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

Rolf Drechsler
University of Freiburg
Freiburg, Germany
drechsle@informatik.uni-freiburg.de

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 CMOS Pass Transistors has in simulation (using a commercially available process model) shown reduced power dissipation characteristic. Experimental results on a set of MCNC benchmarks 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, see e.g. [7] for an excellent overview.

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 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 computations of a switching probability at an internal BDD vertex can be manipulated through purely local operations on the graph. Based on the local graph operations, we develop a heuristic approach for minimizing the overall switching activity and then perform a set of experiments evaluating the effectiveness of the minimization method as well as the effect on the size of the resultant BDD. To further emphasize the applicability of our approach, we develop a mapping method for *Pass Transistor Logic* (PTL) circuits for a commercially available CMOS process. For a given set of inputs, the process specific transistor models allow not only the functionality of the resulting circuit to be verified, but also an estimate of power dissipation to be derived.

2 Power Dissipation in CMOS

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.

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

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)
- P_{sw_k} is the switching activity at circuit node k

Equation 1 shows that the dynamic power dissipation component is dependent on the switching activity parameter, P_{sw_k}. Past research has shown that the switching activity parameter is highly dependent on temporal correlations of circuit input signal values [12]. 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, P_{sw_k} may be approximated by the switching probability, $\hat{P_{sw_k}}$.

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

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

3 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*:

$$f = \overline{x} f_0 + x f_1 \quad (3)$$

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 [13, 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 Estimation

An output probability of a function, f, denoted as $P[f]$ is the probability that f has a value of "1" at some arbitrary time of observation [14, 6, 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 the corresponding cofactors f_0 and f_1. In terms of a BDD, this relationship is shown in Figure 1.

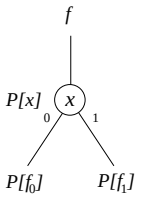

Figure 1: Switching Probability in BDD Vertex.

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 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.

$$(1 - P[x])(1 - P[f_0]) + P[x](1 - P[f_1]) \quad for \;\; f = 0 \quad (4)$$

$$(1 - P[x])P[f_0] + P[x]P[f_1] \quad for \;\; f = 1 \quad (5)$$

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 $t2$:

$P[f^{t1} \cap f^{t2}]$	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

Table 1: Probabilities at Subsequent Observations.

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:

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

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

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

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 [11] or to a "pass-transistor" based circuit as described in [2, 5, 15]. 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 MUX 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 [8]. 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 Equation 8, 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 [9]. The key property is that a local exchange of variables in a BDD can be done solely by redirecting edges locally in the diagram. Since our main concern is to minimize the overall switching probability (activity) of the resulting circuit, we show that the switching probability can be computed by local operations during sifting.

Consider a function f represented by a BDD. The switching probability of f (as represented by a root node in a BDD) is independent of the variable ordering, as switching probability is a functional property of f. 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 the functional property of switching probability, 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.

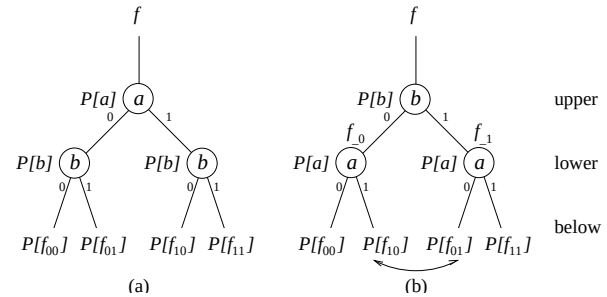

Figure 2: BDD Local Variable Exchange and Effect on Switching Probabilities.

Finally we need to show that the switching probabilities of the nodes (f_0 and f_1) at levels denoted by the word "lower" in Figure 2 can be computed locally during sifting.

From Equation 8 we derive:

$$P_{sw}[f_0] = 2((1-P[a])(1-P[f_{00}]) + P[a](1-P[f_{10}])))((1-P[a])(P[f_{00}]) + P[a](P[f_{10}])) \quad (8)$$

$$P_{sw}[f_1] = 2((1-P[a])(1-P[f_{01}]) + P[a](1-P[f_{11}]))((1-P[a])(P[f_{01}]) + P[a](P[f_{11}])) \quad (9)$$

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, [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 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.

6 Power Minimization

Given the results described above, we can now state the BDD based algorithm for the minimization of total estimated power dissipation due to switching 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 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 the input signals to 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.

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.

```
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.

6.2 Heuristic Minimization Algorithm

The proposed heuristic minimization algorithm, iteratively seeks a variable order reducing the circuit's switching probability weighted by the fan-out cost for each node. We outline the procedure in Figure 3.

The sifting and re-calculation of output and switching probabilities 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 fan-out changes for nodes below in the diagram can be handled. Figure 4 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 probabilities and reference counters may change, hence also 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 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 5. 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 5.*

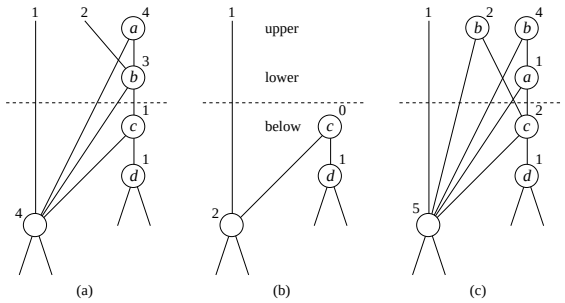

Figure 5: Reference Count Update During Sifting.

fan-out	Stage 1 p/n	Stage 2 p/n	Stage 3 p/n	Stage 4 p/n	Stage ... ...
1	4/2	4/2	-	-	-
> 1	4/2	8/4	8/4	-	-
> 3	4/2	16/8	16/8	-	-
> 6	4/2	8/4	32/16	32/16	-
> 12	4/2	16/8	64/32	64/32	-
...	...	...	...	...	...

Table 2: Driver width in lambda.

7 CMOS PTL Mapping

As mentioned in Section 5 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 subcircuit shown in Figure 6. 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 as shown in Table 2. (This assumes the worst case situation, all outputs of the same polarity.) The number of inverter stages ranges from 2 (as shown in Figure 6) (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).

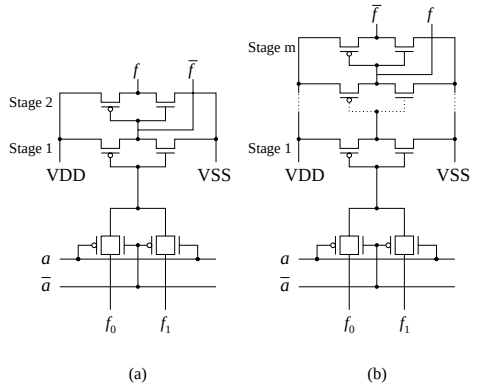

Figure 6: PTL Mapping of BDD Node.

8 Experimental Results

We have implemented a prototype evaluation of this technique based on the CUDD 2.3.0 [16] package for BDD manipulation with custom data types for storing the internal switching properties of the BDD nodes.

In the first set of experiments (Table 3), the output probabilities of the support 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 Ord." show the size and estimated dynamic power dissipation ($\hat{PD}$) 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 $\hat{PD}$:s are unitless values. The column "BDD Ord." is minimized by size and "Power Ord." is minimized by our proposed method (for size reduction, the *group sifting algorithm with convergence* [16] is applied). The column labeled "CPU" shows the run time in seconds for our optimization heuristic. Area optimized circuits "BDD Ord." outperform the initial ones for all benchmarks both with respect to size and estimated power consumption, the latter attributed to the fact that our cost model computes the sum of all internal switching probabilities (weighted by the fan-out of each node), hence fewer nodes leads (in general) to reduced power dissipation. 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, "Power Ord.". This is due to decreased internal switching probabilities in the circuitry obtained by our minimization method.

In the second set of experiments (Table 4), the output probabilities of the support 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 Ord." and "Power Ord." compare the size and estimated dynamic power dissipation for circuits minimized by size and our proposed algorithm respectively. The column labeled "CPU" shows the run time in seconds for our optimization heuristic. On the average, the power dissipation is reduced by 20%, while the size increase is only 12%. For some cases, the power dissipation can be significantly reduced. As an example, consider "chkn", "in2" and "x6dn", where the estimated dynamic power dissipation is reduced to less than 1/2, while the circuit sizes are increased only by 30%, 5% and 13% respectively. These results show that knowledge about the output probabilities of the circuit's input signals can be exploited to sometimes give significant dynamic power dissipation reductions in BDD-mapped circuits. For evaluation of the quality of the presented heuristic, an exhaustive enumeration of all variable orders is performed on benchmarks having up to 10 variables. The worst and best results (optimal) are shown in columns "Worst Ord." and "Optimal Ord." respectively. The experiment shows our algorithm to obtain the optimal results for the these functions. As for other sifting based heuristics, we conclude our method to give high quality results within reasonable run times.

For the PTL implementation we have used transistor models from a commercial 0.35u process. All analog simulations are performed on the process specific models using a commercial "SPICE-like" circuit simulator. The unit transistor width

name	in/out	Naive Ord. *Size*	Naive Ord. $\hat{PD}$	BDD Ord. *Size*	BDD Ord. $\hat{PD}$	Power Ord. *Size*	Power Ord. $\hat{PD}$	CPU sec
5xp1	7/10	74	66	41	32	41	30	0.01
add6	12/ 7	308	272	28	23	28	23	0.01
apex7	49/37	1659	1237	289	176	316	158	11.19
bc0	26/11	589	369	522	320	540	310	2.84
chkn	29/7	741	298	267	132	361	85	3.43
duke2	22/29	972	268	355	107	361	93	2.10
exp	8/18	209	84	169	80	176	62	0.06
in2	19/10	2360	1464	234	116	244	95	3.48
in7	26/10	234	146	79	22	78	20	0.46
inc	7/ 9	76	47	70	45	70	45	0.02
intb	15/ 7	1033	687	537	349	556	305	0.98
misex3	14/14	1300	644	520	224	592	205	1.60
sao2	10/ 4	154	73	80	36	87	34	0.05
tial	14/ 8	1306	1027	579	423	579	423	0.27
vg2	25/ 8	1043	650	80	46	80	46	0.11
x6dn	39/ 5	274	142	240	143	244	122	2.58

Table 3: All Output Probabilities of Inputs are set to 0.5.

name	BDD Ord. *Size*	BDD Ord. $\hat{PD}$	Power Ord. *Size*	Power Ord. $\hat{PD}$	Worst $\hat{PD}$	Opt. $\hat{PD}$	CPU sec
5xp1	42	16	41	15	43	15	0.01
add6	28	14	28	14	-	-	0.01
apex7	289	54	329	47	-	-	23.62
bc0	522	140	551	131	-	-	1.90
chkn	267	77	348	33	-	-	2.05
duke2	355	79	399	72	-	-	1.04
exp	169	48	174	39	73	39	0.06
in2	234	73	247	25	-	-	0.69
in7	79	6	86	5	-	-	0.40
inc	70	20	75	19	45	19	0.02
intb	537	137	577	124	-	-	1.14
misex3	520	150	592	122	-	-	0.51
sao2	80	13	89	10	66	10	0.04
tial	579	242	691	227	-	-	0.89
vg2	80	25	80	24	-	-	0.23
x6dn	240	78	272	28	-	-	8.57

Table 4: Output Probabilities set {0.1, 0.9} Alternating.

is 0.35u. Transistor sizings from Table 2 give approximately 100p seconds rise and fall times. This results in a total delay of 150p seconds (select to output) for a PTL MUX implementation with a fan-out of 4 (without internal routing parasitics). To gain confidence in our mapping tool, the functional behavior for some smaller circuits from the *LGSynth93* benchmark suite ("5xp1", "majority" and "xor5") has been successfully verified against the specification using exhaustive simulation of input vectors. For the experiments below we have applied pseudo-random values {'0','1'} (according to output probability) for the support variables. The pseudo-random vectors are generated from constant seeds, producing exactly the same set of vectors in the same order for each simulation run.

To ensure stable outputs, cycle time (1us) is set well longer than the (combinational) critical path. The total power dissipation is calculated from the average current (through the circuit without routing parasitics) times the VDD (3.3v). Power dissipation for the primary inputs (support variables) are assumed negligible as they only affect the select-inputs of the multiplexors.

Due to the extensive time of analog simulation, only a small selection of interesting benchmarks was chosen. To keep runtimes manageable the number of cycles was limited to 1000 and CPU time limited to approximately 4 hours, for each selected experiment. Columns "PTL" in Table 5 show the energy dissipated in pico watt seconds during the 1000us simulation, (in cases where simulation was earlier aborted due to time limit, an interpolation is given, e.g. "chkn" simulated for only 1000/5 cycles). Note, that neither our minimization method, nor the mapping tool makes any attempt at reducing glitches in the final circuit. However, the effect of glitches is present in the analog simulations. Furthermore, for the larger benchmarks, only a small subset of the input vector space has been applied, therefore the presented simulation results can be used only as rough estimates of the power dissipation. Table 5, gives a comparison between estimated and simulated values, where the output probability of the support variables is set to 0.5. The correlated results confirm our minimization method to be applicable to CMOS PTL circuits.

name	in/out	Naive Ord. $\hat{PD}$	Naive Ord. *PTL*	BDD Ord. $\hat{PD}$	BDD Ord. *PTL*	Power Ord. $\hat{PD}$	Power Ord. *PTL*
5xp1	7/10	66	2422	32	895	30	858
chkn	29/7	298	-	132	729*5	85	626*5
exp	8/18	84	-	80	1587*2	62	1281*2
in2	19/10	1464	-	116	851*5	95	697*5
sao2	10/ 4	73	-	36	1305	34	1077

Table 5: Analog Simulation vs. Estimated Values.

9 Conclusions

A method for the reduction of the overall sum of internal switching probabilities for a BDD based on efficient local variable exchange operations has been presented. When the switching probability is used as an estimate for circuit switching activity in BDD-mapped circuits, it is shown that the 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 be obtained. 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.

Our minimization method has been validated by a straightforward mapping to PTL circuitry. Simulation results (utilizing transistor models from a commercial CMOS process) are well correlated to our estimates, which confirms the applicability of our approach.

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] S. Chakravarty. On the complexity of using BDDs for the synthesis and analysis of Boolean circuits. In *Communication, Control and Computing*, pages 730–739, 1989.

[7] S. Devadas and S. Malik. A survey of optimization techniques targetting low power VLSI circuits. In *Design Automation Conf.*, pages 242–247, 1995.

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

[9] 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.

[10] R. Krieger. PLATO: A tool for computation of exact signal probabilities. In *VLSI Design Conf.*, pages 65–68, 1993.

[11] 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.

[12] R. Marculescu, D. Marculescu, and M. Pedram. Efficient power estimation for highly correlated input streams. In *Design Automation Conf.*, June 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.P. Parker and E.J. McCluskey. Analysis of logic circuits with faults using input signal probabilities. *IEEE Trans. on Comp.*, 24:573–578, 1975.

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

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