Switching Activity Estimation using Limited Depth Reconvergent Path Analysis

José C. Costa
IST/INESC
Lisboa, Portugal

José C. Monteiro
IST/INESC
Lisboa, Portugal

Srinivas Devadas
MIT
Cambridge, MA

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

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.

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 we tried, power estimates with $l = 1$ 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.

I. Introduction

The estimation of average switching activity and power dissipation in digital logic circuits is recognized as an important problem. Average switching activity estimation is computationally difficult, because the input space of the circuit, over which the averaging needs to be done, is very large. Further complications arise due to correlation between internal signals in the logic circuit and logic gate delays. As a result, exact methods are viable only for relatively small circuits, and approximate methods are required for most circuits.

Approximation schemes proposed for power estimation thus far lack some desirable properties. Most schemes are not based on an exact strategy, but based on heuristic rules that model correlation between internal signals in the circuit. While their runtime is typically polynomial, they are rarely parameterizable to improve accuracy at the expense of runtime, and are not calibrated against an exact strategy.

We describe a method of *polynomial simulation* to calculate switching activities in a general-delay combinational logic circuit. This method is a generalization of the exact signal probability evaluation method due to Parker and McCluskey [10], which as been extended to handle temporal correlation and arbitrary transport delays.

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 reconvergence paths meet 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 some long path.

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 we tried, power estimates with $l = 1$ are within 5% error of the exact. However, this error can be higher than 20% for other examples. More robust estimates are obtained with $l = 2$, providing a good compromise between speed and accuracy.

In Section II, 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 III. We introduce in Section IV 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 V. In Section VI, 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 VI.

II. 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.

A. 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 [10] 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 [3]. In this work, both signal spatial and temporal correlation are ignored. The transition density work of Najm [8] introduces temporal correlation, but still ignores correlation between internal signals. Improvements to the basic strategy [4] model some internal correlation, but do not serve as a basis for an exact method. In [6], 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 [13] 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. We use the notion of super-gates to improve the efficiency of our method as well.

B. 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 [9] 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 [12] extends Najm's method by including some correlation coefficients in the probability waveforms.

In [7], 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, the symbolic simulation method is useful in calibrating approximation strategies since it is an exact method, for a given gate delay and capacitance models.

III. Polynomial Simulation

We base the computation of the switching activity at each node in the circuit on the Parker-McCluskey method [10]. 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.

A. 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 $pr(x_i)$, $pr(f) = P_f(pr(x_1), \ldots, pr(x_N))$.

B. Transition Probabilities

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

AND	$0\to0$	$0\to1$	$1\to0$	$1\to1$
$0\to0$	$0\to0$	$0\to0$	$0\to0$	$0\to0$
$0\to1$	$0\to0$	$0\to1$	$0\to0$	$0\to1$
$1\to0$	$0\to0$	$0\to0$	$1\to0$	$1\to0$
$1\to1$	$0\to0$	$0\to1$	$1\to0$	$1\to1$

INVERTER	
$0\to0$	$1\to1$
$0\to1$	$1\to0$
$1\to0$	$0\to1$
$1\to1$	$0\to0$

TABLE I
SIMULATION CALCULUS FOR AN AND GATE AND INVERTER.

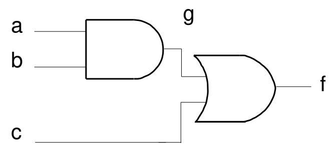


Fig. 1. Unit-delay example circuit.

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 I 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.

$$\begin{aligned} P_h^{00} &= P_g^{11} & \qquad P_h^{01} &= P_g^{10} \\ P_h^{10} &= P_g^{01} & \qquad P_h^{11} &= P_g^{00} \end{aligned}$$

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

$$\begin{aligned} P_h^{00} &= \text{supexp}(P_{g_1}^{00} + (P_{g_1}^{01} + P_{g_1}^{10} + P_{g_1}^{11}) \cdot P_{g_2}^{00} + \\ &\quad P_{g_1}^{01} \cdot P_{g_2}^{10} + P_{g_1}^{10} \cdot P_{g_2}^{01}) \\ P_h^{01} &= \text{supexp}(P_{g_1}^{01} \cdot P_{g_2}^{01} + P_{g_1}^{01} \cdot P_{g_2}^{11} + P_{g_1}^{11} \cdot P_{g_2}^{01}) \\ P_h^{10} &= \text{supexp}(P_{g_1}^{10} \cdot P_{g_2}^{10} + P_{g_1}^{10} \cdot P_{g_2}^{11} + P_{g_1}^{11} \cdot P_{g_2}^{10}) \\ P_h^{11} &= \text{supexp}(P_{g_1}^{11} \cdot P_{g_2}^{11}) \end{aligned}$$

C. 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 1, with unit gate delays we will have, for the various signals, the following polynomial

Polynomial_Simulation ($Network$)
Initialize_Polynomial_Waveforms (PIs ($Network$)) ;
$Gates$ = Topological_Sort($Network$) ;
for each g_i in $Gates$ {
  Δ = delay of g_i ;
  $TimePts$ = NIL(LIST) ;
  for each input g_j of g_i ($gi_1, \cdots, gi_m$) {
    for each time point (k, $P_{g_j}[k]$) of g_j {
      $TimePts$ = InsertInOrder ($TimePts$, (k,$P_{g_j}[k]$)) ;
    }
  }
  for each new time point k in $TimePts$ {
    $P_{g_i}[k+\Delta] = G_i(P_{gi_1}[k], \cdots, P_{gi_m}[k])$;
  }
}

Fig. 2. Pseudo-code for the polynomial simulation algorithm.

waveforms,

$$\begin{array}{ll} a: & P_a[0] \\ b: & P_b[0] \\ c: & P_c[0] \\ g: & P_g[0], P_g[1] \\ f: & P_f[0], P_f[1], P_f[2] \end{array}$$

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

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 2.

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

$$\begin{array}{lcl} P^{00}_{gi_j}[k] & = & P^{00}_{gi_j}[m] + P^{10}_{gi_j}[m] \\ P^{01}_{gi_j}[k] & = & P^{10}_{gi_j}[k] \quad = \quad 0 \\ P^{11}_{gi_j}[k] & = & P^{11}_{gi_j}[m] + P^{01}_{gi_j}[m] \end{array}$$

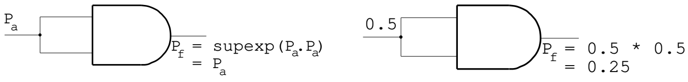
The polynomial group for instant k can equally be computed from the polynomial immediately after instant k.

IV. 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 [11], [2] to reduce the 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.

A. 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



(a) (b)

Fig. 3. Handling spatial correlation.

are common to the support[1] 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 3(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 $pr(x_1) \cdot x_3 + x_2 \cdot x_3$. If additionally we do the same for x_2, the polynomial becomes $pr(x_1) \cdot x_3 + pr(x_2) \cdot x_3 = k \cdot x_3$.

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 3(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* [5].

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 [11], [2] to reduce polynomial complexity. Super-gates are significantly more constrained in that they require the gate to be a dominator 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.

B. 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 4 node d is a dominator for node a. For simplicity assume a unit-delay

[1]The *support* of a logic function is the set of primary inputs that the function depends on.

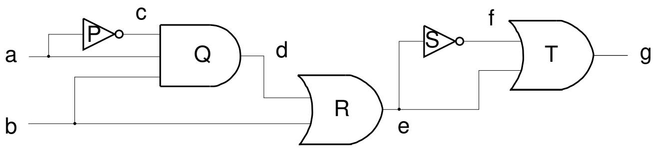


Fig. 4. Variable substitution under a general-delay model.

model, although the following observations apply equally well to the general-delay model. At node d we have polynomials corresponding to instants 1 and 2, both a function of $P_a[0]$ and $P_b[0]$, respectively $P_d[1](P_a[0], P_b[0])$ and $P_d[2](P_a[0], P_b[0])$. If the variable $P_a[0]$ is replaced by its numerical value, thus obtaining $P'_d[1](P_b[0])$ and $P'_d[2](P_b[0])$, the *temporal* correlation between $P'_d[1]$ and $P'_d[2]$ due to a is lost. In this circuit, error will be introduced at node g where, because of the reconvergent path starting at node e, $P'_d[1](P_b[0])$ and $P'_d[2](P_b[0])$ will be operated with each other due to the different delays from e to g.

Also evident in the above example is the error introduced by super-gates. Gates P, Q and R in Figure 4 form a super-gate. However, node e cannot be treated exactly like a primary input since there are three time instants at which e can make a transition. Further, if all variables are substituted, we lose all information about the correlation between these three instants.

V. 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. 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.

A. 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,

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

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 one term of b_1 with a term of b_2 both 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,

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

```
Switching_Activity_Estimation ( Network, l )
for each gate g in ( Network ) {
    Path = NIL(Table) ;
    for each fanout f_j of g {
        Gates = limited_depth_search ( f_j, l ) ;
        for each node h in Gates {
            Insert( Path, h, j, fanin(h) ) ;
        }
    }
    for each duplicate node h in Path with different indexes j {
        while fanin(h) ≠ g {
            Insert(fanin(h).Active_Nodes, g) ;
            h = fanin(h) ;
        }
    }
}
Polynomial_Sim_with_Variable_Substitution ( Network ) ;
```

Fig. 5. Pseudo-code for the limited depth spatial correlation algorithm.

and thus

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

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.

B. 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 2 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 5. 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 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

Circuit Name	$l=2$			$l=3$			$l=\infty$		
	Act. Nodes		CPU	Act. Nodes		CPU	Act. Nodes		CPU
	Avg	Max	(s)	Avg	Max	(s)	Avg	Max	(s)
c1355	1.05	2	0.1	1.08	2	0.2	5.53	33	1.5
c499	1.05	2	0.1	1.08	2	0.1	5.63	33	1.4
add16	1.10	2	0.1	1.50	4	0.1	1.93	3	0.5
alu4	1.30	8	0.2	3.82	14	0.5	7.05	14	2.7
cht	1.08	4	0.1	1.09	4	0.1	1.11	4	0.1
cm163	1.12	2	0.0	1.12	2	0.0	1.12	2	0.0
comp	1.45	9	0.0	2.48	10	0.1	3.75	28	0.1
cordic	1.11	2	0.0	1.28	4	0.0	1.37	5	0.0
count	2.61	26	0.1	2.61	26	0.1	2.61	26	0.2
frg1	1.10	3	0.0	1.67	7	0.1	3.94	22	0.1
i3	1.00	1	0.0	1.00	1	0.0	1.00	1	0.0
i6	1.00	2	0.2	1.32	3	0.2	1.65	4	0.3
mult	1.23	9	0.2	9.59	96	0.4	10.54	16	5.4
mux	1.20	2	0.0	1.47	3	0.0	1.47	3	0.0
x4	1.46	6	0.1	1.55	6	0.2	1.76	6	0.4
z4ml	1.07	2	0.0	1.70	5	0.0	1.57	3	0.0

TABLE II
STATISTICS OF ACTIVE NODES FOUND PER CIRCUIT NODE.

Circuit Name	Symbolic		$l=1$			$l=2$			$l=3$			$l=\infty$		
	P	CPU	P	%	CPU	P	%	CPU	P	%	CPU	P	%	CPU
c1355	N/A		2559		0.5	3910		27.3	3926		29.9	N/A		
c499	N/A		2247		0.3	3146		17.8	3160		20.9	N/A		
add16	960	53.3	965	0.53	0.3	997	3.81	14.5	943	1.74	26.0	958	0.21	36.4
alu4	N/A		4131		4.4	4387		2430.4	N/A			N/A		
cht	930	18.1	938	0.85	0.1	941	1.11	4.8	930	0.00	3.8	931	0.00	3.9
cm163	256	6.1	242	5.65	0.0	256	0.14	1.1	256	0.14	1.1	256	0.14	1.1
comp	568	36.1	580	2.27	0.1	577	1.74	4.0	573	1.04	4001.2	N/A		
cordic	341	8.9	325	4.63	0.0	340	0.08	0.9	340	0.07	1.3	341	0.01	2.4
count	608	29.3	602	0.94	0.3	607	0.09	27.4	607	0.09	27.9	607	0.09	30.3
frg1	816	76.5	837	2.53	0.1	824	0.88	8.4	820	0.41	97.8	N/A		
i3	887	11.1	887	0.00	0.0	887	0.00	1.0	887	0.00	0.9	887	0.00	1.0
i6	2726	67.5	2688	1.37	0.3	2691	1.28	10.8	2719	0.25	13.6	2726	0.00	15.1
mult	N/A		17536		12.9	18024		3374.2	N/A			N/A		
mux	354	5.7	332	6.26	0.0	346	2.29	2.9	353	0.36	6.0	353	0.36	5.3
x4	2089	75.2	1990	4.73	0.5	2083	0.29	39.6	2095	0.28	88.3	2087	0.10	1800.4
z4ml	319	6.0	286	10.27	0.0	317	0.73	1.6	317	0.57	6.2	319	0.03	3.4
			$max = 10.3, avg = 3.3$			$max = 3.8, avg = 1.0$			$max = 1.7, avg = 0.4$			$max = 0.4, avg = 0.1$		

TABLE III
POWER ESTIMATION RESULTS.

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 ($P_{g_i}[k+\Delta]$) ;

Fig. 6. Pseudo-code for the polynomial simulation algorithm with variable substitution.

is done is called. The only difference from the algorithm of Figure 2 is that between lines 12 and 13 we insert the code where variable substitution is done (Figure 6).

VI. EXPERIMENTAL RESULTS

In this section we present power and switching activity estimation results obtained with the approximation algorithm based on limited depth reconvergent path analysis described in the previous sections. We present results for different values of l and compare them with the exact value obtained with symbolic simulation [7].

The first part of the approximation algorithm involves computing for each node in the circuit the set of active nodes, i.e., the variables the polynomials at each node will be a function of. We present statistics on the number of active nodes for our benchmark circuits in Table II. 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. As expected, as l increases, both the average and maximum values increase. 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 $l = 1$ because for the examples we have, there is only one active node for all nodes, though this is not necessarily true for all circuits.

The CPU time we present in this table corresponds only to the algorithm that computes the active nodes (cf. Figure 5). All reported CPU times are in seconds and were obtained on Sun 5/85 with 64M of main memory. 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.

Circuit Name	$l = 1$		$l = 2$		$l = 3$		$l = \infty$	
	max	avg	max	avg	max	avg	max	avg
c1355	N/A		N/A		N/A		N/A	
c499	N/A		N/A		N/A		N/A	
add16	0.32	0.06	0.22	0.05	0.30	0.05	0.12	0.01
alu4	N/A		N/A		N/A		N/A	
cht	0.09	0.01	0.09	0.00	0.01	0.00	0.00	0.00
cm163	0.29	0.04	0.02	0.00	0.02	0.00	0.02	0.00
comp	0.30	0.01	0.17	0.01	0.15	0.01	N/A	
cordic	0.67	0.04	0.05	0.00	0.09	0.00	0.00	0.00
count	0.11	0.01	0.03	0.00	0.03	0.00	0.03	0.00
frg1	0.32	0.02	0.13	0.01	0.07	0.00	N/A	
i3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i6	0.17	0.03	0.17	0.02	0.07	0.01	0.00	0.00
mult	N/A		N/A		N/A		N/A	
mux	0.47	0.04	0.11	0.01	0.05	0.00	0.05	0.00
x4	0.31	0.04	0.15	0.01	0.14	0.01	0.07	0.00
z4ml	0.43	0.07	0.39	0.02	0.04	0.01	0.03	0.00
Avg	0.29	0.03	0.13	0.01	0.08	0.01	0.03	0.00
Max	0.67	0.07	0.39	0.05	0.30	0.05	0.12	0.01

TABLE IV
SWITCHING ACTIVITY ERRORS.

Table III presents the power estimation results obtained with the approximation algorithm using l equal to 1, 2, 3 and ∞. A general delay model was used for all the examples and a supply voltage of 5V 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 [7] and the CPU time (in seconds) taken by this computation. For some of the circuits, this method run out of memory and this is indicated with a "N/A" in the table. In the columns under "$l = 1$" are the results for the approximation algorithm using $l = 1$. 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 "$l = 2$", "$l = 3$" and "$l = \infty$".

At the bottom of the table, we give the maximum and average error over all the circuits, for each value of l. We can observe that both these values decrease very rapidly with l. Also interesting to note is that the average error is very low even for $l = 1$. However, the maximum error is not as low. $l = 2$ brings both values to an acceptable level. For some circuits, we cannot compute estimates with $l = 3$. As it can be seen from Table II, these correspond to situations where the circuits have a large number of active nodes, thus making the polynomials too large.

Note that the error for $l = \infty$ is not zero. This is due to the temporal correlation effects described in Section IV-B. If a zero-delay model is used, $l = \infty$ gives exactly the same results as the symbolic simulation method.

For many applications, a more relevant measure of accuracy is the error in the switching activities of individual signals. In Table IV 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 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.

VII. 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 $l = 2$ an average switching activity error below 0.05 (absolute value) and a power estimation error below 5% is obtained, within acceptable CPU time.

REFERENCES

[1] R. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 35(8):677–691, August 1986.

[2] D. Cheng. *Power Estimation of Digital CMOS Circuits and the Application to Logic Synthesis for Low Power*. PhD thesis, University of California at Santa Barbara, December 1995.

[3] M. Cirit. Estimating Dynamic Power Consumption of CMOS Circuits. In *Proceedings of the International Conference on Computer-Aided Design*, pages 534–537, November 1987.

[4] B. Kapoor. Improving the Accuracy of Circuit Activity Measurement. In *Proceedings of the International Workshop on Low Power Design*, pages 111–116, April 1994.

[5] T. Lengauer and R. Tarjan. A Fast Algorithm for Finding Dominators in a Flowgraph. *ACM Transactions on Programming Languages and Systems*, 1(1):121–141, July 1979.

[6] R. Marculescu, D. Marculescu, and M. Pedram. Switching Activity Analysis Considering Spatiotemporal Correlations. In *Proceedings of the International Conference on Computer-Aided Design*, pages 294–299, November 1994.

[7] J. Monteiro, 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*, 16(1):121–127, January 1997.

[8] F. Najm. Transition Density, A Stochastic Measure of Activity in Digital Circuits. In *Proceedings of the 28^{th} Design Automation Conference*, pages 644–649, June 1991.

[9] F. Najm, R. Burch, P. Yang, and I. Hajj. Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits. *IEEE Trans. on Computer-Aided Design*, 9(4):439–450, April 1990.

[10] K. Parker and E. McCluskey. Probabilistic Treatment of General Combinational Networks. *IEEE Transactions on Electronic Computers*, C-24(6):668–670, 1975.

[11] S. C. Seth, L. Pan, and V. D. Agrawal. PREDICT: Probabilistic Estimation of Digital Circuit Testability. In *Proceedings of the Fault Tolerant Computing Symposium*, pages 220–225, June 1985.

[12] C-Y. Tsui, M. Pedram, and A. Despain. Efficient Estimation of Dynamic Power Dissipation under a Real Delay Model. In *Proceedings of the International Conference on Computer-Aided Design*, pages 224–228, November 1993.

[13] T. Uchino, F. Minami, T. Mitsuhashi, and N. Goto. Switching Activity Analysis using Boolean Approximation Method. In *Proceedings of the International Conference on Computer-Aided Design*, pages 20–25, November 1995.