Low Power Optimization Techniques for BDD Mapped Circuits Using Temporal Correlation

Rolf Drechsler
Computer Science
Univeristy of Bremen
28359 Bremen, Germany
drechsle@informatik.uni-bremen.de

Mikael Kerttu Per Lindgren
EISLAB/Computer Engineering
Luleå University of Technology
Luleå, Sweden
{kerttu,pln}@sm.luth.se

Mitch Thornton
Computer Engineering
Mississippi State University
Mississippi State, MS, USA
mitch@ece.msstate.edu

Abstract

We address the problem of switching activity optimization under temporal correlation. We propose a novel BDD based approximative method for switching activity estimation. Experimental results on a set of MCNC benchmarks show the estimated potential power dissipation reduction by exploiting temporal correlation using proposed method.

1 Introduction

The importance of low power optimization is emphasized with the increased use of battery-powered embedded systems. In order to optimize for low power dissipation we may exploit statistical information about the behavior of the system. The switching activity of a circuit node in a CMOS digital circuit directly contributes to the overall power dissipation. Temporal correlation on the occurring signals can have significant effect on the switching activity and hence the power consumption [12]. *Binary Decision Diagrams* (BDDs) [3, 4, 13] offer efficient means for solving many of the problems occurring in VLSI CAD. A promising approach for low power synthesis of BDD mapped circuits utilizes switching activity estimation for circuit optimization [11]. The approach combines logic synthesis, area minimization and low power optimization together with mapping in a single pass. This approach surpasses the need for circuit extraction and back annotation common to traditional synthesis methods. However, the activity estimation method used lacks the ability to exploit temporal correlation information. This may severely mislead optimization in cases where strong temporal correlation is present.

In this paper we address the problem of switching activity minimization using temporal correlation information. We propose a novel BDD based approximative method and show how it can be applied to low power synthesis of BDD mapped circuits. The power dissipation estimate for a mapped BDD node is based on its switching activity and fanout (capacitive load). The chosen model corresponds to a *Pass Transistor Logic* (PTL) circuits obtained by mapping BDD nodes to PTL multiplexers [15]. The proposed estimation method has been validated by transistor level simulations, showing that the power dissipation due to switching is dominated by the switching of the multiplexor outputs and (as our model assumes) the contribution from internal switching in the multiplexors can be neglected.

To be able to calculate the power dissipation we also need to estimate the capacitive load of all nodes. We approach this problem by using the inherent structure of BDD mapped circuits. This allows us to devise a computationally efficient cost function for low power optimization.

Our synthesis technique utilizes statistical properties of the primary inputs. These can be obtained by efficient functional simulation. We describe an analytic method for extracting statistical properties for next state signals of FSM circuits. In this way, the need for computationally expansive gate level simulation is surpassed, while signal statistics are utilized for low power synthesis.

2 Switching Activity Estimation

In this section we give a review and introduction to signal switching activity estimation. We refer the interested reader to [14] for an excellent in-depth elaboration on the subject. In the following we assume the input signals to be mutually independent (spatially uncorrelated) and that the signals can be modeled as strict-sense stationary (SSS) and mean-ergodic with zero delays [14]. That is, all switching is carried out simultaneously and that signal probability and switching activity does not vary over time. $P(f)$ denotes the probability of f being 1, i.e. the output probability of f. $a(f)$ denotes the activity for f, i.e., the probability of f changing value from one cycle to the next.

In order to devise an improved low power synthesis method for BDD mapped circuits we seek accurate and computationally efficient switching activity estimation methods able to utilize temporal correlation information.

In order to avoid the computational complexity of the exact method [14], we assume that there is no spatial correlation between the cofactors (successors). Thus, our approximation renders the exact result for the case that the cofactors are spatially uncorrelated. In the case where cofactors are positively correlated we obtain an overestimate, as a top variable switching is less prone to cause a true switching of the node's output. The opposite holds for negatively correlated cofactors.

This allows us to apply Theorem 3.1 from [14]. The multiplexor model gives the following derived formula for a single BDD node.

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

3 BDD Mapped Circuits

A BDD can be directly mapped to a multiplexor based circuit as described in [1], to a "timed" circuit as described in [10] or to a "pass-transistor" based circuit as described in [2, 5, 11, 15]. In all cases, the resulting circuit can be considered to be one that is obtained by replacing BDD vertices with small sub-circuits 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 (which represent circuit input signals), and may vary from linear to exponential under different orderings for some functions. Both exact and heuristic methods have been

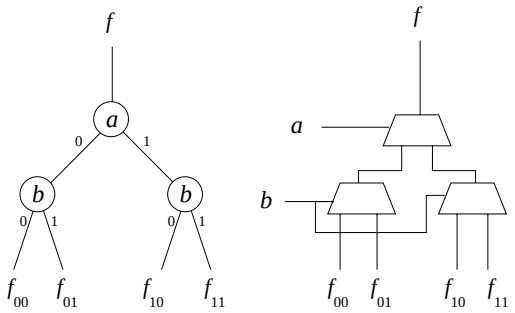

Figure 1: BDD mapped to Multiplexor net-list

developed to tackle this problem. However, in this paper we are not only concerned with the complexity of the circuit resulting from a BDD, but to an even greater extent, the power dissipation.

3.1 Complemented Edges

The use of complemented edges has shown both to reduce BDD complexity and improve performance of operations, [13, 3]. The statements above apply for BDDs using complemented edges by making 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 $a[f]$.

We can utilize these properties to compute local switching probabilities during variable exchange operations on BDDs with complemented edges.

4 Low Power Synthesis

A method for low power synthesis of BDD mapped circuits was first introduced in [11]. The power dissipation of each node was computed by the estimated switching activity and the node's fanout (capacitive load). 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 devised. Since the switching activity estimate, and therefore the cost function, could be implemented solely by local operations on the diagram, the method was shown to be computaionally effective. However, the estimate does not consider any temporal signal correlation, which may severely mislead the optimization into suboptimal circuits.

4.1 Power Dissipation Modeling

We define the cost model based on the total circuit switching activity under a given set of dependent variable output probabilities. In the following we denote the dependent variables as *support* variables. We attempt to minimize the sum of all internal switching probabilities at each BDD vertex.

This model has some assumptions. We assume that the input signals to the resultant circuit are statistically independent. This assumption allows us to use the switching probability computed in the BDD representation as an estimate for the actual switching activity of a circuit.

The approach from [11] is based on the the mapping of each BDD node into a PTL multiplexor circuit, see Figure 2 (the number of driver stages increases with fanout (b) as to balance the speed of the circuit).

We estimate the power dissipation for the mapped node as:

$$PD_n = a(n) * driver(fanout(n)) + leakage(n) \qquad (2)$$

To validate the above formula we have conducted transistor level simulations using models from a commercially available CMOS process. Our results show that the power dissipation of external switching (driving the fanout load capacitance) dominates over the internal switching in the

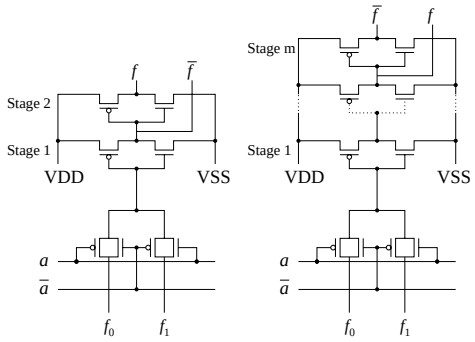

Figure 2: BDD node mapping into PTL multiplexor circuits

multiplexor by a factor of over a 100 to 1 under unity load (a single fanout). Thus, the effect of internal switching can be disregarded.

	Power
switch x, $f_0 = 0, f_1 = 1$	18827
switch x, $f_0, f_1 (f = stable)$	14
x=0, switch $f_0, f_1 = 0$	10688
x=0,$f_0 = 0, switch f_1$	22

Capacitive load and leakage parameters are strongly process dependent. In the following, we have assumed 0 leakage and driver power dissipation to be linear with the fanout (capacitive load). We do not consider any parasitic capacitances due to routing.

4.2 CMOS PTL Mapping

As mentioned in Section **??** a circuit can be derived from a structural mapping of the BDD. In order to verify our minimization method we have developed a simple mapping tool for PTL based CMOS circuits. Each BDD node is mapped to a sub-circuit shown in Figure 2. The select signal (input) is present in both polarities a and $\overline{a}$. f and $\overline{f}$ are always computed (no optimization is applied). The driver is chosen according to the total fan-out of the node. The number of inverter stages ranges from 2 (as shown in Figure 2) (a) and upwards (b) in Figure. The transistor sizings for each inverter stage are given in nominal values for the process, p for P-transistors and n for N-transistors. Values are chosen to ensure balanced rise and fall times. Edges in the diagram are implemented as mere interconnections (without parasitics).

4.3 Approximation Characteristics

The algorithm for low power synthesis of BDD mapped circuits from [11] is based on local variable exchange of the underlying BDD. In Section 2, a number of switching activity estimation methods were discussed. In the following we further analyze their properties and show how they can be applied to low power synthesis for BDD mapped circuits. The total power dissipation of the mapped circuit is computed as:

$$PD_{tot} = \sum_{\forall n} a(n) \times driver(fanout(n)) + leakage(n) \quad (3)$$

Let us consider the EXOR 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 $a(x_1) = 2/3, a(x_2) = 3/4$, Figure 3 (a). The table below shows the estimated switching activity for each BDD node f, f_0 and f_1 and the total estimated power dissipation $Power$. As shown in the table, *Probabilistic* [11] leads to an underestimation, while the proposed local and mux based approximations come closer to the exact result.

	$a(f)$	$a(f_0)$	$a(f_1)$	Power
Over Est. [14] (3.6)	~ 1.42	~ 0.75	~ 0.75	2.92
Exact Est. [14] (3.2)	~ 0.67	~ 0.75	~ 0.75	2.17
Probabilistic [11]	0.5	0.5	0.5	1.5
MUX Approx.	0.58	0.75	0.75	2.08

Let us now change the underlying variable order, Figure 3 (b). (In this example that leads only swapping the input switching activities.) The overall power dissipation for the exact method is reduced

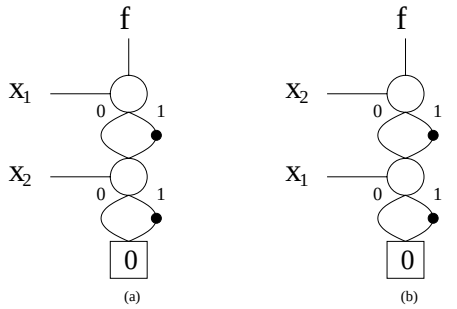

Figure 3: Variable Swap

to 2. Also the other approximative methods indicate a reduction (except for the Probabilistic approach which is unable to utilize the signal activity information).

	$a(f)$	$a(f_0)$	$a(f_1)$	Power
Over Est. [14] (3.6)	~ 1.42	~ 0.67	~ 0.67	2.75
Exact Est. [14] (3.2)	~ 0.67	~ 0.67	~ 0.67	2
Probabilistic [11]	0.5	0.5	0.5	1.5
MUX Approx.	0.54	0.67	0.67	1.88

The switching estimate from section [11] was shown to be computed solely by local operations on the BDD. However, the mux approximation proposed also considers the approximated switching activity of the node's successors the local condition no longer holds. This implies that after a local variable exchange, switching activity estimates need to be propagated towards preceding levels in the diagram. However, as shown in the experimental results, CPU times are reasonable for the set of benchmark function applied.

4.4 Heuristic Minimization Algorithm

The proposed heuristic minimization algorithm, iteratively seeks a variable order reducing the circuit's switching activity weighted by the fan-out cost for each node. We outline the procedure in Figure 4.

The sifting and re-calculation of output probabilities and switching activities is performed solely

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
}

Figure 4: Minimization of Power Dissipation.

through local operations on the BDD representation. The total estimated power dissipation due to switching ($D_{sw}[\_total]$) 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 fan-out changes for nodes below in the diagram can be handled. Figure 5 shows how the total switching probability is updated during sifting. In line 1, we subtract the contribution of the two levels to be sifted ($D_{sw}[\_upper]$+$D_{sw}[\_lower]$) and the contribution of fan-outs from connecting nodes ($D_{sw}[\_below]$). The number of references for connecting nodes are updated (line 2) before applying the sifting (line 3). After the variable exchange is performed, we update the reference counters of the connecting nodes (line 4) and compute the total estimated power dissipation in line 5. Due to the variable exchange, switching activities and reference counters may change, hence also the estimated power dissipation $D_{sw}[\_total]$.

Example 1 *Figure 6 (a) shows a portion of a BDD before sifting. The number at each node denote the number of incoming edges, (i.e, the fan-out in a MUX based mapping). Before sifting we need to determine fan-out changes of the lower levels in the BDD, given as (b) in the Figure 6. Note that only nodes connecting to the "upper" and "lower" levels are updated. After sifting is performed, the new fan-out values (reference counters) of the connecting nodes are computed, as shown in part (c) of Figure 6.*

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 $D_{sw}[\_total]$ += ($D_{sw}[\_upper]$
+ $D_{sw}[\_lower]$ + $D_{sw}[\_below]$)
}

Figure 5: Updating Power Dissipation During Sifting.

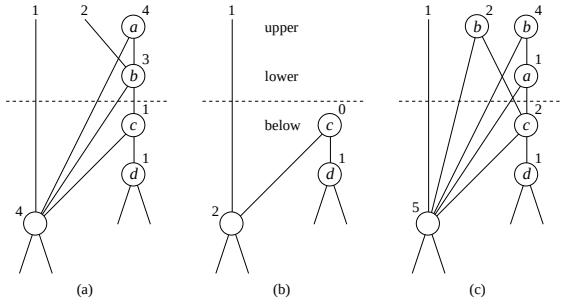

Figure 6: Reference Count Update During Sifting.

5 FSM Analysis

The signal properties for the next state vector is defined from the FSM transition relation together with the properties of the primary input signals. In this section we describe a method to extract this information by modeling the FSM behavior as a Markov chain [9]. There are several approaches for efficient FSM spanning [6]. We have chosen to implement the spanning function in a straight forward way by a depth-first recursive algorithm, which also calculates the transition probability matrix represented by an ADD in the same pass. In [8] and [7] Algebraic Decision Diagram (ADD) were used as the transition probability matrix and the steady state probabilities were calculated in an efficient way. We have implemented our calculations on the ADD in an iterative way, which is sufficient for our purpose of signal activity extraction.

Throughout this section we will give an cook book description of how to do this analysis and also demonstrate it with an example. We have used a set of ISCAS89 benchmarks to test our method.

5.1 Building a BDD

The building is done in two steps. The first part is non-recursive and it builds up small BDD fragments representing the function of a row. The second part is based on a recursive algorithm that starts from the outputs and composes the fragments into a single BDD.

The first part of building the BDD is done by translating each line of the circuit description to a BDD with pseudo variables (wire names). The pseudo variables are referring to some output from an other BDD or an input variable. In Figure 7 you can see the BDD fragment generated from the last line of the example.

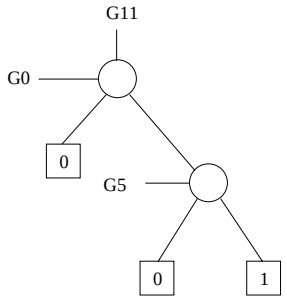

Figure 7: Fragment of the function {G11 = AND(G0,G5)}

When we have a generated BDDs of all the lines we have to compose a single BDD for each output. This is done recursively starting from each output BDD fragment (primary and next state outputs). The recursion terminates when reaching primary inputs or register outputs. When the recursion unfolds the actual function substitutes the pseudo variables and the composed BDD is returned.

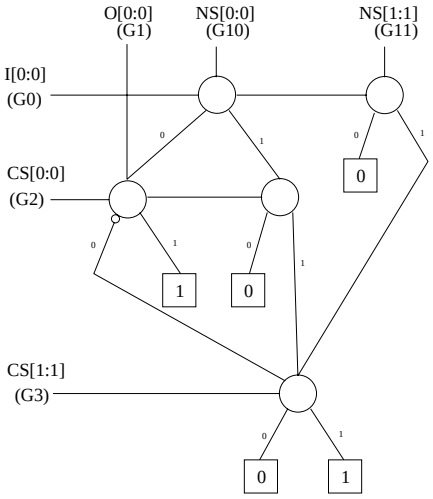

Figure 8: The complete function

5.2 Span FSM states

The completed BDD is used to span the FSM. Starting from the reset state each possible new state is recursively visited (depth first) until reaching an already visited state. During the recursion a transition probability matrix is constructed. The usually sparse matrix is efficiently represented by an ADD (Algebraic Decision Diagram) [7][8]. This matrix is addressed with the Current State as the columns and Next State as the rows. The value in each entry in the matrix (ADD leaf) represents the probability to go from the Current State to the Next State.

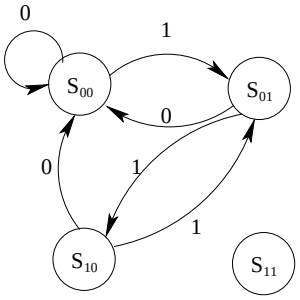

Figure 9: FSM states

Example 2 *When we calculate the transition probabilities the matrix starts empty and a new entries are added during the recursion. We assume that the probability of input I is equal to one is 1/4 (P(I)=1/4).*

	CS_{00}	CS_{01}	CS_{10}	CS_{11}
NS_{00}	0	0	0	0
NS_{01}	0	0	0	0
NS_{10}	0	0	0	0
NS_{11}	0	0	0	0

Here we go from state 00 to state 01 and add the probability of P(I) to row 01 and column 00.

	CS_{00}	CS_{01}	CS_{10}	CS_{11}
NS_{00}	0	0	0	0
NS_{01}	1/4	0	0	0
NS_{10}	0	0	0	0
NS_{11}	0	0	0	0

Finally after we have spanned all reachable states we got the complete matrix.

	CS_{00}	CS_{01}	CS_{10}	CS_{11}
NS_{00}	3/4	3/4	3/4	0
NS_{01}	1/4	0	1/4	0
NS_{10}	0	1/4	0	0
NS_{11}	0	0	0	0

5.3 Calculation of State Probabilities

The ADD obtained by spanning the FSM is used to calculate the steady state probabilities for each state. The FSM can be seen as a Markov chain [7, 8] and this is used in the calculation of the state probabilities. The ADD is multiplied with an initial state probability vector, this represents a matrix multiplication. The initial state vector should have the sum of the entries equal to one and each column should have the sum equal to one.

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

,where A is the matrix represented by the ADD, $\bar{x}$ 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 takes on the value* $\frac{1}{nr_{reachablestates}}$ *except for the unreachable state entries, which takes on the value 0. The total probability in each column of Matrix A is one.*

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

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

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

$$\begin{array}{lcll} P_{SS}(S[1:0]=00) & = & P_{SS}(00) & = & 3/4 \\ P_{SS}(S[1:0]=01) & = & P_{SS}(01) & = & 1/5 \\ P_{SS}(S[1:0]=10) & = & P_{SS}(10) & = & 1/20 \\ P_{SS}(S[1:0]=11) & = & P_{SS}(11) & = & 0 \end{array}$$

6 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.

6.1 Calculation of Bit Probabilities

The state probabilities are used to calculate the bit probabilities for each register. The bit probabilities is calculated by traversing the ADD and utilizing equation 5.

$$(\forall i) P(NS[i:i]) = \sum_{\forall S[N-1:0] \epsilon S[i:i]=1} P_{SS}(S[N-1:0]) \tag{5}$$

Example 4 *For next state bit zero we add all Steady State probabilities, which has one on bit zero. In a similar way we do this for bit one. The computations are shown in equation 6.*

$$\begin{array}{lcll} P(NS[0:0]) & = & P_{SS}(01) + P_{SS}(11) & = & 1/5 \\ P(NS[1:1]) & = & P_{SS}(10) + P_{SS}(11) & = & 1/20 \end{array} \tag{6}$$

6.2 Calculation of Bit Activities

To calculate the activity for each bit we use the ADD with the state transition probabilities and the steady state probabilities calculated earlier. $P_{SS}(n)$ denotes the steady state probability for state n, n[i:i] is the i:th bit of the vector n, A is the Matrix containing the state transition probabilities $(A[NS_k][CS_n] = P(NS_k|CS_n))$, $a(NS[i:i])$ is the activity for the next state bit i and is derived by the following formula.

$$(\forall i) a(NS[i:i]) = \sum_{\forall n} P_{SS}(n) \times \sum_{\forall k \epsilon (k[i:i] \neq n[i:i])} P(NS_k|CS_n) \tag{7}$$

Example 5 *The equation 7 is used to calculate the next state activities in the following example.*

$$\begin{array}{lcl} a(NS[0:0]) & = & P_{SS}(00) \times P(NS_{01}|CS_{00}) \\ & + & P_{SS}(01) \times (P(NS_{00}|CS_{01}) \\ & + & P(NS_{10}|CS_{01})) \\ & + & P_{SS}(10) \times P(NS_{01}|CS_{10}) \\ & = & 3/4 \times 1/4 + 1/5 \times (3/4 + 1/4) \\ & + & 1/20 \times 1/4 = 2/5 \\ a(NS[1:1]) & = & P_{SS}(00) \times 0 \\ & + & P_{SS}(01) \times P(NS_{10}|CS_{01}) \\ & + & P_{SS}(10) \times (P(NS_{01}|CS_{10}) \\ & + & P(NS_{00}|CS_{10})) \\ & = & 0 + 1/5 \times 1/4 \\ & + & 1/20 \times (1/4 + 3/4) = 1/10 \end{array} \tag{8}$$

7 Experimental Results

In [11] the basic low power optimizing method is shown to reduce estimated power dissipation for

		Area Optimized			*Low Power Optimized*		
name	in/out	Prob	Local	Mux	Prob	Local	Mux
5xp1	7/10	32.2	36.3	40.4	30.2	27.7	25.1
add6	12/ 7	23.0	21.9	20.3	23.0	21.9	20.3
apex7	49/37	175.6	191.8	209.8	158.4	161.3	165.1
bc0	26/11	320.0	311.2	330.8	310.3	264.1	229.7
chkn	29/7	132.0	140.4	151.1	84.8	110.1	33.2
duke2	22/29	106.9	114.3	129.2	103.8	111.6	75.9
exp	8/18	79.5	78.9	81.5	61.6	60.7	42.4
in2	19/10	115.5	113.4	115.5	95.5	88.4	67.5
in7	26/10	21.9	22.1	23.5	20.1	18.1	16.8
inc	7/ 9	45.4	52.1	54.4	45.3	37.4	24.6
intb	15/ 7	349.3	336.9	313.8	305.4	284.1	256.6
misex3	14/14	223.9	219.4	234.9	223.9	206.8	203.8
sao2	10/ 4	35.8	36.3	37.2	34.2	32.0	16.6
tial	14/ 8	422.6	438.3	453.2	422.6	430.6	362.9
vg2	25/ 8	50.0	47.2	46.9	50.0	45.3	46.3
x6dn	39/ 5	143.1	124.0	115.2	124.5	110.0	96.0
Sum		2276.8	2284.5	2357.7	2093.6	2009.3	1682.8

Table 1: Area Optimized circuits compared with Low Power Optimized circuits

	Area opt		NonFSM opt		FSM opt		Percent Change	
name	*Size*	$P^{\wedge}D$	*Size*	$P^{\wedge}D$	*Size*	$P^{\wedge}D$	*Size*	$P^{\wedge}D$
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 2: ISCAS89 benchmarks

uncorrelated input signals. The average power estimate reduction was 20 percent with over 50 percent in some benchmarks. The spice simulations using commercial CMOS transistor models also indicates that our model is applicable. We get further potential power reductions by incorporating temporal signal correlation in our synthesis algorithm. As shown in Table 1 the average power estimate reduction for our synthesis tool is 30 percent to the area optimized circuit. This is done with a large activity deviation ($P = 0.5$ and alternating $a_i = 0.1, 0.9, 0.1, ...$), for which the method in [11] only manage to reduce the power estimate by 8.3 percent compared to the area optimizer.

Furthermore we have analyzed the finite state machines as described in section 5 on a set of ISCAS89 benchmarks and extracted statistical information as in in section 6 and applied that information on our synthesis tool. As seen in Table 2 the results show an average power estimate reduction of 43 percent by the new proposed method compared to the area optimized method. The power estimate reductions range from 0 percent to 95 percent, The majority of the tests show a significant power estimate reduction for the FSM optimized circuits compared with the area optimized circuits. The results also show that the power optimized circuits have an increased area of 51 percent on average over the area optimized circuit. In two cases the power optimizer have smaller circuits than the area optimizer. This is due to the heuristic algorithm that the area optimizer utilizes, which may cause it to find a local minimum.

8 Conclusions

The results show that using the information about the temporal correlation when we optimizing the circuit gives good results. Further work should be done with spatial correlation to see if we get a further improvements in the power consumption.

References

[1] S.B. Akers. Binary decision diagrams. *IEEE Trans. on Comp.*, 27:509–516, 1978.

[2] V. Bertacco, S. Minato, P. Verplaetse, L. Benini, and G. De Micheli. Decision diagrams and pass transistor logic synthesis. In *Int'l Workshop on Logic Synth.*, 1997.

[3] K.S. Brace, R.L. Rudell, and R.E. Bryant. Efficient implementation of a BDD package. In *Design Automation Conf.*, pages 40–45, 1990.

[4] R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[5] P. Buch, A. Narayan, A.R. Newton, and A.L. Sangiovanni-Vincentelli. Logic synthesis for large pass transistor circuits. In *Int'l Conf. on CAD*, pages 663–670, 1997.

[6] G. Cabodi, P. Camurati, and S. Quer. Improving symbolic reachability analysis by means of activity profiles. *IEEE Trans. on Comp.*, 19(9):1065–1075, 2000.

[7] G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Markovian analysis of large finite state machines. *IEEE Trans. on Comp.*, 15(12):1479–1493, 1996.

[8] G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Probabilistic analysis of large finite state machines. In *Design Automation Conf.*, pages 270–275, 1994.

[9] G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Markovian analysis of large finite state machines. *IEEE Trans. on Comp.*, 15(12):1479–1493, 1996.

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

[11] P. Lindgren, M. Kerttu, M. Thornton, and R. Drechsler. Low power optimization technique for BDD mapped circuits. In *ASP Design Automation Conf.*, pages 615–621, 2001.

[12] R. Marculescu D. Marculescu and M. Pedram. Efficient power estimation for highly correlated input streams. In *Design Automation Conf.*, 1995.

[13] S. Minato, N. Ishiura, and S. Yajima. Shared binary decision diagrams with attributed edges for efficient Boolean function manipulation. In *Design Automation Conf.*, pages 52–57, 1990.

[14] K. Roy and S. Prasad. *Low-Power CMOS VLSI Circuit Design*. Wiley Interscience, 2000.

[15] C. Scholl and B. Becker. On the generation of multiplexer circuits for pass transistor logic. In *Design, Automation and Test in Europe*, 2000.