Transistor Level Synthesis for Static CMOS Combinational Circuits *

Chia-Pin R. Liu Jacob A. Abraham
Computer Engineering Research Center
The University of Texas at Austin
Austin, TX 78712
{cpliu,jaa}@cerc.utexas.edu

Abstract

This paper introduces a novel framework to synthesize static CMOS circuits at the transistor level. A new class of binary decision diagrams (BDDs) which represent inverting Boolean functions, called Transistor Mapped BDDs (TM-BDDs), is used in the synthesis process. There is a one-to-one correspondence between a transistor netlist and its TM-BDD. Nodes in a TM-BDD represent gate inputs and the edges represent the transistors in the netlist. TM-BDDs can be optimized using BDD operations, and the data structure can retain device aspect ratios and geometries for performance optimization. The synthesis process involves a transformation from logic functions to transistor netlists using TM-BDDs. We show how a transistor netlist can be automatically generated during a depth-first traversal on a TM-BDD. The synthesis process is not only independent of any library, but also capable of generating a cell library for a particular circuit. Experimental results demonstrating the reduction of transistor counts are presented.

1 Introduction

CMOS technology has firmly established a dominant role in today's electronics industry. Automated tools help designers to manipulate more transistors on a chip and shorten the design cycle. In particular, logic synthesis tools have contributed significantly to reducing cycle times. Synthesis can be seen as a set of transformations and optimizations from high-level specifications to low-level implementations. Each transformation between two levels involves restructuring of technology dependent information.

All VLSI designs are ultimately implemented at the transistor level. Technology mapping is a task of transforming an optimized logic network into an interconnection of functional blocks consisting of transistor netlists. In full-custom designs, manual generation of transistor netlists for each functional block is done, but this is an extremely time-consuming and difficult task. In the standard-cell methodology, cell library binding utilizes a specific pre-characterized cell library which may take advantage of optimized cell layout and accurate characterization of each cell. However, with rapid advances in process technologies and circuit modeling techniques, cell libraries may not meet performance goals when migrated to new process technologies. Generating and maintaining a cell library is also a time-consuming process [1]. It is critical, therefore, to create new libraries in a timely fashion and to provide methodologies for evaluating as well as re-targeting designs to new process technologies.

Ideally, technology mapping should not be limited by any particular cell library. Cell libraries should have some degree of flexibility to meet requirements or features of an individual design. Furthermore, cell libraries should be generated based on different classes of designs. Since transistor-level descriptions of a design, such as SPICE netlists, are a refinement of the logic design, synthesis techniques at this level will allow much better optimizations for area, performance and power than techniques at the logic level.

Problem Definition : Given a multi-level logic network at the gate level, **transistor-level synthesis** is the process for automatic generation of transistor netlists, without any predefined cell library. The device aspect ratio (W/L) of each transistor and the device geometry should be preserved for performance analysis, and transistor-level functional cells used in the circuit should be identified for possible further optimization.

Wu [6] and Zhu [8] propose methods optimizing switching networks by utilizing shared transistors or equivalent switching networks. The method can easily reach a local optimum, but fails to deal with large circuits. Jha [4] presents a procedure that can generate transistor netlists for static CMOS XOR (exclusive-OR) gates from BDDs. The

*This research was supported by the Texas Technology Development and Transfer Program under Project 003658-433 at the University of Texas at Austin.

0-7695-0104-4/99 $10.00 © 1999 IEEE

XOR gate is the only gate that can be derived from BDDs in their work. Recent research has shown that PTL (Pass Transistor Logic) can be a promising alternative to static CMOS logic for deep sub-micron design [7]. Buch et al. [2] proposes a comprehensive synthesis flow based on decomposed BDDs for PTL designs. Gavrilov et al. [3] introduce library-less synthesis using SP-BDD for delay/power model and gate-BDD for transistor-level resynthesis. Their approach operates on synthesized circuits and performs a transistor level optimization by transistor restructuring.

This paper introduces a novel approach to perform synthesis for static CMOS combinational circuits at the transistor level. The synthesis process involves a transformation from a logic network into static CMOS transistor netlists using *TM-BDDs (Transistor Mapped Binary Decision Diagrams)* which maintain the device and technology-dependent information for performance modeling and optimization. Our methodology can be applied to full-custom and standard-cell design styles. Our strategy is to use a TM-BDD as an interface between a functional cell and a transistor netlist, since constructing one monolithic BDD for the whole circuit could result in an exponential blowup of the BDD. Because the TM-BDD structure can preserve the device aspect ratios (W/L) and geometries, it can be used to extract performance characteristics such as area, delay or power. The synthesis process is not only independent of any library, but also capable of generating a cell library for a particular circuit.

The remainder of this paper is organized as follows. Section 2 provides an overview of the transformations between logic networks and transistor netlists using TM-BDD. Section 3 describes the synthesis procedure for static CMOS circuits. Section 4 presents the experimental results for ISCAS benchmark circuits. Section 5 concludes the paper.

2 TM-BDD

A static CMOS circuit is a combination of two networks, called pull-up network (PUN) and pull-down network (PDN), as shown in Fig. 1(a). The basic design principle is that PUN and PDN are dual networks. A CMOS transistor netlist can be converted to a graph which is a combination of two subgraphs corresponding to its PUN and PDN (Fig. 1(b)).

The *transistor graph (TG)* is a directed acyclic graph (DAG) correspondent to its transistor netlist. It consists of two subgraphs which are P-subgraph and N-subgraph. The nodes are source/drain nodes in the netlist, including three designate nodes, VDD, GND, and OUT. The edges are transistors that connect between nodes. Accordingly, there are two types of edges known as P-edges and N-edges. The On (Off)-path of G is defined as a sequence of P (N)-edges $P^{on(off)} = \langle OUT, x_1, x_2, \ldots, x_i, VDD(GND) \rangle$, such that x_i is a P (N)-edge. It is easily understood that a TG is a direct mapping of all paths in a netlist. A TG and its original netlist are one-to-one correspondent. To collect all the conduction paths in the TG, we can extract a logic function to characterize its functional behavior.

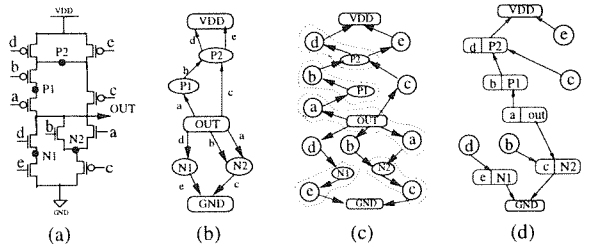


Figure 1. Graph transformation

TG is a concise representation for a netlist. Unfortunately, it is too general to perform complex operations or optimizations. Since a TG is a DAG, there exists two partial orderings among input variables in both P-subgraph and N-subgraph. Intrinsicly, every parallel connection of transistors in PUN corresponds to a serial connection in PDN, and vice versa.

Compiling two partial orderings together among input variables, a total ordering may be obtained. This ordering is the key information to construct the TM-BDD. Based on this ordering, each edge with a variable in a TG will be converted into a decision node in a TM-BDD. Each edge in a TG represents a transistor between two nodes, but those source/drain nodes in a TG are no longer available in a TM-BDD. Nodes in a TM-BDD represent the gate inputs, and the edges, the transistors. A left edge (0-edge) of a node in a TM-BDD corresponds to a P-transistor, and the right, an N-transistor. The ordering information should maintain the correct connections between different decision nodes in a TM-BDD.

DEFINITION The *transistor mapped binary decision diagram* (TM-BDD) $G = (V, E)$ is a BDD which is correspondent to its transistor graph. The decision node is the gate input of a transistor. 0 (1)-edge represents the P (N)-transistor associated with the node. 0 (1)-leaf corresponds to GND (VDD). Only 0 (1)-edge can connect to 1 (0)-leaf. There does not exist a node whose 0-child and 1-child are decision nodes with the same gate input at the same level. The level of a node is defined by letting the root be at level one, and the node be at level *(i+1)* if the maximal level of its parent nodes is *i*.

EXAMPLE. Considering a netlist in Fig. 1(a), the TG is illustrated in Fig. 1(b). The partial ordering in PUN is $[((a \rightarrow b) \parallel c) \rightarrow (d \parallel e)]$, and in PDN, $[((a \parallel b) \rightarrow c) \parallel (d \rightarrow e)]$. A total ordering is obtained as $[a \rightarrow b \rightarrow c \rightarrow d \rightarrow e]$. To illustrate the derivation of an TM-BDD, observe the extended transistor graph (Fig. 1(c)). In Fig. 1(c), every original edge in the TG has been changed into a gate input node

with the incoming and outgoing edges. The drain nodes in Fig. 1(c) should combine with the immediate successor of the gate input node based on the variable ordering. After the nodes merge, a new graphical representation of a TG is formed in Fig. 1(d). Indeed, the upper half of Fig. 1(d) can be transformed into half of a binary tree (Fig. 2(a)), and the lower half into the other (Fig. 2(b)). Combining both of the half binary trees, the unique TM-BDD for the netlist is obtained (Fig. 2(c)). The solid (right) edges in TM-BDD are 1-edges, and dashed (left) edges, 0-edges.

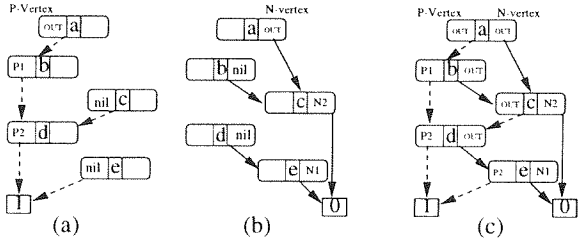
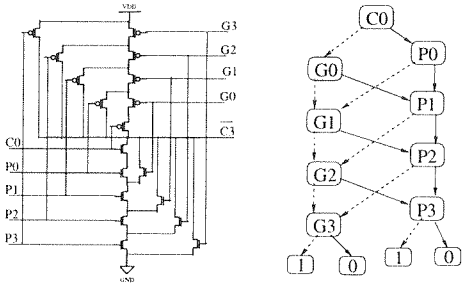


Figure 2. Geometry preserving transformation

Nodes in a TM-BDD represent gate inputs in a netlist, and the edges represent the transistors. A left edge (0-edge) of a node in a TM-BDD represents a P-transistor, and the right (1-edge), an N-transistor. The TM-BDD also preserves the device geometry of a netlist, because each edge corresponds to a unique transistor. The device aspect ratios (W/L) of transistors can also be preserved in the TM-BDD. There is a one-to-one correspondence between a transistor netlist and a TM-BDD, and the transformation between them is a *geometry preserving transformation.*

EXAMPLE. The logic function of the carry generator for a 4-bit lookahead adder is $C_3 = G_3 + P_3(G_2 + P_2(G_1 + P_1(G_0 + P_0 * C_0)))$. The transistor netlist for $\overline{C3}$ is given in Fig. 3(a) [5]. After performing the geometry preserving transformation, the TM-BDD for $\overline{C3}$ can be obtained in Fig. 3(b).



(a) static CMOS netlist for $\overline{C3}$ (b) TM-BDD for $\overline{C3}$

Figure 3. The netlist and TM-BDD

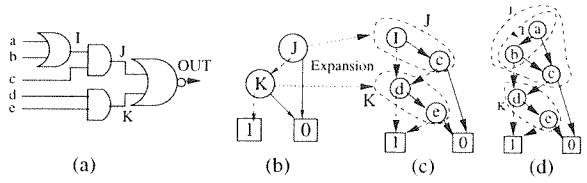


Figure 4. The construction of TM-BDD

3 Synthesis for Static CMOS Circuits

The approach to construct a TM-BDD is to create the individual BDD for each logic gate, and then compose those BDDs, level by level, to form a TM-BDD. In Fig. 4, node expansion is the basic way to construct a TM-BDD. The effect of a node expansion is not only to increase the size of a TM-BDD, but also to affect the critical path of a TM-BDD.

The construction of a TM-BDD can be controlled by the parametric constraint of the lengths of critical paths. In general, the composition of two gates of identical type results in the summation of the lengths of TM-BDDs. It usually increases the length by one while composing two gates of different types. After the TM-BDD construction and optimization phases, the netlist generation procedure proceeds to transform TM-BDDs into transistor netlists.

Unlike an OBDD, a TM-BDD is not canonical. However, since the TM-BDD has some of the properties of BDDs, they are easy to manipulate when performing algebraic factorizations of logic functions. The form and size of a TM-BDD depend on the variable ordering in a netlist. Because of the isomorphism between a netlist and its TM-BDD, the reordering of transistors in a netlist will be reflected in the nodes in a TM-BDD, and vice versa.

Minimizing the number of transistors in a netlist is equivalent to optimizing a TM-BDD by minimizing the number of nodes in it. Reduction of nodes in a TM-BDD can easily be done by applying BDD operations. Generating a transistor netlist from a TM-BDD is an inverse transformation of the TM-BDD. The automatic generation of a netlist from a TM-BDD can be done by performing a depth-first traversal on a TM-BDD.

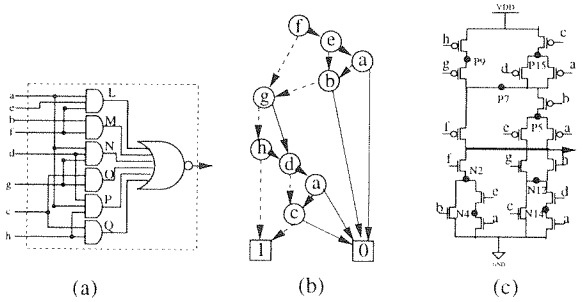


Figure 5. Synthesis using TM-BDD

EXAMPLE. Figure 5 shows a functional cell which is an 8-input AOI complex gate, its optimized TM-BDD and the correspondent transistor netlist. The automatic generation of its PUN netlist can be obtained by performing a depth-first search on 0-edges of the TM-BDD. The DFS sequences on the TM-BDD and the corresponding generations of P-transistors are described as follows : (1). $[f \rightarrow g \rightarrow h \rightarrow 1]$ corresponds to $[OUT \rightarrow f \rightarrow P7 \rightarrow g \rightarrow P9 \rightarrow h \rightarrow VDD]$; (2). $[d \rightarrow c \rightarrow 1]$ corresponds to $[P7 \rightarrow d \rightarrow P15 \rightarrow c \rightarrow VDD]$; (3). $[a \rightarrow c]$ corresponds to $[P7 \rightarrow a \rightarrow P15]$; (4). $[e \rightarrow b \rightarrow g]$ corresponds to $[OUT \rightarrow e \rightarrow P5 \rightarrow b \rightarrow P7]$; (5). $[a \rightarrow b]$ corresponds to $[OUT \rightarrow a \rightarrow P5]$. A similar scenario occurs for the generation of N-transistors in the PDN.

In either full-custom or standard-cell style, generating and maintaining a cell library is a time-consuming process. To speed up the generation of a cell library is very critical for designers as well as the design. Two TM-BDDs represent the same functional cell if they have the same topology. Generating a cell library using TM-BDDs for a particular circuit is easily done by putting all TM-BDDs into equivalence classes. We use a binary tree equivalence checking algorithm on the TM-BDDs to generate a cell library for a particular circuit. The number of equivalence classes of TM-BDDs represents the maximum number of functional cells occurring in a circuit.

4 Experimental Results

We have implemented the algorithms described in this paper in C under Linux, and have applied them to the ISCAS benchmark circuits. The input format of logic networks is in the BENCH format. We first remove redundancies in all benchmark circuits using SIS, then map the logic circuit to TM-BDDs. These are then analyzed to produce the transistor netlists which are output in SPICE format. Table 1 presents the experimental results on a 266MHz Pentium-II processor with 64MB of memory. The second column shows the number of gates after redundancy removal, the third and fourth columns show the number of transistors with conventional technology mapping and using our TM-BDDs, respectively, the fifth column indicates the reduction in transistor count, the sixth column shows the number of different library cells needed, and the last column shows the time for transistor-level synthesis.

5 Conclusions and Future work

Our approach achieves a significant reduction in transistor counts, especially on the larger circuits. On the average, we see a 63% reduction in transistor counts. Specifically, we achieve a 50% reduction on ALU-like circuits, and about 70% reduction on the others. Since the times for synthesis to the transistors are quite small, we plan to add more heuristics and more computation-intensive optimizations to further reduce transistor counts.

Circuit	Gates	Transistors (Tech. Map)	Transistors (TM-BDD)	Reduction	Cell Lib.	Time (sec.)
C17	6	24	22	0.92	4	0.001
C432	222	1100	766	0.70	14	0.02
C499	514	2564	1752	0.68	8	0.13
C880	316	1618	1254	0.77	22	0.12
C1355	482	2084	1624	0.78	6	0.13
C1908	536	2782	2054	0.74	33	0.18
C2670	718	3740	2544	0.68	20	0.38
C3540	1086	5922	4180	0.71	30	0.57
C5315	1733	9828	6300	0.64	50	1.71
C6288	2384	10078	8240	0.82	7	2.51
C7552	2521	13114	8286	0.63	52	3.02
alu2	224	1972	1086	0.55	23	0.02
alu4	456	3660	1992	0.54	29	0.09
dalu	2111	10332	5494	0.53	22	1.99
frg2	1642	10052	4626	0.46	46	1.29
Total	14951	78870	50220	0.63		

Table 1. Experimental results

References

[1] D. Baltus, T. Varga, R. Armstrong, J. Duh, and T. Matheson. Developing a concurrent methodology for standard-cell library generation. In *Proc. of the Design Automation Conf.*, June 1997.

[2] P. Buch, A. Narayan, A. R. Newton, and A. Sangiovanni-Vincentelli. Logic synthesis for large pass transistor circuits. In *Proc. of the ACM/IEEE International Conference on Computer-Aided Design*, November 1997.

[3] S. Gavrilov, A. Glebov, S. Pullela, S. Moore, A. Dharchoudhury, R. Panda, G. Vijayan, and D. Blaauw. Library-less synthesis for static cmos combinational logic circuits. In *Proc. of the ACM/IEEE International Conference on Computer-Aided Design*, November 1997.

[4] N. K. Jha and Q. Tong. Robustly testable static cmos parity tree derived from binary decision diagrams. *IEEE Journal of Solid-State Circuits*, 26(11), November 1991.

[5] J. M. Rabaey. *Digital Interegrated Circuits*. Prentice-Hall, Inc, Upper Saddle River, New Jersey 07458, 1996.

[6] M. Wu and I. N. Hajj. Switching network logic approach to sequenntial mos circuit design. *IEEE Transactions on Computer-Aided Design*, 8(7), July 1989.

[7] K. Yano, Y. Sasaki, and K. Seki. Top-down pass-transistor logic design. *IEEE Journal of Solid-State Circuits*, 31(6), June 1996.

[8] J. Zhu and M. Abd-El-Barr. On the optimization of mos circuits. *IEEE Transactions on Circuits and Systems-I:Fundamental Theory and Applications*, 40(6), June 1993.