Synthesis of High Performance Low Power PTL Circuits

Debasis Samanta, M. C. Dharmadeep, and Ajit Pal
Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur
WB, INDIA 721302
Email: apal@cse.iitkgp.ernet.in

Abstract-***Among the various CMOS logic families, PTL has been recognized as one of the potential alternatives to static CMOS for the synthesis of high performance and low power circuits. Moreover, as BDDs can be readily mapped to PTL circuits, use of BDDs has been synonymous with the synthesis of PTL circuits. Most of the reported works on PTL synthesis are based on the Reduced Ordered BDDs (ROBDDs). We have developed a novel heuristic-based technique for obtaining Reduced Unordered BDDs (RUBDDs), which leads to circuits of smaller size having lesser delay and smaller power consumption compared to the existing results. We propose the technology mapping using the popular LEAP-like cells, such that the PTL circuit synthesis flow has the same flavor as that of the standard cell-based static CMOS circuit synthesis. We have also developed models for the estimation of delay and power consumption of the synthesized PTL circuits and compared those with the static CMOS and other existing PTL-based circuit realizations.***

I Introduction

In recent years, static CMOS has emerged as the technology of choice for VLSI circuit realization, because of the ease of design, robustness, scalability and other advantages that this logic style offers. But, static CMOS uses both pMOS and nMOS transistor networks, taking larger area, incurring more delay and higher power dissipation. As a consequence, in the present era of sub-micron technology, driven by the need for low power and high performance, researchers are looking for better alternatives. Recent studies [1]-[4] show that pass transistor logic (PTL) circuits provide superior performance in terms of area, delay and power. However, PTL circuits have some inherent limitations, such as threshold voltage drop across pass transistors, possibility of sneak paths and higher delay for cascaded pass transistors. While synthesizing PTL circuits, care should be taken to overcome these limitations.

Although PTL is a promising alternative to static CMOS, it demands for a radically different logic synthesis approach. It has been observed that binary decision diagram (BDD) representation of a logic function can be readily mapped onto a PTL network. Moreover, BDD-based synthesis avoids some of the limitations mentioned above. This has made BDD-based approach very attractive for the synthesis of PTL circuits. In earlier works [1] and [5], BDDs of the complete functions called monolithic BDDs, were constructed before technology mapping on PTL cells. The size of the monolithic BDDs grow exponentially with the number of input variables of the functions, making the approach unsuitable for functions of large number of variables, which is common in present-day VLSI circuits. Decomposed-BDD approach [2] has been proposed to overcome this problem. In decomposed-BDD approach, instead of creating a single big monolithic BDD, compact BDDs in terms of intermediate variables are constructed. These smaller BDDs can be optimized with ease and the approach allows synthesis of any arbitrary function with large number of variables. Liu et al. [3] and Chaudhury et al. [4] have proposed performance-oriented and area-oriented optimization techniques based on the decomposed BDDs.

In this paper, we have also used decomposed BDDs for PTL circuits synthesis. In addition to using decomposed BDDs, we have used a new heuristic, based on ratio parameters (RP), for optimal variable ordering. The concept of RP was used earlier [6], [7] for the synthesis of multiplexer-based circuits. The RP-based heuristic helps in obtaining BDDs with smaller number of nodes. However, BDDs generated by this approach may not have the same ordering of variables at the same level along different paths. These BDDs may be termed as Reduced Unordered BDDs (RUBDDs), in contrast to Reduced Ordered BDDs (ROBDDs) commonly used in the existing approaches. As the size of BDDs is more important than the same ordering of variables through different paths, we have adopted this new approach. Efficacy of this heuristic has been established by comparing the sizes of BDDs obtained using our approach with those works reported in [2], [3] and [4].

There exist several pass transistor logic families, such as, CPL, SRPL, DPL [8] and single-rail LEAP cells [1]. Among these logic families, we have selected single-rail MUX like cells, similar to LEAP cells, for technology mapping. Single-rail LEAP like cells provide synthesis flow similar to standard cell-based logic synthesis paradigm. Decomposed BDDs are mapped onto three basic LEAP-like cells, as we commonly do in the standard cell-based approach.

Existing approaches used the number of cells as a measure of area and number of levels as a measure of delay. Using these two parameters as performance metric, results were compared with static CMOS circuits. Instead, we have modeled the PTL cells to estimate power and delay and compared the results with that of the static CMOS circuits. Our results have been found to be superior with respect to all the existing reported results.

Rest of the paper is organized as follows. In Sec. 2, ratio parameters heuristic is presented. In Sec. 3, mapping a BDD to PTL circuit is discussed. Power and delay estimation in PTL circuits is given in Sec. 4. In Sec. 5, detail of the algorithm adopted in our approach is presented. Implementation details and experimental results are given in Sec. 6.

II Ratio Parameter Heuristic

Recursive application of Shannon's expansion theorem [9] on a switching function results into a tree called *binary decision tree,* where each node represent a function. A binary decision tree then can be reduced or compressed into a graph called BDD, using several elimination rules [11]. The tree thus obtained will be termed as *ordered* if splitting variables always follow the same order along any path in the tree, otherwise it will be termed as *unordered*. It is known that the size of a BDD depends on the ordering of the variables. We have proposed the RP-based heuristic technique in order to construct reduced unordered BDD. The concept of RP was used earlier [6], [7] for logic design using multiplexer networks. The RP is purely a functional property, which can be obtained from the completely specified minterm table consisting of only the true vectors of Boolean functions. The RP can be defined as follows:

Definition 1: For an n-variable function *f*, the set of parameters $\left\langle \frac{N_n}{D_n}, \frac{N_{n-1}}{D_{n-1}}, \ldots, \frac{N_1}{D_1} \right\rangle$ are called the ratio parameters (RP) of *f*, where N_i and D_i are the number of 1's and 0's, respectively, in the x_i-th column of the minterm table of *f*.

Example 1: Consider a function $f_1 = \Sigma(0, 1, 4, 6, 8, 10, 11)$. The RP of f_1 can be obtained as shown in Table 1.

Definition 2: If the input to a BDD node is a constant (0, or 1) or a function of just one variable, then it is called *closed input*, otherwise it is called *open input*. In other words, if it is a function of more than one variable, then it is called an *open input*.

Table 1

Minterm	x_4	x_3	x_2	x_1	f_1
0	0	0	0	0	1
1	0	0	0	1	1
4	0	1	0	0	1
6	0	1	1	0	1
8	1	0	0	0	1
10	1	0	1	0	1
11	1	0	1	1	1
RP	$\frac{3}{4}$	$\frac{2}{5}$	$\frac{3}{4}$	$\frac{2}{5}$	

Example 2: 0, 1, x, $\overline{\mathrm{x}}$ are examples of *closed inputs*, whereas *x+y*, $\mathrm{x} \cdot \overline{\mathrm{y}}$ are examples of *open inputs.*

Definition 3: The number of simultaneous occurrences of two or more variables is defined as the number of minterms in which the variables are present with the same variable values.

Example 3: In the minterm table of Table 1, the simultaneous occurrence of $\overline{x}_4 \cdot \overline{x}_1$ is 3 and that of $x_4 \cdot \overline{x}_1$ is 2.

Definition 4: The *k-group* of variables is defined as a group of *k* variables, such that the number of simultaneous occurrences is equal to 2^{n-k}, where *n* is the number of input variables or the number simultaneous occurrences is nil.

Example 4: In the minterm table of Table 1, we find that there are four *k-group*s of size 3, namely, $\overline{x}_3 \overline{x}_2 \overline{x}_1$, $\overline{x}_4 \overline{x}_3 \overline{x}_2$, $\overline{x}_4 \overline{x}_2 \overline{x}_1$ and $x_4 \overline{x}_3 x_2$ since the number of their simultaneous occurrences for each of them is $2^{4-3} = 2$.

Theorem 1: If there is a *k-group* of size *k* in the on-set of the function, then we have a *closed input* at depth *k* in the tree representation of the function.

Definition 5: The level of a Boolean node is defined to be how far it is down from the root node. The root node is in level 0. In a tree structure, the *path length of a closed input* (*PLCI*) at a given level is defined to be the number of levels it is located away from that level.

The basic approach of our RP-based heuristic method is to find a variable for the current level, such that a closed input is available at the minimum value of *PLCI*. If more than one variable satisfy this condition, then a variable among them having maximum number of closed inputs with minimum *PLCI* is selected.

III Decomposition of BDD

There exist two forms of BDDs; monolithic BDD and decomposed BDD of a Boolean network. A monolithic BDD is equivalent to two-level sum of product representation of the Boolean network, whereas decomposed BDD is equivalent to multilevel SOP form of a Boolean network. The disadvantage of decomposed BDD over monolithic BDD is that, decomposed BDD may have some redundancies embedded in it, whereas it is removed in monolithic BDD. However, monolithic BDD representation is not amenable to large Boolean networks, where the size of the BDD may blow up and processing such a BDD is computationally very expensive. On the other hand, if each node in the decomposed BDD is too tiny, then it involves a lot of redundancy, which may lead to inefficient logic synthesis. It is necessary to find out a reasonable trade-off between monolithic BDD and decomposed BDD. In the present work, we have used two techniques; first, replacement of all re-convergent structures into bigger components, and second, partial collapsing of nodes, such that Boolean nodes are neither very large nor very small but are of moderate size in our decomposed BDDs.

IV Mapping a BDD to PTL Circuit

After obtaining a BDD, it can be directly mapped onto PTL cells. One simplest mapping is to map each node in the BDD by a 2×1 MUX cell. A MUX cell can be implemented either with nMOS-pMOS transistors or nMOS transistor only. When the network is implemented using nMOS transistor only, then it uses more number of transistors than the implementation using nMOS-pMOS transistors, because it needs signal in both the input phases for each binary node. However, it produces good results in terms of gate areas [2]. Implementation of only nMOS network has one major limitation; the voltage is V_{DD}-V_T for output "1" and V_{SS} for output "0". As a consequence, it requires restoring the voltage level at some interval in a long chain of pass transistors. This problem is sorted out in [1] using buffer based PTL cell library. The library consists of three function cells Y1, Y2, and Y3 [1]. Use of this PTL cell library has three advantages: (i) the cells take care of the level restoration logic, (ii) avoids the problem of insertion of buffers, and (iii) it provides the same flavor as standard cell-based static CMOS circuits synthesis. This has motivated us to use this library in our work. In the task of the technology mapping, BDDs constructed by our heuristic approach are covered using these PTL cells. We are to cover the BDD by PTL cells so that the total number of cells used is minimized. To do this, any optimal covering technique such as dynamic programming, genetic algorithm etc. can be followed. In our work, we have adopted a greedy approach for this purpose. In this greedy approach, the BDD is first mapped onto as many Y3 cells as possible, then onto Y2 cells and finally the leftover nodes are covered by Y1 cells.

V Power and Delay Estimations

In this section, we propose the models for the estimation of switching power and delay in a PTL circuit.

Calculation of switching power: In PTL circuits, switching power is the main source of power dissipation. The average switching power can be expressed as

$$P_{switching} = \sum_{i=1}^{n} \left[\alpha_i \cdot C_{iL} \cdot V_{DD} + \sum_{j=1}^{\#\mathrm{int}.nodes} \alpha_{ij} \cdot C_{ij} \cdot V_j \right] V_{DD} \cdot f_{CLK} \qquad (1)$$

where, V_{DD} = supply voltage, V_j = voltage swing on node *j*, usually $V_j = V_{DD}$-V_T, V_T being the threshold voltage of nMOS transistor, f_{CLK} = clock frequency, α_i = transition probability at output node of the *i*-th cell, α_{ij} = transition probability at *j*-th node of the *i*-th cell, C_{iL} = output load capacitance of the *i*-th cell and C_{ij} = capacitance at the *j*-th node in the *i*-th cell.

Calculation of switching activity: Suppose *f* is an input switching function. We define the transition activity $\alpha_{0\to1} = \alpha_0 \cdot \alpha_1$, where α_0 and α_1 are the probability that output is 0 and 1, respectively. For a given BDD of *f*, the transition activity $\alpha_{0\to1}(f)$ can be calculated [13], recursively, by expanding it with respect to x_i, say. That is

$$\alpha_{0\to1}(f) = \alpha_{0\to1}(x_i) \cdot \alpha_{0\to1}\left(f_{x_i=1}\right) + \alpha_{0\to1}\left(\overline{x_i}\right) \cdot \alpha_{0\to1}\left(f_{x_i=0}\right) \qquad (2)$$

and $\alpha_{0\to1}(f)$ = 1 if *f* = 0 or 1, that is, *f* is a terminal node.

Using this formula, transition activities at any node in a PTL circuit can be calculated.

Calculation of the capacitances: There are various components of capacitances within a PTL cell. These are:

C_{out} = Capacitance at the output of the cell.
C_1 = Capacitance at the input of the buffer.
C_2 = Internal node capacitance in Y2, Y3.

C_3 = Internal node capacitance in Y3 cell.

The values of theses capacitances can be calculated as given below:

$$C_{out} = C_{gp} + (C_{dp} + C_{dn}) + f_{out} \times (C_{dn} + C_{\text{int}}) \tag{3}$$

$$C_1 = 2C_{dn} + C_{gp} + C_{gn} + C_{dp} \tag{4}$$

$$C_2 = C_3 = 3C_{dn} \tag{5}$$

Here, C_{gp} and C_{dp} are the gate and the drain capacitances of pMOS transistor, respectively; C_{gn} and C_{dn} are the gate and the drain capacitances of nMOS transistor, respectively. f_{out} is the number of fanouts and C_{int} is the interconnect capacitance.

Calculation of Delay

Delay of a PTL circuit can be obtained by computing the critical path in the circuit. Once the critical path is found, total delay can be calculated from it as the sum of the delays in all PTL cells in this critical path. Now, delay in a Y-cell is

Delay(Y) = delay in the buffer + delay in the largest path within the cell

where, delay in the buffer (τ_{buffer}) is given by,

$$\tau_{buffer} = \frac{R_p \cdot C_{out} + R_n \cdot C_{out}}{2} \tag{6}$$

Thus, delays in other varieties of Y-cells can be expressed using RC-delay model as given below:

$$\text{Delay}(Y_1) = \tau_{buffer} + R_n \cdot C_1 \tag{7}$$

$$\text{Delay}(Y_2) = \tau_{buffer} + 2R_n \cdot C_1 + R_n \cdot C_2 \tag{8}$$

$$\text{Delay}(Y_3) = \tau_{buffer} + 2R_n \cdot C_1 + R_n \cdot C_2 \tag{9}$$

where, R_n = on-resistance of an nMOS transistor and R_p = on-resistance of a pMOS transistor.

VI Algorithm

We propose an algorithm to realize PTL circuits of arbitrarily large size. The algorithm *ptlRP* for the realization of PTL circuits is outlined below:

```
Algorithm ptlRP
1.  Partitioning the input circuit into a number of components of moderate sizes.
    1.1 Extract all reconvergent structures in the circuit and merge
        each reconvergent structure into a single node.
    1.2 Perform partial collapsing of a node into its fanouts, if the
        cost gain of the partial collapsing is above a certain
        threshold value.
    1.3 Repeat Step 1.2 until there is no partial collapsing with the
        cost gain above the threshold value.
2.  Create BDD for each partition in the decomposed graph.
    2.1 Let the function $f$ representing the current node be an $n$-input
        function with Boolean variables $x_1, x_2, \dots, x_i, \dots, x_n$.
    2.2 Compute RPs for each Boolean variable $x_i$ in $f$.
    2.3 Case 1: If there is a variable $x_i$ for which $N_i$ or $D_i$ is equal
        to $2^{n-1}$ or 0 then select the variable for expansion.
    2.4 Case 2: If there is a variable $x_i$ whose $N_i$ or $D_i$ is equal to
        $2^{n-2}$ and there is another variable $x_j$ whose $N_j$ or $D_j$ is
        equal to or greater than $2^{n-2}$ and simultaneous occurrence
        of the variable $x_i$ and $x_j$ is equal to $2^{n-2}$ then select the
        variable $x_i$ for expansion with $x_j$ as one of the input of the
        binary node.
    2.5 Case 3: Find the smallest $k$-group. Let the size of it be $k$ ($k$
        $\geq$ 3). Select a variable from the $k$-group, which gives the
        maximum number of closed inputs at the earliest stage. For
        each $x_i$ in k-group, expand into a decision tree.
    2.6 Repeat Step 2.1 to 2.5 for each sub functions obtained to
        get full BDD.
    2.7 Remove all redundant nodes in the BDD so obtained.
    2.8 Remove all duplicate nodes in the BDD.
    2.9 Remove all duplicate terminal nodes in the BDD.
3.  Map the BDDs to PTL cell library.
    3.1 Visit every node in BDD in depth first fashion.
    3.2 Let the current node under visit be $n$.
    3.3 If $n$ is not visited already then map $n$ and its immediate
        successor, if any, cover with best suitable Y-cell.
    3.4 Mark all nodes currently being covered as visited.
4.  Removal of redundant buffers.
    4.1 Visit every node in PTL circuit in depth first fashion.
    4.2 Let the current node under visit be $n$.
    4.3 Remove the buffer of $n$, such that $n$ is not in the critical
        path and removal of the buffer does not increase the delay.
5.  Stop.
```

VII Implementation and Experimental Results

The algorithm has been implemented on Sun Ultra Sparc 10 system having 256MB memory with Solaris operating system and C/C++ compiler embedded with STL and GTL of ATT. We have tested our algorithm with a number of ISCAS benchmark circuits. As these benchmark circuits are quite big (having large number of inputs and outputs), it is not possible to construct and manipulate monolithic BDDs for those circuits. A parser is written to store a benchmark circuit in multilevel decomposed form as DAG. First, the graph is transformed by replacing all reconvergent nodes to their corresponding single nodes. After that, partial collapsing is performed to get Boolean nodes of reasonably larger sizes. For each node in the graph so obtained, BDD is constructed with our algorithm *ptlRP*. After the construction of BDD, we have mapped the BDD to PTL cell library. Finally, removal of redundant buffers is carried out. Switching power and delay of the circuit realized are estimated using the estimation models presented in *Sec 4*. In order to realize static CMOS circuits, we have used the Berkeley SIS tools. The netlist is optimized using *script.rugged* command in SIS. Technology mapping is performed using *44-1.genlib* and with the option of minimum area. Switching power and delay of the static CMOS circuits are calculated using the estimation models proposed in [10]. Value of transistor's parameters are extracted using BSIM3V3 model and for 0.18μ process technology as a particular instance. In our experiment, for all nMOS pass transistors, we have assumed the effective channel length as 0.18μm and channel widths as 0.54μm and 1.08μm for nMOS and pMOS transistors, respectively and the thickness of gate oxide is 40 Å. Further, we have assumed 1.0V as the supply voltage (V_{DD}), 0.2V as the threshold voltage, clock frequency as 100MHz, input transition probability for each primary input as 0.3.

The approach proposed in this paper has been tested on ISCAS benchmark circuits. Experimental results based on the realization of static CMOS circuits using SIS tool, and PTL circuits with our approach are shown in Table 1. In Column 2, 3 and 4 of Table 1, the results of static CMOS circuits, area, delay and energy (power delay product) are shown. For the measurement of area, we have considered the number of transistors as the metric. Area, delay and energy of PTL circuits realized using RP heuristic are presented in Columns 5, 6 and 8 of Table 1, respectively. Percentage reduction in delay and energy of PTL circuits using RP-heuristic compared to their static CMOS counterparts is shown in Column 7 and 9 of Table 1, respectively. Delay is expressed in *ns* (nanosecond) and energy is expressed in *fJ*. From Table 1, it is evident that compared to static CMOS realization, the PTL realization using RP-based heuristic is found to be better in all respects. We find that with respect to static CMOS circuits, percentage reduction in area, delay and energy, on the average, are 32%, 47% and 57%, respectively for PTL circuits based on RP-heuristic.

Unfortunately, previous researches on PTL circuits have reported their experimental results using different performance metric. To compare results on equal footing, we have translated our results to the corresponding parameters reported in those works and compared our results. Table 2 compares our result with that of [2], where percentage reductions in area and delay with respect to static CMOS circuits have been reported without specifying exact values. In Columns 2 and 3 of Table 2, percentage reduction in area and delay in PTL based circuits with respect to static CMOS circuits as reported in [2] are presented. Percentage reduction for the same according to RP-based PTL circuits is furnished in Columns 4 and 5 of Table 2.

In [2], [3] and [4] results were reported in terms of the number of nodes in synthesized circuits. Table 3 compares our result with the results reported in [2], [3] and [4]. The number of nodes according to [2], delay-oriented synthesis presented in [3], area-delay oriented synthesis presented in [4] and based on RP-based heuristic of the present work are presented in Columns 2, 3, 4 and 5 of the Table 3, respectively. It is observed that the experimental results based on our approach are superior with respect to the existing reported results.

Conclusions

In conclusion, we may highlight the three main contributions of this paper. First, unlike the existing approaches based on reduced ordered BDD, our approach is based on the RUBDD obtained using a new RP-based heuristic. RUBDD is obtained using the new heuristic called ratio parameters. As we have seen in the previous section, the sizes of BDDs are smaller in sizes compared to that of existing works. Secondly, the RUBDD is mapped onto LEAP-like cell library, which has the advantage of a small number of cells (only three) with at most two pass transistors in cascade followed by an inverter. This makes the technology-mapping step simpler and gives the approach a flavor of standard cell-based synthesis. Unlike the previous approach [1], we remove buffers between two cells in the non-critical path without affecting the overall performance and thus achieving circuits with lower transistor counts and lower power dissipations. Finally, the LEAP-like cells have been modeled to estimate the delay and power dissipation of the synthesized circuit. As we have reported in the previous section, our results are superior in all respects compared to the existing reported results.

Table 1 Area, Delay and Energy for Static CMOS vs. PTL circuits with our approach

1	2	3	4	5	6	7	8	9
Bench marks	Static CMOS circuits			PTL circuits with RP-based heuristic				
	Area	Delay (ns)	Energy (fJ)	Area	Delay (ns)	% Redn.	Energy (fJ)	%Redn.
C432	692	3.32	407	546	2.08	37	189	53
C499	1880	2.23	820	1428	1.62	27	271	67
C880	1412	2.21	649	988	1.18	46	315	51
C1355	1880	2.61	1045	1203	1.04	60	394	62
C1908	1756	2.91	1068	1088	1.57	46	468	56
C2670	1804	2.94	1452	1010	1.54	48	692	52
C3540	4214	4.57	1868	2782	2.58	43	759	59
C5315	7058	3.65	3033	5364	1.62	55	1261	58
C6288	11222	11.84	4846	6060	4.69	60	2086	57
C7552	8214	2.99	4808	5682	1.66	44	2204	54
	40132			*26151*		*-47%*		*- 57%*

Table 2 Comparison of our results w.r.t. Buch et al. [2]

1	2	3	4	5
Bench marks	Buch et al. [2]		RP-based heuristic	
	%area reduction	%delay reduction	%area reduction	%delay reduction
C432	9	-28	21	37
C499	15	25	24	27
C880	9	44	30	46
C1355	14	57	36	60
C1908	35	30	38	46
C2670	17	39	44	48
C3540	19	36	34	43
C5315	6	54	24	55
C6288	24	42	46	60
C7552	-9	33	29	44
	-14%	***-33%***	***- 32%***	***- 47%***

Table 3 Comparison of sizes of the circuits

1	2	3	4	5
Benchmarks	Buch [2]	Liu [3]	Chau [4]	Using RP
C432	216	265	210	208
C499	235	251	250	234
C880	428	388	359	351
C1355	505	310	269	254
C1908	635	373	366	359
C2670	1077	784	377	752
C3540	1335	1046	1020	734
C5325	2298	1401	1358	1286
C6288	3282	1926	1740	1710
C7552	2931	1718	1719	1714
	12942	***8462***	***7668***	***7602***

Acknowledgement
This work was supported by Intel Corporation, USA, grant #7310. We are thankful to James W. Tschanz of Intel Corporation for his valuable comments and suggestions.

References

[1] Kazuo Yano, Y. Sasaki, K. Rikino, and K. Seki, *Top Down Pass Transistor Logic Design*, IEEE JSSC, Vol. 31, No. 6, pp 792-803, June 1996.

[2] P. Buch, A. Narayana, A. R. Newton, and A. S. Vincentelli, *Logic Synthesis for Large Pass Transistor Circuits*, IEEE/ACM Proc. ICCAD, pp 663-670, Nov. 1997.

[3] Tai-Hung Liu, Adnan Aziz, and J. L. Burns, *Performance Oriented Synthesis for Pass Transistor Logic*, Proc. IWLS, pp. 334-338, June 1998.

[4] R. Chaudhury, Tai-Hung Liu, Adnan Aziz, and J. L. Burns, *Area Oriented Synthesis for Pass Transistor Logic*, IEEE/ACM Proc. ICCAD, pp. 592-596, Nov. 1998.

[5] V. Bertacco, S. Minato, et al., *Decision Diagrams and Pass Transistor Logic Synthesis*, IEEE/ACM Proc. of ICCAD, pp. 256-263, Nov. 1995.

[6] Ajit Pal, *An Algorithm for Optimal Logic Design Using Multiplexers*, IEEE Trans. on Computers, Vol. C-35, No. 8, pp. 755-757, Aug. 1986.

[7] A. Pal, R. K. Gorai, and V. V. S. Raju, *Synthesis of Multiplexers Network using Ratio Parameters and Mapping onto FPGAs*, Proc. 8th Int'l Conf. on VLSI Design, pp. 63-68, Jan. 1995.

[8] A. Bellaur, and M. I. Elmasri, *Low-Power Digital VLSI Design: Circuits and Systems*, Chap. 4, Kluwer Academic Publishers, 1998.

[9] C. E. Shannon, *The Synthesis of Two Dimensional Switching Functions*, Bell System's Tech. Journal, Vol. 28, 1949.

[10] D. Samanta, and A. Pal, *Optimal Dual-V_T Assignment for Low-Voltage Energy Constrained CMOS Circuits*, IEEE/ACM Proc. Joint ASP-DAC/VLSI 2002 Conference, Jan. 2002.

[11] R. E. Bryant, *Graph Based Algorithm for Boolean Function Manipulation*, IEEE Trans. on Computers, Vol. C-35, pp. 677-685, 1986.

[12] Sung-Mo Kang and Yusuf Lebelebigi, *CMOS Digital Integrated Circuits: Analysis and Design (2nd Ed.)*, McGraw Hill, 1999.

[13] A. Ghosh, et al., *Estimation of Average Switching Activity in Combinational and Sequential Circuits*, in IEEE/ACM Proc. of 29th Design Automation Conference, pp. 253-259, July 1992.