Low Power Optimization Technique for BDD Mapped Finite State Machines

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

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

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

Abstract

In modern design flows low power aspects should be considered as early as possible to minimize power dissipation in the resulting circuit. A new BDD-based design style that considers switching activity optimization using temporal correlation information is presented. The technique is developed as an approximation method for switching activity estimation. Experimental results on a set of MCNC and ISCAS89 benchmarks show the estimated reduction in power dissipation.

1 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 behavior 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 [11]. Modern design flows should consider these effects from the very beginning.

Several synthesis tools make use of *Binary Decision Diagrams* (BDDs) [3, 4, 12], 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 be proposed in [10]. The method 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 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 it is shown how it can be integrated with the approach in [10]. 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 [5, 14] 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) the contribution from internal switching in the multiplexers can be neglected.

To be able to calculate 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 modeled as FSMs is described. In this way, the need for computationally expansive gate level simulation is avoided and the signal statistics are utilized for low power synthesis.

2 Switching Activity Estimation

In this section an introduction to signal switching activity estimation is given. (For more details see [13].) In the following it is assumed that the input signals are mutually independent (spatially uncorrelated) and that the signals can be modeled as *Strict-Sense Stationary* (SSS) and mean-ergodic with zero delay [13]. That is, all switching is carried out simultaneously and that signal probability and switching activity do not vary over time. $P(f)$ denotes the probability of f being 1 (the output probability of f). $a(f)$ denotes the activity for f (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 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

*This work was supported in part by the NSF under grants CCR-0000891 and CCR-0097246

correlated an overestimate is obtained since 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 observation allows for the application of Theorem 3.1 from [13]. The formula in Equation 1 can then derived using the multiplexer-based circuit model.

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

In Equation 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 and calculates the activity for each node in the BDD.

3 Low Power Synthesis

3.1 BDD Mapped Circuits

A BDD can be directly mapped to a multiplexer-based circuit as described in [1], to a "timed" circuit as described in [9] or to a "pass-transistor" based circuit as described in [2, 5, 14]. 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. It 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 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.

A method for low power synthesis of BDD mapped circuits was first introduced in [10]. 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, could be implemented 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 shown both to reduce BDD complexity and improve performance of operations, [12, 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 $a[f]$.

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

3.2 Power Dissipation Modeling

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 Figure 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.

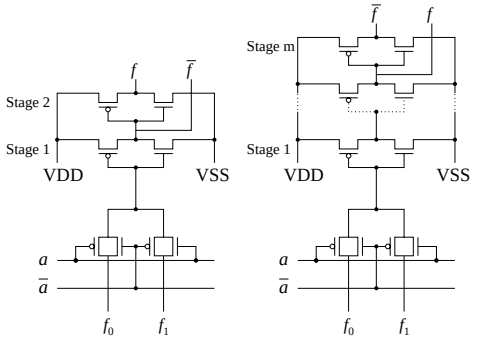

Figure 1: BDD node mapping into multiplexer circuits

The power dissipation for the mapped node n is estimated using the relationship in Equation 2.

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

Equation 2 was validated by conducting transistor level simulations using models from a commercially available CMOS process. The results 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 a 100 to 1 under unity load (a single fanout). Thus, the effect of internal switching can be disregarded.

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

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 are not considered in this model, however this could be included for better accuracy.

3.3 Approximation Characteristics

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

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

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 Figure 2. The following table 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, the technique labeled *Probabilistic* leads to an underestimation, while the proposed multiplexer-based approximation (MUX Approx.) comes closer to the exact result.

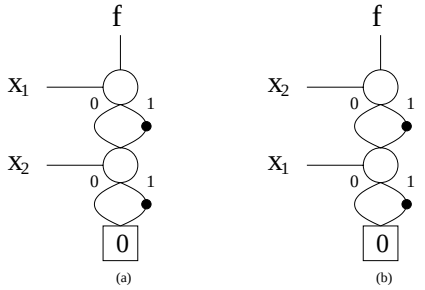

Figure 2: Variable Swap

	$a(f)$	$a(f_0)$	$a(f_1)$	Power
Over Est. [13]	~ 1.42	~ 0.75	~ 0.75	2.92
Exact Est. [13]	~ 0.67	~ 0.75	~ 0.75	2.17
Probabilistic [10]	0.5	0.5	0.5	1.5
MUX Approx.	0.58	0.75	0.75	2.08

When the BDD variable order is changed as shown in Figure 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 the following table (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. [13]	~ 1.42	~ 0.67	~ 0.67	2.75
Exact Est. [13]	~ 0.67	~ 0.67	~ 0.67	2
Probabilistic [10]	0.5	0.5	0.5	1.5
MUX Approx.	0.54	0.67	0.67	1.88

```
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 3: Minimization of Power Dissipation

```
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 4: Updating Power Dissipation During Sifting

The switching estimate in the table labeled *Probabilistic* is computed solely by local operations on the BDD. However, the approximation technique labeled *MUX Approx.* 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 leads to more complexity in the switching activity estimation algorithm, CPU times are reasonable for the set of benchmark functions used in the experiments.

3.4 Heuristic Minimization Algorithm

The proposed heuristic minimization algorithm iteratively seeks a variable order that reduces the mapped circuits' switching activity weighted by the fan-out cost for each node. This procedure is outlined in psuedocode in Figure 3.

The sifting and re-calculation of output probabilities and switching activities is performed solely through local operations on the BDD representation. The total estimated power dissipation due to switching ($D_{sw}[\_total]$) 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 fanout changes for nodes below in the diagram can be handled. Figure 4 shows how the total switching activity is updated during sifting. In line 1, the contribution of the two levels to be sifted ($D_{sw}[\_upper]+D_{sw}[\_lower]$) and the contribution of fanouts from connecting nodes ($D_{sw}[\_below]$) is subtracted. The number of references for connecting nodes are updated (line 2) before applying the sifting (line 3). After the variable exchange is performed, the reference counters of the connecting nodes (line 4) is 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 $D_{sw}[\_total]$.

Example 1 *Figure 5 (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 sifting the fanout changes of the lower level in the BDD shown in Figure 5 (b) need to be determined. 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 illustrated in Figure 5 (c).

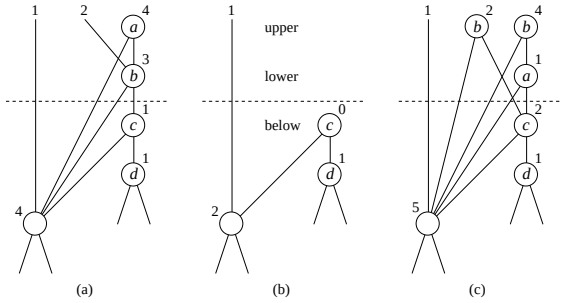

Figure 5: Reference Count Update During Sifting

4 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 from the FSM transition relation together with the properties of the primary input signals. In this section a method to extract this information by modeling the FSM behavior as a Markov chain [8] is described. There are several approaches for efficient FSM spanning [6]. 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 [7] and [8] 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.

4.1 Span FSM States

The BDD representing the next state functions are used to span the FSM. 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 starts empty and new entries are added during the recursion. We assume that the probability that input signal I is at a logic-1 value is 1/4 (P(I)=1/4).*

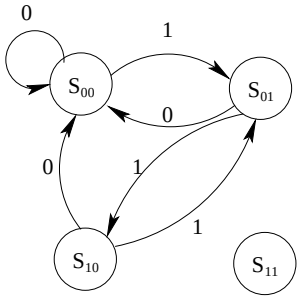

Figure 6: FSM states

$$\begin{array}{c} \\ NS_{00} \\ NS_{01} \\ NS_{10} \\ NS_{11} \end{array} \begin{array}{c} \begin{array}{cccc} CS_{00} & CS_{01} & CS_{10} & CS_{11} \end{array} \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

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

$$\begin{array}{c} \\ NS_{00} \\ NS_{01} \\ NS_{10} \\ NS_{11} \end{array} \begin{array}{c} \begin{array}{cccc} CS_{00} & CS_{01} & CS_{10} & CS_{11} \end{array} \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 1/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

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

$$\begin{array}{c} \\ NS_{00} \\ NS_{01} \\ NS_{10} \\ NS_{11} \end{array} \begin{array}{c} \begin{array}{cccc} CS_{00} & CS_{01} & CS_{10} & CS_{11} \end{array} \\ \left[\begin{array}{cccc} 3/4 & 3/4 & 3/4 & 0 \\ 1/4 & 0 & 1/4 & 0 \\ 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

4.2 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 [8, 7] 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.

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

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 has the value* $\frac{1}{nr_{reachablestates}}$ *except for the unreachable state entries, which have the value 0.*

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

$$\begin{bmatrix} 3/4 & 3/4 & 3/4 & 0 \\ 1/4 & 0 & 1/4 & 0 \\ 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 3/4 \\ 1/5 \\ 1/20 \\ 0 \end{bmatrix} = \begin{bmatrix} 3/4 \\ 1/5 \\ 1/20 \\ 0 \end{bmatrix} \quad (6)$$

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

$$\begin{array}{rcccl} P_{SS}(S[1:0]=00) & = & P_{SS}(00) & = & 3/4 \\ P_{SS}(S[1:0]=01) & = & P_{SS}(01) & = & 1/5 \\ P_{SS}(S[1:0]=10) & = & P_{SS}(10) & = & 1/20 \\ P_{SS}(S[1:0]=11) & = & P_{SS}(11) & = & 0 \end{array} \quad (7)$$

4.3 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 equation 8.

$$(\forall i) P(NS[i:i]) = \sum_{\forall S[N-1:0] \epsilon S[i:i]=1} P_{SS}(S[N-1:0]) \quad (8)$$

To calculate the activity for each bit, the ADD with the state transition probabilities and the steady state probabilities calculated earlier are used. $P_{SS}(n)$ denotes the steady state probability for state n, $n[i:i]$ is the i^{th} bit of vector n, A is the matrix containing the state transition probabilities ($A[NS_k][CS_n] = P(NS_k|CS_n)$), and $a(NS[i:i])$ is the activity for the next state bit i and is given by the formula in Equation 9.

$$(\forall i) a(NS[i:i]) = \sum_{\forall n} P_{SS}(n) \times \sum_{\forall k \epsilon (k[i:i] \neq n[i:i])} P(NS_k|CS_n) \quad (9)$$

5 Experimental Results

Benchmark circuits were synthesized using the low power optimizations described here and also for optimizing with respect to area minimization. As compared to the previous approach in [10], further power reductions were obtained since this method incorporates the use of temporal signal correlations. As shown in Table 1 the average power estimate reduction for the synthesis method described here is 30% as compared to the area optimized circuit. This results were obtained assuming a large activity deviation ($P = 0.5$ and alternating $a_i = 0.1, 0.9, 0.1, ...$). Using the same assumptions with the method in [10] resulted in a power reduction of only 8.3% compared to the area optimizer.

Furthermore we have analyzed the finite state machines as described in Section 4 on a set of ISCAS89 benchmarks and extracted statistical information as in in Section 4.3 and used this information within the synthesis tool. As shown in Table 2 the results indicate an average power estimate reduction of 43% using the new method proposed here compared to 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 is due to the heuristic algorithm that the area optimizer utilizes, which may cause it to get stuck in a local minimum.

6 Conclusions and Future Work

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 power dissipation of 30% as compared to circuits synthesized with area minimization. In the future this technique will be extended to take advantage of any spatial correlation among the input signals. Also, it is planned to consider parasitic capacitances from routing and to incorporate power estimates due to static leakage currents and input signal buffers.

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. Probabilistic analysis of large finite state machines. In *Design Automation Conf.*, pages 270–275, 1994.

[8] 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.

[9] 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.

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

		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 2: ISCAS89 benchmarks

	Area opt		NonFSM opt		FSM opt		Percent Change	
name	*Size*	$\hat{PD}$	*Size*	$\hat{PD}$	*Size*	$\hat{PD}$	*Size*	$\hat{PD}$
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

[10] 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.

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

[12] 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.

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

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