SYNTHESIS FOR MIXED CMOS / PTL LOGIC

Congguang Yang Maciej Ciesielski
Department of Electrical & Computer Engineering
University of Massachusetts, Amherst, MA 01003.

High noise immunity and level-restoring capabilities of static CMOS gates, combined with small area and low power of PTL cells, make a mixed CMOS / PTL design style an ideal alternative to the all-CMOS technology. However, the synthesis of mixed CMOS / PTL circuits imposes a great challenge to the existing synthesis methodology. Neither traditional techniques based on algebraic factorization nor methods based on direct BDD mapping [1] [2] [3] are applicable to this new circuit style.

We have recently proposed a new BDD-based logic optimization method for static CMOS [4]. It is based on iterative BDD decomposition using various *dominators* which correspond to decomposable BDD structures leading to AND, OR, XOR and MUX decompositions. Synthesis results show that the method is very efficient for both AND / OR- and XOR-intensive functions. Since PTL structures can be easily identified on a BDD, our method can be readily extended to perform logic decomposition leading to mixed CMOS / PTL logic implementation. In contrast to other PTL synthesis techniques, based on direct BDD mapping, our method is not limited to decomposition onto PTLs only; its logic decomposition and optimization is driven by the capabilities of both the static CMOS and PTL logic. Our BDD decomposition method can also account for various parameters associated with circuit performance, thus avoiding drawbacks of direct BDD mapping-based synthesis, such as large fanouts and long transistor chains.

The bulk of our BDD decomposition theory has been published in [4]. Table I summarizes the different types of BDD decompositions available; it can be seen that all types of atomic decompositions and their corresponding BDD structures can be easily identified.

Type	BDD Structure	Decomposition
1	*1-dominator*	algebraic AND
2	*0-dominator*	algebraic OR
3	*x-dominator*	algebraic XOR / XNOR
4	*generalized dominator*	Boolean AND / OR
5	*generalized x-dominator*	Boolean XOR / XNOR
6	*cofactor wrt. single node*	simple MUX
7	*cofactor wrt. super node*	*functional* MUX

TABLE I
BDD DECOMPOSITIONS IN [4]

Note that the direct BDD mapping can be seen as a process of performing simple MUX decomposition with respect to each BDD node. As such, it corresponds to decomposition type *6* in Table I. It should also be noted that the *remapping* technique in [5] is a subset of type *1* and *2*. Therefore, our BDD decomposition method is more general than all previously reported methods.

I. IMPLEMENTATION AND RESULTS

A new tool, *BDDlopt.ptl*, has been developed by adding the MUX decomposition capability to *BDDlopt* [4]. A BDD is iteratively decomposed by finding the best decomposition at each iteration. The structural information of a BDD is obtained by a single fast scan on the BDD. All simple dominators (*0-*, *1-* and *x-dominators*), *functional MUX* and *XOR* decompositions are found during the scan. If a simple dominator or a functional MUX decomposition is found, the decomposition is taken immediately. Otherwise, the BDD will be decomposed by either a *generalized-dominator* or a *generalized x-dominator*. The choice between these two is based on the information collected by the BDD scan. A set of factoring trees is built along with the BDD decompositions. Once the BDDs of all the output functions have been decomposed, BDDs are rebuilt on the resulted factoring trees to find all possible logic sharing. Fig. 1 shows an example of a BDD-based synthesis process.

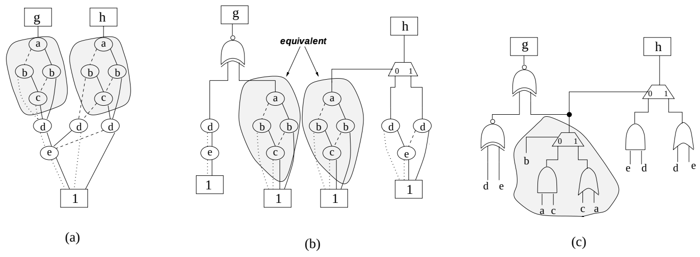

Fig. 1. Example of a BDD-based synthesis process

The experiments have been conducted on UltraSPARC-5 / 320M. The test cases are categorized into two groups: 1) AND / OR-intensive, and 2) XOR-intensive functions. The results of *BDDlopt* have been compared with results obtained by SIS-1.2 running *script.rugged*.

The netlists synthesized by *BDDlopt.ptl* are mapped onto a library, *mcnc_ptl.lib*, which incorporates three new PTL cells. SIS mapper is used for technology mapping. Buffers are inserted only if two PTLs are chained together. Compared with *BDDlopt* [4], the mixed CMOS / PTL circuits generated by *BDDlopt.ptl* offers more area reduction with no CPU time overhead. For XOR-intensive circuits, the average area of mixed CMOS / PTL circuits is only 54% of that of SIS and the delay is only 53% of SIS. For AND / OR-intensive circuits, due to the dominance of AND / OR logic in this group, the improvement of *BDDlopt.ptl* over SIS is marginal. While the area of mixed CMOS / PTL circuits synthesized by *BDDlopt.ptl* is about 2% larger than that of SIS, delay is only 77% of SIS.

Finally, *BDDlopt.ptl* offers a great advantage over SIS in terms of CPU time. It is $3 - 6$ times faster than SIS in both groups.

REFERENCES

[1] K. Yano, Y. Sasaki, K. Rikino, and K. Seki, "Top-Down Pass Transistor Logic Design," *IEEE J. Solid-State Circuits*, vol. 31, no. 6, pp. 792–803, June 1996.
[2] P. Buch, A. Narayan, R. Newton, and A. Sangiovanni-Vincentelli, "On Synthesizing Pass Transistor Logic," in *Intl. Workshop on Logic Synthesis*, 1997.
[3] V Bertacco, S. Minato, P. Verpaetse, L. Benini, and G. De Micheli, "Decision diagrams and pass transistor logic synthesis," in *Intl. Workshop on Logic Synthesis*, 1997.
[4] C. Yang, V. Singhal, and M. Ciesielski, "BDD Decomposition for Efficient Logic Synthesis," in *International Conference on Computer Design*, 1999, pp. 626–631.
[5] S. Yamashita, K. Yano, Y. Sasaki, Y. Akita, H Chikata, K. Rikino, and K Seki, "Pass-Transistor / CMOS Collaborated Logic: The Best of Both Worlds," in *Symposium on VLSI Circuits Digest of Technical Papers*, 1997, pp. 31–32.

This work has been supported by a grant from NSF under contract No. MIP-9613864.