A New Hybrid Methodology for Power Estimation

David Ihsin Cheng+, Kwang-Ting Cheng*, Deborah C. Wang#, and Malgorzata Marek-Sadowska*

Mentor Graphics Corp.+ San Jose, CA 95131

Dept. of Electrical and Computer Engineering* Univ. of California, Santa Barbara, CA 93106

LSI Logic Corp.# Milpitas, CA 95035

Abstract [1]

In this paper, we propose a hybrid approach for estimating the switching activities of the internal nodes in logic circuits. The new approach combines the advantages of the simulation-based techniques and the probability-based techniques. We use the user-specified control sequence *for simulation and treat the weakly correlated data inputs using the probabilistic model. The new approach, on one hand, is more accurate than the probabilistic approaches because the strong temporal and spatial correlations among control inputs are well taken into consideration. On the other hand, the new approach is much more efficient than the simulation-based approaches because the weakly correlated data inputs are not explicitly simulated. In addition, we also propose a heuristic that builds BDDs in terms of internal nodes such that large circuits can be handled. Extensive experimental results are presented to show the effectiveness and efficiency of our algorithms.*

1 Introduction

We can roughly divide the existing techniques on estimating the switching activities into two categories, the simulation-based and the probability-based. In simulation-based approaches (e.g.: [1][11]), the basic idea is quite simple. With either user-provided functional vectors or randomly generated patterns at the primary inputs, we can simulate the circuit repeatedly and monitor the switching activities at the internal nodes of the circuit. Eventually the switching activity of each node in the circuit will converge to the average case. Good techniques, such as the Monte Carlo method [1], do exist for limiting the convergence of switching activities to a user-specified confidence range and terminating the simulation faster.

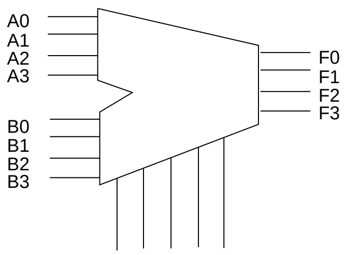
In probability-based approaches (e.g.: [7][4][10][5]), with user-provided parameters that characterize the typical behavior at the primary inputs (PIs), the idea is to propagate these parameters, which are in a probabilistic form, through the whole circuit and thereby obtain the average switching probabilities at the internal nodes of the circuit. One very important issue here is the modeling of the temporal and the spatial correlations among signals. *Temporal correlation* refers to the dependency between the values of a signal in two consecutive time frames; *spatial correlation* refers to the dependency between two separate signals. These correlations exist at internal nodes as well as at PIs.

In general, the simulation-based approaches are more accurate but slower, while the probability-based techniques are less accurate but faster. Moreover, the simulation-based approaches have the advantage that complicated spatial and temporal correlations, no matter at the primary inputs or internal nodes, are automatically taken care of because the actual node transitions are counted. The probability-based approaches, on the other hand, have the advantage of being able to compactly lump lengthy, random, and independent activities into a few, usually one or two, probabilistic parameters.

In modeling the PI activities, we say the simulation-based approaches use a *deterministic* model, to contrast with the *probabilistic* model used in the probability-based approaches. In this paper, we first propose a hybrid approach on the modeling of the PI activities when functional vectors are available. A key motivation of this new approach is the fact that, in most industrial designs, circuits have data path and control path mixed together. The data path inputs are weakly correlated (spatially and temporally) and random in nature, while the control path inputs are highly correlated and typically limited to certain sequences of patterns. We propose to model these two sets of signals separately. Data inputs, due to their random and uncorrelated nature, should be modeled by probabilistic parameters in an implicit form. This is similar to the modeling of PI activities used by the existing probability-based techniques. On the other hand, control inputs, due to their highly correlated nature, stay deterministically in the explicit form given by the functional vectors. This is similar to the modeling of PI activities used by the existing simulation-based techniques. In order to distinguish from the original complete functional vectors, we use the term *control sequence* to refer to the sequence of the control part in the given functional vectors. In other words, in modeling of the PI activities, we keep the part of the control inputs intact in the given functional vectors, but compactly lump the actual patterns in the part of the data inputs into their probabilistic parameters.

Using this hybrid model that separates data inputs and control inputs, we then present a hybrid algorithm that combines the advantages of both the simulation-

[1]This work was done while the first authur was with Univ. of California at Santa Barbara, and was supported in part by the NSF grant MIP 9419119 and California MICRO/LSI/SVR.



S3	S2	S1	S0	M	
0	0	0	0	0	transfer A (load to some register outside)
0	0	0	1	0	A plus B
0	0	0	0	0	transfer A (load to some register outside)
0	1	1	0	0	A xor B
0	0	0	0	1	complement A
1	1	1	0	1	A or B
$s = \frac{1}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{1}{6}$	$\frac{2}{6}$	
$t = \frac{2}{6}$	$\frac{4}{6}$	$\frac{4}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	

Figure 1: A 4-bit ALU

based and the probability-based techniques. The algorithm is extended and generalized from the symbolic simulation technique proposed in [4], for which we will provide a detailed review in later sections. By simulating the control patterns given at the control inputs, we propagate the probabilistic parameters given at the data inputs through the whole circuit. Our algorithm is very accurate and improves the purely probabilistic nature of the original symbolic simulation technique.

To handle large circuits, we also propose a heuristic that builds BDDs in terms of the internal nodes, as opposed to in terms of the PIs, of a circuit. The heuristic is based on the topological analysis presented in [2], which finds the minimally required independent signals for each node in a circuit systematically. We modify the algorithm in [2] to find weakly correlated, rather than completely independent, internal signals for each node. The resulting heuristic can trade accuracy for applicability or run time, and therefore enables us to handle large circuits.

Our hybrid model and algorithm degenerate to a simulation-based approach when a given circuit does not have data path, and degenerate to a probability-based approach either when the circuit does not have control path or when the control sequence is not available.

2 The Model Separating Data and Control

Given a logic circuit, we assume that the control sequence is available. In this section we discuss our new model that separates the control inputs and the data inputs. As an example, Fig. 1 shows a schematic diagram of a 4-bit ALU. The data inputs of the ALU are $A0$-$A3$ and $B0$-$B3$. The control inputs are $S0$-$S3$ (selectors) and m (mode), whose values determine the functionality of the ALU. This ALU was embedded in a larger system we found. Due to the system requirement, this ALU had the fixed control sequence shown in Fig. 1, where the meanings of the control patterns are also shown. The data inputs, on the other hand, will be supplied with random data in the system. We can see that, even in this small example, this control sequence has a spatial correlation across five signals and a temporal correlation across six time frames.

Continuing with the above example, we now look at what traditional techniques would do to model the PI activities. In a simulation-based technique, with the repeating of the six control patterns, we have to generate many random patterns out of the all possible combinations, as well as an even larger number of possible sequences of these combinations, at the data inputs. This large number of random patterns, as mentioned in Section 1, makes a simulation-based technique slow. However, note that in this case the complicated control input correlations (temporal and spatial) are automatically taken care of because the actual control patterns are intact. On the other hand, in a probability-based technique, we need to characterize each PI, regardless of control or data, into a few probabilistic parameters. Let us illustrate the situation with two commonly used parameters, the signal probability and the transition density [7], denoted by s and t respectively. The data inputs, because of their randomness, all have $s = \frac{1}{2}$ and $t = \frac{1}{2}$. The signal probability and the transition density of the control inputs are calculated as shown at the bottom of Fig. 1. Here in the modeling of the control input activities, we can see that these probabilistic characterizations have already implicitly caused a loss of some correlation information. For example, spatially, the occurrence of combination 11111, which should have a probability zero, now has the probability equal to the product of the signal probabilities of all the control inputs. Temporally, for example, the sequence of 00000 followed by 00000, which should have a zero occurrence probability, now obviously has some non-zero probability. In spite of the above inaccuracy in modeling the control inputs, however, note that the lengthy and random nature of the data inputs are compactly lumped into the signal probability and the transition density, which almost fully characterize their activities.

Based upon the understanding of the above phenomena, we propose a hybrid model that separates the data and control inputs. We treat the data inputs probabilistically, and characterize the data inputs with two parameters, the signal probability and the transition density[7]. We treat the control inputs deterministically and keep the control sequence intact.

3 Hybrid Algorithm

With the hybrid model, we present a hybrid algorithm in this section. Since the symbolic simulation technique in [4] is very related to our algorithm, we first provide a brief review of [4] in Section 3.1. In this section we assume the circuit is combinational and all nodes have zero delay. The issues related to sequential circuits will be discussed in Section 5.

3.1 Symbolic simulation

In [4], the transition density of each internal node is analyzed by considering a PI pair, $V(t)$ and $V(t+1)$, where $V(t)$ corresponds to the current time frame and $V(t+1)$ corresponds to the next time frame. More specifically, each PI is first represented by two BDD variables, one for the current time frame, and the other for the next time frame. For each internal node, the symbolic simulation technique builds a Boolean function whose on-set represents all the conditions when

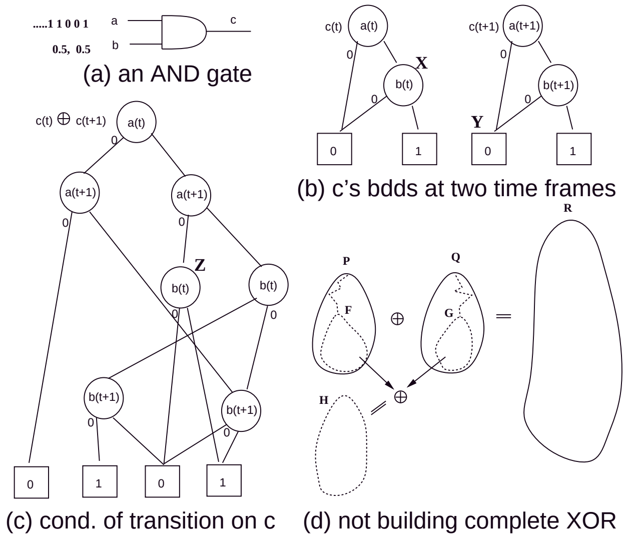


Figure 2: Symbolic simulation and our modifications

the internal node makes a transition. This can be best explained by an example. Fig. 2(a) shows a simple AND gate. Let $a(t)$ and $a(t+1)$ be the BDD variables representing the values of PI a at time frame t and $t+1$, respectively (similarly for signal b). Assuming zero delay, the BDDs of the functions of signal c at time frames t and $t+1$ are shown in Fig. 2(b). Taking the XOR of the BDDs in these two time frames, in Fig. 2(c) we obtain the BDD whose on-set represents all the PI conditions that signal c makes a transition. Given the signal probability and the transition density of PIs a and b, we can evaluate the the transition density of c by linearly traversing the BDD in Fig. 2(c).

3.2 Our algorithm

For each control pattern p_i in a control sequence, we look at the values of a control input in p_i and p_{i+1}. Then we simply pretend that the values of the control inputs are fixed to the values given in p_i and p_{i+1}. For example, suppose signal a in Fig. 2(a) is a control input and has the control sequence as shown, while signal b is a data input having signal probability 0.5 and transition density 0.5. Now we look at the first and the second control patterns of signal a. In this case, we simply pretend signal a is fixed at 1 in the current time frame and also will be fixed to 0 in the next time frame. In terms of the BDD traversal performed in the original symbolic simulation technique, this simply means we only take the paths that consist of $a(t) = 1$ and $a(t+1) = 0$. In Fig. 2(c), this means we only evaluate the probability of the sub-BDD rooted at the node labeled by Z. This evaluated probability corresponds to the transition density of signal c when input a is fixed to the "10" sequence while b is a random data. In this manner, our algorithm iteratively performs the evaluation for every control pattern in the control sequence. Finally, for every internal node, we take the average of the evaluated transition densities as the node transition density.

Since in each run of a control pattern we need to evaluate all the BDDs, at the first glance it may seem to be a formidable task that we need to do n runs, where n is the length of the control sequence. We can greatly improve the efficiency of our algorithm with proper data structures. First, we should order the control inputs as the top variables in all BDDs, while the orderings among control inputs themselves and among data inputs themselves are not important. Recall that, in the original symbolic simulation technique, the final BDD that characterizes *all* the conditions for an internal node making a transition is built by XOR'ing the BDDs corresponding to the current and the next time frames. When control inputs are ordered as the top variables, we can "walk down" these two BDDs according to the deterministic values given in control patterns p_i and p_{i+1} until a BDD node representing a data input is reached. We can then take the XOR of the sub-BDDs rooted at the reached nodes and perform probability evaluation on the resulting BDD, which now only has BDD variables representing the data inputs. As an example, consider the first and the second patterns of signal a in Fig. 2(a) again. Since we have a "10" sequence, we set $a(t) = 1$ and $a(t+1) = 0$. In Fig. 2(b), we therefore can "walk down" the BDDs and reach the BDD nodes labeled by X and Y in the BDDs representing the current(left) and the next(right) time frames, respectively. Taking the XOR of these two sub-BDDs, we have the resulting BDD as the same one labeled by X because the other BDD is Boolean 0. Note that the BDD labeled by X is the same as the one labeled by Z in Fig. 2(c). This is because both of these two BDDs characterize the same condition: the transition density of signal c given that signal a has a "10" transition. In the original symbolic simulation technique, the BDD corresponding to the XOR of two time frames at each node characterizes *all* the conditions when the node makes a transition. Since very often the patterns in a control sequence do not consist of all possible combinations, with our "walk down" method we perform the XOR *when needed*. Depending on the control sequence, this sometimes greatly reduces the time and space complexity. An illustrative diagram is shown in Fig. 2(d) where all the egg-shaped circles represent BDDs. What the original symbolic simulation technique does is XOR'ing the *complete* functions of the current (labeled by P) and the next (labeled by Q) time frames. The resulting BDD (labeled by R) may become very large. In our case, we only XOR the *sub-functions* (labeled by F and G), and many times have much smaller resulting BDD (labeled by H).

There is one more point we need to clarify with the above "XOR when needed" method. It is not unusual at all that some control pattern pairs repeat. When a control pattern pair repeats, we would repeat the same "walk down" process and XOR the same pair of sub-BDDs that we XOR'ed before. This repeating of computation can be saved by building a cache, usually implemented as a hash table, that stores the XOR results.

The time complexity of each run of our algorithm is linear in terms of the BDDs needed. In practice, because of the cache setup, many times we have the situation where only the first few runs reach this time complexity, and the rest runs are extremely fast because they either just walk down the BDDs and take the result from the cache, or easily hit some BDD node whose probability has been calculated during previous

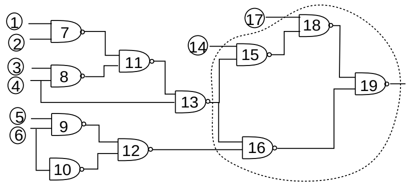


Figure 3: Supergate concept

traversals.

Because we deterministically simulate all control patterns, all the temporal and spatial correlation existing at control inputs are automatically taken care of, just like the case in a traditional simulation-based technique. On the other hand, because we compactly characterize the randomness of data inputs into two parameters, we keep the efficiency, just like the case in a traditional probability-based technique.

4 BDDs in terms of internal nodes

In both the symbolic simulation technique and our hybrid algorithm discussed so far internal signal correlations have not been a problem because complete BDDs, in terms of PIs, are built. In dealing with large circuits where complete BDDs cannot be built, we propose a heuristic that builds BDDs in terms of internal signals. Our heuristic is extended and generalized from the *supergate* analysis [9][2], which will be briefly reviewed in Section 4.1.

4.1 Supergate

Informally, the *supergate* of a node X, denoted by $SG(X)$, is the minimal subcircuit in X's transitive fanin, feeding X, such that the subcircuit's fanins, denoted by $SGFI(X)$, are logically independent ([9]). As an example, consider Fig. 3 and ignore the dotted circle for now. The supergate of node 13 is the subcircuit consisting of $SG(13) = \{13, 11, 8\}$, which is the minimal subcircuit in 13's transitive fanin such that the fanins, $SGFI(13) = \{7, 4, 3\}$, are indeed logically independent. As another example, node 18's supergate consists of the subcircuit of itself only, $SG(18) = \{18\}$, whose fanins are $SGFI(18) = \{17, 15\}$. The supergate of each node is unique ([9]). As illustrated by the above example, some nodes have smaller supergates while others have bigger ones. Node 19 has the largest supergate in Fig. 3, $SG(19) = \{19, 18, 16, 15\}$ and $SGFI(19) = \{17, 14, 13, 12\}$, which is marked by the dotted circle.

[2] presents an efficient algorithm to find the supergate for every node in a circuit. In estimating the switching probabilities, the algorithm therefore enables us to build BDDs in terms of the minimal number of independent internal signals (i.e., SGFI) for each node. For example, in Fig. 3, we build the BDD for node 19 in terms of nodes $SGFI(19) = \{17, 14, 13, 12\}$. After obtaining the transition densities of these four nodes, we can then calculate that of node 19. Since SGFI's are independent by definition, we keep the same accuracy for all nodes, while at the same time building smaller BDDs. The problem with the supergate analysis, however, is that very often the supergates of some nodes in a circuit are still too large to be practical.

4.2 k-level heuristic

Correlations among internal signals are caused by reconvergent fanouts, which in turn cause the supergates of certain nodes to be large. By assuming certain reconvergent signals independent, which sacrifices some accuracy, we can shrink the sizes of large supergates. The important thing on making such assumptions on certain internal signals is that we want to shrink the supergate sizes as much as possible, while sacrificing as little accuracy as possible. Our modification on the supergate analysis is based on the following observations.

Fact 1 *Given a node n, for all the signals in $SGFI(n)$ which are not immediate fanins of n, there exists at least one that is a multiple fanout point and that reconverges inside $SG(n)$.*

For example, in Fig. 3, node 13 is such a reconvergent node among $SGFI(19) = \{19, 18, 16, 15\}$. We will call these nodes *the reconvergent nodes in SGFI*. For a node n with a large supergate, the reconvergent nodes in $SGFI(n)$ must propagate through multiple levels of logic.

Fact 2 *Given a node n, if node i is a reconvergent node in $SGFI(n)$, then assuming node i's multiple fanouts independent will shrink the size of $SG(n)$.*

For example, in Fig. 3, if we make the assumption that node 13's two fanouts are independent, the supergate of node 19 would shrink to $SG(19) = \{19\}$ and $SGFI(19) = \{16, 18\}$.

Our heuristic is based on the above two observations. Let the *distance* between two nodes n_1 and n_2 be the maximal number of levels of nodes along all paths between n_1 and n_2. For each node n, we first find the supergate $SG(n)$ using the algorithm in [2]. If the maximal distance between node n and any node in $SGFI(n)$ is greater than a given threshold k, we regard the supergate as too large. We then find the reconvergent signals in $SGFI(n)$ with a distance to node n longer than k and assume their fanouts as independent. The algorithm in [2] is then called again. Repeatedly, we shrink the supergate size until the given k-level constraint is met. Given a node n whose supergate is large, since the reconvergent signals in $SGFI(n)$ with a long distance to n are the main cause of the largeness of $SG(n)$, assuming independence on these signals, as opposed to any other reconvergent signals in the whole circuit, shrinks $SG(n)$ the most. The lost accuracy on making such independent assumptions is, on the other hand, the least. To see this, we first note that the correlations of signals, when propagating through different paths, are "diluted" by other uncorrelated signals and thus become weaker when reconverged. The degree of dilution is somewhat related to the distance between the reconvergent signals and the target node n. For example, in Fig. 3, consider the two fanout signals of node 13. One of the fanout signals propagates through nodes 15 and 18 (the upper path) and the other through node 16 (the lower path). The two fanout signals finally reconverge at node 19. The probabilities of the fanout signal of the upper path are mixed with the probabilities of nodes 14 and 17, while that of the lower path are mixed with the probabilities of node 12. When arriving at node 19, the correlation of these two signals are therefore much weaker ("diluted") than it was when they first forked. For a given node n, each time in our algorithm we only make the independent assumptions on some signals in $SGFI(n)$ with a distance to n greater than k and all other reconvergent signals inside $SG(n)$ are kept intact.

5 Sequential Circuits

Recall that the symbolic simulation technique associate each PI with two BDD variables, one representing the value of the current time frame and the other representing the next time frame. When dealing with sequential circuits, [4] and [6] propose a *prepending* mechanism. For each present state (PS) line, the BDD variable representing the the current time frame is kept intact, while the BDD variable representing the next time frame is replaced by the corresponding next state function. In other words, the whole next state logic is "prepended" before the symbolic simulation equations. In this manner every setup stays the same as in the combinational case except that we need the probabilities of the PS lines. The PS line probabilities are obtained by iterations over a set of implicit nonlinear equations that characterize the whole next state functions ([6]). Assuming the PS lines are spatially independent, the state probabilities can then be approximated by the PS line probabilities.

Our hybrid algorithm uses the same setup for the PS lines as discussed above. The only difference between our hybrid algorithm and the original symbolic simulation technique is that we propagate the probabilities at the data inputs through the whole circuit while simulating the deterministic values at the control inputs. This applies to both the iterations on finding the PS line probabilities and the final calculation of the transition density of every internal node. When BDDs are built in terms of the PIs, we replace the BDD variable representing the next time frame at every PS line with the corresponding next function. When BDDs are built in terms of internal signals, we simply view the prepended next state logic as part of the whole circuit and hence BDDs could be built across PS line boundary into the internal signals of the prepended logic. Due to space limitation, see [3] for more details and a discussions of the issues related to arbitrary delays.

6 Experimental Results

In this section we present the experimental results on our algorithms. Table 1 shows the characteristics of the circuits we conduct the experiment on. The first five circuits in Table 1 are industrial circuits and the remaining ones are from ISCAS benchmark circuits. Columns "PI", "PO", "nodes", and "FF" list the number of primary inputs, primary outputs, nodes, and latches, respectively. Columns "c_PI" and "c_seq" list the number of control inputs and the length of the control sequence. For the first five industrial circuits, the control inputs and the control sequence are specified by the circuit designers. For the remaining ISCAS circuits, we first optimize them with *script.boolean* in *SIS*, then we map the circuits using *lib2.genlib* library ([8]). Since we do not know the details of what these benchmark circuits are, we randomly selected 1/3 of the PIs, up to maximally 30, as control inputs and randomly generated 100 patterns as control sequence

b	PI	PO	nodes	FF	c_PI	c_seq	low_tr	high_tr
i1	14	8	63	0	6	6	4	59
i2	83	103	799	0	34	47	205	594
i3	19	11	159	15	7	10	17	142
i4	40	20	374	42	10	140	174	200
i5	34	31	1020	53	4	402	602	418
C1355(c1)	41	32	276	0	14	100	66	210
C1908(c2)	33	25	282	0	12	100	65	217
C2670(c3)	233	140	366	0	30	100	10	356
C3540(c4)	50	22	601	0	17	100	62	539
C6288(c5)	32	32	2090	0	11	100	7	2083
C7552(c6)	207	108	1154	0	30	100	13	1141
C880 (c7)	60	26	203	0	21	100	19	184
s1196(s1)	14	14	281	18	5	100	71	210
s1238(s2)	14	14	305	18	5	100	86	219
s1423(s3)	17	5	364	74	6	100	152	212
s5378(s4)	35	49	721	162	12	100	300	421
s9234(s5)	36	39	632	135	13	100	264	368

Table 1: Circuit characteristics

on these selected control inputs. For each of the circuits listed in Table 1, a very long logic simulation was then performed using randomly generated patterns applied to the data inputs with repeated control sequence applied to the control inputs. The transition density obtained from the long simulation was then regarded as the accurate result. To compare the average error of each node in our algorithms against the accurate result, we would like to use this metric of relative error defined by $\sum_{i=1}^{n} \frac{|T_i^e - T_i^a|}{T_i^a}$, where n is the number of nodes in a given circuit and T_i^e and T_i^a are the estimated and the accurate transition density of node i, respectively. However, as was also pointed out by other researchers (e.g.: [5]), the metric tends to overemphasize the nodes with low transition density. We therefore decided to measure our results by separating the nodes into two groups. The nodes whose transition density is low, defined by $T_i^a < 0.08$, are classified into one group, and the remaining nodes into the other. The last two columns of Table 1, "low_tr" and "high_tr", list the number of nodes, respectively, in these two groups. This classification allows us to fairly use the metric of relative error on the higher transitioned ("high_tr") group. For the nodes in "low_tr", we would like to use the metric of absolute error defined by $\frac{\sum_{i=1}^{n} |T_i^e - T_i^a|}{n}$; however for the convenience of comparison using percentage, we divide the above absolute error by the threshold value 0.08.

Table 2 lists the result of our hybrid algorithm compared with the probabilistic symbolic simulation technique, both with complete BDDs built. The circuits that are in Table 1 but not in Table 2 are the ones whose BDDs cannot be built due to memory limitation. In Table 2, we contrast the results of our hybrid algorithm (column "hyb") and the probabilistic symbolic simulation (column "prb"), under the categories of low transition nodes (column "low_tr") and high transition nodes (column "high_tr"). The result of each of the two algorithms is reported by the aver-

	low_tr						high_tr					
	hyb(%)			prb(%)			hyb(%)			prb(%)		
b	av	mx	dv	av	mx	dv	av	mx	dv	av	mx	dv
i1	0.5	0.9	0.6	3.9	4.6	4.0	0.3	1.0	0.4	16.0	179.4	33.4
i2	0.3	3.6	0.6	0.5	11.3	1.9	0.5	2.8	0.7	3.8	18.3	8.9
i3	1.5	8.7	2.3	6.0	19.9	8.2	1.2	9.8	3.2	8.0	43.6	14.4
c1	0.4	2.0	0.6	2.6	98.2	12.7	0.2	2.8	0.4	4.4	95.2	12.4
c2	0.8	4.2	1.2	0.9	4.6	1.4	0.2	1.7	0.3	3.4	61.6	9.5
c4	0.2	1.7	0.5	44.5	246.3	69.4	0.4	2.6	0.5	9.7	258.6	24.2
c7	0.1	0.3	0.2	40.6	170.0	58.6	0.2	1.3	0.2	10.1	77.5	13.0
s1	1.1	15.6	2.8	4.1	48.2	8.6	0.8	10.1	1.7	6.3	55.7	11.0
s2	1.0	10.2	2.0	7.6	257.0	30.7	0.9	11.5	1.8	7.3	60.0	12.3
av	0.6	5.2	1.2	12.3	95.6	21.7	0.5	4.8	1.0	7.7	94.4	15.5

Table 2: Complete BDDs built in terms of PIs

age error (column "av"), the maximal error (column "mx"), and the standard deviation of errors (column "dv"). Take *c4* (*C3540*) as an example. For the low transition nodes (57 nodes, from Table 1), on average the absolute error is 0.000191. Dividing 0.000191 by the threshold value 0.08, we have 0.2% of error on average. The maximal error, 0.001380, and the standard deviation on the errors, 0.000360, are therefore 1.7% and 0.5%, respectively. Similarly, using the probabilistic technique, the average error, the maximal error, and the standard deviation are 0.035578, 0.197040, and 0.055537, which after divided by 0.08 give the entries 44.5%, 246.3%, and 69.4%. For the high transition nodes, the metric of relative error is used. In the case of *C3540*, the average error, the maximal error, and the standard deviation of the errors of the higher transitioned 539 nodes (Table 1) are, respectively, 9.7%, 258.6%, and 24.2%. From Table 2, we can see that our hybrid algorithm not only uniformly improves the average error on both low and high transition nodes, but also reduces the standard deviation and maximal error greatly. The last row in Table 2 shows the overall quantitative improvement.

The results of running our k-level heuristic are listed in Tables 3 for $k = 4$. For comparison, we applied the k-level heuristic to both our hybrid algorithm and the probabilistic symbolic simulation technique. Again we see that our hybrid algorithm improves the probabilistic technique in almost all the cases[2].

7 Conclusions

In this paper, we describe a new model, which separates data inputs and control inputs, for calculating the switching activities of the internal nodes in a logic circuit. The control inputs tend to have very strong temporal and spatial correlations in typical system operations and thus cannot be accurately characterized by a probabilistic model. We use the control sequence, provided by the designers, and the probabilistic characteristics of the data inputs as the input to our approach and developed an algorithm to handle the hybrid model in an integrated manner. Using the control sequence explicitly for simulation increases the estimation accuracy because the strong temporal and spatial correlations at control inputs are well taken into consideration. We also propose a k-level heuristic, based on topological analysis, for handling large circuits. Moreover, our methods do not much increase the computational complexity as compared to probability-based approaches, and are applicable to sequential circuits and circuits with an arbitrary delay model. Experimental results justify that the new method produces results with high accuracy.

[2]Due to space limitation, see [3] for a more extensive comparison, including various k values and CPU/memory usage

	low_tr						high_tr					
	hyb(%)			prb(%)			hyb(%)			prb(%)		
b	av	mx	dv	av	mx	dv	av	mx	dv	av	mx	dv
i1	0.5	0.9	0.6	0.5	0.9	0.6	1.1	20.1	3.6	11.9	95.7	28.3
i2	0.6	8.7	1.2	0.9	11.7	2.0	1.9	32.7	9.0	2.3	44.6	8.9
i3	9.2	50.2	12.6	19.8	60.2	15.2	6.3	20.7	10.3	12.7	29.6	24.3
i4	12.7	35.8	16.4	28.3	100.2	30.8	10.3	88.8	20.2	18.3	21.3	33.9
i5	30.2	133.7	45.6	48.9	190.3	60.3	15.7	64.8	32.3	33.8	75.3	49.9
c1	4.6	9.2	6.0	6.8	108.2	15.2	0.5	13.5	1.6	3.2	90.0	9.5
c2	3.1	11.1	3.9	3.2	15.1	3.9	0.9	10.9	3.7	2.3	32.9	6.0
c3	1.2	4.1	2.0	1.3	5.2	2.4	3.0	30.5	5.7	3.3	40.5	7.0
c4	2.7	45.4	8.9	50.4	344.8	80.3	7.3	162.3	21.5	17.6	294.8	36.4
c5	15.3	119.2	20.9	27.6	125.2	24.4	8.0	67.3	18.3	11.9	92.3	24.9
c6	20.8	115.1	30.7	29.8	135.7	62.9	2.6	109.9	9.0	4.9	200.8	15.8
c7	0.1	0.3	0.2	40.1	174.2	50.0	0.8	11.1	1.9	4.9	77.5	14.4
s1	7.0	58.8	13.0	10.3	58.8	16.0	2.8	94.4	21.9	7.8	66.6	14.1
s2	2.4	20.8	4.5	11.3	170.1	35.1	2.9	23.7	5.3	8.1	104.7	16.5
s3	26.3	200.3	51.3	29.3	190.3	61.3	13.7	123.3	27.3	32.9	162.5	47.2
s4	8.1	145.7	19.4	12.1	155.6	23.7	9.5	206.3	20.6	19.7	236.2	31.9
s5	11.7	170.8	21.7	22.3	210.7	36.8	11.3	148.7	26.5	30.1	217.8	48.8
av	9.2	66.5	15.2	20.1	171.4	30.6	5.8	72.3	14.0	13.3	110.8	24.6

Table 3: BDDs built in terms of internal signals, k=4

References

[1] R.Burch, F.N. Najm, P.Yang, and T.N.Trick, "A Monte Carlo Approach for Power Estimation", *IEEE Tran. on VLSI*, Mar. 1993, pp. 63-71.

[2] David I. Cheng, M. Marek-Sadowska, and K.T. Cheng, "Speeding Up Power Estimation by Topological Analysis," *Proc. CICC-95*, pp.623-626.

[3] David I. Cheng, K.T. Cheng, D.C. Wang, and M. Marek-Sadowska, "A Hybrid Methodology for Power Estimation," *Tech Report, ECE Dept., UC, Santa Barbara.*

[4] A.Ghosh, S.Devadas, K.Keutzer, and J.White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," *Proc. DAC-92*, pp. 253-259.

[5] R. Marculescu, D. Marculescu, and M. Pedram, "Switching Activity Analysis Considering Spatiotemporal Correlations," *Proc. ICCAD-94*, pp. 294-299.

[6] J.Monteiro, S.Devadas, B.Lin, C.Tsui, and M.Pedram, "Exact and Approximate Methods of Switching Activity Estimation in Sequential Logic Circuits," *Proc. Int. Workshop on Low Power Design*, pp. 117-122, Apr. 1994.

[7] F.N. Najm, "Transition Density, A Stochastic Measure of Activity in Digital Circuits," *Proc. DAC-91*, pp. 644-649.

[8] E.Sentovich et al., "SIS:A System for Sequential Circuit Synthesis" *Memo. UCB/ERL M92/41, UC, Berkeley.*

[9] S.C.Seth, B.B.Bhattacharya, V.D.Agrawal, "An exact analysis for efficient computation of random pattern testability in combinational circuits," *Proc. Fault-Tolerant Comput. Symp.*, pp. 318-323, 1986.

[10] C.Y. Tsui, M. Pedram, A.M Despain, "Efficient Estimation of Dynamic Power Consumption under a Real Delay Model," *Proc. ICCAD-93*, pp. 224-228.

[11] M. Xakellis and F. Najm, "Statistical Estimation of the Switching Activity in Digital Circuits," *Proc. DAC-94*, pp. 728-733.