Low-power optimization techniques for BDD mapped circuits using temporal correlation

Techniques d'optimisation pour les faibles puissances pour des diagrammes de décision binaire utilisant la corrélation temporelle

Rolf Drechsler, Mikael Kerttu, Per Lindgren, and Mitchell Thornton*

In modern design flows low-power aspects should be considered as early as possible to minimize power dissipation in the resulting circuit. A new binary decision diagram–based design style that considers switching activity optimization using temporal correlation information is presented. The technique is based on an approximation method for switching activity estimation. In the case of finite state machines, the presented method extracts signal statistics by means of Markov chain analyses. Experimental results on a set of MCNC and ISCAS89 benchmarks show the estimated reduction in power dissipation.

Les aspects relatifs aux faibles puissances devraient être pris en compte dès les premières phases du design en vue de minimiser la dissipation de puissance du circuit résultant. Cet article présente une méthode de design basée sur un diagramme de décision binaire qui traite l'optimisation des commutations via l'information de corrélation temporelle. L'approche repose sur une approximation de l'estimation de l'activité de commutation. Dans le cas des machines à états finis, la méthode extrait les statistiques du signal via une analyse par chaînes de Markov. Des résultats expérimentaux obtenus avec des données de banc d'essai MCNC et ISCAS89 montrent la réduction estimée de la dissipation de puissance.

I. Introduction

The importance of low-power optimization is growing due to the increased use of battery-powered embedded systems. In order to optimize for low power dissipation, statistical information about the behaviour of the system can be exploited. The switching activity of a circuit node in a CMOS digital circuit directly contributes to the overall dynamic power dissipation. Temporal correlation of the occurring input signals can have a significant effect on the switching activity and hence the power consumption [1]. Modern design flows should consider these effects from the very beginning.

Several synthesis tools make use of binary decision diagrams (BDDs) [2]–[4], an efficient data structure used for solving many of the problems occurring in VLSI CAD. BDDs can be directly transformed into circuits if each node of the underlying graph is substituted with a multiplexer. An approach for BDD mapping that also considers low-power aspects has recently been proposed in [5]. The method combines logic synthesis, area minimization and low-power optimization together with mapping in a single pass. This approach eliminates the need for circuit extraction and back annotation common in many traditional synthesis methods. However, the activity estimation method used lacks the ability to exploit temporal correlation information. This limitation can severely affect optimization for low power in cases where strong temporal correlation of input signals is present.

The problem of switching activity minimization using temporal correlation information is addressed in this work. A novel BDD-based approximation method is described, and we show how it can be combined with the approach in [5]. The power dissipation estimate for a mapped BDD node is based on its switching activity and its fanout (corresponding to the capacitive load). The resulting circuit is realized by mapping BDD nodes to multiplexer circuits implemented using CMOS transmission gates and static inverters. Similar BDD mapping methods based on pass transistor logic (PTL) circuits [6]–[7] can also be used. The proposed switching activity estimation method has been validated by transistor-level simulations, showing that the power dissipation due to switching is dominated by the switching of the multiplexer outputs and (as the model used here assumes) that the contribution from internal switching in the multiplexers can be neglected.

To permit calculation of the power dissipation, the capacitive load of all nodes is also estimated. This problem is handled by using the inherent structure of BDD mapped circuits. This allows for devising a computationally efficient cost function for low-power optimization. The synthesis technique utilizes statistical properties of the primary inputs that can be obtained by functional simulation. An analytic method for extracting statistical properties for next-state signals of circuits modelled as finite state machines (FSMs) is described. In this way, the need for computationally expensive gate-level simulation is avoided, and signal statistics are utilized for low-power synthesis.

II. Switching activity estimation

In this section an introduction to signal switching activity estimation is given. (For more details, see [8].) In the following it is assumed that the input signals are mutually independent (spatially uncorrelated) and that the signals can be modelled as strict-sense stationary (SSS) and mean-ergodic with zero delay [8]; that is, all switching is carried out simultaneously, and signal probability values and switching activity do not vary over time. $P(f)$ denotes the probability that f is 1 (the output probability of f), and $a(f)$ denotes the activity for f (the probability that f will change in value from one cycle to the next).

*Rolf Drechsler is with the Department of Computer Science, University of Bremen, 28359 Bremen, Germany. E-mail: drechsle@informatik.uni-bremen.de. Mikael Kerttu and Per Lindgren are with EISLAB/Department of Computer Engineering, Luleå University of Technology, Luleå, Sweden. E-mail: {kerttu,pln}@sm.luth.se. Mitchell Thornton is with the Department of Computer Science and Engineering, Southern Methodist University, Dallas, Texas, U.S.A. E-mail: mitch@engr.smu.edu.

In order to devise an improved low-power synthesis method for BDD mapped circuits, an accurate and computationally efficient switching activity estimation method is needed that is able to utilize temporal correlation. To avoid the high computational complexity of an exact method, it is assumed that there is no spatial correlation between the Shannon cofactors of the function of interest. The approximation technique provides the exact result for the case where the cofactors are spatially uncorrelated. In the case where cofactors are positively correlated an overestimate is obtained, since the switching of a top variable is less prone to cause a true switching of the node's output. The opposite holds for negatively correlated cofactors. This observation allows for the application of Theorem 3.1 from [8]. The formula in (1) can then be derived using the multiplexer-based circuit model:

$$\begin{aligned} a(f) = & \left(\frac{P(f_0) + P(f_1) - 2P(f_0)P(f_1)}{1 - (a(f_0) + a(f_1) - a(f_0)\,a(f_1))} \right. \\ & \left. - \frac{\frac{1}{2}(a(f_0) + a(f_1) - a(f_0)\,a(f_1))}{1 - (a(f_0) + a(f_1) - a(f_0)\,a(f_1))} \right) \\ & \times a(v)\,(1 - a(f_0))\,(1 - a(f_1)) \\ & + \frac{1 - P(v) - \frac{a(v)}{2}}{1 - a(v)} \times a(f_0)\,(1 - a(v))\,(1 - a(f_1)) \\ & + \frac{P(v) - \frac{a(v)}{2}}{1 - a(v)} \times a(f_1)\,(1 - a(v))\,(1 - a(f_0)) \\ & + \frac{1}{2} a(v)\,a(f_0)\,(1 - a(f_1)) \\ & + \frac{1}{2} a(v)\,a(f_1)\,(1 - a(f_0)) \\ & + a(f_0)\,a(f_1)\,(1 - a(v)) \\ & + \frac{1}{2} a(v)\,a(f_0)\,a(f_1). \end{aligned} \tag{1}$$

In (1), v is the input variable, f_0 is the low cofactor, and f_1 is the high cofactor. This formula is used recursively in a bottom-up approach to calculate the activity for each node in the BDD.

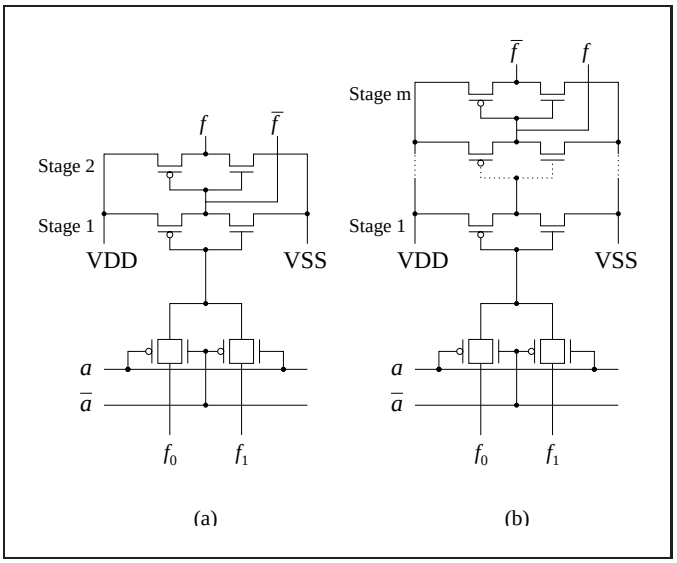

Figure 1: *BDD node mapping into multiplexer circuits.*

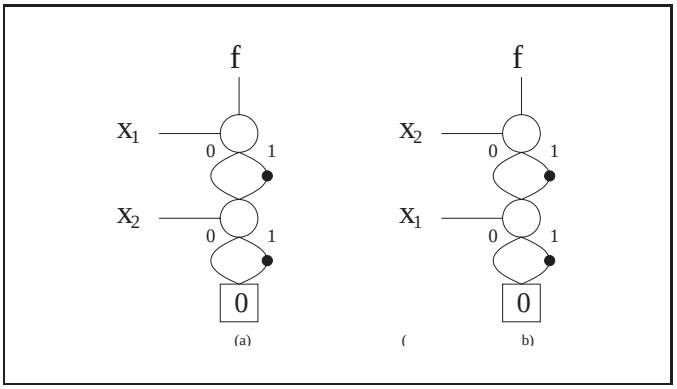

Figure 2: *Variable swap.*

III. Low-power synthesis

In this section, BDD mapping, power dissipation modelling and approximation characteristics are described. Furthermore, the proposed heuristic optimization technique based on the sifting algorithm is shown.

A. BDD mapped circuits

A BDD can be directly mapped to a multiplexer-based circuit as described in [9], to a "timed" circuit as described in [10], or to a "pass transistor"–based circuit as described in [6], [7] and [11]. In all cases, the resulting circuit can be considered to be one that is obtained by replacing BDD vertices with small subcircuits and BDD edges with wires. It is known that the diagram size (and therefore the circuit complexity) is sensitive to the ordering of the function variables. The complexity may vary from linear to exponential under different orderings for some functions. Both exact and heuristic methods have been developed to tackle this problem. However, in the work described here we are concerned not only with the complexity of the circuit resulting from a BDD, but to an even greater extent with the power dissipation.

A method for low-power synthesis of BDD mapped circuits was first introduced in [5]. The power dissipation of each node was computed by the estimated switching activity and the node's fanout. The variable order of the underlying BDD was shown to influence not only the area (number of nodes) but also the internal switching activity. An optimization algorithm based on local variable exchange (sifting) was proposed. Since the switching activity estimate, and therefore the cost function, was implementable solely by local operations on the diagram, the method was shown to be computationally effective. However, the estimation technique did not consider any temporal signal correlation. The technique can also be used with BDDs using complemented edges. The use of complemented edges has been shown to both reduce BDD complexity and improve performance of operations [2], [4]. The statements above apply for BDDs using complemented edges, taking into account the following observations:

1. The output probability, $P[\overline{f}]$, of $\overline{f}$ is equal to $1 - P[f]$.
2. The switching activity, $a[\overline{f}]$, of $\overline{f}$ is equal to $a[f]$.

These properties are used to compute local switching probabilities during variable exchange operations on BDDs with complemented edges.

B. Power dissipation modelling

A cost model based on the total circuit switching activity under a given set of dependent-variable output probabilities is defined. The dependent variables are denoted as *support* variables. We attempt to minimize the sum of all internal switching activities at each BDD vertex. The approach then maps each BDD node into a multiplexer-based circuit as shown in Fig. 1. The number of stages of active buffers is determined by the fanout of each BDD node, which is equivalent to the number of BDD edges pointing to the node.

The power dissipation for the mapped node n is estimated using the relationship in (2):

$$\mathrm{PD}_n = a(n) \times \mathrm{driver}(\mathrm{fanout}(n)) + \mathrm{leakage}(n). \tag{2}$$

Table 1
Power dissipation of external switching vs. internal switching

	Power dissipation
Switch v, $f_0 = 0$, $f_1 = 1$	18 827
Switch v, f_0, f_1 (f = stable)	14
$v = 0$, switch f_0, $f_1 = 0$	10 688
$v = 0$, $f_0 = 0$, switch f_1	22

Table 2
Estimated switching activity for each BDD node and total estimated power dissipation with BDD variable order as shown in Fig. 2(a)

	$a(f)$	$a(f_0)$	$a(f_1)$	Power dissipation
Overestimation [8]	~ 1.42	~ 0.75	~ 0.75	2.92
Exact estimation [8]	~ 0.67	~ 0.75	~ 0.75	2.17
Probabilistic [5]	0.5	0.5	0.5	1.5
MUX approximation	0.58	0.75	0.75	2.08

Table 3
Estimated switching activity for each BDD node and total estimated power dissipation with BDD variable order as shown in Fig. 2(b)

	$a(f)$	$a(f_0)$	$a(f_1)$	Power dissipation
Overestimation [8]	~ 1.42	~ 0.67	~ 0.67	2.75
Exact estimation [8]	~ 0.67	~ 0.67	~ 0.67	2
Probabilistic [5]	0.5	0.5	0.5	1.5
MUX approximation	0.54	0.67	0.67	1.88

Equation (2) was validated by conducting transistor-level simulations using models from a commercially available CMOS process. The results (see Table 1) show that the power dissipation of external switching (driving the fanout load capacitance) dominates over the internal switching in the multiplexer by a factor of over 100 to 1 under unity load (a single fanout). Thus, the effect of internal switching can be disregarded.

Capacitive load and leakage parameters are strongly process-dependent. In the following, leakage current is ignored and driver power dissipation is assumed to be linear with the fanout (capacitive load). Any parasitic capacitances due to routing are also ignored. The power dissipation from the buffering of input signals is not considered in this model.

C. Approximation characteristics

The switching activity estimation method described in (1) is analyzed further to show various properties and demonstrate how it can be applied to low-power synthesis for BDD mapped circuits. The total power dissipation of the mapped circuit is computed as

$$\mathrm{PD}_{\mathrm{tot}} = \sum_{\forall n} a(n) \times \mathrm{driver}(\mathrm{fanout}(n)) + \mathrm{leakage}(n). \qquad (3)$$

Consider the XOR function $f = x_1 \oplus x_2$, given the input probabilities $P(x_1) = 1/2$, $P(x_2) = 1/2$, and the switching activities $a(x_1) = 2/3$, $a(x_2) = 3/4$ as shown in Fig. 2. Table 2 shows the estimated switching activity for each BDD node f, f_0, and f_1, and the total estimated power dissipation. As shown in the table, the technique labelled *Probabilistic* leads to an underestimation, while the proposed multiplexer-based approximation (*MUX approximation*) comes closer to the exact result.

When the BDD variable order is changed as shown in Fig. 2(b), the switching activities are swapped, and the overall power dissipation for the exact method is reduced to 2. Also, the other approximation methods indicate a reduction, as shown in Table 3 (except for the probabilistic approach, which is unable to utilize the signal activity information).

The switching estimate labelled *Probabilistic* in Table 3 is computed solely by local operations on the BDD. However, the approximation technique labelled *MUX approximation* that is proposed here also considers the approximated switching activity of each node's successors, so that the local condition no longer holds. This implies that after a local variable exchange, switching activity estimates need to be propagated toward preceding levels in the diagram. While this approach leads to more complexity in the switching activity estimation algorithm, CPU times are reasonable for the set of benchmark functions used in the experiments.

D. Heuristic minimization algorithm

The proposed heuristic minimization algorithm iteratively seeks a variable order that reduces the mapped circuits' switching activity, weighted by the fanout cost for each node. This procedure is outlined in pseudocode as follows:

```
D_min() {
1 compute D_sw[_total]
2 for each variable {
3   sift to position minimizing D_sw[_total]
4 } repeat until no further improvement
}
```

The sifting and recalculation of output probabilities and switching activities is performed solely through local operations on the BDD representation. The total estimated power dissipation due to switching ($D_{\mathrm{sw}}[\_\mathrm{total}]$) can also be updated by local operations on the two levels sifted (*upper* and *lower*) and nodes connecting to the sifted levels (*below*). By maintaining reference counters (i.e., the number of incoming edges) for each node, the effect of fanout changes for nodes below in the diagram can be handled. The following pseudocode shows how the total switching activity is updated during sifting:

```
D_sift(upper, lower) {
1 D_sw[_total] -= (D_sw[_upper]
  + D_sw[_lower] + D_sw[_below])
2 ref_remove_edges_to(upper,lower)
3 perform local variable exchange
4 ref_add_edges_to(upper,lower)
5 D_sw[_total] += (D_sw[_upper]
  + D_sw[_lower] + D_sw[_below])
}
```

In line 1 above, the contribution of the two levels to be sifted ($D_{\mathrm{sw}}[\_\mathrm{upper}] + D_{\mathrm{sw}}[\_\mathrm{lower}]$) and the contribution of fanouts from connecting nodes ($D_{\mathrm{sw}}[\_\mathrm{below}]$) are subtracted. The number of references for connecting nodes is updated (line 2) before sifting is applied (line 3). After the variable exchange is performed, the reference counters of the connecting nodes (line 4) are updated and the total estimated power dissipation in line 5 is computed. Due to the variable exchange, switching activities and reference counters may change, thereby also changing the estimated power dissipation $D_{\mathrm{sw}}[\_\mathrm{total}]$.

Example 1 *Fig. 3(a) shows a portion of a BDD before sifting. The number at each node denotes the number of incoming edges (the fanout in a multiplexer-based mapping). Before the fanout is sifted, changes of the lower level in the BDD shown in Fig. 3(b) need to be deter-*

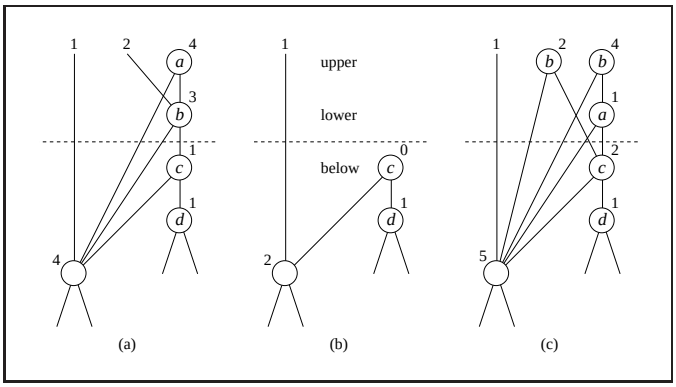

Figure 3: *Reference-count update during sifting.*

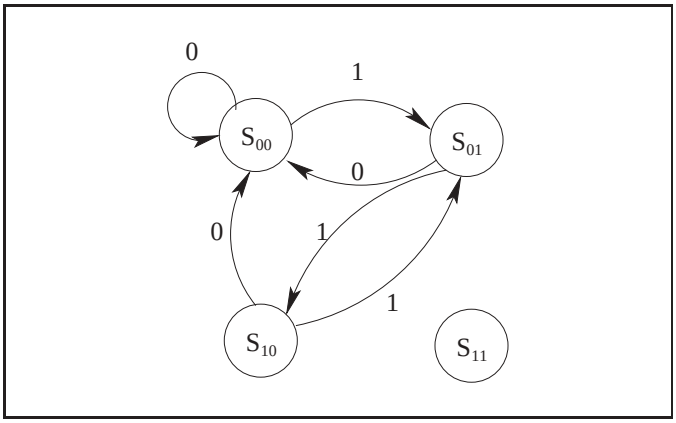

Figure 4: *FSM states.*

mined. Note that only nodes connecting to the upper *and* lower *levels are updated. After sifting is performed, the new fanout values (reference counters) of the connecting nodes are computed as illustrated in Fig. 3(c).*

IV. FSM analysis

The optimization algorithm described here utilizes the statistical information of the input signals. The ability to gather this information is essential for optimizing for low power. The signal properties for the next-state vector are defined using the FSM transition relation together with the properties of the primary input signals. In this section a method to extract this information by modelling the FSM behaviour as a Markov chain [12] is described. There are several approaches for efficient FSM spanning [13]. In this work the spanning function is implemented in a straightforward way by a depth-first recursive algorithm, which also calculates the transition probability matrix represented by an algebraic decision diagram (ADD) in the same pass. In [12] and [14], ADDs were used to represent the transition probability matrix, and the steady-state probabilities were calculated in an efficient way. The calculations described here were implemented using ADDs in an iterative manner, as described in the following.

A. Span FSM states

The BDD representing the next-state functions is used to span the FSM (see Fig. 4). Starting from the reset state, each possible new state is recursively visited (depth first) until an already visited state is reached. During the recursion, a transition probability matrix is constructed. This usually sparse matrix is efficiently represented by an ADD. The matrix is addressed with the current state as the columns and the next state as the rows. The value in each entry in the matrix (corresponding to an ADD leaf) represents the probability of a transition from a current state to a next state.

Example 2 *When the transition probabilities are calculated, the matrix is initially empty and new entries are added during the recursion. We assume the probability that an input signal I is at a logic-*1 *value to be* $1/4$ *(i.e., $P(I) = 1/4$):*

$$\begin{array}{c} \\ \mathrm{NS}_{00} \\ \mathrm{NS}_{01} \\ \mathrm{NS}_{10} \\ \mathrm{NS}_{11} \end{array} \begin{array}{c} \begin{array}{cccc} \mathrm{CS}_{00} & \mathrm{CS}_{01} & \mathrm{CS}_{10} & \mathrm{CS}_{11} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}.$$

There is a transition from state 00 *to state* 01*, and the probability of $P(I)$ is added to row* 01 *and column* 00*:*

$$\begin{array}{c} \\ \mathrm{NS}_{00} \\ \mathrm{NS}_{01} \\ \mathrm{NS}_{10} \\ \mathrm{NS}_{11} \end{array} \begin{array}{c} \begin{array}{cccc} \mathrm{CS}_{00} & \mathrm{CS}_{01} & \mathrm{CS}_{10} & \mathrm{CS}_{11} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}.$$

Finally, after all reachable states are found, the complete matrix is represented:

$$\begin{array}{c} \\ \mathrm{NS}_{00} \\ \mathrm{NS}_{01} \\ \mathrm{NS}_{10} \\ \mathrm{NS}_{11} \end{array} \begin{array}{c} \begin{array}{cccc} \mathrm{CS}_{00} & \mathrm{CS}_{01} & \mathrm{CS}_{10} & \mathrm{CS}_{11} \end{array} \\ \begin{bmatrix} 3/4 & 3/4 & 3/4 & 0 \\ 1/4 & 0 & 1/4 & 0 \\ 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}.$$

B. Calculation of state probabilities

The ADD obtained by spanning the FSM is used to calculate the steady-state probabilities for each state. The FSM is viewed as a Markov chain [12], [14] and is used in the calculation of the state probabilities. The ADD is multiplied with an initial state probability vector. Equation (4) describes this operation mathematically:

$$A\bar{x} = \bar{x}', \tag{4}$$

where A is the matrix represented by the ADD, and $\bar{x}$ and $\bar{x}'$ are the steady-state probability vectors after the iterations. The iteration terminates when $\bar{x}$ and $\bar{x}'$ are within the specified tolerance from each other. The resulting $\bar{x}'$ contains the resulting steady-state probability vector.

Example 3 *The state probability vector is initialized such that each state entry has the value* $1/nr_{\mathrm{reachablestates}}$ *except for the unreachable state entries, which have the value* 0*:*

$$\begin{bmatrix} 3/4 & 3/4 & 3/4 & 0 \\ 1/4 & 0 & 1/4 & 0 \\ 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \\ 0 \end{bmatrix} = \begin{bmatrix} 3/4 \\ 1/6 \\ 1/12 \\ 0 \end{bmatrix}, \tag{5}$$

$$\begin{bmatrix} 3/4 & 3/4 & 3/4 & 0 \\ 1/4 & 0 & 1/4 & 0 \\ 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 3/4 \\ 1/6 \\ 1/12 \\ 0 \end{bmatrix} = \begin{bmatrix} 3/4 \\ 0.21 \\ 0.04 \\ 0 \end{bmatrix}. \tag{6}$$

Finally the solution is found when the result vector is equal to the vector from the previous iteration:

$$\begin{bmatrix} 3/4 & 3/4 & 3/4 & 0 \\ 1/4 & 0 & 1/4 & 0 \\ 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 3/4 \\ 1/5 \\ 1/20 \\ 0 \end{bmatrix} = \begin{bmatrix} 3/4 \\ 1/5 \\ 1/20 \\ 0 \end{bmatrix}. \tag{7}$$

Table 5
ISCAS89 benchmarks

	Area-optimized		Non-FSM-optimized		FSM-optimized		Percent change	
Name	Size	PD	Size	PD	Size	PD	Size	PD
s208.1	40	25	40	25	64	19	60	−24
s27	9	4.1	9	4.1	9	4.1	0	0
s298	73	4.3	74	4.2	77	2.9	5.4	−33
s344	103	12	108	19	148	3.4	44	−72
s349	103	12	108	19	127	3.3	23	−73
s382	120	2.1	122	2.1	120	2.1	0	0
s386	113	44	114	41	114	39	0	−11
s400	120	2.1	122	2.1	120	2.1	0	0
s444	150	43	161	19	156	2.1	4	−95
s510	163	118	168	81	153	61	− 6.1	−48
s526	137	8.4	139	8.1	136	4.6	− 0.7	−55
s641	398	81	384	77	1149	15	289	−81
s713	398	81	384	77	1149	15	289	−81
s820	219	172	261	149	280	108	28	−37
s832	219	174	261	148	294	103	34	−41

Table 4
Area optimization vs. low-power optimization

		Area-optimized		Low-power-optimized	
Name	In/Out	Prob	Mux	Prob	Mux
5xp1	7/10	32.2	40.4	30.2	25.1
add6	12/7	23.0	20.3	23.0	20.3
apex7	49/37	175.6	209.8	158.4	165.1
bc0	26/11	320.0	330.8	310.3	229.7
chkn	29/7	132.0	151.1	84.8	33.2
duke2	22/29	106.9	129.2	103.8	75.9
exp	8/18	79.5	81.5	61.6	42.4
in2	19/10	115.5	115.5	95.5	67.5
in7	26/10	21.9	23.5	20.1	16.8
inc	7/9	45.4	54.4	45.3	24.6
intb	15/7	349.3	313.8	305.4	256.6
misex3	14/14	223.9	234.9	223.9	203.8
sao2	10/4	35.8	37.2	34.2	16.6
tial	14/8	422.6	453.2	422.6	362.9
vg2	25/8	50.0	46.9	50.0	46.3
x6dn	39/5	143.1	115.2	124.5	96.0
Sum		2276.8	2357.7	2093.6	1682.8

The steady-state probabilities (P_{SS}) are shown below:

$$\begin{aligned} P_{SS}(S[1:0]=00) &= P_{SS}(00) = 3/4, \\ P_{SS}(S[1:0]=01) &= P_{SS}(01) = 1/5, \\ P_{SS}(S[1:0]=10) &= P_{SS}(10) = 1/20, \\ P_{SS}(S[1:0]=11) &= P_{SS}(11) = 0. \end{aligned} \tag{8}$$

C. Extracting signal statistics

The transition probability matrix and the steady-state probability vector can be used to calculate the bit probability and the switching activity of the next-state bits. This is accomplished using the ADD and (9):

$$(\forall i) P(NS[i:i]) = \sum_{\forall S[N-1:0] \in S[i:i]=1} P_{SS}(S[N-1:0]). \tag{9}$$

To calculate the activity for each bit, the ADD with the state transition probabilities and the steady-state probabilities calculated earlier is used. $P_{SS}(n)$ denotes the steady-state probability for state n, where $n[i:i]$ is the i-th bit of vector n, A is the matrix containing the state transition probabilities ($A[NS_k][CS_n] = P(NS_k \mid CS_n)$), and $a(NS[i:i])$ is the activity for the next-state bit i and is given by the formula in (10):

$$(\forall i) a(NS[i:i]) = \sum_{\forall n} P_{SS}(n) \times \sum_{\forall k \in (k[i:i] \neq n[i:i])} P(NS_k \mid CS_n). \tag{10}$$

V. Experimental results

The implementation of this technique is based on the CUDD 2.3.0 [15] package for BDD manipulation. All experiments were run on a SUNW,Ultra-5/10 platform running at 333 MHz with 768 MB RAM. No automatic variable reordering was enabled [15].

Benchmark circuits were synthesized using the low-power optimizations described here and also those for optimizing with respect to area minimization. Compared to the previous approach in [5], further power reductions were obtained since this method incorporates the use of temporal signal correlations. As shown in Table 4, the average power estimate reduction for the synthesis method described here is 30% compared to that for the area-optimized circuit. These results were obtained assuming a large activity deviation ($P = 0.5$ and a_i has alternating values of $0.1, 0.9, 0.1, \ldots$). Using the same assumptions with the method in [5] resulted in a power reduction of only 8.3% compared to the area-optimizer results.

Table 4 is organized as follows: *Area-optimized* denotes circuit optimization for minimum area (minimum number of BDD nodes), and *Low-power-optimized* denotes the optimization technique presented in this paper. The two subcolumns denote the estimation technique applied. *Prob* is a probability-based algorithm that does not consider temporal correlation [5]. Finally, the estimation technique described in this paper is found in the column *Mux*.

Furthermore, we have analyzed a set of ISCAS89 finite state machine benchmarks and extracted statistical information as described in Section IV and used this information within the synthesis tool. As shown in Table 5, the results indicate an average power estimate reduction of 43% using the new method proposed here compared to the

results of the area-optimized method. The power estimate reductions range from 0% to 95%. The majority of the tests show a significant power estimate reduction for the FSM-optimized circuits compared with the area-optimized ones. The results also show that the power-optimized circuits have an increased area of 51% on average over the area-optimized circuit. In two cases the power optimizer synthesized smaller circuits than the area optimizer. This outcome is due to the heuristic algorithm that the area optimizer utilizes, which may cause it to get stuck in a local minimum.

VI. Conclusions

A synthesis method that reduces the dynamic power dissipated in a CMOS circuit obtained using a BDD mapping technique was presented. The technique utilizes a switching activity estimate that is based on the structure of the subcircuit used to represent each BDD node. Furthermore, temporal correlation statistics were extracted from the transition functions of a finite state machine and also included in the low-power optimization technique. Experimental results show an average decrease in estimated power dissipation of 30% compared to circuits synthesized with area minimization for combinational benchmarks, and an average decrease of 43% for sequential benchmarks.

Acknowledgements

This work was supported in part by the National Science Foundation under grants CCR-0000891 and CCR-0097246.

References

[1] R. Marculescu, D. Marculescu, and M. Pedram, "Efficient power estimation for highly correlated input streams," in *Proc. Design Automation Conf.*, 1995.

[2] K.S. Brace, R.L. Rudell, and R.E. Bryant, "Efficient implementation of a BDD package," in *Proc. Design Automation Conf.*, 1990, pp. 40–45.

[3] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, vol. 35, no. 8, 1986, pp. 677–691.

[4] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagrams with attributed edges for efficient Boolean function manipulation," in *Proc. Design Automation Conf.*, 1990, pp. 52–57.

[5] P. Lindgren, M. Kerttu, M. Thornton, and R. Drechsler, "Low power optimization technique for BDD mapped circuits," in *Proc. ASP Design Automation Conf.*, 2001, pp. 615–621.

[6] P. Buch, A. Narayan, A.R. Newton, and A.L. Sangiovanni-Vincentelli, "Logic synthesis for large pass transistor circuits," in *Proc. Int. Conf. CAD*, 1997, pp. 663–670.

[7] C. Scholl and B. Becker, "On the generation of multiplexer circuits for pass transistor logic," 2000.

[8] K. Roy and S. Prasad, *Low-Power CMOS VLSI Circuit Design*, Hoboken, N.J.: Wiley Interscience, 2000.

[9] S.B. Akers, "Binary decision diagrams," *IEEE Trans. Comput.*, vol. 27, 1978, pp. 509–516.

[10] L. Lavagno, P. McGeer, A. Saldanha, and A.L. Sangiovanni-Vincentelli, "Timed Shannon circuits: A power-efficient design style and synthesis tool," in *Proc. Design Automation Conf.*, 1995, pp. 254–260.

[11] V. Bertacco, S. Minato, P. Verplaetse, L. Benini, and G. De Micheli, "Decision diagrams and pass transistor logic synthesis," in *Proc. Int. Workshop Logic Synth.*, 1997.

[12] G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Markovian analysis of large finite state machines," *IEEE Trans. Comput.*, vol. 15, no. 12, 1996, pp. 1479–1493.

[13] G. Cabodi, P. Camurati, and S. Quer, "Improving symbolic reachability analysis by means of activity profiles," *IEEE Trans. Comput.*, vol. 19, no. 9, 2000, pp. 1065–1075.

[14] G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Probabilistic analysis of large finite state machines," in *Proc. Design Automation Conf.*, 1994, pp. 270–275.

[15] F. Somenzi, CUDD: CU Decision Diagram Package Release 2.3.0, University of Colorado at Boulder, Boulder, Colo., 1998.