitic and load capacitors connected either to power or to ground routes, according to the bulk connection of the corresponding devices. As shown in Fig. 3, we denote by C_j^p and C_j^g capacitances from node j to the power and ground routes, respectively. In the following, we also use C_j to denote the overall parasitic capacitance connected to node j: $C_j = C_j^p + C_j^g$. Capacitance values are assumed to be constant.

Example 5: Consider the three-input OR gate of Fig. 3. An input transition from $\mathbf{x} = 100$ to $\mathbf{x} = 000$ has three main effects: i) discharging the parasitic capacitances connected to power supply (C_1^p, C_2^p and C_3^p) and the output capacitance connected to ground (C_4^g), ii) charging capacitances C_3^g and C_4^p, and iii) activating temporary conductive paths between the supply and ground routes. Looking at Fig. 3 we can see, however, that discharging phenomena do not contribute to the actual current, since they give rise to pairs of compensating currents ideally flowing through the same power route. For instance, the current from C_3^p to V_{dd} is equal to the current flowing from V_{dd} to C_3^p through the pull-up network. The

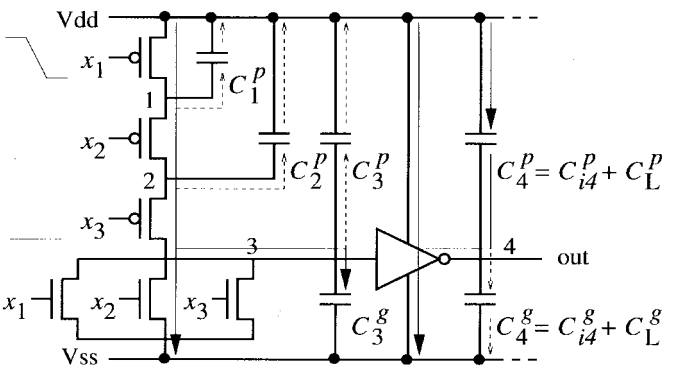
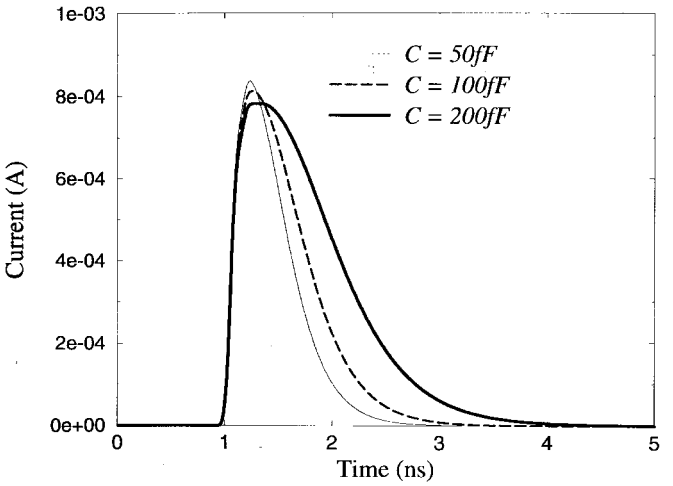
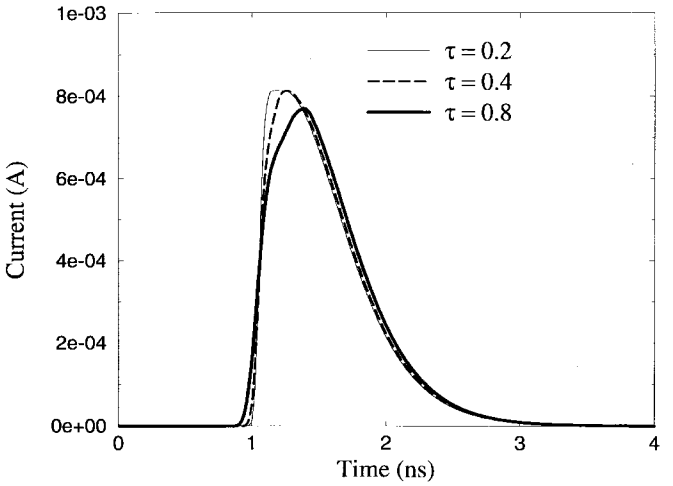


Fig. 3. Schematic representation of a three-input OR gate. Constant capacitors connected to power and ground routes are used to model parasitic and load capacitances. In particular, C_4^p and C_4^g represent the sum of intrinsic output capacitances and external loads.The current flows caused by a falling transition of input signal x_1 are also represented.



(a)



(b)

Fig. 4. Supply current drawn by the OR gate of Fig. 3 during the falling transition of input x_1. Current behaviors are reported for different values of the load capacitance ($C = C_L^p + C_L^g$) and the input falling time (τ).

overall supply current drawn by the OR gate corresponding to the falling transition of x_1 is shown in Fig. 4 for different load capacitances and input slopes.

The supply current $I(t)$ drawn by a CMOS cell corresponding to an input transition (namely, from $\mathbf{x}^i$ to $\mathbf{x}^f$) can always be viewed as consisting of two contributions:

- a *charging* current $I_c(t)$, that increases the total charge of internal and load capacitors and
- a *wasted* current $I_w(t)$ that does not affect the charge status of the cell.

The energy drawn by the cell during the whole transition can accordingly be partitioned into two contributions:

$$E_c = \int_{t^i}^{t^f} (V_{dd} - V_{ss}) I_c(t)\, dt$$
$$E_w = \int_{t^i}^{t^f} (V_{dd} - V_{ss}) I_w(t)\, dt.$$

Apexes i and f are hereafter used to denote the beginning and the end of a given transition, respectively.

In general, it is hard to distinguish between the two portions of supply current (I_c and I_w). Nevertheless, we can easily evaluate E_c by looking at the charge status of the cell. In fact, since $I = \frac{dq}{dt}$, E_c can be expressed as

$$E_c = (V_{dd} - V_{ss}) \int_{t^i}^{t^f} I_c(t)\, dt = (V_{dd} - V_{ss}) \Delta Q_c. \quad (1)$$

ΔQ_c is the total charge provided by the power source to internal and load capacitors, and can be computed using the following equation:

$$\Delta Q_c = \sum_{j \in \mathcal{S}} \Delta q_j - \sum_{j \in \mathcal{S}^p} \Delta q_j^p \quad (2)$$

where

- $\mathcal{S}$ is the set of nodes with a conductive path to V_{dd} for input vector $\mathbf{x}^f$;
- $\mathcal{S}^p$ is the set of nodes with a capacitor to V_{dd};
- Δq_j is the total charge variation at node j: $\Delta q_j = (C_j^p + C_j^g)(V_j^f - V_j^i)$;
- Δq_j^p is the charge variation of C_j^p: $\Delta q_j^p = C_j^p (V_j^f - V_j^i)$.

Sets $\mathcal{S}$ and $\mathcal{S}^p$ are not necessarily disjoint. Nodes belonging to $\mathcal{S} \cap \mathcal{S}^p$ are associated with parasitic capacitances connected to V_{dd} and discharged by input vector $\mathbf{x}^f$. As stated in Example 5, discharging phenomena don't contribute to the overall current, nor to ΔQ_c, since they give rise to two compensating supply currents.

Example 6: Consider the example situation of Fig. 3. All nodes have a constant capacitor to V_{dd}: $\mathcal{S}^p = \{1,2,3,4\}$. For input vector $\mathbf{x}^f = 000$, the set of nodes with a conductive path to V_{dd} is $\mathcal{S} = \{1,2,3\}$. Since there are no capacitors from nodes 1 and 2 to ground, we assume $C_1^g = C_2^g = 0$. Equation (2) becomes

$$\begin{aligned}\Delta Q_c &= C_1^p \Delta V_1 + C_2^p \Delta V_2 + \left(C_3^p + C_3^g\right) \Delta V_3 \\ &\quad - C_1^p \Delta V_1 - C_2^p \Delta V_2 - C_3^p \Delta V_3 - C_4^p \Delta V_4 \\ &= C_3^g \Delta V_3 - C_4^p \Delta V_4 \\ &= \left(C_3^g + C_4^p\right) * (V_{dd} - V_{ss})\end{aligned}$$

ΔV_j has been used to represent $V_j^f - V_j^i$. If $C_3^g = 100$ fF, $C_4^p = 60$ fF and $V_{dd} - V_{ss} = 5$ V, the effective charge variation is of 800 fC.

It is worth noting that E_c does *not* depend on the I/O parameters, and its computation ultimately requires only the knowledge of the charge status (or voltage level) at each node of the cell. The wasted energy ($E_w = E - E_c$), on the other hand, does *not* depend on the internal charge status, and can be expressed as a function of the I/O parameters. Without loss of accuracy, the modeling task can then be partitioned into two easier subtasks: modeling E_c and modeling E_w.

B. The Charging Energy (E_c)

We denote by $\mathcal{N}$ the ordered set of cell nodes, including primary outputs. In order to compute E_c, we need to know the voltage level at each node at the beginning and at the end of any transition. Moreover, we need to dynamically determine the set ($\mathcal{S}$) of nodes connected to power supply. To solve these problems, we keep track of the Boolean conditions enabling the connection of each node to V_{ss}, to V_{dd} and to each other node in the cell.

These conditions make up a *connection matrix* $\mathcal{M}(\mathbf{x})$, with N rows and $N+2$ columns. The square submatrix consists of the first N columns represents the connectivity among the internal ad output cell nodes: entry $m_{i,j}(\mathbf{x})$ is a Boolean function of the cell inputs, taking value 1 for those input configurations for which a conductive path exists between nodes i and j. Columns $N+1$ and $N+2$ are used to represent the connectivity of each node to power supply (p) and ground (g), respectively.

Example 7: For the OR gate of Fig. 3, the elements of the first row of the connection matrix are: $m_{1,1}(\mathbf{x}) = 1$, $m_{1,2}(\mathbf{x}) = x_2'$, $m_{1,3}(\mathbf{x}) = x_2' x_3'$, $m_{1,4}(\mathbf{x}) = 0$, $m_{1,p}(\mathbf{x}) = x_1'$, and $m_{1,g}(\mathbf{x}) = x_1(x_2 + x_3)$. The output node is denoted by 4.

The efficient handling of the connection matrix is obtained by using reduced ordered binary decision diagrams (BDD's) to represent Boolean functions [29]. To this purpose notice that the square sub-matrix consisting of the first N columns of $\mathcal{M}(\mathbf{x})$ is symmetrical, and the BDD-based representation provides a consistent amount of sharing among its entries. It is also worth noting that $\mathcal{M}(\mathbf{x})$ is constructed only once for all, during the cell characterization phase. At run-time, for each input pattern $\mathbf{x}$ the connection status is then obtained from $\mathcal{M}(\mathbf{x})$ in linear time, by simple BDD evaluations.

During logic simulation, the connection matrix is used both to compute E_c and to update the charge status. In particular, the total charge provided by power supply to the internal and load capacitors can be easily evaluated using the column of $\mathcal{M}$ associated with V_{dd}. For a generic cell with N nodes (including primary output nodes), (2) can be rewritten as

$$\Delta Q_c = \sum_{j=1}^{N} m_{j,p}(\mathbf{x}^f) C_j \Delta V_j - \sum_{j \in \mathcal{S}^p} C_j^p \Delta V_j. \quad (3)$$

Example 8: With respect to Fig. 1, the Boolean conditions enabling the connection of each node of the OR gate to power supply are: $m_{1,p}(\mathbf{x}) = x_1'$, $m_{2,p}(\mathbf{x}) = x_1' x_2'$, $m_{3,p}(\mathbf{x}) = x_1' x_2' x_3'$, $m_{4,p}(\mathbf{x}) = x_1 + x_2 + x_3$. All nodes have also

capacitances to V_{dd}. For instance, the charge provided by V_{dd} when the cell inputs switch to $\mathbf{x}^f = 011$ is expressed by:

$$\begin{aligned}\Delta Q_c &= C_1\Delta V_1 + C_4\Delta V_4 - C_1^p\Delta V_1 - C_2^p\Delta V_2 \\ &\quad - C_3^p\Delta V_3 - C_4^p\Delta V_4 \\ &= C_1^g\Delta V_1 - C_2^p\Delta V_2 - C_3^p\Delta V_3 + C_4^g\Delta V_4.\end{aligned}$$

If the initial vector is $\mathbf{x}^i = 111$, the transition to $\mathbf{x}^f = 011$ does not change the steady state voltage at nodes 2, 3, and 4. Hence, $\Delta Q_c = C_1^g\Delta V_1$.

Equation (3) requires the complete knowledge of node voltages at the beginning and at the end of the transition. Node voltages are updated by exploiting the whole connection matrix

$$\begin{aligned}V_i^f &= m_{i,p}(\mathbf{x}^f)V_{dd} + m_{i,g}(\mathbf{x}^f)V_{ss} \\ &\quad + m_{i,\text{float}}(\mathbf{x}^f)\frac{\sum_{j=0}^{N} m_{i,j}(\mathbf{x}^f)C_jV_j^i}{\sum_{j=0}^{N} m_{i,j}(\mathbf{x}^f)C_j} \qquad (4)\end{aligned}$$

where $m_{i,\text{float}}(\mathbf{x}^f)$ takes value 1 whenever node n_i is floating: $m_{i,\text{float}} = m'_{i,p}m'_{i,g}$. In practice, $m_{i,p}(\mathbf{x})$, $m_{i,g}(\mathbf{x})$ and $m_{i,\text{float}}(\mathbf{x}^f)$ are mutually exclusive conditions:

- if i is connected to power supply $(m_{i,p} = 1)$, the new value of V_i is V_{dd};
- if i is connected to ground $(m_{i,g} = 1)$, the new value of V_i is V_{ss};
- if i is floating $(m_{i,\text{float}} = 1)$, the new value of V_i is computed by taking into account the charge sharing with other nodes.

Example 9: Consider a transition to $\mathbf{x}^f = 101$ at the inputs to the OR gate of Fig. 1. At the end of the transition, node 1 is floating and connected only to 2. So, the new value of V_1 is given by the charge sharing between nodes 1 and 2:

$$V_1^f = \frac{C_1V_1^i + C_2V_2^i}{C_1 + C_2}.$$

Notice that (4) also allows us to take implicitly into account the effect of threshold drop on the voltage levels of internal nodes connected to V_{dd} (V_{ss}) through n-channel (p-channel) transistors [30]. For a generic node (say, i) this is done simply by replacing the nominal values of V_{dd} and V_{ss} with values obtained from electrical simulations (namely, V_{dd_i} and V_{ss_i}), that take into account transistor threshold drops.

C. Sequential Elements

In order to construct a model of E_c in CMOS gates, we have always referred to a static combinational cell (namely, a CMOS implementation of a three input OR gate). We extend now the approach to static and dynamic sequential elements.

1) Static Sequential Elements: The gate-level representation of a static sequential circuit is always characterized by the presence of feedback signals. Consider the edge-triggered register of Fig. 5. At gate level, it can be represented either as a net of six interconnected combinational elements (four inverters and two multiplexers) with two external feedbacks (x_1 and x_2), or as a sequence of two latches (LATCH1 and LATCH2) with internal feedback.

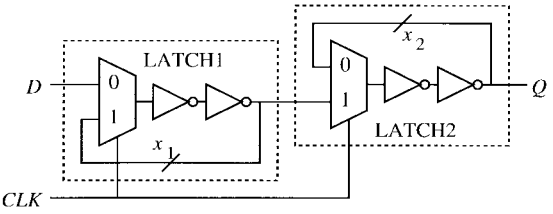


Fig. 5. Static implementation of an edge-triggered D register consisting of two-level sensitive latches.

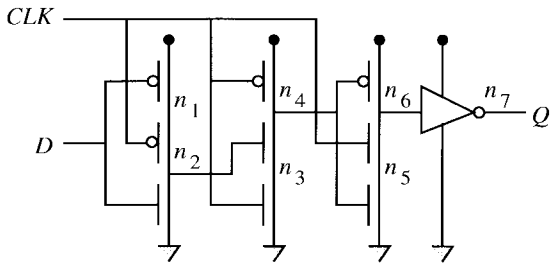


Fig. 6. Dynamic CMOS implementation of an edge-triggered D register.

In the first case, no change is required. Our gate model is directly applied to each component, while feedback signals are explicitly handled by the event scheduling mechanism provided by the simulation platform (needless to say, the initial state of the sequential elements has to be specified in order to obtain significant simulation results).

In the second case, the entire latches need to be modeled as basic building blocks. We refer to the negative level-sensitive latch of Fig. 5 (namely, LATCH1) with inputs D and CLK and output x_1. Because of the internal feedback, neither the functionality nor the connectivity of the cell (i.e., the values of the connection matrix entries) can be inferred from the bare knowledge of the current input pattern. Nevertheless, we want to express the connection matrix as a *combinational* function of Boolean variables. To this purpose, the last value of the feedback signal (x_1) is to be considered as an additional control variable for the connection matrix: $\mathcal{M}(D, CLK, x_1)$.

Notice that x_1 is also the primary output of the cell, and there is a row of the connection matrix associated with it. During simulation, the new value of x_1 is then provided by the model itself. The use of the same signal both as independent and as dependent variable is the implicit representation of the internal feedback. In general, the connection matrix of a sequential element will be a combinational function of both primary inputs and feedback variables. This is the only extension required to deal with sequential components.

2) Dynamic Elements: Dynamic CMOS logics exploit the memory effects associated with the charge retention at the internal (floating) nodes of a cell. On the other hand, in Section II we remarked that the charge status at the internal nodes of a CMOS cell may have a sizable impact on power consumption. In Section III-B we then constructed a state-dependent symbolic model that takes into account charge retentions at internal nodes. As a consequence, our cell model is inherently able to capture dynamic effects associated with internal parasitic capacitances.

From a structural point of view, the only difference between static and dynamic CMOS logics is that in dynamic logic floating nodes may be used to drive CMOS stages.

Example 10: Consider for instance the dynamic edge-triggered register shown in Fig. 6. The second CMOS stage is driven by internal node n_2, that is floating when $D = 0$ and $CLK = 1$. In this case, neither the logic value of n_2 nor the connectivity of the subsequent stage can be inferred from the current values of D and CLK. Nevertheless, the charge status at n_2 is provided by our cell model, and the connectivity of the second CMOS stage can be expressed as a combinational function of CLK and n_2.

In general, to deal with dynamic cells we do not require additional information, but we need to partition the connection matrix and change the sequence of steps involved in model evaluation. Submatrices associated with cascaded stages must be evaluated in sequence, in order to use partial results to drive the subsequent evaluations.

Example 11: The connection matrix of the first stage of the register shown in Fig. 6, is

$$(m_{1,1}, m_{1,2}, m_{1,p}, m_{1,g}) = (1, CLK', D', CLK'D)$$
$$(m_{2,1}, m_{2,2}, m_{2,p}, m_{2,g}) = (CLK', 1, CLK'D', D).$$

Whenever an input event occurs, the matrix is evaluated to obtain V_{n_2} from (4). Then, the connection matrix of the second stage can be evaluated using n_2 as input variable

$$(m_{3,3}, m_{3,4}, m_{3,p}, m_{3,g}) = (1, n_2, n_2CLK', CLK)$$
$$(m_{4,3}, m_{4,4}, m_{4,p}, m_{4,g}) = (n_2, 1, CLK', n_2CLK).$$

In summary, the entries of the connection matrix associated with a generic (combinational or sequential) CMOS cell may be functions of primary inputs, internal variables, and previous values of feedback signals.

D. The Wasted Energy (E_w)

The main contribution to E_w is due to the presence of short circuit currents from power supply to ground. The connection matrix can be used to detect conditions for which there is a transient open path between V_{dd} and V_{ss}. In practice, a wasted current is drawn from power supply whenever a node that was connected to V_{dd} for input vector $\mathbf{x}^i$ is connected to V_{ss} for input vector $\mathbf{x}^f$, or vice versa. For a generic cell with N nodes, this condition is expressed by

$$f_w(\mathbf{x}^i, \mathbf{x}^f) = \sum_{i=1}^{N} [m_{i,p}(\mathbf{x}^i) m_{i,g}(\mathbf{x}^f) + m_{i,g}(\mathbf{x}^i) m_{i,p}(\mathbf{x}^f)]. \tag{5}$$

We remark that for elementary CMOS gates flag f_w can be computed simply by looking at the output activity and (5) can be simplified accordingly. Nevertheless, two-stage cells may have short-circuit currents associated with internal activities that do not affect primary outputs. In this case, the general form of (5) has to be used to evaluate f_w.

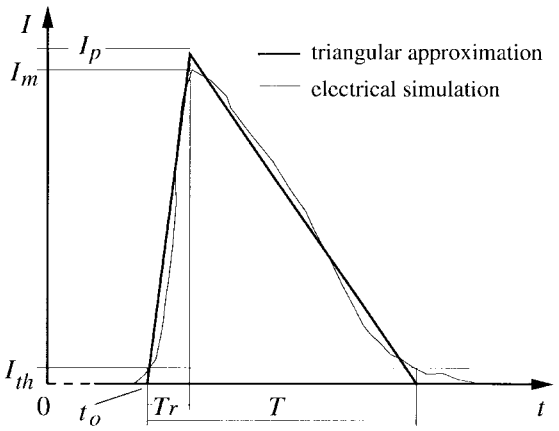


Fig. 7. Triangular approximation of a current pulse. Parameters t_0, T_r, T, and I_p are defined using a threshold $I_{\rm th} = I_m/20$ to filter simulation noise.

If $f_w = 0$, then $E_w = 0$; if $f_w = 1$, instead, E_w depends on fan-in and fan-out conditions, represented by the input falling/rising times $\tau_1, \ldots, \tau_n$ and by the output load C_L. Corresponding to any transition, however, short circuit currents are not influenced by those input (output) parameters associated with input (output) signals that do not change.

Since there are no simple closed-form models for the wasted supply energy, we approximate E_w with a first-order function of the I/O parameters

$$E_w = f_w \cdot (c_1\tau_1 + \cdots + c_n\tau_n + c_{n+1} f_{\rm out} C_L) \tag{6}$$

where $f_{\rm out}$ is a Boolean flag taking value 1 corresponding to output transitions ($f_{\rm out} = out(\mathbf{x}^f) \oplus out(\mathbf{x}^i)$), and the input transition times are set to 0 if the corresponding inputs do not change ($x_i^f = x_i^i \Longrightarrow \tau_i = 0$). Pattern dependence is thus implicitly accounted for.

During characterization, coefficients $c_1, \cdots, c_{n+1}$ are computed by least squares fitting on values obtained by electrical simulations. Notice that modeling E_w requires a number of fitting coefficients that is *linear* in the number of inputs and outputs of the cell.

IV. Modeling Current Pulses

When dealing with time-domain current waveforms, the distinction between charging and wasted contributions is no longer useful to partition the modeling task. On the other hand, the behavior of the *total* current drawn by a CMOS gate corresponding to an input transition can be effectively approximated by an asymmetric triangular pulse. This is shown in Fig. 7 where the current profile obtained by electrical simulation of the three-input OR gate of Fig. 3 is compared with its triangular approximation. Three parameters are then sufficient to describe the approximate shape of a single current pulse: the *rising time* T_r, the *peak value* I_p and the *duration* T. An additional parameter (t_0) is used to denote the initial time of the pulse.

Current modeling then reduces to a twofold issue: 1) finding an operative definition for the pulse parameters and 2) modeling their dependence on input patterns and I/O conditions.

The operative definitions we propose follow two criteria: 1) filtering the noise of the electrical simulations used for characterization and 2) making the triangular pulse as close as possible to the *measured*[1] (nontriangular) one.

We distinguish significant currents from noise by means of a current threshold $I_{\rm th}$ and we define time parameters t_0, T_r, and T considering only current values $I(t) > I_{\rm th}$, as shown in Fig. 7. In particular, accurate estimates have been obtained with a threshold of one twentieth (i.e., 5%) of the maximum measured current I_m.

To define the upper vertex of the triangular pulse, we decided not to use the maximum measured current. This is because the measured current profile is not exactly triangular. Keeping its maximum value as a vertex for the triangular pulse may lead to crude approximations on the overall charge transfer.

We decided, instead, to use a model that preserves the total amount of charge (ΔQ) drawn by the cell during a transition. Since ΔQ represents the area of the current pulse, I_p is defined in order to make the area of the triangular pulse equal to that of the measured one

$$I_p = \frac{2\Delta Q}{T}. \tag{7}$$

The triangular current pulse is then uniquely described by the values of t_0, T_r, T, and ΔQ. To model the dependence of the current profile on the actual switching conditions, we need to find pattern-dependent models for each of these parameters. This is still a very difficult task. Notice, however, that in Section III we have already constructed an accurate model for ΔQ.

If we assume supply voltages to be constant, the amount of charge drawn by the cell during a complete transition is proportional to the overall energy. The model of ΔQ can then be obtained directly from that of E

$$\Delta Q = \frac{E_c + E_w}{V_{dd} - V_{ss}}. \tag{8}$$

This has two main advantages. First, without adding to the global complexity we grant to the current-pulse model the same flexibility we achieved for the model of E (notice that the energy model captures not only the dependence on input patterns and I/O conditions, but also the dependence on the internal charge status). Second, we make the time-domain current waveforms consistent with the overall energy estimate.

V. Modeling Time Parameters

As mentioned in previous sections, power consumption is strictly related with timing: the supply energy drawn by a CMOS cell upon an input transition depends on the input slope and arrival time. Additional time information is required to deal with time-domain current pulses. At the logic level, however, signal slopes are neither represented nor propagated, and simple delay models (such as zero or unit delay) are used for scheduling the events. These approximations have a critical impact on the accuracy of power estimation.

For our energy/current estimates, we need four time parameters to represent the time behavior of a CMOS cell: the propagation delay D (used by the simulation engine for event scheduling), the output falling/raising time $\tau_{\rm out}$ (used for estimating the power consumption of the driven gates), the current rising time T_r (used to determine the position of the current peak) and the current pulse duration T. Incidentally, note that although the pulse duration usually does not correspond to the propagation delay, the initial time of the current pulse (t_0) is always almost coincident with the input arrival time and does not require further attention.

Time parameters are strongly affected by the actual switching conditions. In principle, look-up tables could be constructed based on the results of a large set of electrical simulations. Notice that signal slopes and output loads are analog quantities. Using look-up tables to represent their effects on our parameters would imply to trade-off between large tables and crude discretizations. To avoid both drawbacks, in this section we propose an alternative symbolic model that exploits the use of decision diagrams and linear regressions to provide compact and accurate representations of time parameters as functions of Boolean and analog variables.

Looking at the example of Fig. 4, we notice that the duration of a current pulse is tightly related to the output load, while the location of the peak is mainly related to the input transition time. Furthermore, both dependencies are almost linear. This empirical observation is supported by physical reasons. In first approximation, there is a linear relation between the duration of the current pulse and the amount of charge to be transferred through a conductive path with constant conductivity. Furthermore, the time at which the pull-up (pull-down) network reaches its maximum conductivity is delayed by the input falling (rising) time. Similar considerations apply to propagation delays and output slopes.

Following this observation (supported by the results of electrical simulations run on a large set of CMOS gates and two-stage cells), we use linear equations to approximate the dependence of time parameters on the I/O conditions. The model of T is, for instance,

$$T = c_0 + c_1\tau + c_2 C_L \tag{9}$$

where τ is the input transition time, C_L is the total output load ($C_L = C_{Lp} + C_{Lg}$), possibly including wiring capacitances, and coefficients c_0, c_1, c_2 are to be set in order to fit the results of electrical simulations. If there is more than one input signal switching at the same time, we take the average of the transition times as τ. Notice that only capacitances connected to ground (power) are charged by a supply current corresponding to an output rising (falling) transition. Nevertheless, time parameters always depend on the overall capacitance, since both charging and discharging phenomena actually imply a charge transfer through the pull-up (pull-down) network.

Linear functions of τ and C_L are also used to model T_r, D, and $\tau_{\rm out}$.

[1] Since we always refer to electrical simulations both to characterize library cells and to evaluate the accuracy of our estimates, for the sake of conciseness hereafter we use the term "measured" instead of "provided by electrical simulation."

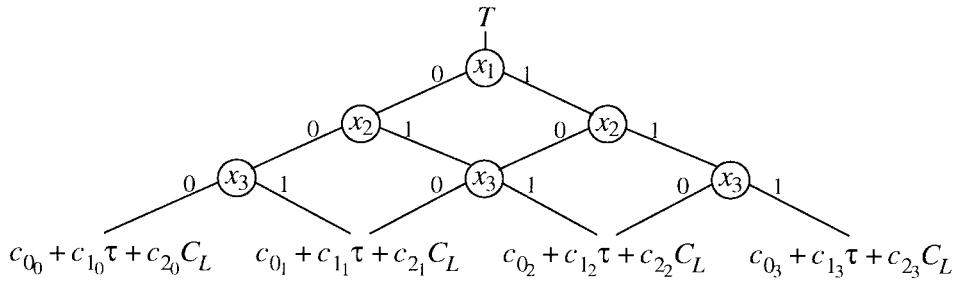


Fig. 8. Decision diagram representing the dependence of T on both Boolean variables (x_1, x_2, x_3) and analog quantities (τ, C_L), for a three-input OR gate.

A. Pattern Dependence

The linear model of (9) provides a good approximation of the actual values of T as long as the driving capability of the pull-up (pull-down) network can be assumed to be constant. In general, however, different driving capabilities are associated with different input transitions, because they activate different conductive paths.

Example 12: Referring to the OR gate of Fig. 3, we consider an input transition from $\mathbf{x}^i = 000$ to $\mathbf{x}^f = 001$ and we compare it to a transition between $\mathbf{x}^i = 000$ and $\mathbf{x}^f = 011$. In the two cases, the same amount of charge is to be transferred through the pull-down network of the first CMOS stage to charge C_{3p} and discharge C_{3g}. However, the driving capability of the pull-down network is not the same, and T will take different values corresponding to the two transitions (e.g., 1.1 ns and 0.9 ns, respectively, for $C_L = 100$ fF).

In principle, different equations should be used for each possible input transition (i.e., for 2^{2n} pairs of input patterns), thus resulting in using $3 \cdot 2^{2n}$ coefficients to model T for a n-input gate. In practice, however, substantial simplifications can be made without loss of accuracy, thanks to two important observations:

Observation 1: Only the last test pattern applied to a CMOS gate affects its driving capability.

Observation 2: The pull-up and pull-down networks of a CMOS gate can assume only a small set of driving capabilities (usually much smaller than 2^n).

Example 13: For instance, the first-stage pull-down network of Fig. 3 has the same conductivity for input patterns $\mathbf{x}^f = 110$, $\mathbf{x}^f = 101$, and $\mathbf{x}^f = 011$.

The complete model of T then consists of a small set of linear equations, associated with clusters of input patterns. Such a model can be effectively represented by a decision diagram [31], in which:

- root is associated with T;
- internal nodes are associated with input variables (decisions being made on the values they take at the end of the actual transition);
- leaves are associated with linear equations (obtained by least squares fitting on the results of electrical simulations).

Fig. 8 shows the model of T for a three input OR gate, with driving capability depending on the number of input signals taking value 1. The structure of the decision diagram is automatically extracted from the transistor-level description of the cell. Similar models are constructed for D, τ_{out} and T_r.

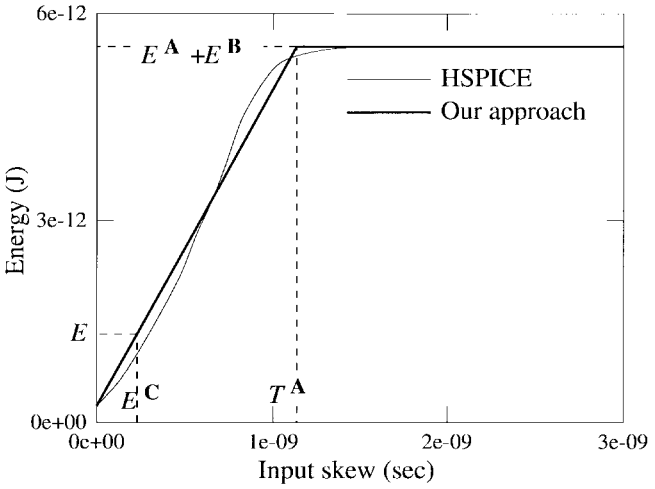


Fig. 9. Effect of the input skew on the energy drawn by a three-input OR gate corresponding to an input transition from $\mathbf{x}^i = 100$ to $\mathbf{x}^f = 010$, through $\mathbf{x}^m = 000$.

VI. Multiple Transitions

So far, we have constructed a refined model of the supply energy/current drawn by a generic CMOS gate corresponding to a transition between two input patterns. We implicitly made the assumption that all switching inputs have the same arrival times (even if they may have different slopes). Unfortunately, in real circuits internal signals are in general slightly misaligned (possibly by short time compared to the transition time of a gate) and may give rise to glitches and overlapping transitions.

Though these phenomena have a sizable effect on power consumption and current flows [23], they have never been modeled at gate-level for two reasons. First, they elude any pre-characterization attempt due to the intractable number of possible combinations of signal skews. Second, the corresponding current waveforms are no longer shaped as triangular pulses.

To handle input misalignments, we propose a method based on the following simple observation.

Observation 3: A misaligned transition of two input signals can be viewed as an intermediate situation between two limiting cases: a simultaneous double transition, and a sequence of two disjoint single transitions.

Since our gate model provides accurate energy/current estimates in both the limiting situations, we approximate any

intermediate case using linear interpolation between the two limits.

A. Supply Energy Interpolation

Referring to a generic CMOS gate with inputs $\mathbf{x}$, assume that a two input transition from input pattern $\mathbf{x}^i$ to $\mathbf{x}^f$ is not perfectly aligned. The misalignment causes an intermediate pattern (say, $\mathbf{x}^m$) to appear at the input of the gate for a short period of time. Assume ΔT to be the delay between the misaligned input transitions (i.e., the input skew). For the sake of simplicity, we use $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ to denote ideal (either single or aligned) input transitions $\mathbf{x}^i \longrightarrow \mathbf{x}^m$, $\mathbf{x}^m \longrightarrow \mathbf{x}^f$ and $\mathbf{x}^i \longrightarrow \mathbf{x}^f$, respectively. We use E to denote the energy drawn during the entire misaligned transition, while we denote by apexes quantities referring to ideal transitions. $T^{\mathbf{A}}$ is the *transient time* associated with transition $\mathbf{A}$.

1) If $\Delta T \ll T^{\mathbf{A}}$, the two transitions are almost aligned. Pattern $\mathbf{x}^m$ never appears at the input of the gate and the energy estimate is $E = E^{\mathbf{C}}$.
2) If $\Delta T > T^{\mathbf{A}}$, the two transitions don't overlap. We have two complete transitions and the total energy estimate is $E = E^{\mathbf{A}} + E^{\mathbf{B}}$.
3) If $0 < \Delta T < T^{\mathbf{A}}$, the two transitions do overlap and we cannot distinguish between their effects. We approximate the total energy by means of a linear interpolation between the two estimates provided by our model for the limiting cases (1) and (2). Namely, $E = (E^{\mathbf{A}} + E^{\mathbf{B}})\frac{\Delta T}{T^{\mathbf{A}}} + E^{\mathbf{C}}(1 - \frac{\Delta T}{T^{\mathbf{A}}})$.

The general model of E is

$$E = \begin{cases} (E^{\mathbf{A}} + E^{\mathbf{B}})\frac{\Delta T}{T^{\mathbf{A}}} + E^{\mathbf{C}}\left(1 - \frac{\Delta T}{T^{\mathbf{A}}}\right) & 0 \le \Delta T < T^{\mathbf{A}} \\ E^{\mathbf{A}} + E^{\mathbf{B}} & \Delta T \ge T^{\mathbf{A}}. \end{cases} \tag{10}$$

The same approach is used to approximate the charge status of the cell at the end of slightly misaligned multiple transitions.

The linear approximation is obviously exact at the boundaries, but its accuracy depends on the definition of $T^{\mathbf{A}}$. In particular, good results have been obtained using the duration of the current pulse (defined at the end of the previous section) as transient time. This means that we consider two input transitions to be disjoint if the corresponding current pulses do not overlap.

Example 14: Consider a misaligned transition from $\mathbf{x}^i = 100$ to $\mathbf{x}^f = 010$ at the inputs of the OR gate of Fig. 3. Assume, in particular, a skew of $\Delta T = 0.2$ ns between the falling edge of x_1 and the rising edge of x_2, giving rise to the intermediate (temporary) pattern $\mathbf{x}^m = 000$. We denote by $\mathbf{A}$ and $\mathbf{B}$ the single transitions $100 \rightarrow 000$ and $000 \rightarrow 010$ and by $\mathbf{C}$ the ideal (aligned) transition $100 \rightarrow 010$. To estimate the total energy drawn by the cell corresponding to the misaligned double transition, we first evaluate the energy consumption associated with the three ideal transitions (directly provided by our model), then we use (10) to compute the actual value of E. In our example, $E^{\mathbf{A}} = 2.7$ pJ, $E^{\mathbf{B}} = 2.5$ pJ, $E^{\mathbf{C}} = 0.3$ pJ, and $T^{\mathbf{A}} = 1.1$ ns. With an input skew of 0.2 ns (10) returns $E = 1.2$ pJ. A comparison with HSPICE is reported in Fig. 9.

B. Current Pulse Interpolation

To describe the linear interpolation of current pulses, we refer to the situation of Example 14. Fig. 10 reports the current waveforms corresponding to the two limiting situations (boldface, capital letters are used to denote the current pulses associated with each ideal transition). As for the supply energy, we estimate the current pulse due to a misaligned input transition starting from the knowledge of those associated with the ideal transitions $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. To do this, we extend to current pulses the interpolation criterion of (10).

Since our logic-level simulation paradigm is inherently event-driven, we construct the shape of the actual current pulse by following an event-driven approach. First, notice that the second input event (i.e., the rising edge of x_2) cannot affect the current behavior before its arrival time. Moreover, when the first event occurs (i.e., when x_1 switches) we do not have any information on the future event on x_2. At time $t_0^{\mathbf{A}}$, the triangular pulse $\mathbf{A}$ provided by the pre-characterized cell-model for a single transition of x_1 is added to the overall current.[2] Let $\mathbf{A}$ have duration $T^{\mathbf{A}}$. When x_2 switches, the overlapping of the two transitions is easily detected by comparing its arrival time ($t_0^{\mathbf{B}}$) with $t_0^{\mathbf{A}} + T^{\mathbf{A}}$.

Instead of adding $\mathbf{B}$ to the overall current, the interpolation procedure is then invoked and a new (virtual) current pulse (namely, $\mathbf{D}$) is constructed having the time parameters of $\mathbf{B}$, and peak value $I_p^{\mathbf{B}}$ such that its area corresponds to the difference between the actual value of ΔQ (obtained from the interpolated value of E) and the already considered charge $\Delta Q^{\mathbf{A}}$

$$T_r^{\mathbf{D}} = T_r^{\mathbf{B}};\ T^{\mathbf{D}} = T^{\mathbf{B}};\ I_p^{\mathbf{D}} = \frac{2(\Delta Q - \Delta Q^{\mathbf{A}})}{T^{\mathbf{B}}}. \tag{11}$$

Notice that $I_p^{\mathbf{D}}$ does not represent a real current and it may also take negative values. Nevertheless, the overall current estimate is a good approximation of the actual behavior provided by electrical simulation. For our example, this is shown in Fig. 10 by pulse $\mathbf{E}$. The intuition behind this procedure is that when the second event occurs (on x_2), we correct the error made by scheduling the full current pulse upon the arrival of the first one (on x_1).

VII. Event-Driven Simulation

During simulation, whenever an event occurs at the inputs of a logic gate, the corresponding model is evaluated. The propagation delay provided by the model is then used to schedule the output event, while the estimated supply energy and current pulse are added to the overall energy and current, respectively.

When simulating a large circuit partitioned in subblocks, we may also be interested in local estimates. We use a simple labeling mechanism to represent subcircuits: gates belonging to the same subcircuit are associated with the same label. The energy (current) drawn by a gate is added not only to the overall energy (current), but also to that

[2] Hereafter we always refer to time-domain current waveforms. The addition of a pulse to the overall current is to be intended as the sum of the corresponding time-continuous functions.

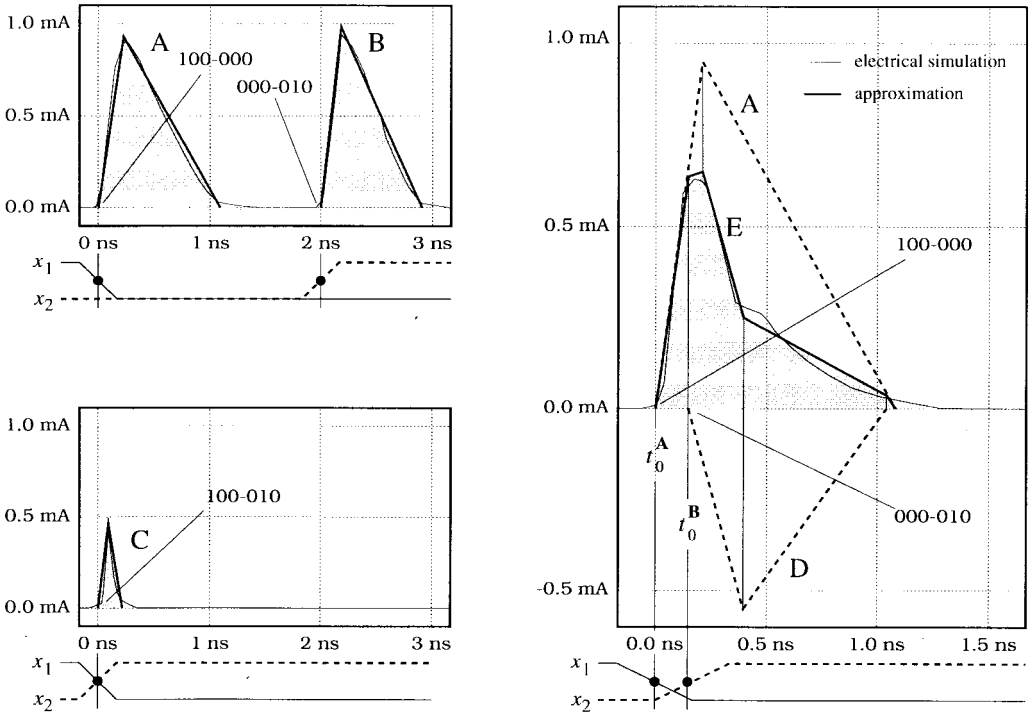


Fig. 10. Approximation of the current pulse associated with a misaligned double transition at the inputs of the OR gate of Fig. 3.

associated with the corresponding subcircuit, thus providing local estimates. Circuit partitioning can be used, for instance, to represent the power distribution network. In this case, subcircuits correspond to sets of gates fed by the same supply route. The overall current drawn by a subcircuit is an estimate of the instantaneous current flowing through the corresponding route.

Adding energy contributions and current pulses is one of the key steps involved in event-driven simulation. However, while energy contributions can be easily added in an event-driven context, current pulses are indeed time-continuous functions that are not directly handled by traditional event-driven simulators. In the following we propose an efficient algorithm that exploits the properties of second-order derivatives to perform effective pulse composition during event-driven simulation.

A. Current Pulse Composition

Consider a current pulse $I(t)$ starting at time t_0 and having duration T. Since the pulse is a time-continuous function, adding it to the overall current $I_{\rm tot}(t)$ would affect $I_{\rm tot}(t)$ for every $t \in [t_0, T]$, thus involving a number of operations related to the ratio between T and the time resolution. Notice however that we approximate $I(t)$ with a triangular pulse (with parameters I_p, T_r, and T) that can be completely described by looking at the instantaneous changes of its slope, occurring at time t_0, $t_0 + T_r$ and $t_0 + T$. Starting from this information, a three-step algorithm can then be used to construct the entire pulse:

1) at t_0 change the slope by I_p/T_r;
2) at $t_0 + T_r$ change the slope by $-I_p/T_r - I_p/(T - T_r)$;
3) at $t_0 + T$ change the slope by $I_p/(T - T_r)$.

The three instantaneous changes in the current slope actually represent an impulsive function that is the second-order derivative of the triangular pulse. Adding impulsive functions is no longer a time-continuous operation. In an event-driven context, corresponding to an input event occurring at time t_0, the three slope changes of the new current pulse are added to the overall second-order derivative at time t_0, $t_0 + T_r$, and $t_0 + T$.

On the other hand, due to the linearity of derivation ($\mathcal{D}$) and integration ($\mathcal{I}$), the following property holds for any pair of functions $f(t)$ and $g(t)$:

$$f(t) + g(t) = \mathcal{I}\{\mathcal{D}\{f(t)\} + \mathcal{D}\{g(t)\}\}. \tag{12}$$

During simulation, we construct the second-order derivative of the overall current by adding the slope changes of the estimated pulses. The second-order derivative is stored in an array of instantaneous impulses, as represented in Fig. 11. The overall current waveform is then obtained at the end of the simulation run by integrating twice with initial conditions $I_{\rm tot}(0) = 0$ and $\mathcal{D}\{I_{\rm tot}\}(0) = 0$.

Example 15: Looking at Fig. 11, let $\mathbf{A}$ and $\mathbf{B}$ represent the current pulses associated with the falling edges of a and b, respectively. At time "0," the event at the input of the AND gate causes pulse $\mathbf{A}$ to be added to the overall current. Hence, the changes in its slope are added to the impulsive second-order derivative of the total current: "+2" at time 0, "−3" at time 1 and "+1" at time 3. When b switches (at time 2), pulse $\mathbf{B}$ is accounted for by adding its second-order derivative at time 2, 3, and 6. Notice that adding a pulse always entails only three steps. Additions are involved if and only if vertices of different pulses overlap at some point (only at time 3 in our example situation). Fig. 11 also reports the overall

TABLE I
EXPERIMENTAL RESULTS ON BENCHMARK CIRCUITS. DATA REFER TO SEQUENCES OF 100 RANDOM GENERATED TEST-VECTORS WITH 20 ns CLOCK PERIOD (MISSING RESULTS MEAN THAT THE CORRESPONDING SIMULATION EXCEEDED 10 h OF CPU AND/OR 20 Mbytes OF RAM)

benckmark			CPU time (s)		P_{avg} (mW)		accuracy (%)			
name	gates	FF's	HSPICE	PPP	HSPICE	PPP	P_{avg}	$I(t)$	$I_p(n)$	$T(n)$
C17	6	—	199.4	1.8	0.435	0.432	0.7	19.3	8.7	6.1
C432	217	—	7867.4	38.8	11.510	10.954	4.8	27.2	7.6	5.4
C499	498	—	21841.8	107.0	21.926	22.884	4.1	18.2	5.6	9.1
C880	343	—	17713.6	65.2	16.262	16.405	0.9	13.8	5.9	5.7
C1908	619	—	—	128.0	—	33.950	—	—	—	—
C7552	2776	—	—	1239.8	—	223.921	—	—	—	—
cmb	49	—	974.6	8.8	1.305	1.329	1.8	14.8	6.6	2.9
decod	54	—	859.8	7.6	1.385	1.400	1.1	22.0	14.7	10.9
parity	75	—	1451.0	13.6	2.381	2.302	3.4	13.7	8.7	5.3
count	113	—	3320.0	17.0	4.985	5.081	1.9	22.5	12.0	6.4
comp	163	—	5450.4	32.8	6.700	6.832	1.9	14.5	8.9	2.6
alu2	359	—	29222.6	67.2	18.010	18.750	4.1	16.2	8.8	5.4
alu4	712	—	—	112.8	—	31.856	—	—	—	—
$s27$	12	3	316.0	3.6	0.600	0.601	0.2	29.6	9.8	2.0
$s208$	72	8	3692.8	10.2	2.097	2.100	0.1	19.8	9.6	5.6
$s953$	342	29	27083.6	40.8	8.768	8.310	5.2	28.9	14.8	7.9
$s1196$	466	18	37358.6	70.8	15.918	15.763	1.0	15.6	5.5	5.3
$s1238$	522	18	—	77.8	—	17.953	—	—	—	—
$s5378$	1346	163	—	161.2	—	35.934	—	—	—	—

current waveform obtained by integrating twice the impulsive function constructed above (represented by the array of its instantaneous values).

In summary, adding a pulse to the overall current entails only three additions (in the worst case). No extra events are generated. Integration is performed off-line once for all. As a consequence, pulse composition does not impose substantial performance degradation on logic simulation.

VIII. IMPLEMENTATION AND EXPERIMENTAL RESULTS

We have developed a power/current simulator, called PPP, based on the algorithms described in this paper. Routines for both automatic construction of connection matrices and least squares fitting have been implemented in C using standard packages for BDD and matrix manipulation. Verilog-XL has been used as event-driven simulation platform. Pre-characterized symbolic models of library cells have been written as C functions and made available from logic simulation using the *programming language interface* (PLI) of Verilog-XL.

We tested our simulator using a low-power CMOS library [32] with complex gates and two-stage cells. Each library cell was characterized according to the model proposed in this paper, using HSPICE to run electrical simulations.

We performed a first set of experiments to verify the single-cell/single-pattern accuracy of our energy model. Each library cell was simulated for all possible test-pairs and for a wide range of fanin and fanout conditions. In the worst case, the average absolute error from HSPICE was of 4%, with a standard deviation of 0.2%. We obtained the same accuracy by applying to each cell a sequence of 100 randomly generated test vectors with 50% of misaligned input transitions.

Global accuracy and performance of PPP were evaluated on a large set of benchmark circuits mapped on the precharacterized test library. Both combinational and sequential circuits were simulated by applying random sequences of 100 test vectors, with 20 ns time period.

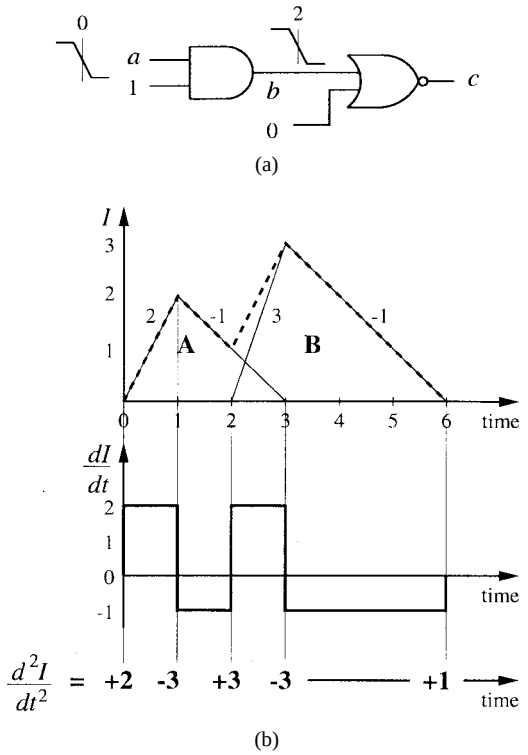


Fig. 11. Current pulse addition in an event-driven context.

Experimental results are reported in Table I. The first three columns contain the circuit name, the number of gates and the number of flip-flops. Columns four and five represent the CPU times required by HSPICE and by PPP, respectively, to simulate the circuit on a DECstation 5000/240. The speedup of PPP with respect to HSPICE was always between two and three orders of magnitude, with an average performance loss of eight times with respect to the simplest gate-level

TABLE II
LOCAL POWER CONSUMPTION OF A NAND-ONLY REALIZATION OF BENCHMARK CIRCUIT $C17$. DATA REFER TO A SEQUENCE OF 100 RANDOM GENERATED TEST VECTORS, WITH A 20 ns CLOCK PERIOD

Cell	Power (mW)		Error	
#	HSPICE	PPP	(mW)	(%)
0	0.032	0.031	−0.001	3
1	0.038	0.037	−0.001	3
2	0.043	0.042	−0.001	2
3	0.156	0.154	−0.002	1
4	0.032	0.030	−0.002	6
5	0.134	0.138	+0.004	3
tot:	0.435	0.432	−0.003	1

simulation with unit delay. Moreover, PPP used about one half of the memory required by HSPICE. Missing results in Table I correspond to simulations that took either more than 10 h of CPU time or more than 20 Mbytes of RAM.

Average power estimates provided by HSPICE and by PPP are reported in columns 6 and 7. The relative error is shown in column 8. Notice that the average power consumption is nothing but the overall supply energy divided by the simulation time. Hence, values shown in column 8 represent also the accuracy of the overall energy estimate. The average error is around 2.5%, with a worst case of 5.2%.

For circuit C17, local power estimates are also reported in Table II. The average power drawn by each internal gate was estimated with a worst case error of 6% from HSPICE.

The last three columns of Table I represent the accuracy achieved on time-domain current simulation. Three parameters are used to represent the accuracy:

- the average absolute error of the time domain current waveform $I(t)$ (represented with 0.1 ns time resolution);
- the average absolute error of the peak current per test pattern $I_p(n)$;
- the average absolute error of the duration of the overall current pulse per test pattern $T(n)$.

The average accuracy of the time-domain current waveforms provided by our approach is of about 20%. This value, however, strongly depends on the time resolution used to represent and compare the results. For larger time steps the average absolute error on $I(t)$ would reduce to the accuracy of the single-pattern energy estimate (around 5%), while for smaller time steps the point-wise comparison would be mainly affected by slight time misalignments between the two waveforms (that actually do not impact more significant measures). We decided to use a high time-resolution (of 0.1 ns) in order to capture all current peaks. The peak value $I_p(n)$ and the duration $T(n)$ of the overall current drawn by the circuit corresponding to each transition at primary inputs have been estimated with 9 and 6% accuracy from HSPICE, respectively. Similar accuracy has been achieved on local current estimates based on random circuit partitionings.

The current waveform obtained for benchmark circuit $alu2$ is reported in Fig. 12 and compared with that provided by HSPICE.

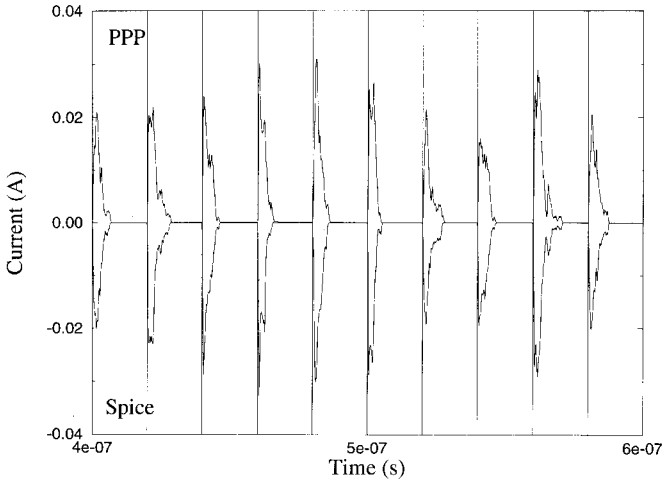


Fig. 12. Comparison between the current waveform provided by PPP and HSPICE for benchmark circuit *alu2*.

A. Model Robustness

The accuracy achieved by PPP on gate-level power and current estimates depends both on the physical significance of the symbolic model and on the statistical significance of the characterization process.

In this paper, we did not obtain a symbolic model by abstracting away the transistor-level description of a gate. On the contrary, we exploited the physical understanding of the power consuming phenomena at the transistor level to partition the modeling task. Our model captures the main contributions to the supply energy/current (dynamic power, short-circuit currents, internal charge redistributions, signal glitches, ...) and their dependence on the I/O parameters, thus providing pattern-dependent power estimates close to those obtained from the electrical simulation of the corresponding transistor netlists.

It is worth noting, however, that a preliminary assumption has been made at the transistor level, by modeling parasitics as constant capacitors lumped to ground and power routes. Results reported in Table I refer to a 2 μ technology library, whose parasitics may be properly represented by lumped capacitances. However, this is not always the case. For deep sub-micron technologies the lumped-capacitance assumption looses its physical meaning and may ultimately impair the accuracy of the gate-level power estimate.

Our model is inherently based on the lumped-capacitance assumption, that allowed us to distinguish between charging and wasted currents and to develop a symbolic model to compute the charging energy (E_c) drawn by a CMOS cell during a complete transition. However, the model is more robust than the assumption it is based on. The reason of its robustness is two-fold. The statistical nature of the characterization procedure automatically allows the wasted-energy estimator (E_w) to compensate the inaccuracy in the charging-energy model. On the other hand, although inaccurate, the symbolic model of the charging energy decreases the variance of the wasted energy contribution. Hence, it enables more accurate fitting.

Moreover, the error introduced by using lumped capacitances to model parasitics can be reduced by carefully setting their values. In practice, capacitances are nothing but fitting parameters that can be used to make the behavior of the transistor-level netlist as close as possible to that of the actual

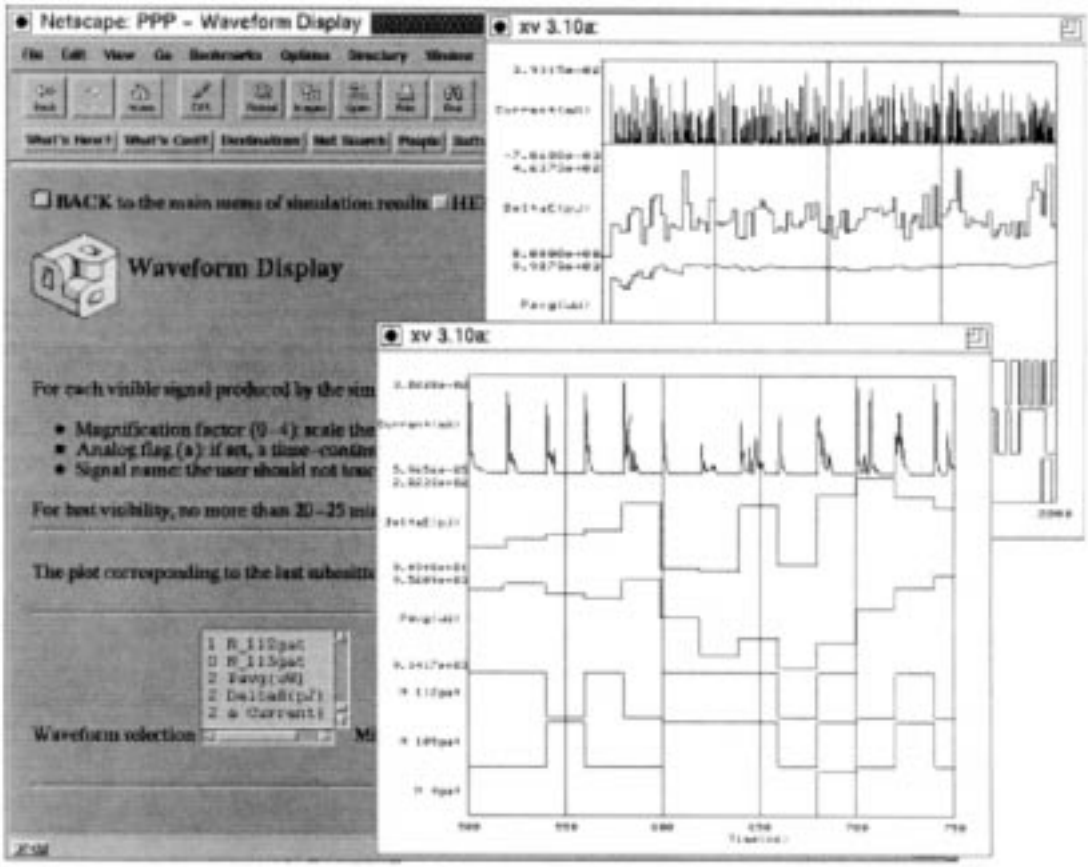


Fig. 13. Web-based graphic user interface of PPP: the waveform display.

circuit. Extracting a significant transistor-level description (with lumped capacitances) from the layout of a circuit is a well-known issue whose solution is out of the scope of this work. However, when characterizing the gate-level power model for a CMOS cell, we can reasonably assume that the transistor-level netlist we use as reference model is a good representation of its actual implementation.

B. Web-Based User Interface

Our simulator has been used as the first building block of a fully integrated synthesis and simulation environment for low-power CMOS circuits [22]. PPP has become the name of the entire environment.

A powerful web-based user interface has been exploited to address critical EDA issues: modularity, platform independence, remote accessibility and user interface standardization. PPP is a modular system consisting of interacting tools that may run on different machines and be provided by different vendors or contributors. The user accesses PPP using his/her own Web browser, with no additional requirements on hardware or software installation. The graphical user interface (GUI) of PPP is exactly the same used for Web navigation and no additional effort needs to be spent in familiarizing with a new GUI when the user first accesses the system.

PPP appears as a tree of highly interactive HTML pages, allowing the users to run synthesis and simulation tools on their own circuits, issue specific commands and analyze (partial) results. Users' commands are interpreted by the server of PPP and sent to specific CAD tools. Both batch and interactive remote execution paradigms are implemented. Results are made available through the Internet by dynamically generated HTML pages. All the details of resource retrieval, data format conversion and tool communication are hidden to the user.

Power/current simulation is performed in four steps. First, the target circuit is transferred from the user's machine to the PPP server. A standard *file transfer protocol* (FTP) is used to this purpose. Second, the user specifies the simulation parameters (time step, test size, ...) and the input patterns to be applied to the circuit. This is done by means of a set of interactive HTML forms. Both random and deterministic simulation can be performed. Third, the simulator is launched. Since no interactive control is required, a batch execution paradigm is used. Last, new HTML pages are automatically generated to make simulation results available to the user. Results are reported in terms of power and current statistics, probability distributions and time-domain signal/current waveforms (as shown in Fig. 13).

PPP is currently available for evaluation at the following URL: `http://akebono.stanford.edu/users/PPP`.

IX. Conclusion

In this work we have presented a new cell-based approach to gate-level power and current simulation, that reduces the gap

between the accuracy of electrical and logic-level techniques. Statistical uncertainties on device characteristics or inaccuracies on wiring capacitance extraction and modeling may lead to mismatches between circuit simulation and measured values that are larger than the inaccuracy of our simulator. As a consequence, our simulation technique gives the designer a level of confidence on power and current estimates comparable to those attainable with computationally expensive circuit simulation. The high local accuracy makes our tool a valuable source of information for optimization algorithms that operate locally within a gate-level network.

The speed-up with respect to electrical-level simulators ranges from two to three orders of magnitude, thus enabling accurate simulation of large circuits in reasonable time. Performance loss with respect to traditional gate-level simulators based on zero/unit delay models is of a factor of 8.

Our technique is based on a symbolic model of CMOS cells that provides consistent estimates of supply energies, current profiles and propagation delays. The dependence on input patterns and I/O conditions is captured without actually using lookup tables. Important effects such as short-circuit currents and internal charge redistributions are also modeled. An interpolation algorithm is used to handle misaligned multiple transitions.

To make the interface with pre-existing design flows (based on Verilog HDL) completely straightforward we have used Verilog-XL as simulation platform. Moreover, we have used our simulator (PPP) as the starting point for the development of an integrated environment for low-power synthesis and simulation, with a Web-based user interface.

Future extensions of our work will be focused on hierarchical power analysis and simulation.

REFERENCES

[1] C. Huang *et al.*, "The design and implementation of Powermill," in *Proc. IEEE Symp. Low Power Electronics*, 1995, pp. 105–110.

[2] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 446–455, Dec. 1994.

[3] C. Y. Tsui, M. Pedram, and A. Despain, "Efficient estimation of dynamic power dissipation under a real delay model," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1993, pp. 224–228.

[4] R. Marculescu, D. Marculescu, and M. Pedram, "Logic level power estimation considering spatiotemporal correlations," in *Proc. IEEE Int. Conf. Comput. Design*, 1994, pp. 294–299.

[5] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE J. Solid-State Circuit*, vol. 29, no. 6, pp. 663–670, 1994.

[6] P. Landman and J. Rabaey, "Architectural power analysis, the Dual Bit Type method," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 173–187, June 1995.

[7] R. S. Martin and J. Knight, "Power-Profiler: Optimizing ASIC's power consumption at the behavioral level," in *Proc. Design Automation Conf.*, 1995, pp. 42–47.

[8] O. Coudert, R. Haddad, and K. Keutzer, "What is the state of the art in commercial eda tools for low power?," in *Proc. Int. Symp. Low Power Electron. Design*, 1996, pp. 181–187.

[9] R. Burch, F. Najm, P. Yang, and I. Hajj, "Pattern independent current estimation for reliability analysis of CMOS circuits," in *Proc. Design Automat. Conf.*, 1988, pp. 294–299.

[10] H. Kriplani, F. N. Najm, and I. N. Hajj, "Pattern independent maximum current estimation in power and ground buses of CMOS VLSI circuits: Algorithms, signal correlations, and their resolution," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 998–1012, Aug. 1995.

[11] B. J. George *et al.*, "Power analysis and characterization for semi-custom desing," in *Proc. Int. Workshop Low Power Design*, 1994, pp. 215–218.

[12] J.-Y. Lin *et al.*, "A cell-based power estimation in CMOS combinational circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1994, pp. 304–309.

[13] H. Sarin and A. McNelly, "A power modeling and characterization method for logic simulation," in *Proc. IEEE Custom Integr. Circuits Conf.*, 1995, pp. 363–366.

[14] S. Chowdhury and J. Barkatullah, "Current estimation in MOS IC logic circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1988, pp. 212–215.

[15] A. Deng, Y. Shiau, and K. Loh, "Time domain current waveform simulation of CMOS circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1988, pp. 208–211.

[16] S. Chowdury and J. Barkatullah, "Estimation of maximum currents in MOS IC logic circuits," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 642–654, June 1990.

[17] U. Jagau, "SIMCURRENT—An efficient program for the estimation of the current flow of complex CMOS circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1990, pp. 396–399.

[18] F. Rouatbi, B. Haroun, and A. J. Al-Khalili, "Power estimation tool for sub-micron CMOS VLSI circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1992, pp. 204–209.

[19] A. Nabavi-Lishi and N. Rumin, "Delay and bus current evaluation in CMOS logic circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1992, pp. 198–203.

[20] A. Nabavi-Lishi and N. Rumin, "Inverter models of CMOS gates for supply current and delay evaluation," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 1271–1279, Oct. 1994.

[21] A. Deng, "Power analysis for CMOS/BiCMOS circuits," in *Proc. Int. Workshop on Low Power Design*, 1994, pp. 3–8.

[22] L. Benini, A. Bogliolo, and B. Riccò, "Distributed EDA tool integration: The PPP paradigm," in *Proc. IEEE Int. Conf. Computer Design*, 1996, pp. 448–453.

[23] L. Benini, M. Favalli, and B. Riccò, "Analysis of hazard contribution to power dissipation in CMOS IC's," in *Proc. Int. Workshop Low Power Design*, 1994, pp. 27–32.

[24] System Science's PowerSim, (http://techweb.cmp.com/ techweb/eet/eda/News/PowerSim.html).

[25] Veritools' Power_Tool, (http://www.veritools-web.com/ power_t.htm).

[26] Synopys' DesignPower, (http://www.synopsys.com/products /power/power.html).

[27] Senté's Watt Watcher/Gate, (http://www.powereda.com/wwinfo .htm).

[28] Viewlogic's POET, (http://www.viewlogic.com/news/ prpoet.html).

[29] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, pp. 677–691, Aug. 1986.

[30] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, 2nd ed. Reading, PA: Addison-Wesley, 1992.

[31] A. Bogliolo, L. Benini, G. D. Micheli, and B. Riccò, "Accurate logic level power estimation," in *Proc. IEEE Symp. Low Power Electron.*, 1995, pp. 40–41.

[32] T. Burd, "Current estimation in MOS IC logic circuits," Univ. of California at Berkeley, M.S. Rep. UCB/ERLM94/89.

Alessandro Bogliolo was born in Urbino, Italy, in 1968. In 1992, he received the Laura degree in electrical engineering from the University of Bologna, Bologna, Italy. In the same year, he joined the Department of Electronics (DEIS) of the University of Bologna, where he is presently working toward the Ph.D. degree in electrical engineering and computer science.

In 1996, he was a Visiting Scholar at the Computer Systems Laboratory (CSL), Stanford University, Stanford, CA. His research interests are in the area of power modeling and simulation of digital IC's. He is also interested in reliability, fault-tolerance, and computer-aided design of low-power IC's.

Luca Benini received the Laurea degree from the University of Bologna, Bologna, Italy, and the M.S. degree in electrical engineering from Stanford University, Stanford, CA. In 1997, he received the Ph.D. degree in electrical engineering from Stanford University.

He is currently a Visiting Scientist with the HP Laboratories, Palo Alto, CA, and he holds a Postdoctoral position at Stanford University. His research interests are in the area of computer aided-design of digital integrated circuits, with particular emphasis on power modeling and simulation and on logic and behavioral synthesis targeting low power.

Giovanni De Micheli (S'79–M'82–SM'89–F'94) received the nuclear engineer degree from Politecnico di Milano, Italy, in 1979, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California at Berkeley, in 1980 and 1983, respectively.

He is a Professor of Electrical Engineering and of Computer Science at Stanford University, Stanford, CA. Previously, he held positions with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, with the Department of Electronics of the Politecnico di Milano, Italy, and with Harris Semiconductor, Melbourne, FL. His research interests include several aspects of the computer-aided design of integrated circuits and systems, with particular emphasis on automated synthesis, optimization and validation. He is author of *Synthesis and Optimization of Digital Circuits* (New York: McGraw-Hill, 1994), co-author of *High-level Synthesis of ASIC's under Timing and Synchronization Constraints* (New York: Kluwer, 1992), and co-editor of *Hardware/Software Co-Design* (New York: Kluwer, 1995) and of *Design Systems for VLSI Circuits: Logic Synthesis and Silicon Compilation* (Amsterdam, The Netherlands: Martinus Nijhoff, 1986).

Dr. De Micheli was received a Presidential Young Investigator award in 1988. He received the 1987 IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN/CIRCUITS AND SYSTEMS Best Paper Award and two Best Paper Awards at the Design Automation Conference, in 1983 and in 1993. He is the Program Chair (for Design Tools) of the 1996–1997 Design Automation Conference. He was Program and General Chair of International Conference on Computer Design (ICCD) in 1988 and 1989, respectively. He was also CoDirector of the NATO Advanced Study Institutes on Hardware/Software Codesign, held in Tremezzo, Italy, 1995, and on Logic Synthesis and Silicon Compilation, held in L'Aquila, Italy, 1986.

Bruno Riccò was born in Parma, Italy, in 1947. He received the electrical engineering degree from the University of Bologna, Bologna, Italy, in 1971, and the Ph.D. degree from the University of Cambridge, Cambridge, UK, in 1975.

While at the University of Cambridge, he worked at the Cavendish Laboratory. In 1980, he became a Full Professor of Applied Electronics at the University of Padova, Italy. In 1983, he joined the Department of Electronics at the University of Bologna. In 1981, he was a Visiting Scholar at Stanford University, Stanford, CA, and later, at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY. His scientific interests concern solid-state devices, fault modeling, testing, and simulation of digital integrated circuits.