Design Automation for Embedded Systems, 9, 19–52, 2004.

Power Estimation Using Probability Polynomials

JOSÉ COSTA jccc@algos.inesc-id.pt
LUIS SILVEIRA lms@inesc-id.pt
Institute for Systems and Computer Engineering, Technical University of Lisbon, R Alves Redol, 9 Lisbon, Portugal

SRINIVAS DEVADAS devadas@mit.edu
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar Street, Cambridge, MA 02139, USA

JOSÉ MONTEIRO jcm@inesc-id.pt
Institute for Systems and Computer Engineering, Technical University of Lisbon, R Alves Redol, 9 Lisbon, Portugal

Abstract. We describe a method of polynomial simulation to calculate switching activities in a general-delay logic circuit. This method is a generalization of the exact signal probability evaluation method due to Parker and McCluskey, which has been extended to handle temporal correlation and arbitrary transport delays. The method can target both combinational and sequential circuits.

Our method is parameterized by a single parameter l, which determines the speed-accuracy tradeoff. l indicates the depth in terms of logic levels over which spatial signal correlation is taken into account. This is done by only taking into account reconvergent paths whose length is at most l. The rationale is that ignoring spatial correlation for signals that reconverge after many levels of logic introduces negligible error. When $l = L$, where L is the total number of levels of logic in the circuit, the method will produce the exact switching activity under a zero delay model, taking into account all internal correlation.

We present results that show that the error in the switching activity and power estimates is very small even for small values of l. In fact, for most of the examples, power estimates with $l = 0$ are within 5% of the exact. However, this error can be higher than 20% for some examples. More robust estimates are obtained with $l = 2$, providing a good compromise between speed and accuracy.

Keywords: power estimation, switching activity, logic circuits, probability polynomials, transition probability, correlation

1. Introduction

Power dissipation has become a major parameter for many VLSI designs. Two independent factors have led to this power concern. One factor is the autonomy of portable devices. For a portable device to be successful in the market, it must have long autonomy and small size. Large batteries cannot be used in order to keep the size small, hence the need for low power designs. The other factor is related to heat dissipation problems. As the feature size reduces, the design density increases, thus increasing the number of power dissipating devices per area. Also, the smaller sizes reduce the propagation delay, allowing for higher clock frequencies, which in turn increase power dissipation.

During the design process, several alternative designs may need to be evaluated and compared. In particular, when targeting low power circuits, a power estimation tool is required to obtain a fast estimation for the different designs.

Power estimation can be done at different levels of design. The higher the abstraction level, the faster the estimation process, yet the lower the accuracy. In our work we focus

more on gate-level power estimation. Although the absolute power estimate may not be very accurate, the estimates are accurate enough in relative terms, permitting a safe comparison of different designs.

Despite the increase in the static power consumption due to leakage current, the switching activity of the gates is still the primary factor in the power consumption of CMOS circuits [17]. Under some generally accepted simplifying assumptions [7], power dissipation at the output of some gate i in a gate-level circuit description can be approximately computed using

$$P_i = \frac{1}{2} \cdot C_i \cdot V_{\mathrm{DD}}^2 \cdot f \cdot N_i \tag{1}$$

where V_{DD} is the supply voltage, f is the clock frequency, C_i represents the load capacitance of the gate and N_i is the gate's switching activity, i.e., the average number of gate output transitions per clock cycle. Under this model, the power estimation process reduces to computing the switching activity (N_i) of the gates in the circuit, as the other parameters can be easily extracted from the circuit.

Many different techniques have been proposed to compute the switching activity in combinational logic circuits [14]. There are two major approaches: *simulation-based* techniques, which simulate the circuit with as many input vectors as needed to achieve some predefined accuracy; *probabilistic* techniques, which propagate user-specified input probabilities through the circuit. Probabilistic techniques can in principle be more efficient since they only require the propagation of a single value through the circuit. However, issues such as spatial and temporal correlation may hinder the effectiveness of the methods.

We describe a method of *polynomial simulation* to calculate switching activities in a general-delay logic circuit [5] which has been extended to handle sequential circuits. This method is a generalization of the exact signal probability evaluation method due to Parker and McCluskey [18], which has been extended to handle temporal correlation and arbitrary transport delays. This method can be applied to both combinational and sequential circuits.

Our method is parameterized by a single parameter l, which determines the speed-accuracy tradeoff. l indicates the depth in terms of logic levels over which signal correlation is taken into account. This is done by only taking into account reconvergent paths whose length is at most l. When $l = L$, where L is the total number of levels of logic in the circuit, the method will produce the exact switching activity under a zero delay model, taking into account all internal signal correlation. Under a generic delay model, the method although very close, is still not exact due to temporal correlation issues.

The rationale behind our approximation scheme is that spatial correlation between internal signals is more important when paths reconverge within a few logic levels. This observation implies that only small errors are introduced when signal independence is assumed for two or more signals, which share input variables and meet after a large number of logic levels.

We present results that show that the error in the switching activity and power estimates is very small even for small values of l. In fact, for combinational circuits and for most of the examples we tried, power estimates are within 5% error of the exact. However, this error can be higher than 20% for other examples. Robust estimates are obtained with $l = 2$, providing a good compromise between speed and accuracy.

In the particular case of sequential circuits previously proposed techniques were limited in the size of the circuits they could handle. Using the approximation proposed will allow

sequential power estimation to be applied to very large circuits. The results for sequential power estimation show that with $l = 2$, the error in the switching activity estimation is less than 6%, with very fast execution times.

This paper is organized as follows. In Section 2, we describe several issues on the computation of signal statistics. In Section 3, we survey previous work on probabilistic switching activity estimation and discuss how it relates to our own. We describe the polynomial simulation method in Section 4. We introduce in Section 5 the concept of dominators and super-gates, concepts used in our approximation scheme. The approximation algorithm based on limited circuit depth signal correlation is presented in Section 6. In Section 7, we extend the method to handle sequential circuits. In Section 8, we provide a set of experimental results that show that with this approximation method very accurate power and node switching estimates can be achieved even for small values of l. We present some conclusions in Section 9.

2. Issues on the Computation of Signal Statistics in Logic Circuits

The accuracy of the results in power estimation depend on how three factors are handled. Those factors are the spatial and temporal correlation, and the delay model used. A more detailed analysis of these issues can be found in [11].

2.1. Spatial Correlation

In signals that are not independent the correlation between those signals is called spatial correlation.

Consider the circuit in Figure 1. We are interested in computing the probability of signal z having the logic value 1, p_z^1. So, we have the following expressions:

$$p_w^1 = p_x^1 \cdot p_y^1$$
$$p_z^1 = p_w^1 \cdot p_y^1.$$

From Figure 1 we can see that the inputs of gate z are not independent. Both inputs depend on the signal y. Assume that the primary inputs, x and y, are independent and $p_x^1 = p_y^1 = 0.5$. If we propagate only numerical values we get $p_w^1 = 0.25$ and $p_z^1 = 0.125$. In this case the spatial correlation is not taken into account. If we do not replace p_w^1 by its numerical value, consequently taking into account the spatial correlation, we obtain $p_z^1 = p_x^1 \cdot p_y^1 \cdot p_y^1 = p_x^1 \cdot p_y^1$. The final result is thus $p_z^1 = 0.25$ which is the correct value since the circuit can be reduced

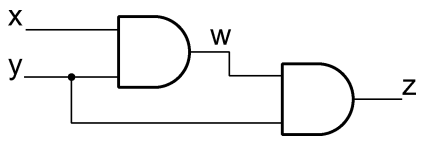

Figure 1. Simple circuit with spatial correlation between signals.

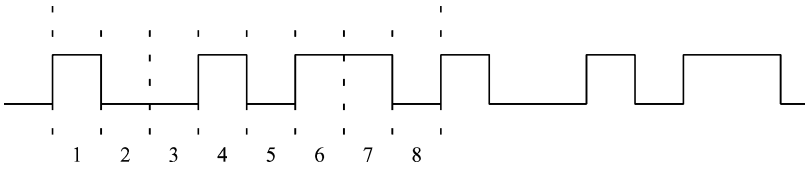

Figure 2. Periodic signal.

to an AND gate. Thus, not accounting for spatial correlation can introduce significant errors in the calculations.

The problem with spatial correlation is that the information that has to be stored in memory (in our example we had to store the probability polynomial of signal w) increases when we go through the circuit from primary inputs to primary outputs. Spatial correlation is related to the existence of reconvergent paths. Thus, taking into account spatial correlation is virtually impossible in circuits with a great number of reconvergent paths.

2.2. Temporal Correlation

In power estimation methods we are interested in the number of transitions of the signals. With the number of transitions per clock cycle we can use Expression 1 to compute the power. In probabilistic methods we use the probability of a signal making a transition to compute the number of transitions. The temporal correlation give us information about the transitions of a signal.

In Figure 2 we have a signal with a period of eight clock cycles. The vertical lines show us the clock. We can see that the probability of the signal being at logic level 1 is $p^1 = 0.5$. Assuming only that, thus ignoring temporal correlation, we can compute the transition probabilities:

$$\begin{aligned} p^{00} &= p^0 \cdot p^0 = 0.25 \\ p^{01} &= p^0 \cdot p^1 = 0.25 \\ p^{10} &= p^1 \cdot p^0 = 0.25 \\ p^{11} &= p^1 \cdot p^1 = 0.25 \end{aligned}$$

where p^{00}, p^{01}, p^{10} and p^{11} correspond to the probability that the signal stays low, makes a low to high transition, makes a high to low transition and stays high, respectively.

By taking a closer look at the signal in Figure 2 we can see that for each period of the signal, we have one cycle where the signal stays low, three cycles with a low to high transition, three cycles with a high to low transition and one cycle where the signal stays high. Thus, being eight the number of clock cycles per period, we have:

$$\begin{aligned} p^{00} &= 1/8 = 0.125 \\ p^{01} &= 3/8 = 0.375 \\ p^{10} &= 3/8 = 0.375 \\ p^{11} &= 1/8 = 0.125. \end{aligned}$$

So, by not using all the information of the signal, in this case the temporal correlation, some error is introduced. Using temporal correlation, the transition probabilities are correctly

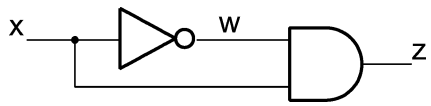

Figure 3. Circuit with glitches.

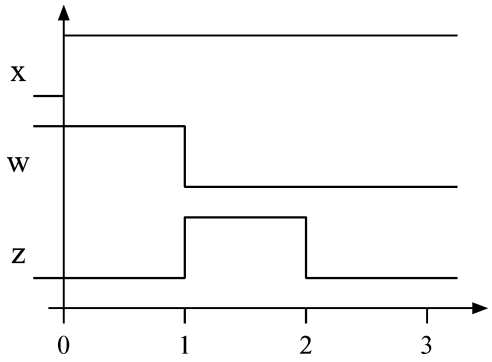

Figure 4. Signals from circuit of Figure 3.

computed. To obtain the number of transitions per clock cycle we just add the two probabilities p^{01} and p^{10}. So, in this example the average number of transitions per clock cycle is $p^{01} + p^{10} = 0.75$.

2.3. Delay Model

The simplest way to model the gate delay is to assume zero delay in all the gates. Meaning that the delay of the signal from the input of a gate to its output is zero and all the gates switch instantaneously. So, each gate has a maximum of one transition per clock cycle. One of the main causes of power dissipation in digital circuits is due to glitches [6]. These glitches occur due to the fact that a non zero delay model can cause the appearance of two, or more, inputs of a gate to have transitions at different time instants.

In Figure 3 we have a circuit in which we assume that the delay for both gates is 1 (unit delay model). A low to high transition in the primary input x causes an undesirable glitch in the primary output z. This can be seen in Figure 4.

Thus, the modeling of gate delays in power estimation is of crucial significance. To obtain exact results we must use a generic delay model with the gate delays given by a library of gates. On the other hand, the use of a generic delay model increases the memory necessary for the computation and the time consumed.

3. Previous Work on Logic Level Power Estimation

There has been a great deal of work in the area of power estimation in the past few years. We describe some representative approaches in this section.

3.1. Zero-Delay Signal Probability Evaluation

Signal probability evaluation methods compute the probability that a Boolean function will evaluate to a 1 on a randomly applied input vector. They model Boolean functionality and disregard circuit delays. The earliest method of signal probability evaluation is the Parker-McCluskey method [18] upon which our method is based. Various other methods to approximate signal probability for testability applications have been proposed.

The use of probabilities to estimate power was first proposed by Cirit [4]. In this work, both signal spatial and temporal correlation are ignored. The transition density work of Najm [13] introduces temporal correlation, but still ignores correlation between internal signals. Improvements to the basic strategy [8] model some internal correlation, but do not serve as a basis for an exact method. In [10], signal probability evaluation and power estimation is based on pairwise correlations between signals. This results in efficient estimation schemes, however, correlation between triplets of signals is ignored. Our method takes into account correlation between two or more signals; our approximations are based on the depth of reconvergence between these multiple signals. The Boolean Approximation Method [22] uses the first term in the Taylor series expansion to efficiently compute signal probabilities taking into account some internal correlation.

Recent work by Cheng generalizes the Parker-McCluskey method to handle transition probabilities by using four-valued variables rather than Boolean variables [2]. The proposed method can be used to obtain exact switching activities for the zero delay model, but no generalization to handle gate delays was made. Methods to improve the efficiency of zero delay switching activity estimation based on the notion of super-gates were described by Cheng.

3.2. General-Delay Switching Activity Estimation

Methods limited to zero-delay models do not account for spurious transitions (glitching) at the output of a gate. Due to different input path delays, gates may switch more than once during a clock cycle. In order to model general-delay transport delays, Najm proposes in [15] propagating probability waveforms through the circuit. These represent the time instants where nodes can toggle, together with information about static signal probability between these instants. Still, correlation between internal signals is ignored. Tsui [21] extends Najm's method by including some correlation coefficients in the probability waveforms.

In [12], Boolean functions representing all possible logical values at each time point for each gate are computed, and the probability of switching activity is evaluated by XOR'ing consecutive time instants. The method relies on the creation of a symbolic network which can become quite large. To perform exact switching activity estimation, BDDs [1] have to be created for each output of the symbolic network, which can be very time-consuming. To handle transition probabilities at primary inputs the method requires constraints on the BDD ordering, which further reduces efficiency. However, when assuming that the primary inputs have no temporal correlation, the symbolic simulation method is useful in calibrating approximation strategies since it is an exact method for a given gate delay and capacitance models.

4. Exact Method

We base the computation of the switching activity at each node in the circuit on the Parker-McCluskey method [18]. A desirable feature of this method is that spatial correlation of internal signals is accurately taken into account. In this section we describe this method and its extension to handle temporal correlation and generic delays.

4.1. The Parker-McCluskey Method

Consider a Boolean function f with inputs $x_1, \ldots, x_N$. The Parker-McCluskey method generates a polynomial that represents the probability that the gate output is a 1, for each gate in the circuit. It follows basic rules for propagating polynomials through logic gates.

Definition 1. Given a polynomial $P(x_1, \ldots, x_N)$, the function supexp(P) is defined as the polynomial resulting from replacing each ${x_i}^k \in P$ with x_i for all $k > 1$.

For example, if $P = {x_1}^2 + x_1 \cdot x_2$, supexp($P$) $= x_1 + x_1 \cdot x_2$.

Given a polynomial P_g for gate g, if g is an input to an inverter, the polynomial for the output of the inverter is $1 - P_g$. Given polynomials P_{g_1} and P_{g_2} at the inputs of an AND gate h, the polynomial for the output of the AND gate will be $P_h = \text{supexp}(P_{g_1} \cdot P_{g_2})$. For an OR gate, $P_h = 1 - \text{supexp}((1 - P_{g_1}) \cdot (1 - P_{g_2})) = P_{g_1} + \text{supexp}((1 - P_{g_1}) \cdot P_{g_2})$.

We begin with the primary input polynomials x_1 through x_N, and traverse the circuit from inputs to outputs to obtain $P_f(x_1, \ldots, x_N)$. Given a probability value for each x_i, namely pr(x_i), pr(f) $= P_f(\text{pr}(x_1), \ldots, \text{pr}(x_N))$.

4.2. Transition Probabilities

The Parker-McCluskey algorithm can be generalized to work with transition probabilities [2].

Each input x_i has four probability variables corresponding to the input staying low, making a rising transition, making a falling transition, and staying high. These are x_i^{00}, x_i^{01}, x_i^{10}, and x_i^{11}, respectively. For each gate g, we now have four polynomials P_g^{00}, P_g^{01}, P_g^{10}, and P_g^{11}, corresponding to the probability that the gate stays low, makes a rising transition, makes a falling transition, and stays high, respectively. We will refer to these four polynomials as the *polynomial group* for a gate.

Table 1 gives the simulation tables of an AND gate and an inverter. These tables can be used to obtain the basic rules for computing the polynomial group for the output of each gate. The polynomial group for the output of an inverter h with input g is simply a re-ordered version of the input polynomial group.

$$P_h^{00} = P_g^{11} \quad P_h^{01} = P_g^{10}$$
$$P_h^{10} = P_g^{01} \quad P_h^{11} = P_g^{00}$$

Table 1. Simulation Calculus for an AND Gate and Inverter

AND	**0→0**	**0→1**	**1→0**	**1→1**	**Inverter**	
0→0	0→0	0→0	0→0	0→0	0→0	1→1
0→1	0→0	0→1	0→0	0→1	0→1	1→0
1→0	0→0	0→0	1→0	1→0	1→0	0→1
1→1	0→0	0→1	1→0	1→1	1→1	0→0

For an AND gate h with inputs g_1 and g_2 we will compute:

$$P_h^{00} = \text{supexp}\left(P_{g_1}^{00} + \left(P_{g_1}^{01} + P_{g_1}^{10} + P_{g_1}^{11}\right) \cdot P_{g_2}^{00} + P_{g_1}^{01} \cdot P_{g_2}^{10} + P_{g_1}^{10} \cdot P_{g_2}^{01}\right)$$
$$P_h^{01} = \text{supexp}\left(P_{g_1}^{01} \cdot P_{g_2}^{01} + P_{g_1}^{01} \cdot P_{g_2}^{11} + P_{g_1}^{11} \cdot P_{g_2}^{01}\right)$$
$$P_h^{10} = \text{supexp}\left(P_{g_1}^{10} \cdot P_{g_2}^{10} + P_{g_1}^{10} \cdot P_{g_2}^{11} + P_{g_1}^{11} \cdot P_{g_2}^{10}\right)$$
$$P_h^{11} = \text{supexp}\left(P_{g_1}^{11} \cdot P_{g_2}^{11}\right)$$

4.3. Gate Delay Effects and Polynomial Waveforms

We propose an important generalization of the Parker-McCluskey method to handle gate delays in this section. This will directly lead to an exact power estimation algorithm, since we just have to sum up the values of appropriate polynomials to obtain the average switching activity at any gate in the circuit.

We will always be manipulating polynomial groups henceforth, and for clarity, we will represent the polynomial group $\{P_g^{00},\ P_g^{01},\ P_g^{10},\ P_g^{11}\}$ as P_g. At each gate output we will have a waveform of polynomial groups, termed a *polynomial waveform*, where each group represents the conditions at the gate output at a particular time instant. We denote the polynomial group for gate g at time instant t as $P_g[t]$.

For example, in the simple circuit of Figure 5, with unit gate delays we will have, for the various signals, the following polynomial waveforms,

$$\begin{aligned} x&: \ P_x[0] \\ y&: \ P_y[0] \\ v&: \ P_v[0] \\ w&: \ P_w[0],\ P_w[1] \\ z&: \ P_z[1],\ P_z[2] \end{aligned}$$

representing the different time instants that each input/gate can make transitions.

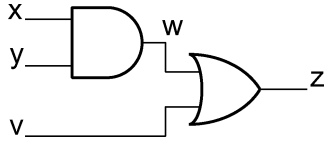

Figure 5. Unit-delay example circuit.

```
    Polynomial_Simulation ( Network )
1.  Initialize_Polynomial_Waveforms ( PIs ( Network ) );
2.  Gates = Topological_Sort( Network ) ;
3.  for each g_i in Gates {
4.     Δ = delay of g_i ;
5.     TimePts = NIL(LIST) ;
6.     for each input g_j of g_i ( g_{i_1}, ..., g_{i_m} ) {
7.        for each time point (k, P_{g_j}[k]) of g_j {
8.           InsertInOrder ( TimePts, (k,P_{g_j}[k]) ) ;
9.        }
10.    }
11.    for each new time point k in TimePts {
12.       P_{g_i}[k + Δ] = G_i(P_{g_{i_1}}[k], ..., P_{g_{i_m}}[k]) ;
13.    }
14. }
```

Figure 6. Pseudo-code for the polynomial simulation algorithm.

We need a polynomial simulation algorithm that can simulate a gate-level network with arbitrary gate delays. Given primary input polynomial waveforms the algorithm should generate polynomial waveforms for each gate output. Such an algorithm is described in pseudo-code in Figure 6.

The simulator processes one gate at a time, moving from the primary inputs to the primary outputs of the circuit. For each gate g_i, an ordered list of the possible transition times of its inputs is first obtained. Then, possible transitions at the output of the gate are derived, taking into account transport delays from each input to the gate output.

It is possible that the polynomial for some input g_{i_j} has not been computed for a given time point k. This simply means that node g_{i_j} does not make a transition at this particular instant. In this case, the polynomial group for g_{i_j} at instant k is obtained from the latest existing polynomial group for g_{i_j} prior to k. If the instant corresponding to this polynomial is m, then

$$P^{00}_{g_{i_j}}[k] = P^{00}_{g_{i_j}}[m] + P^{10}_{g_{i_j}}[m]$$
$$P^{01}_{g_{i_j}}[k] = P^{10}_{g_{i_j}}[k] = 0$$
$$P^{11}_{g_{i_j}}[k] = P^{11}_{g_{i_j}}[m] + P^{01}_{g_{i_j}}[m]$$

The polynomial group for instant k can equally be computed from the polynomial immediately after instant k.

5. Graph Dominators and Super-Gates

The Parker-McCluskey algorithm cannot be used on large circuits, since it involves "collapsing" the circuit into two levels. *Super-gates* have been proposed [2, 19] to reduce the

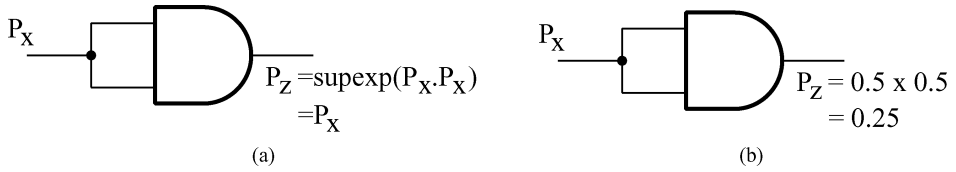

Figure 7. Handling spatial correlation. (a) Taking into account spatial correlation. (b) Ignoring spatial correlation.

size of the polynomials and still obtain an exact solution. We review this concept together with the more generic concept of *graph dominators* in this section.

5.1. Zero-Delay Model

In propagating signal probabilities through a logic circuit, spatial correlation measures how the probabilities of the inputs to a gate are related. In a logic circuit, this is determined by what primary inputs are common to the support (The *support* of a logic function is the set of primary inputs that the function depends on.) of the inputs to the gate. If the supports are disjoint, then the probabilities of the inputs are independent.

In the Parker-McCluskey method, spatial correlation is handled by the *supexp* operator (cf. Definition 1). All polynomials are a function of the primary inputs. When some gate depends on logic signals that share some primary inputs, the method is able to detect the common variables and the exponent is suppressed, as depicted in Figure 7(a).

The complexity of the polynomials can be reduced by substituting some variables by their probability values. This procedure reduces the number of variables in some terms of the polynomial, creating a constant factor for that term. For example, if we substitute the probability of x_1 in polynomial $x_1 \cdot x_3 + x_2 \cdot x_3$, we obtain the polynomial $\mathrm{pr}(x_1) \cdot x_3 + x_2 \cdot x_3$. If additionally we do the same for x_2, the polynomial becomes $\mathrm{pr}(x_1) \cdot x_3 + \mathrm{pr}(x_2) \cdot x_3 = k \cdot x_3$.

However, in this process we have lost information about the polynomial depending on the substituted variables. If these variables are present in any reconvergent path in the transitive fanout of the current gate, some error is introduced since the probability of the same variable will be multiplied, as in Figure 7(b). On the other hand, if we determine that some variable will not be present in any reconverging signal, then under a zero-delay model the method is still exact (this may not be true for a general delay model, which we analyze in the next section).

It is useful to introduce the concept of *graph dominator* [9].

Definition 2. A vertex v *dominates* another vertex $w \neq v$ in a directed graph G if every path from the root vertex to w contains v.

Thus, if we determine that a given gate g is the dominator of some primary input i as seen from a primary output, then we can substitute the probabilities corresponding to this input i in the polynomials at gate g. Under a zero-delay model no error is introduced since we know that no reconvergent signal in the transitive fanout of g will depend on i.

Super-gates have previously been proposed [2, 19] to reduce polynomial complexity. Super-gates are significantly more constrained in that they require the gate to be a dominator

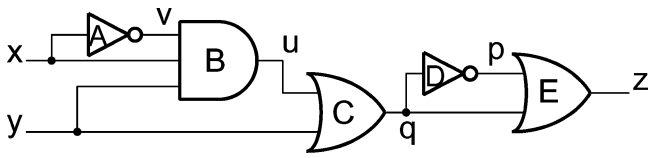

Figure 8. Variable substitution under a general-delay model.

for all the primary inputs in its support. However, when found, super-gates have the important property that the polynomials are reduced to the independent term (i.e., constants) and thus can be treated as primary inputs.

5.2. General-Delay Model

Under a general-delay model, substituting variables at dominator nodes is no longer an exact procedure. For every node in the circuit there will be a polynomial corresponding to each time point where the node can make a transition. These polynomials will necessarily be a function of some common variables. It is possible that in the transitive fanout of a node, polynomials corresponding to different time points are operated together. If a variable has been substituted by its probability value, an error will be introduced because correlation due to this variable has been ignored.

To illustrate this point, in the somewhat contrived circuit of Figure 8 node u is a dominator for node x. For simplicity assume a unit-delay model, although the following observations apply equally well to the general-delay model. At node u we have polynomials corresponding to instants 1 and 2, both a function of $P_x[0]$ and $P_y[0]$, respectively $P_u[1](P_x[0], P_y[0])$ and $P_u[2](P_x[0], P_y[0])$. If the variable $P_x[0]$ is replaced by its numerical value, thus obtaining $P'_u[1](P_y[0])$ and $P'_u[2](P_y[0])$, the *temporal* correlation between $P'_u[1]$ and $P'_u[2]$ due to x is lost. In this circuit, error will be introduced at node z where, because of the reconvergent path starting at node q, $P'_u[1](P_y[0])$ and $P'_u[2](P_y[0])$ will be operated with each other due to the different delays from q to z.

Also evident in the above example is the error introduced by super-gates. Gates A, B and C in Figure 8 form a super-gate. However, node q cannot be treated exactly like a primary input since there are three time instants at which q can make a transition. Further, if all variables are substituted, we lose all information about the correlation between these three instants.

6. Approximation Based on Limited Depth Spatial Correlation

It has been our experience that dominators (and consequently super-gates) are not very common in a general logic circuit [5]. In most circuits, due to a high degree of reconvergent paths, dominators of primary inputs exist only close to the primary outputs. This severely restricts their usefulness in the switching activity estimation process.

We describe a parameterizable approximation scheme, based on approximate dominators, that is able to handle large circuits and still obtain accurate estimates for power and switching activity.

6.1. Basis for the Approximation

One important observation behind our approach is that spatial correlation is more important if the reconvergence of paths happens within a few logic levels. Consider two paths starting at some primary input a that reconverge at some node b. The polynomials at the inputs b_1, b_2 of b will in general have some terms dependent on the polynomials at a and other terms independent of them,

$$P_{b_1} = \alpha + \beta P_a \quad P_{b_2} = \gamma + \delta P_a$$

where α, β, γ and δ are functions of other primary inputs.

When node b is close to a in terms of logic levels, most terms in P_{b_1} and P_{b_2} will contain P_a, thus $\alpha \ll \beta$ and $\gamma \ll \delta$. On the other hand, if b is at a high logic level, the fraction of terms that depend on P_a is smaller. Therefore, $\alpha \gg \beta$ and $\gamma \gg \delta$.

Substituting the probability value of a will always introduce some error because in generating the polynomial at b, the probability of a is squared when multiplying terms of b_1 and b_2 containing P_a. With no variable substitution (exact case) we compute,

$$P_{b_1} \times P_{b_2} = \alpha\gamma + (\alpha\delta + \beta\gamma + \beta\delta)P_a$$

When we substitute the value of P_a we obtain the polynomials,

$$P'_{b_1} = \alpha + \beta \mathrm{pr}(P_a)$$
$$P'_{b_2} = \gamma + \delta \mathrm{pr}(P_a)$$

and thus

$$P'_{b_1} \times P'_{b_2} = \alpha\gamma + (\alpha\delta + \beta\gamma)\,\mathrm{pr}\,(P_a) + \beta\delta(\mathrm{prob}(P_a))^2$$

Therefore, the error is only present in the last term ($\beta\delta$). For a low logic level of b ($\alpha \ll \beta$ and $\gamma \ll \delta$) the relative weight of this term may be large, leading to a high relative error. For a high logic level of b ($\alpha \gg \beta$ and $\gamma \gg \delta$), we have a smaller relative error.

In Figure 9 an example is shown to illustrate the relative error due to substituting the probability polynomial by its value in the calculations. We use static probabilities to simplify analysis. Being p_x the probability of signal x having the logical level 1 and assuming $p_{x_1} = p_{x_2} = p_{x_2} = p_{x_2} = 0.5$ we have

$$p_{w_1} = p_{x_2} - p_{x_1} p_{x_2}$$
$$p_{z_1} = p_{x_1} + p_{x_2} - p_{x_1} p_{x_2} - p_{x_1} p_{x_2} + p_{x_1} p_{x_1} p_{x_2}$$

Taking into account spatial correlation we have

$$\begin{aligned} p_{z_1} &= p_{x_1} + p_{x_2} - 2p_{x_1} p_{x_2} + p_{x_1} p_{x_2} \\ &= p_{x_1} + p_{x_2} - p_{x_1} p_{x_2} \\ &= 0.5 + 0.5 - 0.25 \\ &= 0.75 \end{aligned}$$

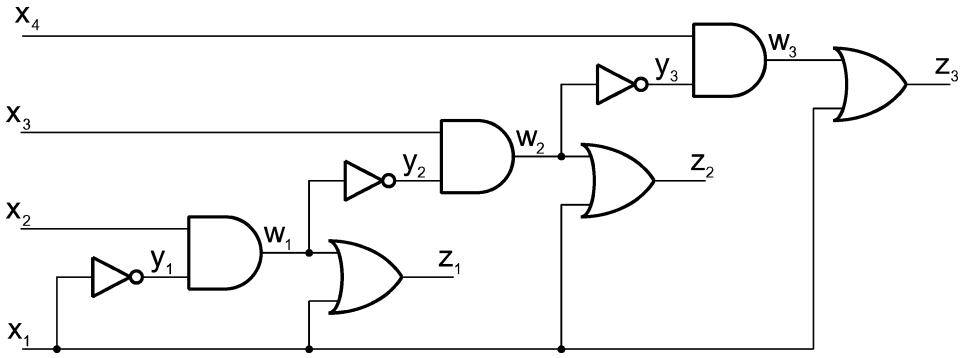

Figure 9. Example circuit to demonstrate approximation.

Not taking into account spatial correlation we have

$$\begin{aligned} p_{z_1} &= p_{x_1} + p_{x_2} - 2p_{x_1}p_{x_2} + p_{x_1}p_{x_1}p_{x_2} \\ &= 0.5 + 0.5 - 0.5 + 0.125 \\ &= 0.375 \end{aligned}$$

Doing the same for signals z_2 and z_3 we get

$$\begin{aligned} p_{z_2} &= p_{x_1} + p_{x_3} - p_{x_2}p_{x_3} + p_{x_1}p_{x_2}p_{x_3} - p_{x_1}p_{x_3} + p_{x_1}p_{x_2}p_{x_3} - p_{x_1}p_{x_1}p_{x_2}p_{x_3} \\ p_{z_3} &= p_{x_1} + p_{x_4} - p_{x_3}p_{x_4} + p_{x_2}p_{x_3}p_{x_4} - p_{x_1}p_{x_2}p_{x_3}p_{x_4} - p_{x_1}p_{x_4} + p_{x_1}p_{x_3}p_{x_4} \\ &\quad - p_{x_1}p_{x_2}p_{x_3}p_{x_4} + p_{x_1}p_{x_1}p_{x_2}p_{x_3}p_{x_4} \end{aligned}$$

Taking into account spatial correlation and replacing the probabilities by their values we have

$$\begin{aligned} p_{z_2} &= 0.625 \\ p_{z_3} &= 0.5625 \end{aligned}$$

Not taking into account spatial correlation we get

$$\begin{aligned} p_{z_2} &= 0.6875 \\ p_{z_3} &= 0.59375 \end{aligned}$$

The error ε in probability of z_1, z_2 and z_3 for not taking into account spatial correlation is

$$\begin{aligned} \varepsilon_{z_1} &= 0.125 \\ \varepsilon_{z_2} &= 0.0625 \\ \varepsilon_{z_3} &= 0.03125 \end{aligned}$$

It can be seen that the error introduced by not taking into account spatial correlation is decreasing to negligible values the farther away the reconvergence of paths happens.

6.2. Description of the Approximation Algorithm

In our approximation scheme the user specifies one parameter l. This parameter determines the depth in terms of logic levels from each node a that will be searched in order to determine if two paths starting at a will reconverge. Spatial correlation corresponding to two paths starting at a that reconvergence within l logic levels will be accurately taken into account. If reconvergent paths meet after l logic levels then they are assumed to be independent, thus the polynomials will be simplified by variable substitution and some error will be introduced.

The approximation algorithm is divided in two parts. We first determine the *active nodes* for each node in the circuit. Active nodes are nodes where two (or more) reconvergent paths begin and these nodes need to be active until the paths meet. These will be the variables in the polynomials at each node. In the second part of the algorithm, a polynomial simulation routine similar to the one described in Figure 6 is used. The difference is that the information about active nodes will be used to simplify the polynomials.

The pseudo-code for the algorithm that determines the active nodes is described in Figure 10. The algorithm works by taking each node g and doing a limited depth first search (DFS) of l levels for each fanout of g. While doing the DFS, we build a table that stores information about the h node found, a number j that identifies for which fanout the DFS is being done and the fanin of h. This fanin information will allow us to backtrack the path without doing another DFS, for the case when reconvergence is found. After all the fanouts are done, we go through the table to check which nodes in the table have two or more

```
Switching_Activity_Estimation ( Network, l )
1.  for each gate g in ( Network ) {
2.      Path = NIL(Table) ;
3.      for each fanout f_j of g {
4.          Gates = limited_depth_search ( f_j, l ) ;
5.          for each node h in Gates {
6.              Insert( Path, h, j, fanin(h) ) ;
7.          }
8.      }
9.      for each duplicate node h in Path with different indexes j {
10.         while fanin(h) ≠ g {
11.             Insert(fanin(h).Active_Nodes, g) ;
12.             h = fanin(h) ;
13.         }
14.     }
15. }
16. Polynomial_Sim_with_Variable_Substitution ( Network ) ;
```

Figure 10. Pseudo-code for the limited depth spatial correlation algorithm.

13. for each variable h in $P_{g_i}[k+\Delta]$ not in $g_i.Active\_Nodes$ {
14. substitute h with $pr(h)$ in $P_{g_i}[k+\Delta]$;
15. }
16. simplify ($P_{g_i}[k+\Delta]$) ;

Figure 11. Modification to the algorithm of Figure 10 in order to handle variable substitution.

different numbers of DFS, indicating that this is a node where reconvergent paths meet. We can now use the fanin information to go back in the path and in doing so inserting in the table of active nodes of each node in the path the node which is being processed.

After the active nodes for all nodes in the circuit have been computed, the modified polynomial simulation where variable substitution is done is called. The only difference from the algorithm of Figure 6 is that between lines 12 and 13 of Figure 6 we insert the code where variable substitution is done (Figure 11).

7. Sequential Power Estimation

In general, integrated circuits include some storage elements, such as registers, so they present a sequential behavior. A generic sequential circuit is depicted in Figure 12. Estimating power consumption of sequential circuit has the increased difficulty that the probability of the state lines has to be taken into account.

As in the case of combinational circuits, both simulation- and probabilistic-based methods have been proposed. All observations made to the circuit in Figure 12 are applicable to all kinds of sequential circuits, and also to the particular case of pipelined circuits.

7.1. Simulation-Based Sequential Power Estimation

Whereas in the case of combinational circuits, the number of input vectors required to guarantee some maximum error for the switching activity of every node in the circuit

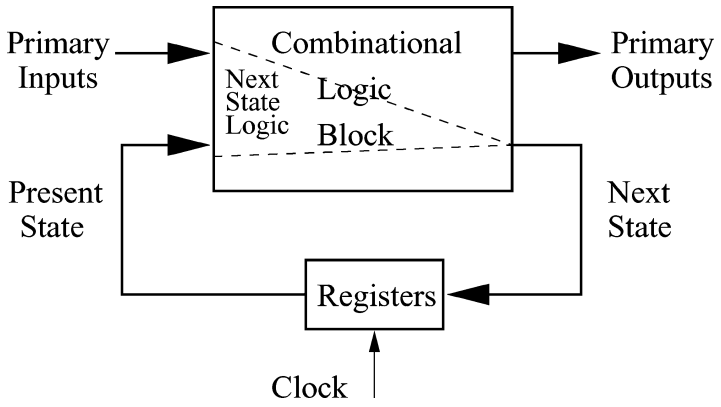

Figure 12. Generic sequential circuit.

can be computed, for sequential circuits that is substantially more difficult since one must guarantee that all the possible states have been visited *in a representative manner*. Basically this means that the fraction of the time that the circuit was in some state during the simulation process has to be proportional to the stationary probability of that state. This is a value that is not known beforehand, may not be possible to compute, and in any case, is what simulation-based methods want to avoid computing.

Still, some techniques have been proposed [3, 16]. The approach is to have more than one simulation in parallel and average among them, so that several different branches of state traversal can be followed. Also, the setup phase (initial part of the simulation during which transitions are not accounted) is larger to allow the circuit to be close to its stationary regime.

7.2. Probabilistic Sequential Power Estimation

For probabilistic methods, the additional difficulty of estimating the power dissipation of a sequential circuit is that the state line probabilities have to be computed beforehand. Once this has been done, the process is exactly the same as for combinational circuits, except that the state lines are now treated as primary inputs with probabilities given from the previous step.

In order to do this, one can set up and solve the Chapman-Kolmogorov system of equations [20]:

$$\begin{cases} p_{s_i} = \sum\limits_{\forall j : s_j \to s_i} p_{PI_{ji}} p_{s_j}, \quad 1 \le i \le K-1 \\ \sum\limits_{i=1}^{K} p_{s_i} = 1 \end{cases} \tag{2}$$

where p_{s_i} represents the stationary probability of state s_i and $p_{PI_{ji}}$ the probability of the input condition that triggers a transition from state s_j to state s_i.

The problem is that this approach requires that the state transition graph (STG) has to be extracted from the circuit. For a circuit with N registers, it is possible that the number of states is $K = 2^N$. Therefore, for the majority of the circuits of interest the STG cannot be obtained.

It has been proposed in [20] that the state *line* probabilities be computed instead of the state probabilities. The advantage is that in a circuit with N registers there are only N variables to compute (as opposed to 2^N). The disadvantage is that the spatial correlation among state lines is lost. However, it is shown in [20] that the error incurred with this approximation is less than 5% for all tested benchmark circuits.

Two different methods are proposed in [20]. The Picard-Peano method simply extracts the next state logic block (see Figure 12), builds BDDs for all state lines, propagates the input and state line probabilities using these BDDs to obtain new values for the state line probabilities (using 0.5 as the initial value), repeating this process until all the state line probabilities converge. The Newton-Raphson method also extracts the Hessian matrix from the previous BDDs. The Newton-Raphson method converges in fewer iterations, but each iteration takes longer than the Picard-Peano.

The method proposed in this paper is based on the Picard-Peano method. However, the method to compute the probabilities does not require BDDs, and therefore is applicable to larger circuits.

7.3. Sequential Power Estimation Based on Polynomials

The work proposed in this paper extends the method of polynomial simulation presented in Sections 4 and 6 to work with sequential circuits. The proposed method starts by computing the static state line probabilities of the next state logic block (see Figure 12). Using those static probabilities, the transition probabilities for the state lines are determined. The switching activity for all the nodes in the circuit is then computed. In these three steps polynomial simulation is used.

7.3.1. Static Probabilities of the State Lines

The static probabilities of the state lines are computed using the method of Picard-Peano, as described in Section 7.2. However, instead of using BDDs for this computation, the new method is based on probability polynomials. Considering that the polynomial simulation method described in the previous section uses transition probabilities, this algorithm has to be modified to handle static probabilities. The way to do this is to assign static probabilities to the inputs of the next state block. Hence, we have the following transition probabilities at the inputs: $p^{00} = p^0$, $p^{01} = p^{10} = 0$, and $p^{11} = p^1$. By assigning the value zero to p^{01} and p^{10}, one guarantees that what will be propagated are static probabilities, and the polynomials resulting from the propagation give the static probabilities. Those polynomials are propagated through the next state block the number of times necessary for the static probabilities of the state lines to converge.

7.3.2. Transition Probabilities of the State Lines

The methods described above can only compute the static state line probabilities. In order to account for temporal correlation, one has to use transition probabilities of the state lines. The transition probabilities consist of four values, p^{00}, p^{01}, p^{10} and p^{11}, which are respectively the probabilities of a signal staying at zero, making a low to high transition, a high to low transition and staying at one. In the case of the next sate logic block, the probability of a signal staying at zero, p^{00}, for example, is the product $(1 - p_{PS})(1 - p_{NS})$, which is equal to say that is the probability of the next state signal being zero knowing that the present state signal is zero.

7.3.3. Algorithm

The algorithm for switching activity estimation using polynomials in sequential circuits is presented in Figure 13. It starts by computing the static probabilities of the state lines using the method by Picard-Peano. In computing the static probabilities of the state lines

```
1.  nsLogic = Get_Next_State_Logic(Network);
2.  Initialize p^1_{NS_i} = 0.5 ∀i in nsLogic;
3.  iter = 0;
4.  do {
5.     p^1_{PS_i} = p^1_{NS_i} ∀i;
6.     Polynomial_Simulation(nsLogic);
7.     iter++;
8.  } while (∃i: p^1_{NS_i} − p^1_{PS_i} > ε)
9.  for PS_i ∈ {present state lines in nsLogic} {
10.    PS_i = 0;
11.    Compute p_{NS_i}
12.    p^{00}_{NS_i} = (1 − p^1_{PS_i}) · (1 − p^1_{NS_i})
13.    p^{01}_{NS_i} = (1 − p^1_{PS_i}) · p^1_{NS_i}
14.    PS_i = 1;
15.    Compute p_{NS_i}
16.    p^{10}_{NS_i} = p^1_{PS_i} · (1 − p^1_{NS_i})
17.    p^{11}_{NS_i} = p^1_{PS_i} · p^1_{NS_i}
18. }
19. Polynomial_Simulation(Network);
```

Figure 13. Pseudocode for the approximate sequential power estimation.

only the next state logic block is used. The next state logic is obtained by deleting in the original circuit the nodes that are not part of the next state lines support. The iterations start by initializing the present state line probabilities. The probability polynomials are then propagated through the next state logic. The resulting probabilities in the next state lines are then compared with the values in the present state lines. The algorithm will iterate until the difference between all corresponding present and next state lines is below ϵ.

The computation of the transition probabilities is the next step in the algorithm. For each state line, i, two passes are made through the next state logic block. In the first, the input PS_i is set to zero and the other input probabilities have the values that resulted from the first part of the algorithm. The probabilities are propagated from the inputs to the outputs of the next sate logic block and the value of the static probability for NS_i, p_{NS_i}, is obtained. With p_{NS_i} we can get the transition probabilities $p^{00}_{NS_i} = (1 - p_{PS_i})(1 - p_{NS_i})$ and $p^{01}_{NS_i} = (1 - p_{PS_i})p_{NS_i}$ A similar computation is done to get the values p^{10}_{NS} and p^{11}_{NS}. In this case the input PS_i is set to one and after obtaining the value p_{NS_i} the remaining transition probabilities for NS_i, $p^{10}_{NS} = p_{PS_i}(1 - p_{NS_i})$ and $p^{11}_{NS} = p_{PS_i}\, p_{NS_i}$.

The resulting transition probabilities are then applied at the primary inputs of the circuit. Finally, the probability polynomials are propagated through the original circuit, thus computing the switching activity of the circuit.

Table 2. Statistics of all Combinational Circuits

Circuit	pi	po	Nodes	Lits	Levels	Circuit	pi	po	Nodes	Lits	Levels
9symml	9	1	99	267	12	decod	5	16	28	54	3
C1355	41	32	514	1032	23	des	256	245	681	6101	4
C17	5	2	6	12	3	example2	85	66	159	400	8
C1908	33	25	880	1497	40	f51m	8	8	51	125	23
C2670	233	140	1161	2043	32	frg1	28	3	82	204	10
C3540	50	22	1667	2934	47	frg2	143	139	522	2010	8
C432	36	7	106	259	23	i1	25	16	24	57	6
C499	41	32	208	598	15	i10	257	224	2497	5376	54
C5315	178	123	2290	4369	49	i2	201	1	77	289	8
C6288	32	32	2416	4800	124	i3	132	6	46	172	3
C7552	207	108	3466	6098	43	i4	192	6	110	356	6
C880	60	26	190	526	25	i5	133	66	161	293	12
add_cla_16	32	16	214	357	23	i6	138	67	318	699	5
add_rpl_16	32	16	214	350	35	i7	199	67	406	912	6
alu2	10	6	198	501	25	i8	133	81	1183	4626	8
alu4	14	8	378	943	33	i9	88	63	353	1453	7
apex6	135	99	411	1007	17	k2	45	45	225	2928	2
apex7	49	37	137	316	15	lal	26	19	57	132	10
b1	3	4	5	12	3	majority	5	1	8	17	4
b9	41	21	76	170	8	mult8	16	16	176	896	21
c8	28	18	61	166	8	mux	21	1	23	65	7
cbp.32.4	65	33	489	825	84	my_adder	33	17	112	241	35
cbp.64.4	129	65	977	1649	164	pair	173	137	877	2195	25
cc	21	20	40	79	6	pcle	19	9	41	96	9
cht	47	36	93	213	5	pcler8	27	17	66	137	9
cm138a	6	8	14	34	4	pm1	16	13	30	63	6
cm150a	21	1	22	64	7	rot	135	107	390	901	21
cm151a	12	2	12	34	6	sct	19	15	42	99	25
cm152a	11	1	9	30	4	t481	16	1	374	932	17
cm162a	14	5	29	66	7	tcon	17	16	8	32	1
cm163a	16	5	21	57	10	term1	34	10	97	229	10
cm42a	4	10	17	38	3	too_large	38	3	165	401	14
cm82a	5	3	11	29	5	ttt2	24	21	123	294	14
cm85a	11	3	29	69	6	unreg	36	16	67	150	5
cmb	16	4	30	75	5	vda	17	39	304	754	14
comp	32	3	65	148	11	×1	51	35	174	421	10
comp16	35	3	48	240	16	×2	10	7	29	71	6
cordic	23	2	28	82	6	×3	135	99	332	1345	9
count	35	16	80	194	18	×4	94	71	211	493	16
cu	14	11	32	79	5	z4ml	7	4	23	57	9
dalu	75	16	488	1271	21						

8. Experimental Results

This section presents the results obtained using the approximation algorithm based on limited depth reconvergent path analysis. Power and switiching activity estimation results are presented for different values of l and are compared with the value obtained with symbolic simulation [20]. Results are presented for both combinational and sequential circuits. In this section we also present statistics on the number of active nodes for each

Table 3. Statistics of Active Nodes Found per Circuit Node

Circuit name	pi	l = 2			l = 3			l = 4			l = ∞		
		Active nodes		CPU	Active nodes		CPU	Active nodes		CPU	Active nodes		CPU
		Avg	Max	(s)	Avg	Max	(s)	Avg	Max	(s)	Avg	Max	(s)
C4 99	41	1.05	2	0.5	1.08	2	0.5	1.13	2	0.6	5.53	33	5.4
C5315	178	1.12	3	2.6	1.33	4	2.9	1.43	9	3.3		N/A	
C6288	32	1.19	2	2.6	1.57	2	2.9	3.41	20	3.6		N/A	
C7552	207	1.05	3	3.8	1.36	9	4.3	1.62	27	5.0		N/A	
C880	60	1.12	3	0.5	2.84	16	0.6	5.66	24	0.7	7.76	29	5288.4
add_rpl_16	32	1.00	1	0.2	1.25	2	0.2	1.42	2	0.2	1.42	2	0.7
cm151a	12	1.00	1	0.1	1.08	2	0.1	1.08	2	0.1	1.08	2	0.1
cm163a	16	1.00	1	0.0	1.00	1	0.1	1.00	1	0.0	1.00	1	0.0
des	256	2.89	6	11.9	2.89	6	11.9	2.89	6	12.6	2.89	6	13.1
f51m	8	1.07	2	0.1	2.20	6	0.1	3.46	8	0.1	3.61	8	3.3
i9	88	1.57	13	1.1	5.95	9	2.8	5.95	9	5.9	5.95	9	14.2
my_adder	33	1.12	2	0.1	1.66	3	0.2	2.08	3	0.2	2.08	3	0.4
rot	135	1.05	3	0.9	1.31	5	0.9	1.87	10	1.0	4.14	39	2.4
×4	94	1.03	3	0.6	1.19	5	0.7	1.61	6	0.7	2.23	9	0.8
z4ml	7	1.07	2	0.1	1.70	5	0.1	1.87	5	0.1	1.90	5	0.1

Table 4. Power Estimation Results for Combinational Circuits

Circuit name	Symbolic		$l = 0$			$l = 1$			$l = 2$		
	P	**CPU (s)**	**P**	**%**	**CPU (s)**	**P**	**%**	**CPU (s)**	**P**	**%**	**CPU (s)**
C499	N/A		2559.4	–	14.6	3298.6	–	17.1	3910.5	–	17.4
C5315	N/A		21553.2	–	150.3	21553.2	–	200.1	21185.3	–	239.1
C6288	N/A		572925.4	–	519.4	552937.7	–	878.5	249790.2	–	1379.6
C7552	N/A		30676.3	–	219.3	30676.3	–	280.3	31213.9	–	351.6
C880	N/A		2486.8	–	20.9	2741	–	34.3	2376	–	53.8
add_rpl_16	1440.3	24.7	1108	23.1	11.0	1108	23.1	11.9	1108	23.1	11.8
cm151a	200.2	1.0	181.2	9.5	1.4	188.5	5.8	1.5	199.6	0.3	1.6
cm163a	291.9	1.6	244.9	16.1	1.8	284.7	2.5	1.9	293	0.4	1.9
des	30623.6	815.7	29554.5	3.5	106.9	30010.8	2.0	1819.7		N/A	
f51m	906.2	15.2	708.4	21.8	5.5	728.2	19.6	8.0	826.2	8.8	9.8
i9	11309.7	257.0	12046.5	6.5	48.6	12046.5	6.5	103.0	12223.4	8.1	261.4
my_adder	1026.7	17.8	1055.3	2.8	7.0	1053.3	2.6	8.2	1052.6	2.5	10.6
rot	N/A		3628.5	–	25.5	3588.5	–	31.9	3641.5	–	45.0
×4	1870.5	22.4	1803.2	3.6	13.2	1833.5	2.0	15.1	1818.5	2.8	15.1
z4ml	319.1	2.0	286.3	10.3	2.0	299.1	6.3	2.2	316.8	0.7	2.2
			max = 23.1, avg = 2.8			max = 23.1, avg = 2			max = 23.1, avg = 1.4		

Table 5. Power Estimation Results for Combinational Circuits (Cont.)

Circuit name	Symbolic		$l = 3$			$l = 4$			$l = \infty$		
	P	**CPU (s)**	**P**	**%**	**CPU (s)**	**P**	**%**	**CPU (s)**	**P**	**%**	**CPU (s)**
C499		N/A	3925.8	–	18.2	3456.9	–	20.7		N/A	
C5315		N/A	28184.7	–	233.2		N/A			N/A	
C6288		N/A	13371	–	1805.2		N/A			N/A	
C7552		N/A	41440.5	–	908.1		N/A			N/A	
C880		N/A		N/A			N/A			N/A	
add_rpl_16	1440.3	24.7	1282.2	11.0	12.0	1436.9	0.2	12.3	1436.9	0.2	12.8
cm151a	200.2	1.0	199.5	0.4	3.1	199.5	0.4	3.1	199.5	0.4	3.1
cm163a	291.9	1.6	293	0.4	1.9	293	0.4	1.9	293	0.4	1.9
des	30623.6	815.7		N/A			N/A			N/A	
f51m	906.2	15.2		N/A		897	1.0	730.3	906.2	0.0	357.8
i9	11309.7	257.0		N/A			N/A			N/A	
my_adder	1026.7	17.8	1025.5	0.1	16.4	1023.7	0.3	16.0	1023.7	0.3	15.6
rot		N/A	3564.3	–	1442.7		N/A			N/A	
×4	1870.5	22.4	1822.2	2.6	18.9	1831.4	2.1	182.7	1849.2	1.1	313.8
z4ml	319.1	2.0	275.7	13.6	4.9	273.9	14.2	4.6	273.2	14.4	3.8
			max = 13.6, avg = 0.9			max = 14.2, avg = 0.7			max = 14.4, avg = 0.5		

circuit. All reported CPU times are in seconds and were obtained on a Sparc Ultra I running at 170 MHz with 384 M of main memory.

8.1. Combinational Power Estimation

The benchmark circuits used in combinational power estimation are presented in Table 2 with statistics of the number of primary inputs (pi), primary outputs (po), number of nodes (nodes), number of literals (lits) and number of levels (levels) for each circuit. In the following tables only the results for some relevant circuits are presented here due to the lack of space.

The first part of the approximation algorithm involves computing for each node in the circuit the set of active nodes, i.e., the variables whose polynomials at each node will be a function of. We present statistics on the number of active nodes for our benchmark circuits in Table 3. For different values of l, we give the average (Avg) and maximum (Max) number of active nodes over all nodes in each circuit. ∞ corresponds to the maximum number of logic levels in the circuit, thus detecting all reconvergent paths. The second column shows

Table 6. Number of Combinational Circuits for Which it was not Possible to Obtain Results of a Total of 79 Circuits

Symbolic	$l = 0$	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = \infty$
18	0	1	3	18	26	30

Table 7. Switching Activity Errors for Combinational Circuits

Circuit name	$l = 0$		$l = 1$		$l = 2$		$l = 3$		$l = 4$		$l = \infty$	
	Max	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**
C499	–		–		–		–		–		–	
C5315	–		–		–		–		–		–	
C6288	–		–		–		–		–		–	
C7552	–		–		–		–		–		–	
C880	–		–		–		–		–		–	
add_rpl_16	0.97	0.23	0.97	0.23	0.97	0.23	0.4	0.1	0.06	0.01	0.06	0.01
cm151a	0.56	0.09	0.3	0.05	0.07	0.01	0.08	0.01	0.08	0.01	0.08	0.01
cm163a	0.53	0.1	0.2	0.03	0.11	0.01	0.11	0.01	0.11	0.01	0.11	0.01
des	0.82	0.04	0.26	0.01	N/A		N/A		N/A		N/A	
f51m	1.05	0.27	0.87	0.25	0.56	0.12	N/A		0.18	0.02	0	0
i9	0.25	0.09	0.25	0.09	0.3	0.1	N/A		N/A		N/A	
my_adder	0.25	0.05	0.25	0.05	0.21	0.05	0.13	0.01	0.08	0.01	0.08	0.01
rot	–		–		–		–		–		–	
×4	0.32	0.04	0.38	0.02	0.64	0.02	0.52	0.02	0.43	0.02	0.43	0.01
z4ml	0.43	0.08	0.44	0.06	0.39	0.02	0.65	0.11	0.64	0.11	0.64	0.12
Max	1.14	0.27	1.21	0.25	1.44	0.23	0.65	0.11	0.64	0.11	0.64	0.12
Avg	0.3	0.04	0.24	0.03	0.21	0.02	0.11	0.01	0.08	0.01	0.06	0

the number of primary inputs in each circuit. As expected, as l increases, both the average and maximum values increase. To a small number of circuits the opposite can happen, as in circuit $i9$. This is due to the fact that in those circuits there is a large number of reconvergent paths for a low value of l but for a larger l that number decreases. An interesting observation is that, even for large values of l, the average number of active nodes is relatively small. Yet, the maximum number can be large. We do not show statistics for $l = 0$ and $l = 1$ because in those cases the number of active nodes for all the nodes will be zero and one, respectively.

The CPU time we present in this table corresponds only to the algorithm that computes the active nodes (cf. Figure 10). As it can be seen from the table, for $l = 2$ and $l = 3$, the time spent in doing the depth search for reconvergent paths is very small, typically less than 1s. Even for $l = \infty$ we can still execute this operation using small amounts of CPU time.

Tables 4 and 5 presents the power estimation results obtained with the approximation algorithm using l equal to 0, 1, 2, 3, 4 and ∞. A general delay model was used for all the

Table 8. Statistics of all Sequential Circuits

Circuit	pi	po	Nodes	latches	Lits	Levels
s1196	14	14	310	18	819	16
s1238	14	14	329	18	868	16
s1269	18	10	259	37	719	30
s1423	17	5	374	74	880	43
s1512	29	21	309	57	756	18
s208	10	1	41	8	107	8
s27	4	1	8	3	16	4
s298	3	6	64	14	157	8
s3271	26	14	690	116	1773	23
s3330	40	73	423	65	1024	16
s344	9	11	82	15	192	13
s349	9	11	80	15	193	12
s382	3	6	87	21	211	9
s386	7	7	67	6	176	6
s400	3	6	99	21	230	9
s420	18	1	84	16	215	16
s444	3	6	94	21	218	9
s499	1	22	116	22	308	11
s510	19	7	136	6	370	7
s526	3	6	110	21	263	9
s635	2	1	111	32	303	32
s641	35	24	101	17	237	22
s6669	83	55	1469	231	3667	68
s713	35	23	105	17	243	22
s820	18	19	171	5	430	12
s832	18	19	177	5	451	11
s838	34	1	178	32	463	25
s938	34	1	178	32	463	25
s953	16	23	229	29	592	11
s967	16	23	233	29	596	10
s991	65	17	193	19	528	37

Table 9. State Lines Static Probability Errors

Circuit	$l = 0$	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = \infty$
s1196	0.028	0.015	0.008	0.007	0.003	0.003
s1238	0.026	0.017	0.008	0.003	0.002	0.001
s208	0.096	0.014	0	0	0	0
s27	0.007	0.007	0.007	0	0	0
s298	0.054	0.038	0.007	0.024	0.024	0.024
s3330	0.009	0.007	0.006	0.002	0.002	0.001
s344	0.053	0.029	0.028	0.013	0.013	0.013
s349	0.08	0.043	0.054	0.042	0.039	0.039
s382	0.014	0.002	0	0	0	0
s386	0.021	0.021	0.019	0.001	0.001	0.001
s400	0.014	0.002	0	0	0	0
s420	0.052	0	0	0	0	0
s444	0.014	0.002	0	0	0	0
s499	0.001	0.001	0.001	0.001	0.001	0
s510	0.059	0.043	0.035	0.029	0.021	0.012
s526	0.007	0.004	0	0	0	0
s635	0.001	0	0	0	0	0
s641	0.02	0.019	0.009	0.01	0.009	0.008
s713	0.02	0.02	0.008	0.011	0.01	0.009
s820	0.017	0.011	0.007	0.003	0.001	0.001
s832	0.017	0.01	0.008	0.004	0.001	0.001
s838	0.026	0.004	0	0	0	0
s938	0.026	0.004	0	0	0	0
s953	0.112	0.123	0.119	0.111	0	0
s967	0.071	0.067	0.036	0.012	0.002	0.001
s991	0.113	0.045	0.004	0.012	0.002	0.001

examples and a supply voltage of 5 V and clock frequency of 20 MHz was assumed. A probability of 0.25 was used for all primary input events.

The two columns under "Symbolic" show the power (in μW) computed using the symbolic simulation method of [12] and the CPU time (in seconds) taken by this computation. For some of the circuits, this method runs out of memory and this is indicated with a "N/A" in the table. In the columns under "$l = 0$" are the results for the approximation algorithm using $l = 0$. Again we show the power dissipation results and the CPU time for this method. Under "%" is the percentage error of the power estimation relative to the symbolic method. Similarly for the columns under "$l = 1$", "$l = 2$", "$l = 3$", "$l = 4$" and "$l = \infty$".

At the bottom of the table, we give the maximum and average error over all the circuits presented in Table 2, for each value of l. We can observe that the average error decreases with l. The maximum error can be very large but that is due to only two circuits of all the circuits tested. And that can be seen by the fact that the average error is very low, even for a low value of l. In fact, for $l = 2$ the error is typically below 5%, and in many cases below 1%.

Note that the error for $l = \infty$ is not zero. This is due to the temporal correlation effects described in Section 5.2. If a zero-delay model is used, $l = \infty$ gives exactly the same results as the symbolic simulation method. The computation times are equivalent to those obtained with the symbolic method. For several circuits, when using the symbolic method, it was not

Table 10. Power Estimation Results for Sequential Circuits

Circuit name	Symbolic		$l = 0$			$l = 1$			$l = 2$		
	P	CPU (s)	P	%	CPU (s)	P	%	CPU (s)	P	%	CPU (s)
s1196	2971.3	186.6	2898.3	2.5	26.6	2911.4	2.0	34.0	2924.1	1.6	42.2
s1238	2991.5	152.7	2957.8	1.1	28.4	2957.5	1.1	35.7	2961	1.0	38.1
s1269		N/A	4798.7	–	66.7	4170.2	–	137.0	4907.1	–	483.5
s1423		N/A	2039.4	–	78.3	1837.4	–	114.1	1852.4	–	126.3
s1512		N/A	1603.3	–	611.3	1364.7	–	421.9	1346.2	–	490.8
s208	236.5	3.1	399.7	69.0	23.2	196.8	16.8	11.4	202.7	14.3	3.4
s27	61.3	0.6	73.6	20.0	1.1	66.4	8.3	1.1	66.4	8.3	1.2
s298	325.6	5.3	421.9	29.6	8.5	360	10.6	10.2	335.3	3.0	9.1
s3271		N/A	10284.8	–	120.4	13172.9	–	203.2	14374.5	–	229.7
s3330	3221.1	463.6	3371.4	4.7	376.4	3279.3	1.8	507.6	3293.4	2.2	520.3
s344	585.1	16.1	640.7	9.5	8.5	632.8	8.1	10.3	620.8	6.1	12.7
s349	624.7	15.3	719.9	15.2	9.0	702.5	12.5	10.7	675.4	8.1	15.7
s382	333.9	8.2	327.3	2.0	12.6	313.5	6.1	12.3	316.4	5.2	13.0
s386	454.1	6.0	426.1	6.2	8.0	437.8	3.6	9.0	431.9	4.9	8.4
s400	345.2	9.3	346.4	0.3	15.6	324.5	6.0	16.9	323.9	6.2	14.0
s420	327.1	7.6	744.9	127.7	48.7	279.1	14.7	7.1	284.6	13.0	7.2

(*Continued on next page*.)

Table 10. (*Continued*).

Circuit name	Symbolic		$l = 0$			$l = 1$			$l = 2$		
	P	CPU (s)	P	%	CPU (s)	P	%	CPU (s)	P	%	CPU (s)
s444	323.6	8.2	323.2	0.1	13.3	302.9	6.4	15.9	302.3	6.6	13.0
s499	375.3	15.4	375.2	0.0	12.3	390.2	4.0	15.1	389.7	3.8	15.4
s510	926.7	13.1	1391.4	50.1	19.3	951.8	2.7	20.8	952.4	2.8	21.5
s526	280.8	10.1	280.6	0.1	13.6	267.7	4.7	16.1	263.6	6.1	15.9
s635	178.2	23.9	179.7	0.8	14.1	173	2.9	17.8	173.7	2.5	18.0
s641	649.2	190.8	707.7	9.0	12.2	670.1	3.2	16.2	670.5	3.3	17.4
s6669		N/A	33883.8	–	369.3	32708.1	–	808.0	39822.6	–	904.1
s713	694.1	169.8	742.5	7.0	12.4	703.3	1.3	16.6	711.3	2.5	17.6
s820	916.6	20.7	962.7	5.0	16.8	924.5	0.9	19.5	911.8	0.5	19.4
s832	921.0	22.1	986.8	7.1	18.6	930.5	1.0	20.8	931.9	1.2	21.4
s838	483.1	20.5	1591.5	229.4	107.7	434.8	10.0	81.9	457.2	5.4	17.7
s938	483.1	20.4	1591.5	229.4	109.2	434.8	10.0	83.6	457.2	5.4	17.8
s953	1204.2	45.3	1692.9	40.6	34.0	1473.1	22.3	39.5	1466.1	21.7	44.4
s967	1161.3	43.8	1530.1	31.8	48.4	1348.5	16.1	115.0	1306.4	12.5	92.0
s991	4034.0	174.7	2956	26.7	28.8	3854.4	4.5	44.0	3758.2	6.8	47.1
			max = 229.4, avg = 29.8			max = 22.3, avg = 5.9			max = 21.7, avg = 5		

Table 11. Power Estimation Results for Sequential Circuits

Circuit name	Symbolic		$l = 3$			$l = 4$			$l = \infty$		
	P	**CPU (s)**	**P**	**%**	**CPU (s)**	**P**	**%**	**CPU (s)**	**P**	**%**	**CPU (s)**
s1196	2971.3	186.6	2926.3	1.5	390.2		N/A			N/A	
s1238	2991.5	152.7	2970.9	0.7	198.6		N/A			N/A	
s1269		N/A		N/A			N/A			N/A	
s1423		N/A	1764.2	–	247.8		N/A			N/A	
s1512		N/A		N/A			N/A			N/A	
s208	236.5	3.1	202.7	14.3	3.4	202.8	14.2	3.5	203.2	14.1	58.6
s27	61.3	0.6	66.4	8.3	1.2	66.4	8.3	1.2	66.4	8.3	1.2
s298	325.6	5.3	361.4	11.0	10.9	359.9	10.5	17.6	359.9	10.5	16.9
s3271		N/A		N/A			N/A			N/A	
s3330	3221.1	463.6	3242.1	0.7	648.7		N/A			N/A	
s344	585.1	16.1	643.7	10.0	64.0	644	10.1	41.0	643.5	10.0	60.8
s349	624.7	15.3	692.6	10.9	840.4	698.2	11.8	54.8	698.5	11.8	64.0
s382	333.9	8.2	315.6	5.5	20.9	315.6	5.5	21.3	315.6	5.5	20.5
s386	454.1	6.0	430.7	5.2	102.0	427.2	5.9	15.6	427.1	5.9	16.6

(*Continued on next page.*)

Table 11. (*Continued*).

Circuit name	Symbolic		$l = 3$			$l = 4$			$l = \infty$		
	P	**CPU (s)**	**P**	**%**	**CPU (s)**	**P**	**%**	**CPU (s)**	**P**	**%**	**CPU (s)**
s400	345.2	9.3	323.2	6.4	68.7	323.2	6.4	20.2	323.2	6.4	20.0
s420	327.1	7.6	284.6	13.0	7.2	284.6	13.0	7.3	284.6	13.0	249.0
s444	323.6	8.2	301.6	6.8	19.9	301.6	6.8	18.5	301.6	6.8	20.3
s499	375.3	15.4	389.5	3.8	23.5		N/A		389.7	3.8	204.1
s510	926.7	13.1	948.9	2.4	87.4	999.2	7.8	179.8	1003.2	8.3	158.0
s526	280.8	10.1	263.9	6.0	23.4	263.9	6.0	26.3	263.9	6.0	31.7
s635	178.2	23.9	173.7	2.5	18.3	173.7	2.5	18.5	173.7	2.5	171.7
s641	649.2	190.8	661.7	1.9	18.4	656.1	1.1	1154.8		N/A	
s6669		N/A		N/A			N/A			N/A	
s713	694.1	169.8	708.4	2.1	19.0	701.6	1.1	668.3		N/A	
s820	916.6	20.7	911.7	0.5	80.3	918.1	0.2	466.8	920	0.4	1848.4
s832	921.0	22.1	932.8	1.3	80.8	926.9	0.6	667.6	927.8	0.7	698.5
s838	483.1	20.5	455.3	5.8	21.6		N/A			N/A	
s938	483.1	20.4	455.3	5.8	21.9		N/A			N/A	
s953	1204.2	45.3	1441.1	19.7	548.3		N/A		1230.9	2.2	1585.2
s967	1161.3	43.8	1205.6	3.8	5284.2		N/A		1205.8	3.8	1701.5
s991	4034.0	174.7		N/A			N/A			N/A	
			max = 19.7, avg = 5.8			max = 14.2, avg = 6.6			max = 14.1, avg = 6.7		

Table 12. Switching Activity Errors for Sequential Circuits

Circuit name	$l = 0$		$l = 1$		$l = 2$		$l = 3$		$l = 4$		$l = \infty$	
	Max	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**
s1196	0.31	0.03	0.36	0.02	0.32	0.02	0.17	0.01	N/A		N/A	
s1238	0.38	0.02	0.4	0.02	0.31	0.02	0.1	0.01	N/A		N/A	
s1269	–		–		–		–		–		–	
s1423	–		–		–		–		–		–	
s1512	–		–		–		–		–		–	
s208	0.49	0.17	0.25	0.06	0.25	0.05	0.25	0.05	0.25	0.05	0.25	0.05
s27	0.16	0.07	0.12	0.04	0.12	0.04	0.12	0.04	0.12	0.04	0.12	0.04
s298	0.31	0.1	0.2	0.04	0.19	0.02	0.22	0.04	0.22	0.04	0.22	0.04
s3271	–		–		–		–		–		–	
s3330	0.41	0.04	0.42	0.03	0.43	0.03	0.38	0.02	N/A		N/A	
s344	0.26	0.06	0.21	0.05	0.17	0.04	0.2	0.04	0.2	0.05	0.2	0.04
s349	0.31	0.09	0.21	0.05	0.19	0.04	0.2	0.05	0.2	0.05	0.21	0.05
s382	0.2	0.03	0.19	0.02	0.18	0.02	0.18	0.02	0.18	0.02	0.18	0.02
s386	0.3	0.03	0.14	0.02	0.14	0.02	0.26	0.02	0.3	0.02	0.3	0.02
s400	0.2	0.03	0.19	0.02	0.26	0.02	0.26	0.02	0.26	0.02	0.26	0.02
s420	0.5	0.21	0.25	0.04	0.25	0.03	0.25	0.03	0.25	0.03	0.25	0.03
s444	0.2	0.03	0.19	0.02	0.26	0.02	0.26	0.02	0.26	0.02	0.26	0.02

(*Continued on next page*.)

Table 12. (*Continued*).

Circuit name	$l = 0$		$l = 1$		$l = 2$		$l = 3$		$l = 4$		$l = \infty$	
	Max	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**	**Max**	**Avg**
s499	0.7	0.03	0.68	0.03	0.68	0.03	0.68	0.03	N/A		0.67	0.03
s510	0.5	0.13	0.22	0.04	0.2	0.03	0.14	0.03	0.19	0.03	0.19	0.03
s526	0.27	0.03	0.17	0.01	0.17	0.01	0.17	0.01	0.17	0.01	0.17	0.01
s635	0.1	0	0.11	0	0.1	0	0.1	0	0.1	0	0.1	0
s641	0.29	0.03	0.24	0.02	0.34	0.03	0.32	0.03	0.32	0.02	N/A	
s6669	–		–		–		–		–		–	
s713	0.26	0.03	0.23	0.03	0.27	0.02	0.27	0.02	0.28	0.02	N/A	
s820	0.28	0.03	0.14	0.02	0.14	0.01	0.14	0.01	0.09	0.01	0.09	0.01
s832	0.4	0.03	0.14	0.02	0.14	0.02	0.26	0.02	0.1	0.01	0.1	0.01
s838	0.5	0.22	0.25	0.02	0.25	0.01	0.25	0.01	N/A		N/A	
s938	0.5	0.22	0.25	0.02	0.25	0.01	0.25	0.01	N/A		N/A	
s953	0.98	0.15	0.66	0.12	0.65	0.12	0.58	0.11	N/A		0.34	0.03
s967	0.61	0.08	0.49	0.05	0.47	0.04	0.3	0.04	N/A		0.34	0.03
s991	1.31	0.24	0.8	0.1	0.7	0.11	N/A		N/A		N/A	
Max	1.31	0.24	0.8	0.12	0.7	0.12	0.68	0.11	0.32	0.05	0.67	0.05
Avg	0.41	0.08	0.29	0.04	0.29	0.03	0.25	0.03	0.21	0.03	0.24	0.03

possible to obtain results due to lack of time or memory. The number of circuits for which it was not possible to obtain results was larger using the symbolic method than using the approximation method with $l = 0, 1, 2$. That can be seen in Table 6.

For many applications, a more relevant measure of accuracy is the error in the switching activities of individual signals. In Table 7 we present the maximum and average error for the switching activity estimation over all the signals of each circuit. The average was computed by summing the absolute value of the switching probability error relative to the symbolic simulation method for all signals and dividing by the total number of signals. At the bottom of the table we have the average and maximum of the values for each column.

We can see that the average switching activity error is again very low even for low values of l and that it reduces as l increases. However, for low values of l, switching activity values for some of the nodes may present significant errors. Yet, since the average error is low, the number of nodes with high error is clearly small. Also note that the maximum error can be large even for $l = \infty$, indicating that most of the error is caused by ignoring temporal correlation.

8.2. Power Estimation in Sequential Circuits

This section presents power and switching activity estimation results obtained using the approximation algorithm based on limited depth reconvergent path analysis applied to sequential circuits. Results are presented for different values of l and are compared with the value obtained with the method of [20].

The benchmark circuits used in sequential power estimation are presented in Table 8 with statistics of the number of primary inputs (pi), primary outputs (po), number of nodes (nodes), number of latches (latches), number of literals (lits) and number of levels (levels) for each circuit.

The results presented in Table 9 simply compare the static probabilities of the state lines obtained with the approximate polynomial simulation and the exact method using BDDs. The table shows the average of the difference of the probabilities for each state line. Values for l equal to 0, 1, 2, 3 and ∞ are presented. As it can be observed, the probability errors are very low, and reduce significantly with l.

Tables 10 and 11 present the power estimation results obtained with the approximation algorithm, the % difference to the method of [20] and the CPU time (in seconds) taken by this computation. A general delay model was used for all the examples and a supply voltage of 5 V and clock frequency of 20 mHz was assumed. A probability of 0.25 was used for all primary input events. An "N/A" entry indicates that either the memory or CPU time limits were exceeded. At the bottom of the table is indicated the maximum and average error over all the circuits, for each value of l. The average error is relative to only those circuits for which there are results for all values of l. It can be observed that, with the exception of $l = \infty$, these values decrease with l. For $l = \infty$, the average value increases but that is due to only two circuits. For the rest of the circuits, the error for $l = 3$ is equal to $l = \infty$, which means that for these circuits there is no advantage in taking into account correlation for paths that reconverge after three or more logic levels. For $l = 0$, we can see that the confidence in the results is very low. But the error decreases significantly for $l = 1$. Note

that for $l = 2$ results were obtained for all the circuits tested, with an average error below 6%. This average value could be lower if all the circuits were taken into account and not only those for which there are results for all values of l.

Table 12 presents the maximum and average error for the switching activity estimation over all the individual signals of each circuit. The average was computed by summing the absolute value of the switching probability error relative to the symbolic simulation method for all signals and dividing by the total number of signals. At the bottom of the table the average and maximum of the values for each column are indicated. It can be observed that the average and maximum switching activity error decreases with l. However, the switching activity error may present significant errors. Yet, since the average error is low, the number of nodes with high error is clearly small.

9. Conclusions

We have described an approximation scheme to estimate the switching activity in a logic circuit described at gate level. Our method is parameterized by a single value l which indicates the depth in terms of logic levels over which reconvergent paths (i.e., spatial correlation) is considered. We have presented results that show that in many cases we can ignore spatial correlation and still obtain reasonably accurate switching activity estimates. However, this is not true for all circuits. We showed that for the benchmark circuits we used, with $l = 2$ a power estimation error below 5% is obtained for virtually all combinational circuits and below 6% for sequential circuits, within acceptable CPU time.

References

1. Bryant, R. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, vol. 35, no. 8, pp. 677–691, 1986.
2. Cheng, D. Power Estimation of Digital CMOS Circuits and the Application to Logic Synthesis for Low Power. PhD thesis, University of California at Santa Barbara, Dec. 1995.
3. Chou, T.-L. and K. Roy. Statistical Estimation of Sequential Circuit Activity. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, Nov. 1995, pp. 34–37.
4. Cirit, M. Estimating Dynamic Power Consumption of CMOS Circuits. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, Nov. 1987, pp. 534–537.
5. Costa, J., J. Monteiro, and S. Devadas. Switching Activity Estimation Using Limited Depth Reconvergent Path Analysis. In *Proceedings of the International Symposium on Low Power Electronics and Design*, Aug. 1997, pp. 184–189.
6. Favalli, M., and L. Benini. Analysis of Glitch Power Dissipation in CMOS ICs. In *Proceedings of the International Symposium on Low Power Electronics and Design*, April 1995, pp. 123–128.
7. Glasser, L. and D. Dobberpuhl. *The Design and Analysis of VLSI Circuits*. Addison-Wesley, 1985.
8. Kapoor, B. Improving the Accuracy of Circuit Activity Measurement. In *International Workshop on Low Power Design*, April 1994, pp. 111–116.
9. Lengauer, T. and R. Tarjan. A Fast Algorithm for Finding Dominators in a Flowgraph. *ACM Transactions on Programming Languages and Systems*, vol. 1, no. 1, pp. 121–141, 1979.
10. Marculescu, R., D. Marculescu, and M. Pedram. Switching Activity Analysis Considering Spatiotemporal Correlations. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, Nov. 1994, pp. 294–299.
11. Monteiro, J. and S. Devadas. *Computer-Aided Design Techniques for Low Power Sequential Logic Circuits*. Kluwer Academic Publishers, 1996.

12. Monteiro, J., S. Devadas, A. Ghosh, K. Keutzer, and J. White. Estimation of Average Switching Activity in Combinational Logic Circuits Using Symbolic Simulation. *IEEE Transactions on Computer-Aided Design*, vol. 16, no. 1, pp. 121–127, 1997.
13. Najm, F. Transition Density, A Stochastic Measure of Activity in Digital Circuits. In *Proceedings of the 28th Design Automation Conference*, June 1991, pp. 644–649.
14. Najm, F. A Survey of Power Estimation Techniques in VLSI Circuits (*Invited Paper*). *IEEE Transactions on VLSI Systems*, vol. 2, no. 4, pp. 446–455, 1994.
15. Najm, F., R. Burch, P. Yang, and I. Hajj. Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits. *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 4, pp. 439–450, 1990.
16. Najm, F., S. Goel, and I. Hajj. Power Estimation in Sequential Circuits. In *Proceedings of the 32nd Design Automation Conference*, June 1995, pp. 635–640.
17. Nebel, W., and J. Mermet (eds.). *Low Power Design in Deep Submicron Electronics*. Kluwer Academic Publishers, 1997.
18. Parker, K. and E. McCluskey. Probabilistic Treatment of General Combinational Networks. *IEEE Transactions on Electronic Computers*, vol. C-24, no. 6, pp. 668–670, 1975.
19. Seth, S.C., L. Pan, and V.D. Agrawal. PREDICT: Probabilistic Estimation of Digital Circuit Testability. In *Proceedings of the Fault Tolerant Computing Symposium*, June 1985, pp. 220–225.
20. Tsui, C.-Y., J. Monteiro, M. Pedram, S. Devadas, A. Despain, and B. Lin. Power Estimation Methods for Sequential Logic Circuits. *IEEE Transactions on VLSI Systems*, vol. 3, no. 3, pp. 404–416, 1995.
21. Tsui, C.-Y., M. Pedram, and A. Despain. Efficient Estimation of Dynamic Power Dissipation under a Real Delay Model. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, Nov. 1993, pp. 224–228.
22. Uchino, T., F. Minami, T. Mitsuhashi, and N. Goto. Switching Activity Analysis using Boolean Approximation Method. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, Nov. 1995, pp. 20–25.