Report on the Dagstuhl-Seminar
Computer Aided Design and Test
Decision Diagrams – Concepts and Applications
(27. - 31.1.1997)

Introduction

The fourth workshop **Decision Diagrams – Concepts and Applications** in the series *Computer Aided Design and Test* at the IBFI SchloßDagstuhl was organized by Bernd Becker (Univ. Freiburg), Randy Bryant (Carnegie Mellon Univ.), Masahiro Fujita (Fujitsu Labs. San José) and Christoph Meinel (Univ. Trier). It was attended by 39 scientists.

Binary Decision Diagrams (BDDs) have found widespread use in computer-aided design for digital circuits. They form the heart of many tools for formal verification and are also commonly used in logic synthesis, circuit testing and in the verification of communication protocols.

The organizers took the opportunity to bring together researchers from different areas in computer science, electrical engineering and industry. The common aim of all researchers is to deepen the knowledge of the – widely accepted – BDD data structures, to improve existing techniques and to explore new fields of application.

Twenty-eight lectures were presented at the workshop covering the following theoretical and practical topics:

- Formal verification of combinational and sequential circuits
- Minimization of BDDs
- Lower bounds techniques
- New BDD-concepts
- Improvements of BDD-algorithms
- Applications using BDD-based data-structures

In addition there was an open discussion on the future directions of BDD research whose aim was to exchange the most recent problems, needs and challenges about industries and universities.

SchloßDagstuhl and its staff provided a very convenient and stimulating enviroment for the workshop. The organizers wish to thank all those who helped in establishing this excellent research experience.

Program

Monday

Amit Narayan
Formal Verification of Combinational Circuits

Stephan Waack
On Parity Ordered Binary Decision Diagrams

Warren A. Hunt, Jr.
The Use of OBDDs in a General-Purpose Theorem Prover

———

Nils Klarlund
Cache Miss Considerations in BDD Algorithms

Rajeev K. Ranjan
Computer Architecture-Based Solutions for Overcoming Complexity of BDD Algorithms

Fabio Somenzi
Remembrance of Things Past

Tuesday

Forrest Brewer
Implicit Support Determination for an Incompletely Specified function

Antoine Rauzy
Handling Boolean Reliability Models by means of BDDs

Thorsten Theobald
Local Re-Encodings and Linear Sifting

Tsutomu Sasao
Ternary Decision Diagrams – Survey

———

Detlef Sieling
Variable Orderings and the Size of OBDDs for Partially Symmetric Boolean Functions

Nicole Göckel
Evolutionary Minimization Approaches for OBDDs

Alan J. Hu
A Crazy Idea for Variable Ordering

Ingo Wegener
Optimal OBDDs for Tree-like Circuits

Wednesday

Beate Bollig
On the Power of Partitioned-OBDDs

Amit Narayan
Partitioned-ROBDDs: A Compact, Canonical and Efficiently Manipulatible Representation for Boolean Functions

Ken McMillan
A Conjunctively Decomposed Representation for Boolean Functions

Vigyan Singhal
A Heuristic Algorithm for Satisfiability of a Conjunction of BDDs

---

Excursion to Saarburg

Thursday

Gianpiero Cabodi
Disjunctive Partitioning and Partial Squaring: An Effective Approach for Reachability Analysis of Large FSMs

Coen A. J. van Eijk
Exploiting Functional Dependencies in Sequential Equivalence Checking

Andreas Kuehlmann
Equivalence Checking Exploiting Structural Design Properties

Michael Martin
Polynomial Formal Verification of Multipliers

Anna Slobodová
A Reducibility Concept for the Problems Defined in Terms of OBDDs

Andreas Hett
Fast and Efficient Construction of BDDs by Reordering Based Synthesis – The MORE-Approach

Jean-Christophe Madre
The Architecture of TiGeR

Discussion
What drives BDD-research?

Friday

Masahiro Fujita
: Verification of Arithmetic Circuits by Comparing Two Similar Circuits

Geert Janssen
: Experiences with BDDs in IBM's BSN Project

Alexander Saldanha
: Using BDDs in Logic Simulation and Low Power Logic Synthesis

---

More detailed information including some full papers can be found on the WWW-pages with the URL:

`http://www.informatik.uni-trier.de/Design_and_Test_97/`

Formal Verification of Combinational Circuits

Amit Narayan
University of California at Berkeley, USA
(with Jawahar Jain, M. Fujita and A. L. Sangiovanni-Vincentelli)

With the increase in the complexity of present day systems, proving the correctness of a design has become a major concern. Simulation based methodologies are generally inadequate to validate the correctness of a design with a reasonable confidence. More and more designers are moving towards formal methods to guarantee the correctness of their designs. This talk surveyed some state-of-the-art techniques used to perform automatic verification of combinational circuits.

The current approaches for combinational verification can be classified into two categories: functional and structural. The functional methods consist of representing a circuit as a canonical decision diagram (DD). Two circuits are equivalent if and only if their DDs are equal (isomorphic). The structured methods consist of identifying related nodes in the circuits and using them to simplify the problem of verification. Both approaches were presented with a discussion of their merits and drawbacks.

On Parity Ordered Binary Decision Diagrams

Stephan Waack
Universität Göttingen

I consider a data structure for Boolean functions, which I call Parity-OBDDs or POBDDs, which combines the nice algorithmic properties of the well-known OBDDs with a considerably larger descriptive power.

Beginning from an algebraic characterization of the POBDD-complexity I sketched in my talk the algorithm which minimizes the number of nodes of a given POBDD. The running time is $O(n \cdot \mathrm{SIZE}(B)^3)$, if Gaussian elimination is used. An equivalence test algorithm can be easily constructed now.

Finally, I gave the basic idea of an algorithm which checks the equivalence of two POBDDS associated with efficient variable orderings.

The Use of OBDDs in a General-Purpose Theorem Prover

Warren A. Hunt, Jr.
Computational Logic Inc, Austin, USA

We have integrated a BDD-like decision procedure into the ACL2 theorem-proving system. This procedure constructs BDD-like structures from general

ACL2 terms by categorizing ACL2 terms as either BDD-constructors, IF terms, or leaves. The IF nodes correspond to conventional OBDD nodes. At the top of an ACL2 BDD may be a collection of BDD-constructors, such as the parity operation cones. The leaves may contain any ACL2 term

The ACl2 logic has been designed so that it is an extension of a useful subset of Common Lisp. The ACL2 logic is a first-order, essentially quantifier-free logic of total recursive functions. The rules of inference include mathematical reduction, and the logic is organized to support several extension principles, including recursive function definition, and support for modular reasoning. The ACL2 system has been primarily implemented by Matt Kaufmann and J. Moore.

An ACL2 term is either a constant, a variable symbol, or a function application whose actual parameters are terms. Formal parameters for functions are untyped, so any kind of parameter can be supplied. ACL2 BDD terms are constructed in a bottom-up fashion, just as with OBDDs, but there are three kinds of nodes: leaves, IF expressions, and BDD constructors. A leaf term is an ACL2 term that contains no embedded IF term or BDD-constructor. There are several interacting procedures used in the construction of BDD terms: unconditional rewriting, IF-lifting, constant subterm evaluation, and basic BDD simplification techniques for ensuring the uniqueness of the BDD representation.

Cache Miss Considerations in BDD Algorithms

Nils Klarlund
AT&T Labs Research, USA
(with Theis Rauhe)

Within the last few years, CPU speed has greatly overtaken memory (RAM) speed. For this reason, implementation of symbolic algorithms – with their extensive use of pointers and hashing – must be reexamined.

In this talk, we introduce the concept of *cache miss complexity* as an analytical tool for evaluating algorithms that depend on pointer chasing. This measure suggests different data structures from those traditionally used.

Our ideas have been implemented in a BDD package, which is used in a decision procedure for the Monadic Second-order logic on finite strings.

Experimental results show that on large examples, our implementation runs tour to five times faster than a widely used BDD implementation.

We believe that the method of cache miss complexity is of general interest to any

implementor of symbolic algorithms.

Computer Architecture-Based Solutions for Overcoming Complexity of BDD Algorithms

Rajeev K. Ranjan
University of California, Berkeley, USA

BDD based algorithms for manipulating Boolean functions are characterized as being highly memory intensive with little locality. For the complex designs, the BDD representation and manipulation of Boolean functions suffer from two limitations: i) memory – BDDs do not fit in the available memory, and ii) computational – for large BDDs even the polynomial complexity of algorithms result in excessive time.

A lot of research has been done to overcome these two limitations. These can be classified under i) application specific (partitioning of the network, partitioned transition relation), ii) algorithmic (variable ordering, re-encoding), and iii) modifying the decision diagram (ADD, BMD, BMD*, etc).

In this work, a survey of techniques has been presented which target to solve this problem from computer architecture perspective.

First, we look at how memory hierarchy has been exploited to reduce the elapsed time for a BDD manipulation. In particular, the techniques for reducing number of page faults have been presented. It was shown that CAL package developed at University of California, Berkeley improves upon the conventional depth-first based package (by David Long) by a factor of 100 for large examples.

Further to improve the computation time, use of parallel computers has been reviewed. We have put the research done in this area in a unified framework to discuss the merits and demerits of various approaches. The parameters involved are the selection of hardware, distribution of data, generation and distribution of workloads, communication paradigm and the management of global data structures (unique table, computed table etc), traversal technique etc. It was observed that a combination of shared memory multiprocessor selection with the breadth-first traversal of the operand BDDs offered the best solution.

Remembrance of Things Past

Fabio Somenzi
University of Colorado at Boulder, USA
(with Srilatha Manne and Dirk Grünwald)

Recent work on breadth-first manipulation of binary decision diagrams has resulted in algorithms with improved memory access locality. The page faults have

thus been greatly reduced. The question naturally arises as to whether these improvements are also reflected in reduced cache misses. We have conducted experiments to answer this question. We have run instrumented versions of the same program using three BDD packages: two are DFS packages (CMU and CUDD) and one is BFS (CAL). We have measured cache and TLB misses from the simulation, as well as from direct execution of the programs. Our results indicate that there are small differences in TLB and cache misses between the BFS and DFS approaches. Implementation here seems to outweigh algorithmic differences. Many differences in speed are easily explained by the different ways in which BFS-based packages and DFS-based packages deal with the computed table. We also show how small implementation details can substantially affect the locality behaviour of a BDD package. As an example we report results obtained by partially sorting the list of free nodes.

Implicit Support Determination for an Incompletely Specified function

Forrest Brewer
University of California at Santa Barbara, USA
(with Andrew Crews)

This talk was presented in two parts. The first reviewed the implicit construction of a BDD containing all valid support sets for an incompletely specified function. Several speed up techniques were presented, from value caching to k-of-n bound imposition. Benchmark results indicate that the run time is bounded by $n^2/2$ – i. e. the algorithm is not worse than conjunction in the worst case. In the second part, this algorithm was applied to the problem of support minimization of each transition bit of a state machine via encoding. The technique used a step-wise refinement approach which specifically allowed for don't care bits in the state codes and non-minimal code-lengths. Techniques were described for isolating the information required from a transition function and for moving such information to other transitions in an attempt to reduce the required support for each bit of the machine. An example was described in which the logic was reduced to 2 (AND) gates and an inverter, using one extra encoding bit, compared to the 37 literal minimum bit solution of Jedi.

Handling Boolean Reliability Models by means of BDDs

Antoine Rauzy
LaBRI-CNRS, Université Bordeaux I, France

In this talk, we first present some Boolean models to assess the reliability of industrial systems: fault trees, reliability diagrams, reliability networks, etc.

We show that the two main problems are the evaluation of the probability of a function on the one hand and the determination of prime implicants on the other hand. Both are computationaly hard. We review theorems that allow to design BDD based algorithms to solve them.

By the way, we introduce a new notion of implicants: minimal p-cuts. Intuitively, minimal p-cuts are minimal positive parts of prime implicants. They are of a particular interest in the reliability analysis framework because they describe what is faulty in a system. We show some experimental evidence that they are far more easy to compute than prime implicants, which is confirmed by a theoretical analysis.

We introduce the notion of truncated computations of prime implicants and minimal p-cuts, and we show that it is interesting for practical purposes.

Local Re-Encodings and Linear Sifting

Thorsten Theobald
Universität Trier
(with Christoph Meinel and Fabio Somenzi)

Ordered binary decision diagrams are the state-of-the-art representation of switching functions. In order to keep the sizes of the OBDDs tractable, heuristics and dynamic reordering algorithms are applied to optimize the underlying variable order. When finite state machines are represented by OBDDs the state encoding can be used as an additional optimization parameter. In this talk we analyze local encoding transformations which can be applied dynamically. In particular, we discuss the use of an XOR-transformation and show why this transformation is most suitable among the set of all transformations.

Going a step further, we present an algorithm called Linear Sifting for the general optimization of decision diagrams that combines the efficiency of Sifting and the power of linear transformations. We show that this algorithm is applicable to large examples and that in many cases it leads to substantially more compact decision diagrams when compared to simple variable reordering.

Finally, we discuss applications of the new technique to synthesis and verification.

Ternary Decision Diagrams – Survey

Tsutomu Sasao
Department of Computer Science and Electronics
Kyushu Institute of Technology, Japan

This talk surveys seven types of TDDs: General_TDD, SOP_TDD, ESOP_TDD, AND_TDD, Prime_TDD, EXOR_TDD, and Kleene_TDDs, and introduces a unifying terminology.

After showing theorems on complexities, we compare the size of these TDDs by using benchmark functions. Finally, outlines of the current problems are shown.

Variable Orderings and the Size of OBDDs for Partially Symmetric Boolean Functions

Detlef Sieling
Universität Dortmund
(with Ingo Wegener)

The problem of computing optimal variable orderings for OBDDs for a special type of functions namely partially symmetric functions is considered. A function is called partially symmetric with respect to a partition $V_1, \ldots, V_k$ of the set of variables iff the function is invariant under permutations of the variables in each set $V_1, \ldots, V_k$. The sets $V_1, \ldots, V_k$ are called symmetry sets. An obvious heuristic for the OBDD variable ordering problem is to arrange the variables of each of the symmetry sets in adjacent levels. But there are examples where this heuristic does not lead to minimum OBDD size. By considering random functions from particular classes of partially symmetric functions it is shown that this heuristic leads to minimum OBDD size with probability close to 1 if the symmetry sets are of equal size. On the other hand with probability close to 1 the heuristic does not lead to minimum size OBDDs if the symmetry sets are of very different size. For the proof of these results some structural properties of OBDDs for partially symmetric functions are used. By these properties it is also possible to reduce the problem of computing optimal variable orderings to the search for shortest paths in a grid graph. This approach leads to an efficient minimization algorithm if the number of symmetry sets is small.

Evolutionary Minimization Approaches for OBDDs

Nicole Göckel
Universität Freiburg
(with Rolf Drechsler and Bernd Becker)

Evolutionary Algorithms are used to find variable orderings that minimize the size of an OBDD for a given Boolean function. A first approach uses "classical" principles of evolutionary algorithms in combination with operators that are based on dynamic reordering of variables. The restriction to small population sizes and the use of sifting and partial inversions improve this approach: it is also applicable to large problem instances. For several benchmark functions (circuits) the previously best known sizes could be improved.

Due to the large execution times of evolutionary algorithms a second different approach is focused on the development of "efficient" and in some sense problem specific heuristics, which can bridge the gap between execution time and quality of

the OBDD sizes. A model of learning heuristic based on evolutionary algorithms is reviewed. It is applied to construct heuristics that improve the variable ordering of OBDDs.

Finally, relations between these approaches are discussed.

A Crazy Idea for Variable Ordering

Alan J. Hu
University of British Columbia, Canada
(work done at Fujitsu Labs of America)

I presented a simple variable ordering heuristic for ROBDDs. The heuristic has serious problems, but I think there are some interesting ideas in it, which are worthy of discussion. Although my motivation was based on high-level model checking, my presentation and experimental results were for combinational circuits.

The idea is very simple. Expert humans outperform variable ordering heuristics, provided the human has insight about the function being represented. Such insight usually takes the form of noticing and grouping together "related" variables. So, the goal is to have a heuristic that looks for related variables.

Let $f(x_1, \ldots, x_n)$ be a Boolean function. For any i, j, let P_{00} be the 1-probability of f given $x_i = 0$ and $x_j = 0$. Define P_{01}, P_{10} and P_{11} similarly. A crude measure of relatedness is $\mid P_{00} + P_{11} - P_{01} - P_{10} \mid$. Computing 1-probabilities is hard, but we can estimate them by random simulation of the circuit. Thus, we get a tuple of pairwise relation estimates. Choosing a variable order to keep related variables together is essentially the graph bandwidth problem, so I resort to a greedy heuristic.

The heuristic works extremely well on C432, quickly producing an order much better than the DFS heuristic (Fujita, Malik et. al.) and almost as good as the very expensive methods (simulated annealing, entropy, etc.). On C1908, however, the results are terrible. Worse, the quality of the orders worsens with increased simulation (and therefore more accurate probability estimates), suggesting fundamental problems with the heuristic.

Optimal OBDDs for Tree-like Circuits

Ingo Wegener
Universität Dortmund
(with Martin Sauerhoff and Ralph Werchner)

The variable ordering problem for OBDDs is known to be hard and important. Many heuristics to obtain an initial ordering and to improve it by reordering have been designed. We look for classes of circuits such that an optimal variable

ordering can be computed efficiently. Such a class is the class of circuits over the full binary basis (including EXOR) which graphtheoretically are a tree. Our algorithm works in linear time (less then 13 CPU seconds for 2^{19} variables). The hardest tree-like circuit leads to an OBDD with complemented edges of size at most n^β where $\beta = \log_4(3+\sqrt{5}) < 1.2$. this bound is tight, since the Reed-Muller-tree needs OBDDs of size $\Omega(n^\beta)$.

Hence, the variable ordering problem is solved for tree like circuits. The algorithm may be used as heuristic to find better initial variable orderings for almost tree-like or arbitrary circuits. Circuit partition may lead to tree-like subcircuits and not tree-like parts can be substituted by pseudovariables.

On the Power of Partitioned-OBDDs

Beate Bollig
Universität Dortmund
(with Ingo Wegener)

Data structures for Boolean functions are used in many applications. Recently a lot of BDD-models have been introduced, i.e., partitioned OBDDs by J. Jain (1992). All models are more or less successsful for some problems. So it is more and more important to know which models are good for a given function, that means which BDD-model has a small size for such a function. Therefore, it may be helpful to know which functions are easy for a BDD-model but difficult for another model. So we are looking for tight lower bounds for BDD-models to get some intuition how to find an adequate BDD-model for a given function. We also compare FBDDs, kIBDDs and kOBDDs with partitioned-OBDDs. Let e.g. P-FBDDs denote the class of sequences of functions representable by polynomial size FBDDs. We prove the following results:

- P-partitioned-$(k-1)$OBDD $\subsetneq$ P-partitioned-kOBDD
- P-partitioned-kOBDD $\subsetneq$ P-kIBDD
- P-FBDD $\not\supseteq$ P-partitioned-kOBDD
- P-kOBDD $\not\supseteq$ P-partitioned-kOBDD

  if $k \leq \frac{(\log n)^{1/2}}{\log\log n}$.

Partitioned-ROBDDs: A Compact, Canonical and Efficiently Manipulatible Representation for Boolean Functions

Amit Narayan
University of California at Berkeley, USA
(with Jawahar Jain, M Fujita, A. L. Sangiovanni-Vincentelli)

In this talk we present partitioned-ROBDDs. Partitioned-ROBDDs divide the Boolean space into k partitions and represent a function over each partition as a separate ROBDD. We show that partitioned-ROBDDs are canonical and can be efficiently manipulated. Further, they can be exponentially more compact then monolithic ROBDDs and even Free BDDs in representing Boolean functions. More over, at any given time, only one partition needs to be manipulated which further increases the space efficiency.

In addition to proving the power of partitioned ROBDDs on special classes of functions, we provide automatic techniques to generate them. We show that for large circuits our techniques are more efficient in space as well as time over monolithic ROBDDs. Using these techniques some complex industrial circuits, with sizes of up to 17k gates, could be verified for the first time.

A Conjunctively Decomposed Representation for Boolean Functions

Ken McMillan
Cadence Berkeley Labs, USA

Presents a canonical representation for Boolean functions based on a successive decomposition using projection and a simplification operator. The representation is shown to exploit functionally determined and "conditionally independet" variables. This is illustrated by means of a model checking example in which BDDs scale exponentially, while the conjunctive form scales quadratically. Algorithms are presented on the conjunctive form which have proved effective in practice, though exponential in the worst case.

A Heuristic Algorithm for Satisfiability of a Conjunction of BDDs

Vigyan Singhal
Cadence Berkeley Labs, USA
(with Adnan Aziz)

We are interested in solving the following problem: Given $n+p$ Boolean variables (primary variables $\vec{x} = x_1, x_2, \ldots, x_m$ and intermediate variables $\vec{y} = y_1, y_2, \ldots, y_p$)

and Boolean functions $f_1(\vec{x},\vec{y}),\ldots,f_n(\vec{x},\vec{y}), g_1(\vec{x},y_2,\ldots y_p),\ g_2(\vec{x},y_3,\ldots y_p)$ and $g_p(\vec{x})$ and constraints $f_1 = f_2 = \ldots = f_n = 1, g_1 = y_1, g_2 = y_2, \ldots g_p = y_p$ return a truth assignment for the Boolean variables (if one exists) that satisfies all the constraints. In our application (test pattern generation during large combinational verification), the functions are represented as reduced ordered BDDs. Thus, we have a satisfiability problem on BDDs (as opposed to clauses). We would like to solve this problem without performing any BDD operations such as bdd_and or bdd_compose.

Our heuristic algorithm is based on clausal SAT heuristics which perform randomised local search. The straightforward extension of this to BDDs fails miserably. We show how we can effectively tailor the cost function as well as the randomisation step to direct the search towards useful regions of the search space. Moreover, we need a fast implementation of the greedy step to make our algorithm effective. Our preliminary results indicate that our randomised local search strategy is usually able to beat our alternative strategy for our problem instances.

Disjunctive Partitioning and Partial Squaring: An Effective Approach for Reachability Analysis of Large FSMs

Gianpiero Cabodi
Politecnico di Torino, Italy
(with Paolo Camurati, Luciano Lavagno and Stefano Quer)

Reachability analysis of FSMs is usually operated by means of breadth-first traversals of state transition graphs, and they often experience the following space/time limitations:

- size of BDDs required to represent state sets
- large internal BDDs for traversal operations
- long chains of state sets.

We propose disjunctive partitioning to deal with large reachable state sets, peak BDDs required by symbolic operations, and complexity of operations. We adopt selective traversals of subsets of the transition relation to bypass the problem of large state sets in itermediate traversal steps, and partial squaring to collapse long chains of states in the state transition graph to much shorter ones.

Exploiting Functional Dependencies in Sequential Equivalence Checking

Coen A. J. van Eijk
Eindhoven University of Technology, The Netherlands

In many practical applications sequential equivalence checking is used to compare

sequential circuits with similar state encodings. This observation can be used to improve the efficiency of BDD-based verification methods. In this talk, we discuss techniques to automatically detect and utilize these similarities. The proposed techniques are based on the concept of functionally dependent state variables. These variables are characterized by the property that their value can always be determined from the values of the other state variables. Therefore, dependent state variables can be removed from the problem representation without loss of information, and this has several positive effects on the performance of a BDD-based verification method: It may result in smaller BDD representations and reduce the number of iterations required to prove equivalence.

Equivalence Checking Exploiting Structural Design Properties

Andreas Kuehlmann
IBM T.J. Watson Research Center, Yorktown Heights, USA

In this talk we present a new verification technique for equivalence checking which is specifically targeted to large combinational circuits with some structural similarities. The approach combines the application of BDDs with circuit graph hashing, automatic insertion of multiple cut frontiers, and a controlled elimination of false negative verification results caused by the cuts. Two ideas fundamentally distinguish the presented technique from previous approaches. First, originating from the cut frontiers, multiple BDDs are computed for the internal nets of the circuit, and second, the BDD propagation is prioritized by size and discontinued once a given limit is exceeded.

Polynomial Formal Verification of Multipliers

Michael Martin
Universität Halle
(with Martin Keim, Bernd Becker, Rolf Drechsler and Paul Molitor)

Until recently, verifying multipliers with formal methods was not feasible, even for small input word sizes. About two years ago, a new data structure called Multiplicative Binary Moment Diagram (*BMD) was introduced [Bryant and Chen, DAC 95] for representing arithmetic functions over Boolean variables. Based on *BMDs verification methods were proposed by which verification of multipliers with input word sizes of up to 256 bits is now feasible. Only experimental data has been provided for these verification methods until now.

We presented a formal proof that logic verification of a class of multipliers, the Wallace-tree like multipliers, with *BMDs is polynomially bounded in both space and time. These bounds are $O(n^2)$ wrt. space and $O(n^4)$ wrt. time and have

been obtained by analyzing a verification method called Backward Construction [Hamaguchi et. al., ICCAD 95] when applied to Wallace-tree like multipliers. Additionally we provide a direct application of our theoretical results in the presence of design errors.

A Reducibility Concept for the Problems Defined in Terms of OBDDs

Anna Slobodová
ITWM Trier
(with Ch. Meinel)

Reducibility concepts are fundamental in computability and complexity theory. Usually, they are defined as follows. A problem Π is reducible to a problem Σ if Π can be computed using a program or device for Σ as a subroutine. This classical approach fails for restricted computational models. In the case of OBDDs, it allows merely to use the almost unmodified original program for the subroutine.

We propose a new reducibility concept for OBDDs. We say that Π is reducible to Σ if an OBDD for Π can be constructed by applying a sequence of elementary operations to an OBDD for Σ. In contrast to the classical approach, this new reduction is able to reflect the real needs of a reducibility concept on the context of OBDD-based complexity classes. It allows to reduce those problems to each other which are computable with the same amount of OBDD-resources, i.e. their OBDD-representations have "similar" size, and it gives a tool to carry over lower und upper bound results.

Fast and Efficient Construction of BDDs by Reordering Based Synthesis – The MORE-Approach

Andreas Hett
Universität Freiburg
(with Rolf Drechsler and Bernd Becker)

In this talk we present a new approach to symbolic simulation with BDDs. Our method uses Reordering Based Synthesis (RBS) by means of specialized level exchange and shift operations on free-floating operation nodes. This concept allows the integration of Dynamic Variable Reordering (DVO) even within a single synthesis operation (e.g. an AND-Operation). Thus huge peak sizes during the construction can often be avoided and we obtain a method that, with no penalty in runtime, is more memory efficient then ITE operator based packages.

The Architecture of TiGeR

Jean-Christophe Madre
Synopsis Inc, Mountain View, USA
(with Olivier Coudert)

The talk presented the work done by the above mentioned persons at Digital

Equipment Corporation's Paris Research Laboratory in 1993 and 1994.

We described the architecture and features of a software libray designed to address a class of problems encountered in the following applications: logic synthesis and verification, Esterel code optimization and verification, fault tree analysis. This library includes a BDD package, a ZBDD and an ADD package, an FSM optimization and verification package. We presented the algorithms developed specially for this library, including original FSM traversal algorithm performing on the fly redundant state variable removal, later refined by C. A. J. van Eijk (see this workshop).

Verification of Arithmetic Circuits by Comparing Two Similar Circuits

Masahiro Fujita
Fujitsu Labs of America, USA

The talk proposes a verification method for arithmetic circuits. It consists of the two steps:

1. Generate two similar circuits by using recurrence equations for arithmetic operation from the given circuit to be verified.

2. Worst case analysis of inputs, apply existing Boolean comparison programs to the problems generated in 1.

Different from BMD based verification, the proposed method can be applied even if the circuits have errors (there is no blow-up of BDDs etc.).

Experiments show its usefulness and more techniques may be developed for other types of logic functions.

Experiences with BDDs in IBM's BSN Project

Geert Janssen
Eindhoven University, The Netherlands

The BSN (Boolean Specification Networks) project dates back to 1990. Its initial aim was to provide a hardware description language and associated toolset for the design of moderately sized sequential circuits. The toolset includes a schematic editor, a simulator, a design browser, various conversion programs from and to the BSN language, and two formal verification tools. All tools share a number of common software libraries, such as a parser and network data manipulation. The combinational verification tool uses BDDs to represent the Boolean functions in

the network. To cope with large designs, a simple scheme to avoid BDD size explosion is adopted: either the user supplies a set of cutpoints or the program tries to determine them. Typically, the type of problem posed consists of a rather high-level specification and a low-level implementation. This makes the finding of so-called one-sided cutpoints a lot easier. The program will automatically try to match up the specification cutpoints with corresponding cutpoints in the implementation. The second verification tool is a CTL model-checker. It is based on the usual implicit image and pre-image computation approach. It allows for non-determinism to be present in the design description and properties may be augmented with fairness constraints.

Using BDDs in Logic Simulation and Low Power Logic Synthesis

Alexander Saldanha
Cadence Berkeley Labs, USA
(with Luciano Lavagno, Patrick McGeer and Ken McMillan)

A novel approach to fast logic simulation based on BDDs is proposed. The method derives a program that evaluates a BDD representation of a set of functions. Optimization is achieved by computing multiple outputs in a BDD, converting a BDD to a MDD to reduce the number of memory lookups, and techniques for decomposition to control BDD size. Analogously, an approach to the design of low power circuits derived from BDD is also presented.