Symbolic Analysis Methods for Masks, Circuits, and Systems

Randal E. Bryant*
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Symbolic representations of systems can achieve a high degree of compaction relative to more explicit forms. By casting an analysis task in terms of operations on a symbolic representation, large and complex systems can be analyzed efficiently. This paper summarizes research in applying symbolic analysis methods to systems at several levels of abstraction.

1 Introduction

For many aspects of system design, the amount of analysis that can be performed prior to construction is limited by the system size and complexity. For example, every 32-bit register increases the number of possible states by a (multiplicative) factor of 4 billion. A modern microprocessor may contain over 5 million transistors, and its mask representation may consist of 50 million rectangles. Even when it is feasible to construct and maintain an representation of such a system, it becomes too cumbersome to analyze in a meaningful way.

By representing a system in a symbolic form, we can effectively decouple the complexity of the representation from that of the system being represented. Furthermore, by developing efficient algorithms that operate on these symbolic representations, it becomes possible to analyze the system comprehensively.

This paper summarizes recent research in symbolic analysis methods for digital systems at three levels of abstraction ranging from the high level system state space to the low level physical design. Data structures such as *Ordered Binary Decision Diagrams* (OBDDs) have provided important tools in these efforts [3, 8]. OBDDs represent Boolean functions as directed acyclic graphs, providing a high level of compaction relative to a tree or tabular representation. Operations on Boolean functions can be implemented by efficient graph algorithms operating on OBDDs. Many mathematical domains can be expressed in terms of Boolean functions, and hence OBDDs provide a powerful technique for many forms of system analysis.

*This research was supported by the Advanced Research Project Agency, ARPA Order 4976, by the National Science Foundation, Grant MIP-8913667, and by the Semiconductor Research Corporation, contract 91-DC-068

2 System State Space Analysis

Many aspects of system behavior can be expressed as properties of finite state machines. By introducing nondeterminism, properties such as concurrency, asynchrony, and "don't care" conditions can be incorporated into the machine models. Conventional methods of analyzing state machines construct explicit models of the state space, such as transition tables or state graphs. Such representations are feasible when analyzing very small systems but become totally impractical for systems with a significant amount of data storage or consisting of multiple, interacting state machines.

Recently, researchers have shown that OBDDs can be used to represent and analyze finite state machines. Typically, the Boolean next state functions for the individual state elements are derived from a gate-level or register-transfer description of the circuit. From these a state transistion relation can be derived. This relation is expressed as a Boolean function over a set of variables representing the input, the old state and the new state. The function yields 1 under those conditions where the given input could cause a transition from the old state to the new state. For systems involving multiple, interacting state machines, a product construction can be used to derive a transition relation for the entire system given relations for the component machines. Based on this relation, iterative methods can be used to derive information such as the reachable state set. Again, this set is expressed as a Boolean function over a set of variables representing the system state yielding 1 when the given state is reachable.

Researchers have shown that symbolic state machine analysis can determine whether two sequential circuits are equivalent, even when they use different state encodings [1], and to determine whether a system satisfies an abstract property expressed as a formula in temporal logic [9]. Machine models with over 10^{20} states have been constructed, far larger than would be possible with an explicit representation. These methods have been used to analyze complex system behaviors, such as cache coherency protocols [11], They have proved effective at detecting subtle design errors that had escaped detection despite extensive simulation.

3 Transistor-Level Circuit Analysis

As described above, symbolic state machine models can be constructed from a gate-level or register-transfer description of a digital circuit. Many circuits, particularly high performance microprocessors, are designed directly at a transistor level. Circuit design techniques such as domino logic and different bus structures cannot be represented accurately in terms of logic gates. For such circuits, there is no straightforward way to derive a state machine representation, or even to compare the circuit to a register-transfer representation.

We have developed methods for deriving a logic representation of a transistor circuit by switch-level circuit analysis [5]. This method uses a variant of a switch-level simulation algorithm, where the circuit behavior is computed symbolically, rather than for a specific set of input patterns. It can capture many subtle aspects of circuit behavior such as bidirectional signal flow, stored dynamic charge, and different signal strengths. This analysis requires solving systems of equations in a Boolean algebra to compute the effects of the paths between nodes formed by conducting transistors. Most significantly, this analysis can use weak symbolic manipulation methods based on networks of Boolean operations [4] or logic gates [7]. Solving the equations by a direct method based on Gaussian elimination avoids the need for a convergence test and hence the need to determine Boolean equivalence.

This symbolic analysis generates a logic representation of a circuit expressed in terms of Boolean operations or conventional logic gates. From this one can extract and analyze an abstract state machine representation [2] or apply symbolic simulation techniques to verify correct circuit operation [6]. Thus, symbolic transistor circuit analysis bridges the gap from a purely structural system representation to one from which high level behavior can be computed.

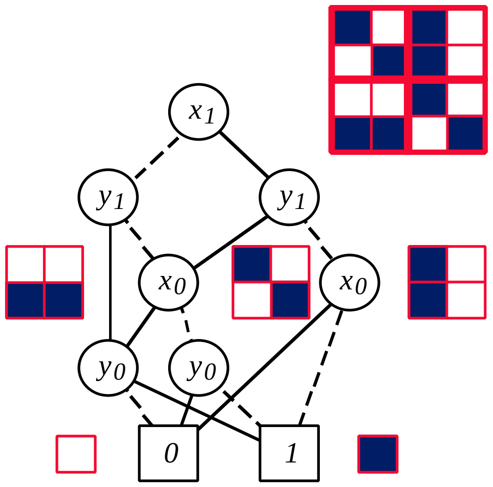

Figure 1: **OBDD Representation of Image**. An image can be viewed as a Boolean function over a set of index variables.

4 Layout-Level Analysis

Most recently, researchers have begun to apply symbolic analysis methods to even lower levels of abstraction. For example, design-rule checking (DRC) remains a major computational task for semiconductor manufacturers. Most DRC programs are based on an explicit representation of the mask geometry, e.g., lists of rectangles or polygons. Modern checkers exploit the hierarchy in the layout representation to avoid multiple checking of repeated cells, but a large amount of low-level geometric processing is still required. Furthermore, certain information, such as electrical connectivity, is difficult to extract from these unstructured representations.

If we consider a mask represented in a rasterized format (as it would be for E-beam lithography), then it can be viewed as a function $I(x, y)$ mapping discrete coordinate values x and y to 0 or 1. If we now represent x and y as n-bit binary numbers, then I becomes a Boolean function over variables $x_{n-1}, \ldots, x_0$ and $y_{n-1}, \ldots, y_0$. Hence, such an image can be represented as an OBDD. Figure 1 shows an example of the OBDD representation of a 4×4 image with $n = 2$. The overall OBDD represents the image shown at the top. If we divide this image into four quadrants, we see that there are three unique 2×2 subimages, each represented by a subgraph of the OBDD as shown in the figure. Each of these subimages can in turn be decomposed into quadrants, yielding 1×1 images represented by leaf values 0 and 1. This example illustrates key properties of OBDDs that make them attractive for representing mask geometry. For a variable ordering $x_{n-1}, y_{n-1}, \ldots x_0, y_0$, each pair of levels in the OBDD effectively divides the image into four quadrants, as one would find in a quadtree representation. Unlike a quadtree, however, there is only one OBDD node for each unique subimage—in essence the tree becomes a maximally reduced directed acyclic graph. Thus any repeated structure in the layout due to a hierarchical design is automatically exploited to reduce the size of the representation.

We have experimented with generating OBDD representations for a number of university-designed VLSI chips [10] and found that OBDDs do indeed provide a compact representation of layouts. Furthermore, we have devised algorithms to perform the various steps of design rule checking by operating on these representations. As with other OBDD algorithms, these can exploit the sharing of nodes in the data structure to avoid repeated computations. Thus, the program effectively performs a hierarchical DRC.

Bit-mapped design rule checkers have fallen out of favor because of the enormous size of explicit image representations. Conventional wisdom states that doubling the resolution of a bit-mapped image requires quadrupling its size. For OBDD representations, this conventional wisdom does not hold. In particular, once we refine down to a subimage that is entirely black or white, a simple leaf value suffices. Thus, the image can be represented at a very fine level of resolution to handle such features as non-rectilinear geometry and complex design rules.

5 Conclusion

Symbolic analysis methods, especially those based on OBDDs, have proven highly effective for a range of tasks in analysis and synthesis. Researchers are devising clever techniques to cast a problem into a discrete form that in turn can be encoded as Boolean functions. As our design-rule checking example shows, a surprising variety of tasks can be handled in this manner.

References

[1] C. Berthet, O. Coudert, and J. C. Madre, "New Ideas on Symbolic Manipulation of Finite State Machines," *ICCD*, 1990, 224–227.

[2] S. Bose, and A. L. Fisher, "Verifying pipelined hardware using symbolic logic simulation," *ICCD*, 1989, 217–221.

[3] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Computers*, Vol. C-35, No. 8 (August, 1986), 677–691.

[4] R. E. Bryant, "Algorithmic aspects of symbolic switch network analysis," *IEEE Trans. CAD/IC*, Vol. CAD-6, No. 4 (July, 1987), 618–633.

[5] R. E. Bryant, "Boolean analysis of MOS circuits," *IEEE Trans. CAD/IC*, Vol. CAD-6, No. 4 (July, 1987), 634–649.

[6] R. E. Bryant, D. E. Beatty, and C.-J. H. Seger, "Formal Hardware Verification by Symbolic Ternary Trajectory Evaluation," *28th Design Auto. Conf.*, June, 1991, 397–402.

[7] R. E. Bryant, "Extraction of Gate Level Models from Transistor Circuits by Four-Valued Symbolic Analysis," *ICCAD*, 1991, 350–353.

[8] R. E. Bryant, "Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams," *ACM Computing Surveys*, Vol. 24, No. 3 (September, 1992), 293–318.

[9] J. R. Burch, E. M. Clarke, and K. McMillan, "Symbolic model checking: 10^{20} states and beyond," *Fifth Annual IEEE Symp. Logic in Computer Science* 1990, 428–439.

[10] Y.-A. Chen, personal communication, Carnegie Mellon University, 1993.

[11] K. L. McMillan, *Symbolic model checking: an approach to the state explosion problem*, PhD thesis, School of Computer Science, Carnegie Mellon University, 1992.