Low Power Optimization Technique for BDD Mapped Circuits*

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

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

Abstract

The minimization of power consumption is an important design constraint for circuits used in portable devices. The switching activity of a circuit node in a CMOS digital circuit directly contributes to overall power dissipation. By approximating the switching activity of circuit nodes as internal switching probabilities in Binary Decision Diagrams *(BDDs), it is possible to estimate the dynamic power dissipation characteristic of circuits resulting from a structural mapping of a BDD. A technique for minimizing the overall sum of switching probabilities is presented. The method is based on efficient local operations on a BDD representing the functionality of the circuit to be realized. The resulting circuit that is obtained by mapping the BDD to structural netlist has a reduced power dissipation characteristic. Experimental results are given for this technique.*

1 Introduction

The popularity of small, portable communications and computing devices has contributed to an increasing interest in producing digital circuits optimized for low power dissipation. The design of low power consumption circuits can allow for the production of devices that operate longer for a given amount of battery power, are more reliable due to reduced heat generation and have lower packaging costs. These facts motivate designers to place emphasis on optimization for low power dissipation.

The power dissipation characteristic for CMOS based digital circuitry results from a static and a dynamic component. The static component consists of contributions from "leakage" and "standby" currents while the dynamic component is attributed to "switching" and "capacitive" currents. The dynamic components only occur during the transition of internal circuit nodes from one logic level to another.

At the architectural level, power dissipation has been reduced by the inclusion of automatic power management techniques, scaling down the supply voltage and the clock frequency and using more sophisticated packaging techniques that reduce the chip and package capacitance. At the device level, there have been advances in the development of new CMOS properties that reduce the static currents and for the development of "low-power" cell libraries. However, methods that focus on reducing the *internal* circuit switching activity have not been as prevalent.

Here, we propose a heuristic method to reduce the estimated internal switching activity which reduces the overall amount of dynamic switching current. The method is based on local, and hence efficient, *Binary Decision Diagram* (BDD) operations. Since BDDs may be used as structural

*This work was supported in part by NSF grant CCR-9633085

descriptions of digital circuits, we focus on developing techniques that optimize the BDD such that the resultant circuit obtained through a direct BDD mapping has a smaller overall switching current.

The remainder of the paper is organized as follows. In the next two sections we briefly survey the power dissipation characteristic of CMOS digital circuitry and basic properties of BDDs. Next, we show how the output probability can be used to estimate the switching activity in a BDD-based circuit. Furthermore, we show that the computation of a switching probability at an internal BDD vertex can be manipulated through purely local operations on the graph. Based on these local graph operations, we develop an approach for minimizing the overall switching activity and then perform a set of experiments evaluating the effectiveness of the method and the effect on the size of the resultant BDD.

2 Power Dissipation in CMOS Digital Circuits

Circuits based on NMOS and PMOS transistors using "CMOS technology" are known for good power dissipation characteristics since very little current flows internally while a circuit is at some logic level. The small amount of current that does flow is termed the "static" component and is due to leakage currents (i.e. reverse bias currents in the FETs) and subthreshold currents which are the small magnitude currents flowing from V_{dd} to ground. During a logic-level transition, additional "dynamic" currents exist that can be classified as "capacitive" currents which are those that are required for charging/discharging capacitive loads during transitions and as "switching" currents which occur on DC paths between the supply rails during logic transitions. The total power dissipation, PD, can then be described as a sum of the contributions from each of these currents as given in Equation 1.

$$PD = V_{dd}(i_l + i_{sub}) + V_{dd}^2 \cdot f_{clk} \cdot \sum_{k=1}^{N} E_{sw_k} \cdot C_k + V_{dd} \cdot \sum_{k=1}^{N} E_{sw_k} \cdot i_{sc_k} \tag{1}$$

Where, the following notation is used:

- V_{dd} is the Supply voltage (Volts)
- i_l is the leakage current (Amps)
- i_{sub} is the subthreshold current (Amps)
- f_{clk} is the circuit clock frequency (Hz)
- N is the total number of internal circuit nodes
- C_k is the capacitive load at circuit node k (farads)
- i_{sc_k} is the short circuit current due to dynamic switching at circuit node k (Amps)
- E_{sw_k} is the switching probability at circuit node k

Equation 1 shows that the dynamic power dissipation component is dependent on the switching activity parameter, E_{sw_k}. Past research has shown that the switching activity parameter is highly dependent on temporal correlations of circuit input signal values [8]. As an example, a combinational circuit that is responsible for generating next-state values in a synchronous finite state machine exhibits a definite correlation between previously produced logic values and those to be produced in the near future since only a subset of all possible states are reachable given the current state. If it is assumed that all circuit input signals are statistically uncorrelated and are completely independent, the switching activity, E_{sw_k}, may be approximated by the switching probability, P_{sw_k}.

In the work presented here, we develop a technique for minimizing the switching probability value, P_{sw_k}, in a digital circuit that is generated based on the structural information of a BDD. We are thus, minimizing an estimate of the total power dissipation, $\hat{PD}$ as shown in Equation 2.

$$\hat{PD} = V_{dd}(i_l + i_{sub}) + V_{dd}^2 \cdot f_{clk} \cdot \sum_{k=1}^{N} P_{sw_k} \cdot C_k + V_{dd} \cdot \sum_{k=1}^{N} P_{sw_k} \cdot i_{sc_k} \tag{2}$$

3 Ordered Binary Decision Diagrams

Let f_0 (f_1) denote the *cofactor* of f with respect to $\overline{x}$ (x). A Boolean function, $f : B^n \rightarrow B$, can then be represented by the following formula commonly known as the *Shannon Decomposition*:

$$f \quad = \quad \overline{x} f_0 \oplus x f_1 \tag{3}$$

Consider a rooted, *Directed Acyclic Graph* (DAG), G, having terminals 0 and 1 and non-terminals (internal nodes) labeled with binary decomposition variables, x. Each internal node has two exiting edges that point to cofactor subgraphs, f_0 and f_1. The edge with the attribute, 0(1), points to the subgraph representing f_0 (f_1). In this work, we only consider *ordered* BDDs [4] where each variable can occur only once on each path and in the same order for any possible path. Vertices having the same decomposition variable are considered to be at the same *level* in the diagram. Diagram levels are enumerated from the root (top level) toward the terminals (bottom level). Furthermore, we assume that the BDDs are fully *reduced* in that the following rules of "reduction" have been applied:

- There exist no two subgraphs expressing the same function (i.e., no two subgraphs are graph-isomorphic).
- There exist no redundant nodes (i.e. $f_0 = f_1$ does not occur).

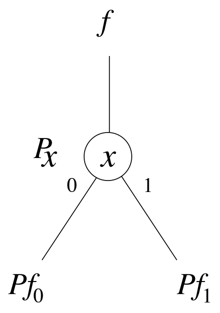
A single graph can be used to represent both f and $\overline{f}$, where the latter function is identified by a complement attribute on the incoming edge [9, 3]. For a given ordering, a reduced BDD provides a canonical representation of the function under some restrictions for the use of complemented edges. The restrictions are that only one terminal is to be used (e.g., the 0-terminal), and complementation is only allowed to occur on one type of the outgoing edges (e.g., the 1-edge).

The form of BDD described above can be used to also represent multiple-output functions by allowing each single output to be rooted arbitrarily in the resulting *shared*-BDD.

4 Switching Probability

An output probability of a function, f, denoted as $P[f]$ is the probability that the function has a value of "1" at some arbitrary time of observation [10]. Consider a function f having the output probability $P[x]$ for the input variable x and the output probabilities $P[f_0]$ and $P[f_1]$ for its' corresponding cofactors f_0 and f_1. In terms of a BDD, this relationship is shown in Figure 1.

We seek the switching probability $P_{sw}[f]$ of f. Switching occurs if and only if the value of f changes from 0 to 1 or 1 to 0. We note that the probability that a function is 0-valued is given



(a)

Figure 1: Switching Probability in BDD Vertex

Table 1: Enumeration of the Probabilities for f Values at Subsequent Observations

$P[f^{t1} \cap f^{t2}]$	f^{t1}	f^{t2}
$((1-P[x])(1-P[f_0])+P[x](1-P[f_1]))^2$	0	0
$((1-P[x])(1-P[f_0])+P[x](1-P[f_1]))((1-P[x])P[f_0]+P[x](P[f_1]))$	0	1
$((1-P[x])P[f_0]+P[x](P[f_1]))((1-P[x])(1-P[f_0])+P[x](1-P[f_1]))$	1	0
$((1-P[x])P[f_0]+P[x](P[f_1]))^2$	1	1

as the probability that the variable, x and the cofactor f_0 are 0-valued or that the variable, x is 1-valued but the cofactor, f_1 is 0-valued. A similar statement can be made for the probability that $f = 1$. These relationships are given in the following equations.

$$(1-P[x])(1-P[f_0])+P[x](1-P[f_1]) \qquad for f = 0 \tag{4}$$
$$(1-P[x])(P[f_0])+P[x](P[f_1]) \qquad for f = 1 \tag{5}$$

Now, consider the value of f at two different observation times f^{t1} and f^{t2}. f is considered to "switch" if the value of $f^{t1} \neq f^{t2}$. Table 1 enumerates the possible states of f at two subsequent observation times, $t1$ and $t2$:

Due to the definition of "switching", $f^{t1} \neq f^{t2}$, we can derive the probability of switching based on the output probabilities given in Table 1. Hence the switching probability $P_{sw}[f]$ of f can be computed as:

$$P_{sw}[f] = P[f^{t1} = 0 \cap f^{t2} = 1] + P[f^{t1} = 1 \cap f^{t2} = 0] \tag{6}$$

Using the expression in Table 1 and substituting them into Equation 6, we have the result:

$$P_{sw}[f] = 2((1-P[x])(1-P[f_0])+P[x](1-P[f_1]))((1-P[x])(P[f_0])+P[x](P[f_1])) \tag{7}$$

5 Circuits Based on BDD Mappings

An Ordered Binary Decision Diagram (OBDD) [4] can be directly mapped to a MUX based circuit as described in [1], to a "timed" circuit as described in [7] or to a "pass-transistor" based circuit as described in [2]. 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 (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 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.

As discussed in a previous section, one of the main factors of power drain in a CMOS digital circuit is the switching probability of each subcircuit output. In order to provide the background for our technique, we state the following Lemma:

Lemma 1 *Consider a level l in the diagram and its corresponding set of nodes N_l having decomposition variable x_l. The output probability of each node in N_l is unaffected by the variable ordering above and below l.*

Proof 1 *This follows from the properties of BDDs [5]. The cofactors to be implemented at level l are independent of the ordering above l. The cofactors implemented by level $l+1$ remain unaltered as they are derived from l only. As the output probability of nodes at N_l are solely defined from the output probabilities of nodes at N_{l+1} and the probability of x_l we have the lemma.* □

Now, let us consider the switching probability of nodes N_l at level l. As defined in 6, the switching probability depends only on the output probability of the cofactors ($P[N_{l+1}]$) and ($P[x_l]$) of variable x_l under our assumptions. In the next section we show how this property can be utilized in a reordering method for BDDs.

5.1 Local Variable Exchange

Many state of the art heuristics for BDD minimization are based on "sifting" operations which are popular due to the ease in which local variable exchanges can be accomplished [6]. The key property is that a local exchange of variables in a BDD can be done solely by redirecting edges (i.e. pointers) locally in the diagram. Since our main concern is to minimize the overall sum of switching activity of the resulting circuit nodes, we show that the switching probability can be computed by local operations during sifting.

Lemma 2 *The switching probability of f is independent of the variable ordering.*

Proof 2 *The switching probability of a function is a property of the function defined solely from it's ON and OFF set and the output probabilities of the dependent variables. Hence, the variable order cannot influence the switching probability of f and we have the lemma.* □

Consider a function f represented by a BDD. We can now show that changes in internal switching probabilities can occur due to local BDD variable exchanges as illustrated in Figure 2. Due to the reduction rules that are applied after a local variable exchange, some vertices and edges may be eliminated resulting in fewer intermediate switching probability values. From formula 2 it follows that the switching probability of f remains unaffected by local variable exchanges. Furthermore, switching probabilities of subfunctions below the exchanged levels are also preserved, since the cofactors at levels indicated by the word "below" are intact during sifting. This holds for exchanging arbitrary (neighboring) levels in the diagram.

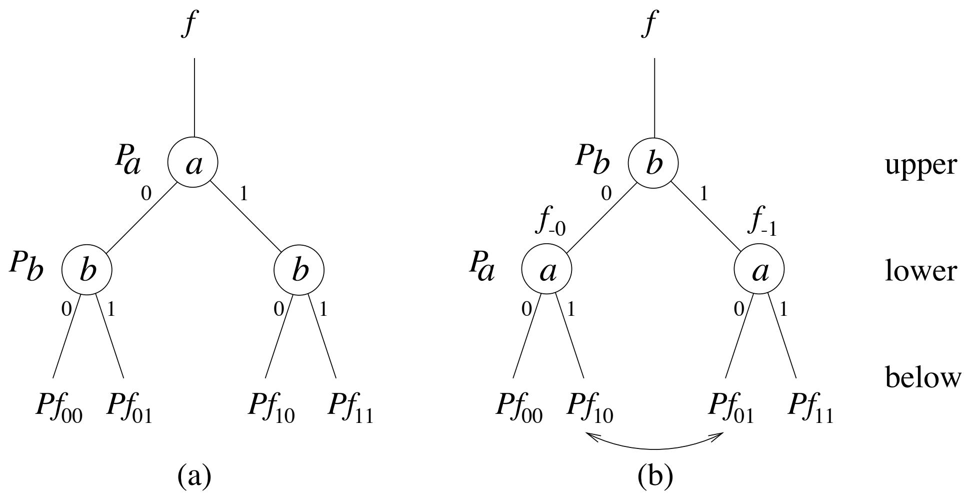


Figure 2: BDD Local Variable Exchange and Effect on Switching Probabilities

Finally we need to show that the switching probabilities of the nodes at levels denoted by the word "lower" in Figure 2 can be computed locally. From Equation 6 we derive:

$$P_{sw}[f_{-0}] = 2((1 - P[a])(1 - P[f_{00}]) + P[a](1 - P[f_{10}))((1 - P[a])(P[f_{00}]) + P[a](P[f_{10}])) \qquad (8)$$
$$P_{sw}[f_{-1}] = 2((1 - P[a])(1 - P[f_{01}]) + P[a](1 - P[f_{11}]))((1 - P[a])(P[f_{01}]) + P[a](P[f_{11}])) \qquad (9)$$

As the output probabilities $\{P[f_{00}], P[f_{10}], P[f_{01}], P[f_{01}]\}$ are unaltered during sifting, the operation is local.

5.2 Complemented edges

The use of complemented edges has shown both to reduce BDD complexity and improve performance of operations, [9, 3]. The Lemmas 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 probability $P_{sw}[\overline{f}]$ of $\overline{f}$ is equal to $P_{sw}[f]$.

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

5.3 Cross-Point BDD Nodes

In order to keep the variable exchange operation local, we leave the initial BDD representation quasi-reduced where nodes are allowed to connect only to the next level in the diagram. This will infer the occurrence of "cross-point nodes" which are essentially connections between levels and would otherwise be eliminated through the reduction rules. Cross-point nodes use output and switching probabilities of their corresponding child nodes since they are "virtual" vertices that are included only for ease in implementing the local variable exchange operations and they do not contribute to the circuitry obtained by structurally mapping the BDD. The use of cross-point nodes tremendously simplifies the sifting procedure at the expense of maintaining redundant nodes during minimization. Furthermore, cross-point nodes are useful to estimate the cost of resulting circuit interconnections since they represent longer conductor paths when the BDD mapped circuit undergoes routing. Figure 3 shows the effect of sifting on a quasi-reduced BDD.

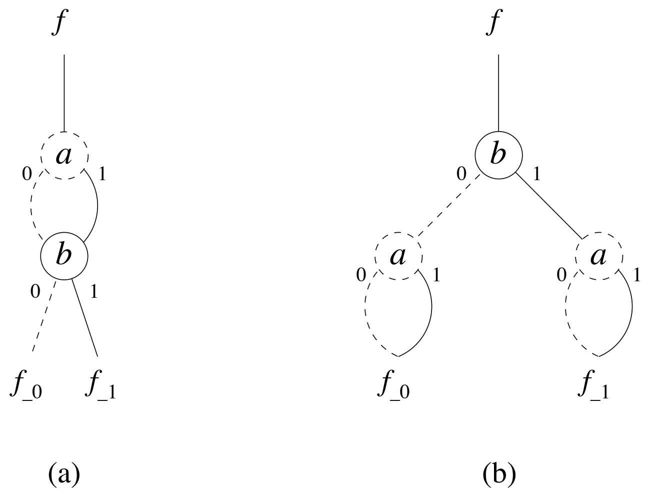


Figure 3: Sifting of a BDD with Cross-Point Nodes

6 Switching Probability Minimization

Given the results described above, we can now state the BDD based algorithm for the minimization of total estimated switching activity in a circuit based on BDD mapping. The algorithm is similar to those of BDD minimization based on local variable exchange but the cost measure is different.

6.1 Cost Model

We define the cost model based on the total circuit switching activity under given a set of dependent variable output probabilities. We attempt to minimize the sum of all internal switching probabilities at each BDD vertex. We neglect those values at cross-point nodes as they have no effect on the circuit obtained through a structural mapping of the BDD.

This model has some assumptions. We assume the input signals of the resultant circuit to be statistically independent from each other and uncorrelated in a temporal sense. This is the assumption that allows us to use the switching probability computed in the BDD representation as an estimate for the actual switching activity of a circuit.

Furthermore, we use a linear model for fan-out cost. This model can be refined if more information is known about the target architecture properties, such as gate or inverter sizing, for cell library or full custom implementations respectively. Also, we apply a unit cost for the load of each fan-in. This measure can be further refined using technology dependent capacitance measures weighted by the estimated length of the interconnection (as the number of cross-point nodes traversed).

Finally we consider only the dynamic power dissipation component of the circuitry. A technology dependent measure for the static power dissipation of each subcircuit could also be applied easily if the target circuit architecture is known in advance.

```
SP_min() {
1       compute $\sum P_{sw}[total]$
2       for each variable {
3               sift to position minimizing $\sum P_{sw}[total]$
4       } repeat until no further improvement
}
```

Figure 4: Minimization of Total Switching Probability

```
ref_update(level, i, ref) {
1       if (index == terminal) return
2       if (cross-point) {
3               BDD[level][i].ref += ref
4               ref_update(level + 1, BDD[level][i].low, ref)
5       } else {
6               BDD[level + 1][BDD[level][i].low].ref += ref
7               BDD[level + 1][BDD[level][i].high].ref += ref
8       }
}
```

Figure 5: Update of Reference Counters

6.2 Heuristic Minimization Algorithm

The heuristic minimization algorithm proposed here iteratively seeks a variable order reducing the overall sum of the circuit's switching activity. We outline the procedure in Figure 4. The sifting and re-calculation of output and switching probabilities is performed solely through local operations on the BDD representation. The total switching activity can also be updated by local operations, with the exception that cross-point nodes leading to lower levels in the diagram are neglected. In order to update the total switching probability we need to determine fan-out changes of the lower levels in the diagram. Figure 5 outlines the recursive procedure. Note that the recursion is terminated on the occurrence of a non cross-point node. This prevents us from traversing the whole diagram below the two levels sifted. Figure 6 shows how the total switching probability is updated during sifting. In line 1, we subtract the total switching probability dependent on the two levels to be sifted. $P_{sw}[below]$, is the switching probability of nodes connecting directly or through cross-points to the lower sifting levels. Note, for sifting based minimization targeting BDD size, there is no need to consider $P_{sw}[below]$. However, targeting the total switching probability calls for this consideration. The fan-out of lower levels might change during sifting, and thus, the total sum of internal switching probabilities of the circuit will also. After a variable exchange is performed, we compute the new switching probability in line 9. In Figures 4 5 and 6, we use the notation, $\sum P_{sw}$, to indicate the total sum of switching probabilities at each of the BDD subgraphs (i.e. the *total*, *lower* or *upper*).

Example 1 *Figure 7 (a) shows a portion of a BDD before sifting. The numbers 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 7. Note that only nodes below the dotted line in the diagram connect to the lower sifting level and are updated. After sifting is performed, the new fan-out values of the nodes below the dotted line are*

```
SP_sift(upper, lower) {
    $\sum P_{sw}[total]$ -= ($\sum P_{sw}[upper]$ + $\sum P_{sw}[lower]$ + $\sum P_{sw}[below]$)
    for each node i at lower level {
        ref_update(lower, i, -BDD[lower][i].ref)
    }
    perform local variable exchange
    for each node i at lower level {
        ref_update(lower, i, BDD[lower][i].ref)
    }
    $\sum P_{sw}[total]$ += ($\sum P_{sw}[upper]$ + $\sum P_{sw}[lower]$ + $\sum P_{sw}[below]$)]
    }
}
```

Figure 6: Updating Switching Probability During Sifting

computed, as shown in part (c) of Figure 7. The fan-out changes occurring below the dotted line cause the total switching probability to change. This is accounted for by $P_{sw}[below]$.

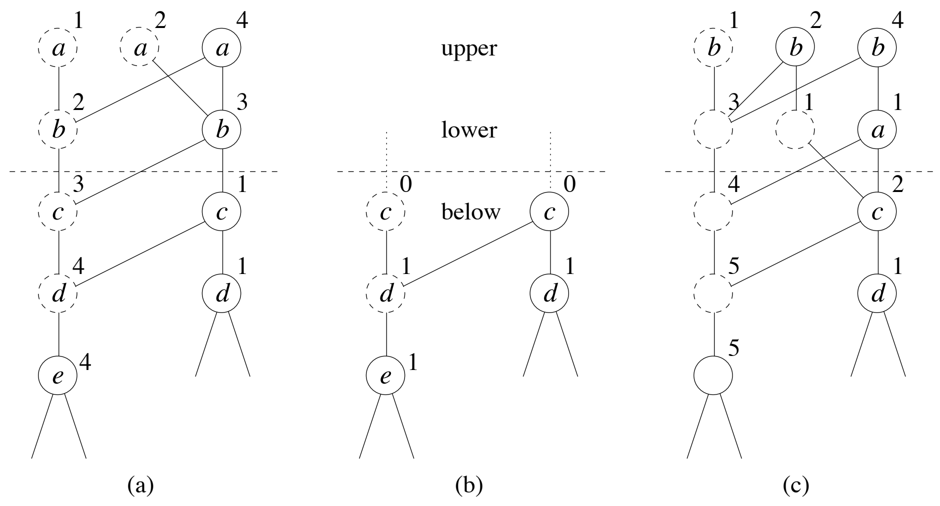


Figure 7: Reference Count Update During Sifting

7 Experimental Results

We have implemented a prototype evaluation of this technique based on the CUDD 2.3.0 [11] package for BDD manipulation with custom data types for the quasi-reduced BDD.

In the first set of experiments (Table 2), the output probabilities of the dependent variables are set to 0.5 (i.e. we assume each variable is equally likely to have a value of '0' or '1'). The columns labeled "Naive Order" show the size and total switching probability sums under an initial variable ordering obtained as the order in which the variables appear in the `.pla` files from the *LGSynth93* benchmark suite. Note that the total switching probability sums are unitless values. The columns "BDD Order" and "Power Order" compare the size and total switching activity estimates for circuits minimized by size and our proposed method respectively (for size reduction, the *group*

		Naive Order		BDD Order		Power Order	
name	in/out	*Size*	*Power*	*Size*	*Power*	*Size*	*Power*
5xp1	7/10	74	66	41	32	41	30
add6	12/ 7	308	272	28	23	28	23
apex7	49/37	1659	1237	289	176	316	158
bc0	26/11	589	369	522	320	540	310
chkn	29/7	741	298	267	132	361	85
duke2	22/29	972	268	355	107	361	93
exp	8/18	209	84	169	80	176	62
in2	19/10	2360	1464	234	116	244	95
in7	26/10	234	146	79	22	78	20
inc	7/ 9	76	47	70	45	70	45
intb	15/ 7	1033	687	537	349	556	305
misex3	14/14	1300	644	520	224	592	205
sao2	10/ 4	154	73	80	36	87	34
tial	14/ 8	1306	1027	579	423	579	423
vg2	25/ 8	1043	650	80	46	80	46
x6dn	39/ 5	274	142	240	143	244	122

Table 2: All Output Probabilities of Dependent Variables are set to 0.5.

sifting algorithm with convergence [11] is applied). Optimized circuits outperform the initial ones for all benchmarks. We observe a moderate reduction of switching activity in the circuits after power minimization compared to the size reduced ones. This is attributed to the fact that our cost model computes the sum of all internal switching probabilities and there are obviously fewer values to be summed when the total number of vertices is reduced. The more interesting outcome of these experiments is that in some cases, allowing the BDD to increase in size slightly (and thus increasing the size of the underlying BDD-mapped circuit) can cause a further reduction in estimated dynamic power dissipation.

In the second set of experiments (Table 3), the output probabilities of the dependent variables are alternatively set to 0.1 and 0.9 (i.e., $\{P[x_1] = 0.1, P[x_2] = 0.9, P[x_3] = 0.1, \ldots\}$). Columns "BDD Order" and "Power Order" compare the size and switching activity for circuits minimized by size and our proposed algorithm respectively. On the average, the switching activity is reduced by 20%, while the size increase is only 12%. For some cases, the switching probability can be significantly reduced. As an example, consider "chkn", "in2" and "x6dn", where the switching activity is reduced by less than 1/2, while the circuit sizes are increased only by 30%, 5% and 13% respectively. These results show that knowledge about the statistics and correlation of the circuit input signals can be exploited to sometimes give significant dynamic power dissipation reductions in BDD-mapped circuits.

In order to evaluate the quality of the presented heuristic, an exhaustive enumeration is performed on benchmarks having up to 10 variables. The worst and best results are shown in columns "Worst Order" and "Best Order" respectively. The experiment shows our algorithm to obtain the optimal results for the smaller tested functions.

8 Conclusions

A method for the reduction of the overall sum of internal switching probabilities for a BDD has been presented based on efficient local variable exchange operations. When the switching probability is

		BDD Order		Power Order		Worst Order	Optimal Order
name	in/out	*Size*	*Power*	*Size*	*Power*	*Power*	*Power*
5xp1	7/10	42	16	41	15	43	15
add6	12/ 7	28	14	28	14	-	-
apex7	49/37	289	54	329	47	-	-
bc0	26/11	522	140	551	131	-	-
chkn	29/7	267	77	348	33	-	-
duke2	22/29	355	79	399	72	-	-
exp	8/18	169	48	174	39	73	39
in2	19/10	234	73	247	25	-	-
in7	26/10	79	6	86	5	-	-
inc	7/ 9	70	20	75	19	45	19
intb	15/ 7	537	137	577	124	-	-
misex3	14/14	520	150	592	122	-	-
sao2	10/ 4	80	13	89	10	66	10
tial	14/ 8	579	242	691	227	-	-
vg2	25/ 8	80	25	80	24	-	-
x6dn	39/ 5	240	78	272	28	-	-

Table 3: Output Probabilities of Dependent Variables Alternate Between 0.1 and 0.9.

used as an estimate for circuit switching activity in BDD-mapped circuits, it is shown that dynamic power dissipation can be reduced using the technique. The second set of experiments suggests that if statistical information is known about the nature of the circuit input signals prior to using the minimization technique, significant reductions in internal switching activity and hence, dynamic power dissipation can occur. Furthermore, it is shown that the increase in the size of the resulting circuits is relatively small as compared to that obtained through the use of a BDD size reduction technique.

Our model can be tailored to the target technology at hand, further increasing the quality of the overall power dissipation estimate through the inclusion of static terms and more knowledge about the internal capacitances.

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] S.J. Friedman and K.J. Supowit. Finding the optimal variable ordering for binary decision diagrams. In *Design Automation Conf.*, pages 348–356, 1987.

[6] N. Ishiura, H. Sawada, and S. Yajima. Minimization of binary decision diagrams based on exchange of variables. In *Int'l Conf. on CAD*, pages 472–475, 1991.

[7] L. Lavagno, P. McGeer, A. Saldanha, and A.L. Sangiovanni-Vincentelli. Timed shannon circuits: A power-efficient design style and synthesis tool. Technical report, CADENCE, Berkeley Laboratories, 1994.

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

[9] 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.

[10] K.P. Parker and E.J. McCluskey. Analysis of logic circuits with faults using input signal probabilities. *IEEE Trans. on Comp.*, 24:573–578, 1975.

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