An Efficient Algorithm for Low Power Pass Transistor Logic Synthesis*

Rupesh S. Shelar, Sachin S. Sapatnekar
Department of Electrical and Computer Engineering,
University of Minnesota, Minneapolis, MN 55455.
Email: {rupesh,sachin}@ece.umn.edu

Abstract

In this paper, we address the problem of power dissipation minimization in combinational circuits implemented using pass transistor logic (PTL). We transform the problem of power reduction in PTL circuits to that of BDD decomposition and solve the latter using the max-flow min-cut technique. We use transistor level power estimates to guide the BDD decomposition algorithm. We present the results obtained by running our algorithm on a set of MCNC benchmark circuits, and show on an average of 47% power reduction over these circuits; the comparison with the previously proposed low power pass transistor logic synthesis algorithms shows an average improvement of over 23% over the best previously published approach.

1 Introduction

Power dissipation is becoming a critical problem in modern day deep sub-micron circuits, especially in case of circuits that are used in portable battery-operated devices. The problem of power optimization at various levels of abstraction has been addressed by numerous researchers. At the logic level, power optimizations include techniques such as gated clocks and precomputation; the latter include the use of the observability *don't cares* to disable the clock signal at the input registers [1]. Using a similar approach, Ruan *et al.* propose bipartitioned codec architecture in which output values are encoded using the minimum number of bits, and then decoded using the decoder in the next clock cycle, or computed conditionally [2, 3]. A limitation of the precomputation scheme [1] is the addition of extra logic to the circuit, while the bipartitioning codec approach may not be always optimal. Among the recently popular logic families, pass transistor logic (PTL) is promising for low power applications as compared to conventional static CMOS because of its lower transistor count, lower capacitances due to the possible use of NMOS transistor as pass transistors and therefore, good performance and lower power dissipation [4, 5]. For power optimization of PTL circuits, Lindgren *et al.* propose the use of sifting [6] which reduces switching activity in BDD mapped PTL circuits [7]. Another approach by Tavares *et al.* proposes employing split cofactors based on the Shannon expansion at the root of BDD's, with the variable corresponding to root node being used as a control input to disable the inverters [8]; the disabled inverters cannot make low-to-high transition, resulting in reduced switching activity. This approach differs from an approach of independent of cofactors by Alidina *et al.* in disabling inverters rather than disabling registers and also in algorithm as [1] uses area-efficient co-factors while [8] uses the power-efficient cofactors.

In this paper, we propose a BDD decomposition technique to minimize the power dissipation in combinational logic under the assumption that all primary inputs and primary outputs are registered. We use the switching probability estimation technique proposed in [7] to estimate the switching probabilities in PTL circuits and also take into account the capacitance driven by each node in the PTL circuit, unlike [7] which uses a linear fanout model. Unlike the previous approaches [1, 8], which use a single variable to disable the inputs of independent co-factors, we decompose the logic function using the max-flow min-cut technique to find the cut in the BDD that minimizes the power dissipation; the cut yields a subset of variables used as inputs to select logic that is used to disable the part of the circuit that does not perform useful computation in a given clock cycle. Our decomposition-based implementation model is more flexible than the bipartitioning codec architecture proposed in [3] and allows us to find optimum decomposition; optimality of decomposition is ensured due to the use of Ford-Fulkerson algorithm [9] to find the min-cut.

The organization of the rest of the paper is as follows. In Section 2, we describe the power estimation technique used for PTL circuits. In Section 3, we illustrate the BDD decomposition and propose an algorithm for minimizing the power dissipation in Section 4. In Section 5, we discuss the experimental results obtained using the algorithm, followed by concluding remarks in Section 6.

*This work was supported in part by SRC under award 99-TJ-692.

2 Power Model

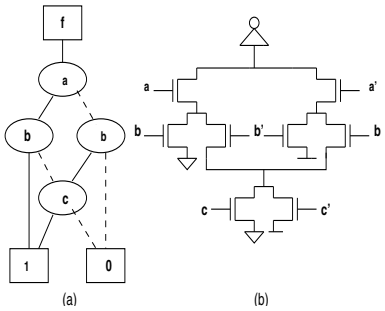

Figure 1. Correspondence between a BDD node and its PTL implementation (a) BDD for f=ab+c(ab'+a'b), (b) Corresponding PTL Implementation.

Figure 1(a) shows the BDD for the function f = ab+c(ab'+a'b), while Figure 1(b) shows the corresponding PTL implementation in which every node in the BDD is translated into a 2-input multiplexer. Given the probabilities for the inputs, the switching probability of a function f can be expressed in terms of the probabilities of its cofactors, f_x and $f_{\overline{x}}$, where x is an input. Equations (1) and (2) express the probability of f being 1 and 0, respectively, while switching probability is given by Equation (3).

$$p_{(f=1)} = p_{(x=1)} \times p_{(f_x=1)} + p_{(x=0)} \times p_{(f_{\overline{x}}=1)} \quad (1)$$

$$p_{(f=0)} = p_{(x=1)} \times p_{(f_x=0)} + p_{(x=0)} \times p_{(f_{\overline{x}}=0)} \quad (2)$$

$$p_{(f_{switching})} = 2 \times p_{(f=1)} \times p_{(f=0)} \quad (3)$$

The switching probabilities for the nodes in BDD shown in Figure 1 computed assuming uniform input probabilities (i.e., $p(a = 1) = p(b = 1) = p(c = 1) = 0.5$) are shown inside the corresponding nodes in Figure 2(a). The triplet (p_{sw}, p_1, p_0) associated with each node corresponds to the switching probability, the probability of the node being evaluated to 1, and the probability of node being evaluated to 0, respectively. For instance, the nodes in Figure 2(a) corresponding to the nodes labeled 'b' shown in Figure 1(a), have the switching probability $p_{sw} = 3/8$, a probability of being evaluated to 1, p_1, of $3/4$, and a probability of being evaluated to 0, p_0, of $1/4$. The capacitances driven by each node can be computed by examining the PTL implementation. For instance, the node labeled 'c' shown in Figure 1(a) drives four source capacitances (C_s) of NMOS transistors. The capacitance driven by each node is shown in Figure 2(b), where C_i is the input capacitance of an inverter. Once the switching probabilities and capacitances are known, the dynamic power can be obtained by using the following formula, where, V_{dd} is supply voltage, f is the clock frequency, P_{sw} and C_{sw} are switching probabilities and capacitances, respectively.

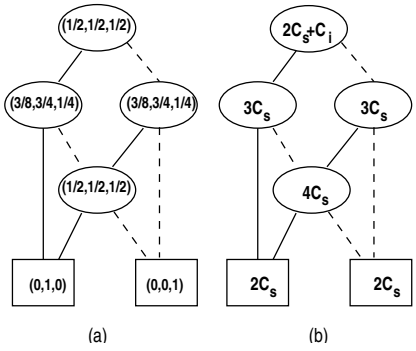

Figure 2. Power estimation in PTL circuits (a) Switching probability estimation, (b) Capacitance estimation.

$$Power = (\Sigma_{\forall nodes} P_{sw} C_{sw}) V_{dd}^2 f \quad (4)$$

3 Decomposition for Low Power

Figure 3 shows a general combinational logic circuit with registered inputs and outputs. Assuming PTL implementation of the combinational logic, we observe that switching activity occurs in the entire PTL network during every clock cycle, although parts of the PTL network may not perform useful computation. This can be observed from a property of BDD's that for any assignment of inputs, only one path from root to terminal node is active, so that the PTL implementation of this path performs useful computation for a given assignment, while the rest of the PTL network still dissipates power because of the switching of its inputs. Therefore, reduction in power dissipation can be achieved if we disable the part of the PTL network that does not perform useful work. Figure 4 shows

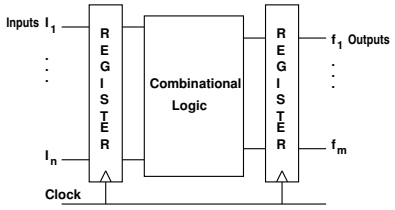

Figure 3. Combinational logic with registered inputs and outputs

the decomposition-based implementation model in which a subset of inputs is used to generate latch enable signals for the input registers. The enable signals constitute the 'select logic' block while the other combinational logic blocks,

CL_1 through CL_k, are PTL implementations of logic derived from original BDD's, as explained in the following subsection. Multiplexers are used to select the outputs from the block that performs useful computation in a given clock cycle. In this case, we observe that only the select logic and multiplexers are active all the time while other combinational blocks are not. Greater power reduction can be achieved if select logic and multiplexers dissipate small power and if the combinational blocks CL_1 through CL_k are active with small probabilities. In this case, the total power dissipation in the combinational logic is given by Equation (5), where p_i is the probability of combinational block CL_i being active and P_{CL_i} is the power dissipation in combinational block CL_i.

$$P_{Decomp} = P_{SelectLogic} + P_{Muxes} + \Sigma_{i=1}^{k} p_i P_{CL_i} \quad (5)$$

While the above equation is similar to the equation used in [3], Equation (5) is more general in the sense that k is allowed to take any value, unlike [3], where k is restricted to 2, in the bipartitioning codec architecture.

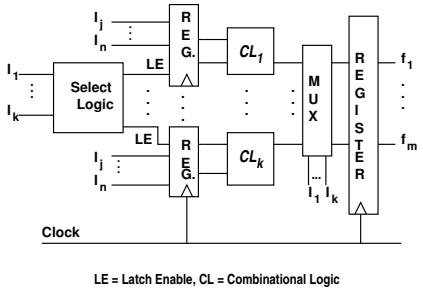

Figure 4. Decomposition model for implementation

3.1 Example

Consider the optimized BDD on 6 inputs for the carry output function for a 3-bit adder as shown in Figure 5(a). If we map the BDD to PTL using the implementation model of Figure 3, then the entire PTL network will dissipate power in each clock cycle. On the other hand, if we take a cut across the BDD containing the shaded nodes as shown in Figure 5(a), then we can decompose the BDD into smaller BDD's and build the original function using multiplexers and PTL implementation of these BDD's.

The process of decomposition can be explained as follows. To generate the BDD's for select logic we introduce dummy terminal nodes V_1, V_2, V_3 as shown in Figure 5(b) and encode them using scheme such as minimum-bit encoding or one-hot encoding. This is similar to the BDD decomposition proposed in [10] for performance oriented PTL synthesis. Figure 5(c) shows the select functions obtained by one-hot encoding, i.e., to generate o_1, we set $V_1 = 1, V_2 = 0, V_3 = 0$. Similarly, BDD's for o_2,o_3 are

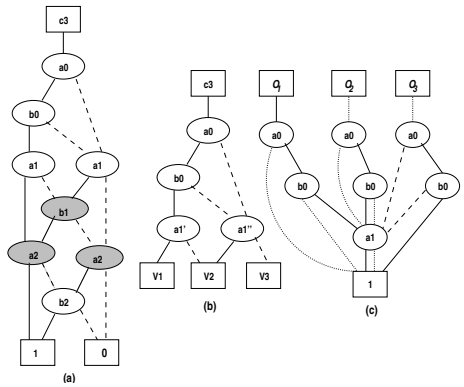

Figure 5. (a)BDD for carry function for 3-bit adder, (b) Introducing dummy nodes in the original BDD, (c) BDD's for select logic after one-hot encoding of dummy nodes

obtained. Functions o_1, o_2 and o_3 are used as latch enables for the registers and also as select inputs for the multiplexer to select among three combinational logic blocks whose BDD's are shown in Figure 6. As shown in Figure 6,

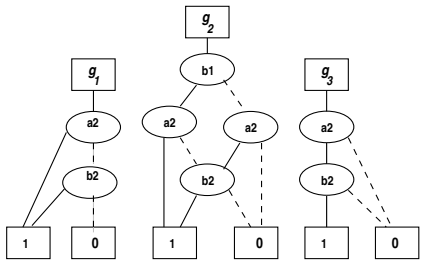

Figure 6. BDD's for functions in combinational logic blocks

BDD's do not share the nodes with the same functionality; such nodes are duplicated, which may cause an area overhead. The decomposed implementation for the carry output function excluding 'select logic' is shown in Figure 7. In this case, total power dissipation in combinational logic can be computed as follows: probabilities of o_0, o_1, o_2 being evaluated to 1 are computed using uniform probability assumption at the primary inputs, which can be relaxed for the known input probabilities.

$$\begin{aligned} P_{c3} &= P_{o_1,o_2,o_3} + P_{(3:1)mux} + p_{(o_1=1)} P_{CL_1} \\ &\quad + p_{(o_2=1)} P_{CL_2} + p_{(o_3=1)} P_{CL_3} \quad (6) \\ P_{c3} &= P_{o_1,o_2,o_3} + P_{(3:1)mux} + 0.125 P_{CL_1} \\ &\quad + 0.5 P_{CL_2} + 0.375 P_{CL_3} \quad (7) \end{aligned}$$

It is easily seen that the power dissipation in the combina-

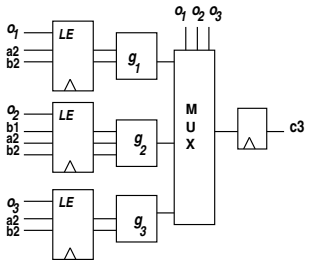

Figure 7. Decomposed implementation of Carry function

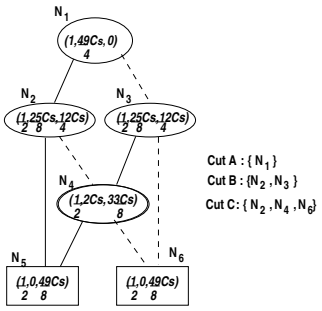

Figure 8. Estimating the cost of nodes

tional logic varies depending on the cut and that there are a large number of candidate cuts in a given BDD. Our objective is to find a cut in such that power given by Equation (5) is minimized. We propose an algorithm to find an optimum cut in the following section.

4 Algorithm

We represent a BDD as a directed acyclic graph (DAG) where the nodes are identical to the nodes of BDD, and the edges are identical to the edges of BDD edges, and are assigned a direction corresponding to variable ordering, from a lower indexed variable to higher indexed variable. As explained in section 2, the switching probability of a node depends on its cofactors and probability information of the input variable at that node. The power dissipation at each node can be computed by traversing a graph in depth-first search (DFS) manner. Similarly, the probability of node being selected as well as the power dissipation in select logic can be computed using DFS. The total power dissipation of PTL implementation of a function rooted at a given node is just the sum of power dissipation of its cofactors and power dissipation at that node[1].

The cost of each node is calculated as the sum of the power dissipation in select logic and product of probability of node being selected and power dissipation at that node. The cost estimation for the BDD shown in Figure 1 (a) is shown in Figure 8; the probability and capacitance estimations have already been shown in Figures 2 (a) and 2 (b), respectively. In Figure 8, we use $C_i = K \times C_s$ and choose $K = 10$ for the sake of simplicity. The triplet $(p_{select}, P, P_{select})$ associated with each node in Figure 8 corresponds to probability of the node being selected, power dissipation in the PTL network rooted at given node in terms of switching capacitance and power dissipation in the select logic. As an example, Figure 8 lists the three node cuts[2], namely Cut A, Cut B, Cut C containing one, two and three nodes, respectively. The cost of a cut is simply the sum of the cost of each node in the cut. After evaluating the cost of each node, the DAG can be converted into a flow network. Ford-Fulkerson algorithm [9] is then used to find the minimum cut that corresponds to the implementation with minimum power dissipation. In case of Figure 8, the minimum cut is Cut B with the cost $\frac{73Cs}{8}$, while the cost of Cut A and Cut C is $\frac{49Cs}{4}$ and $\frac{253Cs}{16}$. The pseudo-code of the overall algorithm is as shown in Figure 9.

```
Input:  G(V,E) = Digraph corresponding to given
BDD, V = Nodes, E = Edges.
Output:  S_cut = Optimum cut-set.
Steps:
/* Estimate the power dissipation using DFS.*/
1.  PowerEstimate(G);
/* Estimate the select probability.*/
2.  EstimateSelectProbability(G);
/* Estimate power dissipated in select logic.*/
3.  EstimateSelectLogicPower(G);
4.  For vεV Do /* Estimate the cost.  */
v->Cost = v->SelectLogicPower +
v->SelectProbability × v->PowerDissipation;
/* Convert to a flow network.*/
5.  G_Flow= CreateFlowNetwork(G(V,E));
/* Find an optimum cut.*/
6.  Ford-Fulkerson(G_Flow, G, S_cut);
```

Figure 9. Pseudocode for Algorithm to find optimum cut

Once the cut is determined, the vertices in the cut are replaced by dummy terminal nodes, which can be assigned unique codes to generate select logic, and also to generate the combinational logic blocks as illustrated in section 3. The following proposition states the time complexity of our algorithm.

Proposition 4.1 *The algorithm shown in Figure 9 takes $O(N^3)$ time to find an optimum cut, where N is the number of nodes in the original BDD.*

[1] In this analysis, signal correlations are ignored. While this may lead to some inaccuracies, it is generally considered as an acceptable approximation. This technique can be substituted by any other technique for probability computation that considers correlation.

[2] Note that cuts have been enumerated for illustrative purposes only and that our algorithm finds the minimum cut without any such enumeration.

Proof 4.1 The Step 1 to Step 5 take $O(N)$ time since Step 1, 2 and 3 used depth-first search that takes $O(\|V\| + \|E\|)$ time on a graph, where V is a set of nodes and E is a set of edges. In case of BDD's, all nodes, except the terminal nodes, have two fanout edges, and therefore, $O(\|V\| + \|E\|) = O(N)$. The Step 6 takes $O(\|V\|\|E\|^2)$ for Edmond-Karp implementation of Ford-Fulkerson algorithm [9]. Therefore, the algorithm takes $O(N^3)$ time to find minimum cut in the worst case.

Comment: Although time complexity of our algorithm is $O(N^3)$, a tighter upper bound can be obtained since we have observed that algorithm took less than a second in case of most of the MCNC benchmarks on Sun Ultra-60 machine and we also observed that the actual run times do not increase cubically. This is because the nodes that are not in the min-cut have higher costs as we move away from the cut and therefore, number of flow augmentations performed by Edmonds-Karp implementation are far less than $O(N^2)$. Since the Ford-Fulkerson algorithm takes $O(N)$ time for each augmentation, this more accurately reflects the trend of run times for the circuits in the MCNC suite.

One can observe that each cut corresponds to implementation with power dissipation equal to the cost of cut, and since the Ford-Fulkerson algorithm cut results in minimum cut, the decomposed implementation obtained by the algorithm in Figure 9 is the implementation that has the minimum power dissipation under our approximations. Specifically, some of the inverters in the select logic can be relocated and new inverters can be added or removed leading to inaccuracies in the capacitance estimation. However, these inaccuracies tend to be small since the area occupied by the select logic in the implementation is small as compared to the other combinational logic blocks. In this work, we have not taken into account area overhead due to node duplication and register duplication. This can be rectified and the a similar algorithmic framework can be used to trade off area and power of the decomposed implementation.

5 Experimental Results

The above algorithm has been implemented as a C++ program. The BDD package CUDD [11] was used for generating BDD's, along with sifting [6] for variable ordering for all our experiments. We assume the use of NMOS transistors as pass transistors and use of inverters after every three pass transistors in series. The size of each transistor was assumed to be $0.5\mu/0.25\mu$; capacitances are measured using area and perimeter coefficients for source and drain junctions for TSMC 0.25μ CMOS process [12]. The primary inputs were assumed to have 50% probability of being at either 0 or 1 throughout the experiments; the supply voltage and clock frequency was assumed to be 2.5V and 1GHz, respectively. We estimated the power dissipation in combinational logic for both the undecomposed and decomposed implementations of several MCNC benchmark circuits, as shown in Table 1. In the Table 1, Columns 2 through 6 show number of inputs/outputs, the power dissipation of a regular implementation, the power dissipation of decomposed implementation, the power reduction and CPU time on Sun Ultra-60 machine, respectively. The CPU time includes time for generation of BDD's, variable ordering, estimation and decomposition. The various benchmarks used here include variety of circuits, from arithmetic logic units to random logic. We observe significant power reductions in all of the cases, with an average reduction of 47.35%. We also observe that the power reduction is more significant in case of the circuits like ex4p, alu2, 9symml for which the number of outputs are relatively small as compared to number of inputs. On the other hand, in case of 5xp1, misex1 the reduction in power dissipation is relatively lower, and the number of outputs are relatively larger as compared to number of inputs. This correlation can be explained by observing that in case of circuits with a large number of outputs, the number of combinational logic blocks that may remain active in a given clock cycle is likely to be larger, resulting in a lower potential for power reduction.

Example	# of I/O	Regular Power(mWatt)	Decomposed Power(mWatt)	Reduction (%)	CPU time Seconds
alu2	10/6	1.48	.447	69.9	5.3
alu4	14/8	4.14	3.48	15.86	45
9symml	9/1	0.12	0.05	52.45	0.09
rd53	5/3	0.11	0.05	54.64	0.06
rd73	7/3	0.22	0.1	51.81	0.15
rd84	8/4	0.31	0.14	54.11	0.28
comp	32/3	0.90	0.41	53.76	1.59
5xp1	7/10	0.29	0.21	27.36	0.29
cordic	23/2	0.28	0.14	50.52	1.6
parity	16/1	0.21	0.1	51.59	0.17
cm162a	14/5	0.15	0.07	51.26	0.1
inc	7/9	0.39	0.18	52.63	0.42
t481	16/1	0.16	0.09	38.12	0.7
z4ml	7/4	0.135	0.82	39.25	0.07
f51m	8/8	0.35	0.18	47.86	0.48
misex1	8/7	0.16	0.1	35.52	0.11
c8	28/18	0.21	0.14	35.0	0.54
ex4p	128/28	3.72	1.23	66.78	19.28
i8	133/81	4.11	1.97	51.95	69.99
Average				47.35	

Table 1. Comparison of regular implementation with our decomposition-based implementation.

Table 2 shows a comparison of our algorithms with previously proposed algorithms [7, 8] for reducing power dissipation in PTL circuits. The experimental results reported in [7, 8] are based on the same assumption of uniform prob-

Example	Power Reduction(%)		
	Our	[7]	[8]
9symml	52.45	-	18.4
alu2	69.9	-	43.0
cordic	50.5	-	44.6
cm162a	51.26	-	39.4
f51m	47.85	-	32.6
parity	51.59	-	4.00
t481	38.13	-	10.2
z4ml	39.25	-	19.50
5xp1	27.36	6.25	-
inc	52.63	0	-
duke2	20.83	13.08	-
Average	45.61	6.44	26.46

Table 2. Comparison of our decomposition-based implementation with the methods of Lindgren *et al.* [7] and Tavares *et al.* [8].

ability of primary inputs and both report switching activity reductions and not the actual power reductions. However, we assume that the switching activity reductions reported in [7, 8] are translated to the same power reductions when the BDD's are mapped onto PTL. In case of [7], power dissipation reductions are considered with respect to the minimum size BDD and we use the same minimum size BDD as the BDD to be decomposed by our algorithm. Column 2 in Table 2 shows the power reductions by our algorithm while Columns 3 and 4 shows the power reductions obtained by the algorithms proposed in [7,8]. The '-' entry in Column 3 and Column 4 means that results were not available for the particular example in [7, 8]. We observe that our algorithm performs better in all the cases. As compared to the average power reduction of 26.46% by algorithm in [8] over the first 8 benchmarks, our algorithm obtains an average of 50.11% power reductions over the same benchmarks. Over the last 3 benchmarks, our algorithm obtains an average power reduction of 33.6%, as compared to the average power reduction of 6.44% obtained by [7].

6 Conclusion

In this paper, we have presented an efficient algorithm based on BDD decomposition for low power pass transistor logic synthesis. The results of our algorithm are encouraging since they show an average power reduction of 47.35% over a variety of MCNC benchmark circuits. The power reductions obtained by our algorithm, averaged over the MCNC benchmarks, are 23.65% and 27.16% higher than the power reductions obtained by previously proposed low power pass transistor logic synthesis algorithms [7] and [8], respectively. Therefore, our algorithm can serve as a viable alternative for low power pass transistor logic synthesis. The same framework of our algorithm can be used to consider area-power trade-offs in the decomposed implementations by considering the costs of node duplication and register duplication.

Acknowledgment

The first author would like to thank Michel Berkelaar of Magma Design Automation for valuable suggestions on PTL synthesis and other useful interactions.

References

[1] M. Alidina *et al.* Precomputation-Based Sequential Logic Optimization for Low Power. In *Proc. ICCAD*, pages 74–81, Nov. 1994.

[2] S.-J. Ruan *et al.* A Bipartition-Codec Architecture to Reduce Power in Pipelined Circuits. In *Proc. ICCAD*, pages 84–90, Nov. 1999.

[3] S.-J. Ruan *et al.* A Bipartition-Codec Architecture to Reduce Power in Pipelined Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* , 20(2):343–349, Feb. 2001.

[4] K. Yano *et al.* A 3.8ns CMOS 16 x 16 multiplier using complementary pass transistor logic. *IEEE Journal of Solid State Circuits*, 25(2):388–395, Apr. 1990.

[5] K. Yano, Y. Sasaki, and K. Rikino. Top-Down Pass-Transistor Logic Design. *IEEE Journal of Solid-State Circuits*, 31(6):792–803, June 1996.

[6] R. Rudell. Dynamic Variable Ordering for Ordered Binary Decision Diagrams. In *Proc. ICCAD*, pages 42–47, Nov. 1993.

[7] P. Lindgren, M. Kerttu, M. Thornton, and R. Drechsler. Low Power Optimization Technqiue for BDD Mapped Circuits. In *Proc. ASP-DAC*, pages 615–621, Jan. 2001.

[8] R. Tavares and M. Berkelaar. Reducing Switching Activity in Pass Transistor Circuits. In *Proc. IWLS*, Jun. 1999.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. Prentice-Hall India, New Delhi, 1998.

[10] R. S. Shelar and S. S. Sapatnekar. Recursive Bipartitioning of BDD's for Performance Driven Pass Transistor Logic Synthesis. In *Proc. ICCAD*, Nov. 2001. (To Appear).

[11] F. Somenzi. CUDD: CU Decision Diagram package, Release 2.3.0. http://vlsi.colorado.edu/ fabio/CUDD/.

[12] MOSIS Parametric Test Results for TSMC 0.25μ CMOS Runs. http://www.mosis.org/cgi-bin/cgiwrap/umosis/swp/params/tsmc-025/t04r-params.txt.