MULTI-TERMINAL BDD SYNTHESIS AND APPLICATIONS

Petr Mikušek

Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
email: imikusek@fit.vutbr.cz

1. INTRODUCTION

Design of digital systems is based on various specifications of Boolean functions, most often in a form of Boolean expressions or in PLA format. Recently also binary decision diagrams (BDDs) became popular means of design, verification and testing. Conversion of a single Boolean function into a BDD was studied intensively [4] including related optimization problems. As the ordering influences the size and shape of the diagram, we should find among all possible orderings of variables the one that produces a optimal diagram.

Another machine representation of single-output Boolean functions uses binary decision diagrams (BDDs), which can have many forms, [4]. Ordered BDDs (OBDDs) use the same order of variables along all paths; for a given variable order there exists a unique reduced OBDD with a minimum number of decision nodes.

Generalization of BDDs to multiple-output Boolean functions are so called word-level BDDs, among them e.g. Multiterminal BDDs (MTBDDs), BDDs for characteristic function BDD_for_CF [1]. Optimum synthesis of these diagrams, basically optimum ordering of variables with respect to a certain goal, is being solved with the aid of heuristic approaches. If the variable ordering is given, then the diagram may be obtained by decomposing the function repeatedly.

In this paper we present a heuristic technique of the iterative decomposition of incompletely specified multiple-output Boolean functions. Its main contribution is that the bottom-up synthesis of MTBDD does not require knowledge of optimum ordering of variables, because the order of variables is generated concurrently. Obtained MTBDDs can be used in hardware (LUT cascades), firmware (branching microprograms) and software implementation of combinational and sequential functions.

This research has been carried out under the financial support of the research grants "Design and hardware implementation of a patent-invention machine" GA102/07/0850, "Safety and security of networked embedded system applications" GA102/08/1429, "Mathematical and Engineering Approaches to Developing Reliable and Secure Concurrent and Distributed Computer Systems" GA102/09/H042, and "Security-Oriented Research in Information Technology" MSM 0021630528.

2. MTBDD CONSTRUCTION BASED ON THE DISJUNCTIVE ITERATIVE DECOMPOSITION

In this section we will present a heuristic technique of a sub-optimal MTBDD construction. It is generalization of the approach taken in [2]. Let us recall that under a decomposition of function $F(x_1, x_2, \ldots, x_n) = F(X)$ we understand a serial disjunctive separation of F into two functions G and H such that $F(X) = H(U, G(V))$ where U, V are disjunctive subsets of X, $U \cap V = \emptyset$, $U \cup V = X$, and $|U| + \log_2 |G| < |X|$, $|V| < |X|$, [3]. We will refer to G and H as to residual and detached functions, respectively.

Decomposition can be applied iteratively to a sequence of residual functions with a decreasing number of variables. We will select always a single input variable ($|U| = 1$), from now on denoted as a detached variable, that will be removed from a residual function in such a way that the width or cost of the diagram be minimized. More general techniques like non-disjunctive decomposition or multi-variable decomposition ($|U| > 1$) can be explored in future.

A single variable will be removed from the function in one decomposition step. Starting with variable x_1, we first create a list of all compatible cube pairs of remaining input variables, their products and ordered pairs of function values

$$[F(0, x_2, x_3, x_4), F(1, x_2, x_3, x_4)] \qquad (1)$$

produced by them, for example on four variable function. We have to include also pairs (1) generated by single cubes with x_1 uncertain, that generate pairs of identical function values. This approach based on compatible cube pairs differs from the one creating sets of compatible cubes covering all minterms [3]; it was taken because it seems to be computationally less demanding.

The next phase of the decomposition step is merging pairs of output cubes in the previous list to as few compatible classes as possible. The reason for reducing the number of compatible classes to the minimum is a direct impact on the number of outputs of the residual function $G(x_2, x_3, x_4)$.

The problem of finding maximal compatibility classes of output cube pairs and then the selection of a set of maximal

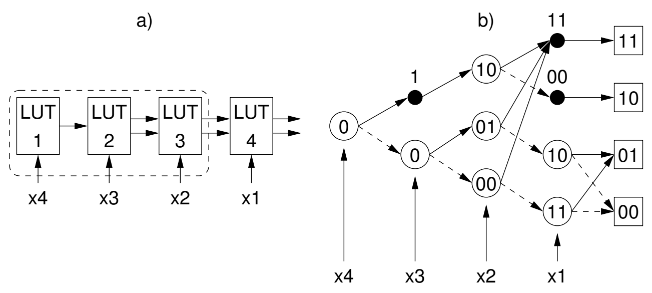

Fig. 1. Result of iterative decomposition: a) MTBDD, b) a generic LUT cascade

classes, with minimum cardinality, so that all input cubes are covered, can be solved by several methods [3].

By now, we have obtained a sequence of detached functions that can be implemented by a generic cascade of small lookup tables (LUTs). By combining LUTs in the generic cascade, shorter cascades with larger LUTs can be obtained, Fig. 1a. Construction of the MTBDD is equally easy. Each LUT is converted to a layer of the MTBDD, Fig. 1b.

3. HEURISTIC ITERATIVE DECOMPOSITION

The remaining question not addressed as yet is, which variable should be used in any given step. We use a heuristics that strives to optimize one level of the MTBDD at a time. There are more sophisticated heuristics which move the window of several variables from the root to leaves and optimize the position of several variables at a time [4]. However, if desired, our approach can be extended to groups of variables as well.

There are three parameters of MTBDDs that can be optimized: size (cost), width and an average path length APL. For firmware implementation of MTBDDs, only cost optimization is of interest. In our heuristics we minimize the number of regular nodes level by level, from leaves to the root. We expect that the total cost will be close to the minimum total cost. That is why we talk about suboptimal MTBDD synthesis.

LUT cascades require slightly modified optimization criterion. The main concern is LUT size and therefore the cascade width. At each step a variable is selected, that generates the minimum number of LUT rows. In the case of a tie the lowest cost criterion is applied: a variable producing the lowest number of rows with distinct function values in the pair is taken. In the case of a tie again, a variable is selected randomly.

To aid MTBDD synthesis, the program HIDET (Heuristic Iterative Decomposition Tool) has been developed [2]. In the meantime it accepts only a restricted class of multiple-output Boolean functions, namely integer functions of Boolean variables specified by cubes that are pairwise uncompatible. Surprisingly, quite a few functions can be specified this way, arbiter and allocators functions among others [2].

The above mentioned restriction simplifies the process of creating compatible cube pairs and their merging. It is not necessary to explore all cube pairs, but only those with detached variable value 0 and 1 and separately cubes with uncertain detached variable. On the other hand, the restriction disables application of the heuristics on a standard benchmark set, because specification of most of the benchmark circuits contains compatible cubes. The next version of HIDET under construction is therefore addressing the more general case, too.

4. CONCLUSION

The presented method of MTBDD/LUT cascade synthesis of multiple-output Boolean functions aided by HIDET tool proved to be suitable for synthesis of a restricted class of combinational and sequential designs with up to around 20 input and state variables. Arbiters, as well as other digital systems frequently used in practice, have relatively low complexity, what makes their cost-effective cascade implementations possible.

Firmware implementation of a MTBDD is usually a matter of trade-off between performance and the size of memory storing the microcode. The memory size can be derived as an aggregate size of all dispatch tables, and the performance is given by the number of dispatch tables in series from the root to leaves of the MTBDD.

Future research should address a tool accepting a more general specification of multiple-output Boolean functions in a form PLA matrix (fr functions). Also the quality of heuristic optimization of variable ordering should be put under a test against results obtained by exhaustive testing of all permutations of variables in problems of reasonable size. Comparison with other design techniques will be undertaken as soon as HIDET-2 is up and running. Security and safety oriented applications will be the nearest target.

5. REFERENCES

[1] S. N. Yanushkevich, D. M. Miller, V. P. Shmerko, and R. S. Stankovic, *Decision Diagram Techniques for Micro- and Nanoelectronic Design*. CRC Press, 2006.

[2] T. Sasao and M. Matsuura, “BDD representation for incompletely specified multiple-output logic functions and its applications to functional decomposition,” in *Design Automation Conference*, June 2005, pp. 373–378.

[3] P. Mikušek and V. Dvořák, “On lookup table cascade-based realizations of arbiters,” in *11th EUROMICRO Conference on Digital System Design*, Sept. 2008, pp. 795–802.

[4] J. A. Brzozowski and T. Łuba, “Decomposition of boolean functions specified by cubes,” University of Waterloo, Tech. Rep., 1997.