Buffer Minimization in Pass Transistor Logic

Hai Zhou
Advanced Technology Group
Synopsys, Inc.
Mountain View, CA 94043

Adnan Aziz
Department of ECE
University of Texas
Austin, TX 78712

ABSTRACT

With the shrinking feature sizes and increasing transistor counts on chips, the push for higher speed and lower power makes it necessary to look for alternative design styles which offer better performance characteristics than static CMOS. Among them, pass transistor logic (PTL) circuits give great promise.

Since the delay in a pass-transistor chain is quadratically proportional to the number of stages, and a signal may degenerate when passing through a transistor, buffers are necessary to guarantee performance and restore signal strength in PTL circuits. In this paper, we first analyze the effects of buffer insertion on a circuit and give the sufficient and necessary condition for safe buffer insertion. Then the buffer minimization problem is formulated, which asks for a minimum number of buffers to make sure that no path has length longer than a given upper bound. Although NP-hard in general, we show that, when buffers are required on multiple fan-outs, it can be solved linearly. We also consider the case when buffers are inverters, where phase assignment need to be done with buffer insertion. Experiments are done on MCNC logic synthesis and optimization benchmarks; compared with a level-by-level insertion, a large number of buffers are saved.

1. INTRODUCTION

At present, technical limits of widely used existing circuit families, such as static CMOS, threaten to hold back the development of microprocessors that can operate at speeds significantly above 500 MHz. For example, at such high speeds, it is predicted that the power dissipation of high-end integrated circuits fabricated using traditional static CMOS circuits will exceed the practical limits of ceramic packages. This makes it necessary to look for alternative design styles which can offer better performance characteristics than static CMOS. Among them, pass transistor logic (PTL) circuits give great promise.

Traditionally, hand-crafted pass transistor logic has been successfully used to implement digital systems which are smaller, faster, and more energy efficient than static CMOS implementations for the same designs [5; 6; 9; 11; 14]. But the lack of a systematic methodology restricts the use of pass transistors in industry circuits. Recently, there have been a number of attempts developing synthesis tools targeting PTL. A top-down design flow for pass-transistor logic, called LEAP (Lean Integration with Pass Transistors), was proposed in [13]. There, designs are first converted into monolithic BDD representations and then mapped to cells of a pass-transistor library consisting of three function cells and four inverters with various drive capabilities. The approach in [2] also utilizes the natural, efficient mapping between BDD and PTL. But, to control the BDD size, it uses decomposed BDDs instead of monolithic BDDs. Furthermore, in stead of mapping to pass-transistor cells, it converts BDD directly into pass-transistor net-list.

Since the delay in a pass-transistor chain is quadratically proportional to its length, and a signal may degenerate when pass through a transistor (e.g., 1 through n-MOS or 0 through p-MOS), buffers are necessary to guarantee performance and restore signals. In [13], buffers are inserted in a straightforward level-by-level fashion. This definitely guarantees the performance of the designs, but may incur a large number of buffers than necessary. In [2], the height of each decomposed BDD is upper-bounded, and a buffer is inserted at the root of each decomposed BDD. As we can see, since the height bound is only 3 or 4, it over-restricts the flexibility of BDD synthesis, and does not guarantee a minimum number of buffers, either.

In this paper, we first demonstrate that buffer insertion on some positions can introduce sneak paths in a PTL circuit. We analyze how a sneak path is introduced and give the condition a node must satisfy to have a safe buffer insertion. We show that checking this condition is NP-complete. It has already been established that PTL circuits derived from BDDs are sneak free; this is one of the primary benefits of using BDD based approaches to PTL synthesis. Our results show that buffering can introduce sneak paths; we also prove that, in PTL circuits derived from BDDs, the safe buffer insertion positions can be easily computed. This reinforces the use of BDD based PTL synthesis.

Given a PTL circuit and a set of candidate buffer insertion positions, we need to consider the following buffer minimization problem. That is, insert a minimum number of buffers among these candidate positions such that the maximum number of consecutive transistors is not greater than a given bound. We show that this problem is in general NP-hard. But in reality, buffers are usually required on multiple fan-out nodes for drive strength. In this case, the problem can be solved linearly.

Usually, buffers are implemented by inverters as they are superior in area, timing, and power. Inverter insertion will change a signal to its complement. Furthermore, in the original circuit, a complemented signal may be needed when passed from one stage to another stage. Therefore, besides separating long paths into short paths, correct phases must also be decide for the whole circuit. We define this as the inverting buffer minimization problem. We deal with this problem in two phases: a non-inverting buffer insertion and a global inverter minimization.

The rest of the paper is organized as follows. In Section 2, the effect of buffer insertion on a circuit is analyzed and conditions for safe buffer insertion are given. The buffer minimization problem when using non-inverting buffers is solved in Section 3 and the corresponding inverting buffer minimization problem is dealt in Section 4. Experimental results on MCNC logic synthesis and optimization benchmarks are given in Section 5. Section 6 concludes the paper.

2. CONDITIONS OF SAFE BUFFER INSERTION

A major difference between pass-transistor logic and other logic styles (such as static CMOS logic, dynamic CMOS logic) is that, in other logic styles, at most one kind of value (0 or 1) can appear in a transit path (i.e. a sequence of drain/source connected transistors); but in PTL, a path may transmit either a 0 or 1 at a different time. This may introduce a connection between 1 and 0 in PTL circuits. As illustrated in Figure 1, when $A = X = Y = 1$ and $B = 0$, there will be a steady current flow from A to B. We call the steady connection between 0 and 1 a *sneak path*. A sneak path in a circuit wastes power and the resulting heat may even lead to catastrophic failure. Therefore, sneak paths should be avoided.

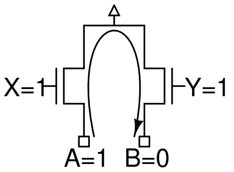

Figure 1: A steady path from 1 to 0

Definition 1. *A PTL circuit without sneak paths is called a* safe *circuit.*

A sneak path only happens when transit paths are wired together. Conventionally, this is called a *wired*-OR. We say the output is in a *floating* state when control signal is disabled in a pass-transistor. Representing a floating state by a Z and a sneak path by an X, the truth table of a wired-OR can be written as follows.

	0	1	Z
0	0	X	0
1	X	1	1
Z	0	1	Z

Table 1: Truth table of wired-OR

Given a safe PTL circuit, buffers are inserted in transit paths to enhance pass signals. But can we guarantee a safe circuit after any insertion? The answer is no.

The reason is that, in reality, buffers are not "ideal". That is, besides transferring a weak 1 or 0 to a strong 1 or 0, it can not transfer a Z to a Z. Instead, it transfers a Z to either 0 or 1. Therefore, a buffer may transform a safe wired-OR of 0 (or 1) with Z to a unsafe wired-OR of 0 with 1. Figure 2 shows a safe circuit, which, when a buffer is inserted at node N, will become unsafe.

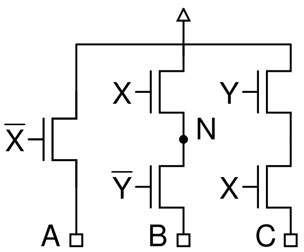

Figure 2: A buffer inserted at N makes circuit unsafe

In order to characterize the positions where buffer insertions are safe, we need the following definition.

Definition 2. *A safe PTL circuit is* redundant *if bridging a transistor (that is, connecting its source and drain) or floating a node (that is, cutting off all paths to that node) gives a safe circuit implementing the same function.*

To illustrate this concept, a redundant circuit and the circuit after bridging a transistor and floating a node are shown in Figure 3.

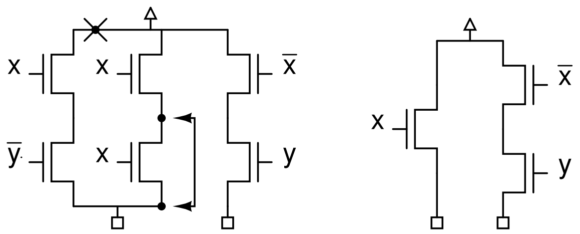

Figure 3: A redundant circuit and its equivalent circuit

Based on the definition, we have the following lemma.

Lemma 1. *Given an irredundant and safe circuit, buffer* *insertion before wired-*OR *or between two consecutive pass-transistors is unsafe.*

PROOF. Suppose there is a buffer inserted before a wire-OR but the circuit is still safe. This means that the input of the buffer can not be floating, or when it floating, other branches are also floating. Since the original circuit is safe, this also means the signal after the wired-OR is totally decided by the signal at the input of the buffer. Therefore, the original circuit can be simplified by floating all other branches, which is a contradiction.

Similarly, suppose there is a buffer inserted between two consecutive pass-transistor but the circuit is still safe. When the pass-transistor before the node is off, there can not be any path from the node to any other non-floating node. Therefore, bridging the pass-transistor before the node will not change any signal at any non-floating node. This contradicts the irredundancy of the original circuit. □

Equivalently, the lemma says that, in an irredundant and safe circuit, only places after wired-OR could be safe buffering places. However, it is not true that inserting buffers after wired-OR always results in a safe design. If under some inputs a node after a wired-OR is floating but there is a path from it to a non-floating node, inserting a buffer on the node is also not safe. Therefore, we have the following theorem.

THEOREM 1. *In an irredundant and safe circuit, the safe buffer insertion positions are the nodes after wired-*OR *such that when it is floating there is no path from it to any non-floating node.*

Unfortunately, the above condition is not easy to check:

THEOREM 2. *Given an irredundant and safe circuit, deciding whether a node is not safe for buffer insertion is NP-complete.*

PROOF. Given an irredundant and safe circuit, by guessing an assignment on control variables, the floating of a node and the existence of a path to a non-floating node under this assignment can be checked in polynomial time. This means the problem is in NP.

We show the problem is NP-hard by transforming 3-SAT problem [3] to it. Given an instance of 3-SAT with n clauses, each has 3 literals, we can construct a circuit illustrated in Figure 4 as follows. For each clause, we have from 1 to node A a path with 3 pass-transistors. The control signals of these pass-transistors are the complements of literals in the clause. That is, if a clause is composed of $\bar{x}_2, x_4$ and $\bar{x}_7$, then the control signals of the path will be $x_2, \bar{x}_4, x_7$. Under this construction, it is easy to see that a path is off if and only if the corresponding clause is satisfied. Therefore, the instance of 3-SAT is satisfiable if and only if A could be floating, or in other word, inserting a buffer on A is not safe. □

However, for PTL circuits synthesized from BDDs, this is not a problem.

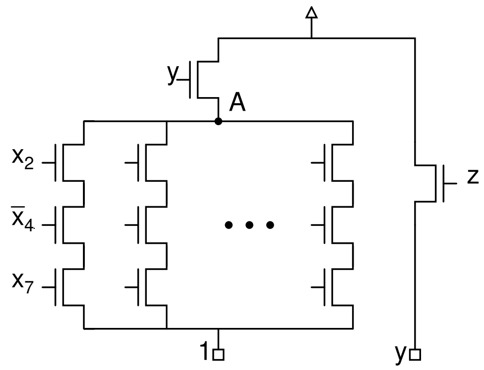

Figure 4: Transforming 3-SAT to a circuit

LEMMA 2. *In a PTL circuit synthesized from BDDs, a buffer can be safely inserted after any wired-*OR.

PROOF. Since each node in a BDD represents an everywhere defined Boolean function, each wire-OR in the circuit is always evaluated to a fixed voltage. Based on Theorem 1, a buffer can be safely inserted on this position. □

3. BUFFER MINIMIZATION

3.1 General buffer minimization

When circuit level issues are considered, a pass-transistor has both capacitance and resistance. Hence, the delay in a transistor chain is quadratically dependent on the number of transistors [8]. It is known for today's static CMOS that transistor chains longer than 3 or 4 in series can be unacceptably slow [10]. Therefore, the main purpose of buffer insertion in a PTL circuit is to make sure that the maximum number of consecutive pass-transistors is not greater than a given upper bound. We hope to use as few buffers as possible to achieve this objective. Therefore, we have the following problem.

PROBLEM 1 (BUFFER MINIMIZATION). *Given a PTL circuit, the set of candidate buffer insertion positions, and a constant K, insert a minimum number of buffers such that the maximum number of consecutive transistors is not greater than K.*

Given a buffer minimization problem, we can model it by an edge-weighted DAG (Directed Acyclic Graph) $G = (V, E)$ as follows. The vertices V represent the candidate buffer insertion positions, primary inputs, and primary outputs. If there is a path of transistors from u to v without going through other vertex, we set $(u, v) \in E$ and use the maximum length of all these paths as its weight. For a problem whose circuit and candidate positions are shown in Figure 5(a), its corresponding DAG is shown in Figure 5(b).

If there is an edge in the DAG whose weight is greater than K, it means there is a path in the original circuit whose length is greater than K but no buffer can be inserted on the immediate nodes. Therefore, there is no feasible solution to the buffer minimization problem. Since this can be

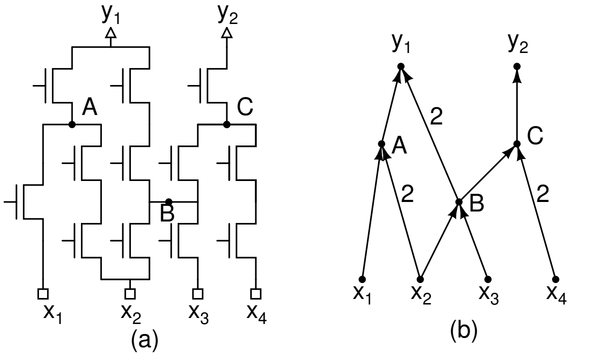

Figure 5: Modeling buffer minimization problem by a DAG

simply checked, we assume it will not be the case in the sequel. Using the DAG presentation, the decision version of the buffer minimization problem can be formulated as follows.

Problem 2 (Buffer Minimization–Decision). *Given an edge-weighted DAG $G = (V, E)$ and two constants K and B, is there $V' \subseteq V$ such that $|V'| \le B$ and after splitting each $v \in V'$, that is, dividing v into two nodes v_1, v_2 and connecting in-coming edges to v_1 and out-going edges to v_2, there is no path whose length is greater than K?*

Based on this formulation, we can show the following result.

Theorem 3. *The Buffer Minimization Problem is NP-hard.*

Proof. This can be shown by a reduction from the INDUCED SUBGRAPH WITH PROPERTY Π [3]. It is defined as follows.

> INSTANCE: Graph $G = (V, E)$, positive integer $M \le |V|$.
> QUESTION: Is there a subset $V' \subseteq V$ with $|V'| \ge M$ such that the subgraph induced by V' has property Π?

The problem is NP-hard for any property Π that holds for arbitrarily large graphs, does not hold for all graphs, and is "hereditary," i.e., holds for all induced subgraphs of G whenever it holds for G. It is still NP-hard when G is restricted to a DAG.

For a proof of Theorem 3, we define Π to be the following property: the longest path length is not greater than a given constant N. It can be easily checked that the property fulfills the above conditions.

Given such an instance with a DAG $G = (V, E)$, constant M, and our property Π, an instance of the buffer minimization problem can be constructed as follows. Copy G and add two vertices s and t. Add edges from s to all vertices with 0 in-degrees in G and edges from all vertices with 0 out-degrees in G to t. Call this graph G'. Set each edge length to be 1, and let $B = |V| - M, K = N + 2$. We will show that the instance of INDUCED SUBGRAPH WITH PROPERTY Π has a yes answer if and only if the instance of the buffer minimization problem has a yes answer.

Suppose the answer to the instance of buffer minimization is yes, that is, there is a subset $V' \subseteq V \cup \{s, t\}$ with $|V'| \le B = |V| - M$ such that, after splitting vertices in V', the longest path length is not greater than $K = N + 2$. Denote the graph after splitting by G''. The paths in G'' start and end with only vertices in $V' \cup \{s, t\}$. Therefore, deleting these vertices makes the longest path length not greater than N. But deleting them also gives a subgraph of G induced on $V - V'$. Since $|V - V'| = |V| - |V'| \ge M$, this means the answer to the INDUCED SUBGRAPH WITH PROPERTY Π is yes. The other direction can be similarly proved. $\square$

3.2 Drive capability consideration

Besides the quadratic dependence of delay on the transistor chain length, a signal may degenerate when passing through transistors. Its strength then may not be strong enough to drive multiple fan-outs. Therefore, in reality, it is usually required to put a buffer on a multiple fan-out node. If the fan-out number is large, this buffer can then be sized accordingly.

As shown in Figure 6, the requirement of buffers on multiple fan-out nodes will split a DAG into separate *leaf-DAG*s. Here, a *leaf-DAG* is a DAG where only the vertices with 0 in-degree can have multiple out-going edges. In this situation, the buffer minimization problem becomes the following problem.

Problem 3 (Buffer Minimization for leaf-DAG). *Given an edge-weighted leaf-DAG and a constant integer K, insert a minimum number of buffers such that there is no path whose length is greater than K.*

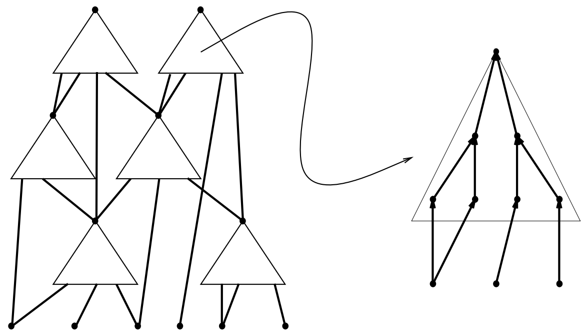

Figure 6: Buffers on multiple fan-out nodes separate circuits into leaf-DAGs

Contrary to the buffer minimization problem for general DAG, the above problem can be solved by the algorithm described in Figure 7. It is a greedy algorithm and runs in linear time. The correctness is given by the following theorem.

```
Algorithm  Greedy Labeling
   for each leaf v
       l(v) = 0;
   for each internal node v bottom-up {
       l(v) = max_{(u,v)∈E}(l(u) + w(u,v));
       if (l(v) + w(v, parent(v)) > K) {
           insert a buffer on v;
           l(v) = 0;
       }
   }
```

Figure 7: The greedy labeling algorithm

THEOREM 4. *The Greedy Labeling algorithm gives optimal solution for the the buffer minimization problem for leaf-DAG.*

PROOF. Since no buffer will be inserted on the leaves, buffer insertion on a leaf-DAG is equal to buffer insertion on the tree when leaves are duplicated. In any path whose length is greater than K, there must be a buffer. Since the number of buffers needed on a graph is no smaller than the number needed on its subgraph, inserting the buffer on the highest level will leave a subgraph comparing with inserting on other nodes, therefore, gives optimal solution. □

4. INVERTING BUFFERS

In reality, buffers are usually implemented by inverters as they have better area, timing, and power characteristics. However, insertion of inverting buffers is more difficult, as they change signals to their complements. Furthermore, in the original circuit, there may already be inverters on transit paths [2]. It will be convenient to refer to, and to represent inverters as "bubbles." Due to the nature of pass-gates, we can get a complemented signal with the same function cell by complementing all the pass signals of the cell, as shown in Figure 8. This gives us the freedom to redistribute inverters around the circuit.

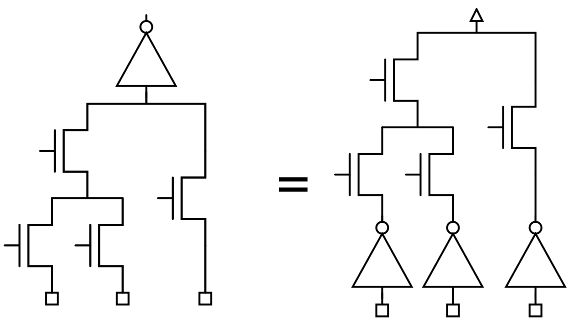

Figure 8: Inverter can be moved freely in PTL circuit

We assume that either a signal or its complement can be chosen to pass at a primary input, and that a primary output can be implemented with arbitrary polarity. This is reasonable because we must have two polarities for control signals, and some of primary outputs are used as control signals.

It follows from these assumptions that the inverting buffer assignment problem can be defined as follows.

PROBLEM 4. **(Inverter Minimization with Phase Assignment)** *Given a PTL circuit, a set of candidate buffer insertion positions, and a constant K, insert a minimum number of inverters in the circuit and decide the phase assignments at primary inputs, such that, each primary output implements the original function or its complement, and the maximum number of consecutive transistors is not greater than K.*

We deal with this problem in two phases. First, ignore all existed bubbles and solve the buffer minimization problem on the circuit. At a position where a buffer is needed, two consecutive bubbles are inserted. One of them is fixed on the position and is called a *hard* bubble; the other together with the existed ones can be moved around and are called *soft* bubbles. Even number of consecutive soft bubbles will cancel. A bubble at a primary input or primary output can be deleted by changing the polarity of the input or output. Therefore, the following problem needs to be solved in the second phase.

PROBLEM 5. *Given a PTL circuit with bubbles and a set of candidate buffer insertion positions, insert a minimum number of inverters only among candidate positions and decide phase assignments at primary inputs, such that, each primary output implements the original function or its complement.*

Generally speaking, the first phase can not be solved efficiently. But if buffers are required on multiple fan-out nodes, it can be solved linearly.

Consider the second phase. Similarly, it can be modeled by a DAG. The vertices represent the candidate buffer insertion positions and the primary inputs and outputs. If there is a path from u to v without going through any other vertex, then we have an edge (u, v). If all such paths have odd (even) numbers of bubbles, then there is 1 (0) bubble on the edge. If some have odd numbers but some have even numbers, then we have two edges between u and v, one with no bubble and one with one bubble.

Treating each internal vertex as a Boolean function and assuming it allows simultaneous polarity change on fan-ins and fan-outs, the above problem is actually the *global phase assignment* problem in the literature. It was shown to be NP-complete by a transformation from MAX-CUT [12]. Heuristic algorithms were given in [4; 12].

5. EXPERIMENTAL RESULTS

We implement the two phase based algorithm described in previous section for the inverting buffer minimization problem. It is required that each multiple fan-out node needs at least one buffer. Experiments are done on MCNC logic synthesis and optimization benchmarks.

Based on discussions in section 2, BDDs are used for PTL synthesis. We use a simple threshold based BDD decomposition heuristic [1]; for simplicity, we do not optimize their representations. Since our algorithm is not based on any specific structure, it can be used on PTL circuits synthesized from other BDD methods, or even non-BDD methods if candidate insertion positions are given.

Based on work of I-Min Liu [7], we choose three as a upper bound on the number of stages in PTL circuits. For comparison, an algorithm which inserts buffers level by level is also implemented. That is, in topological order, a buffer is inserted wherever a node fans out to more than two nodes or there are already three or more serial stages. Wherever a node needs to provide two phases, an addition inverter is introduced. For our algorithm and the level-by-level approach, the numbers of needed buffers and inverters are reported in Table 2. Considering all reported circuits, our algorithm uses 21% fewer buffers and 34% fewer inverters.

Table 2: Experimental results

	Our algorithm		Level-by-level	
circuit	#buffers	#inverters	#buffers	#inverters
C432	767	1158	930	1760
C2670	335	422	442	740
C3540	1836	2499	2300	3913
C5315	2093	2921	2832	4478
C6288	7464	10108	9485	14349
C7552	2776	4003	3393	5262
s1	452	484	553	986
s27	2	6	5	11
s208	26	42	29	60
s298	45	45	54	86
s344	159	160	176	329
s349	159	160	176	329
s382	53	53	60	95
s1196	452	530	565	923
s1238	452	530	565	923
s1423	2459	2466	2737	4769
s1488	212	220	240	397
s1494	212	220	240	397
Total	19502	26027	24782	39807

6. SUMMARY

In this paper, we give the condition for safe buffer insertion in PTL circuits and deal with the problem of inserting minimum number of buffers such that the number of consecutive transistors is upper bounded. We show that this problem is NP-hard in general and can be solved in linear time if a buffer is always needed at a multiple fan-out node. Although we ignored the true timing on electrical basis and used a simple-minded pass-transistor path length for delay control, the problem defines a fundamental issue in buffering PTL circuits and was also considered in [2; 13]. Built on the greedy algorithm for leaf-DAGs, our approach for inverting buffer minimization is an effective and efficient heuristic algorithm. But, on non-critical paths, a tight bound on the number of consecutive transistors may be relaxed, and a buffer driving multiple fan-outs may need to be sized accordingly. Therefore, future work may include non-critical path area recovering and buffer sizing.

7. REFERENCES

[1] R.K. Brayton et al., VIS: A system for Verification and Synthesis. *Computer Aided Verification,* 1996.

[2] P. Buch, A. Narayan, A.R. Newton, and A. Sangiovanni-Vincentelli, Logic Synthesis for Large Pass Transistor Circuits. *IEEE/ACM International Conference on Computer Aided Design,* 1997.

[3] M.S. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, New York, 1979.

[4] A. Jain and R.E. Bryant, Inverter Minimization in Multi-Level Logic Networks. *IEEE/ACM International Conference on Computer Aided Design,* 1993.

[5] T. Kuroda and T. Sakurai, Overview of Low-Power ULSI Circuit Techniques. *IEICE Trans. Electron.,* E78-C(4):334-343, April 1995.

[6] F.S. Lai and W. Hwang, Differential Cascade Voltage Switch with Pass-Gate (DCVSPG) Logic Tree for High Performance CMOS Digital Systems. *International Symposium on VLSI Technology, Systems, and Applications,* 1993.

[7] I-M. Liu, Personal communication. 1997.

[8] I-M. Liu, T.-H. Liu, H. Zhou, and A. Aziz, Simultaneous PTL Buffer Insertion and Sizing for Minimizing Elmore Delay. *International Workshop on Logic Synthesis,*, 1998.

[9] A. Parameswar, H. Hara, and T. Sakurai, A High Speed, Low Power, Swing Restored Pass-Transistor Logic Based Multiply and Accumulate Circuit for Multimedia Applications. *proc. Custom Integrated Circuits Conf.,* May 1994.

[10] J. Rabaey, *Digital Integrated Circuits,* Prentice-Hall, Inc. 1996.

[11] T.S. Sheungm and K. Asada, Regenerative Pass-Transistor Logic: A Circuit Technique for High Speed Digital Design. *IEICE Trans. Electron.,* E79-C(9):1274-1283, September 1996.

[12] A. Wang, Algorithms for Multilevel Logic Optimization. *PhD thesis, U. C. Berkeley,* April, 1989.

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

[14] K. Yano, T. Yamnaka, T. Nishida, M. Satio, K. Shimohigashi, and A. Shimizu, A 3.8-ns CMOS 16x16-b Multiplier Using Complementary Pass-Transistor Logic. *IEEE Journal of Solid-State Circuits,* 25(2):388-395, April 1990.