Design recovery for incomplete combinational logic

Travis E. Doom
Computer Science and Engineering
Wright State University,
Dayton, OH 45435-0001
doom@cs.wright.edu

Anthony S. Wojcik, Moon-Jung Chung
Computer Science and Engineering
Michigan State University,
East Lansing, MI 48824-1027
{wojcik,chung}@cse.msu.edu

Abstract

Motivated by the problem of reengineering legacy digital circuits for which design information is missing or incomplete, this paper presents a new technique for representing the relationships among the internal components of a combinational circuit. This technique proves to be a powerful tool for redesign, capable of representing internal Boolean relationships in a fully or partially specified multiple-output combinational circuit with a single data structure.

1. Introduction

The problem of reengineering of digital circuits is to take a given design and to respecify or remanufacture a circuit without repeating the entire design process [7]. A striking example for the need for modernization of legacy systems is the evolution underway within DoD [1].

In many cases, CAD tools used to develop the initial design provide information concerning the functionality and design of a circuit that simplify the reengineering process. Unfortunately, many existing devices were developed without the use of a comprehensive CAD process. Of course, even when CAD documentation is available, many systems are designed or modified manually, leading to potentially conflicting information [6]. In other cases, the CAD documentation that once existed for a design may be difficult to obtain. Detailed information about "legacy" systems undergoing reengineering is often not available [2, 5]. In these cases, the reengineering process begins with only partial information.

In order for any reengineering methodology to overcome these complications, it is necessary to be able to recognize the functionality of any component (or module) in the context of the overall circuit function. Even though the circuit component's design may only be partially specified and the functionality of the system of which it is a part only partially defined, it is often the case that additional information about the system being reengineered may be available. It is critical that such additional information be used in an attempt to determine the functionality of the partially specified component. This recognition of functionality may be inherently impossible in an implementation for which only partial information is available, but it may be deducible in many situations. In the case where recognition is impossible due to lack of critical information, a representation of all known or deduced information would be invaluable for continued reasoning about functionality.

Traditional binary decision diagram (BDD) representations of circuit functionality are only applicable when the functionality of the circuit is fully specified. We demonstrate a mechanism which allows the representation of circuit functionality when such functionality is only partial defined. We allow the inclusion of a selected subset of the circuit's input, output, and internal net variables as BDD decision variables, and thus are capable of representing any Boolean relationship among these structures. We present the results of an algorithm capable of utilizing these relationships to deduce new ones among these variables.

2. Representing structure

The function of a combinational circuit is generally represented as a set of Boolean functions, each of which describes the logical behavior of one of the circuit's outputs. This notation concisely represents the behavioral functionality of the circuit, representing the function only in terms of the circuit's primary inputs and primary outputs. Note that any number of circuit implementations exist which satisfy this behavioral functionality. Let us now consider a description which is capable of representing relationships between structural components of a particular implementation.

Consider a combinational circuit consisting of k internal components. The variables representing the inputs or outputs to any circuit component are **net variables**. Each component i in a combinational circuit defines the relationship between the component's n_i inputs and its m_i outputs. Denote component i's **input variable vector** as $\mathbf{x_i}$ and its **output variable vector** as $\mathbf{y_i}$ where each element in $\mathbf{x_i}$ and $\mathbf{y_i}$ represents the value of a net variable. The **function of component** i can therefore be represented as $f_i(\mathbf{x_i}) = \mathbf{y_i}$.

The **characteristic function of component** i, $\mathcal{X}_i^C : \{0,1\}^{|\mathbf{x_i}|+|\mathbf{y_i}|} \rightarrow \{0,1\}$, is defined to be:

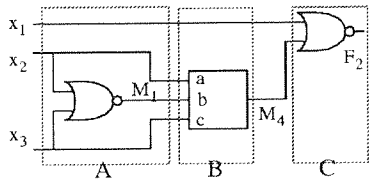
$$\mathcal{X}_i^C(\mathbf{x_i}, \mathbf{y_i}) = 1 \text{ iff } \mathbf{f_i}(\mathbf{x_i}) = \mathbf{y_i} \quad (1)$$

The **structure function** of a k component combinational circuit over the net variables $\mathcal{N}$ is defined to be $\mathcal{X}^S : \{0,1\}^{|\mathcal{N}|} \rightarrow \{0,1\}$ where:

$$\mathcal{X}^S(\mathcal{N}) = \prod_{i=1}^{k} \mathcal{X}_i^C(\mathbf{x_i}, \mathbf{y_i}) \quad (2)$$

The structure function is a Boolean function whose value is false only for those assignments of Boolean values to net variables that contradict the functional constraints imposed by the circuit

0-7695-0104-4/99 $10.00

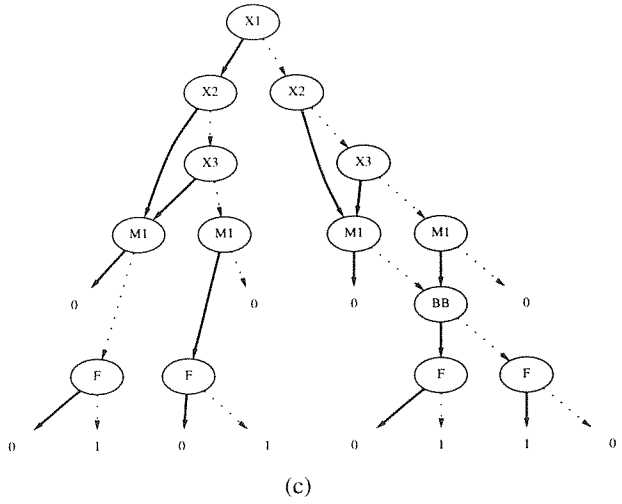


$$F_2 = \overline{X_1} \cdot \overline{BB(X_2, M_1, X_3)}$$

(a)

X_1	X_2	X_3	M_1	BB	F_2	$\mathcal{X}^S$
1	1	*d*	0	*d*	0	1
1	0	1	0	*d*	0	1
1	0	0	1	*d*	0	1
0	1	*d*	0	1	0	1
0	1	*d*	0	0	1	1
0	0	1	0	1	0	1
0	0	1	0	0	1	1
0	0	0	1	1	0	1
0	0	0	1	0	1	1
		otherwise				0

(b)



(c)

Figure 1. Schematic, structural truth table, and structural BDD for `partial simplecircuit`. (a) Partial schematic and functional description of `simplecircuit`; (b) Truth table for $\mathcal{X}^S$; (c) BDD representing `simplecircuit`'s structure function. The blackbox net variable BB is used to represent the value of the unspecified structure output M_4.

structures. Hence, *the value of the complete structure function is false for net variable assignments which could not be observed in the correctly functioning circuit.*

We refer to such a BDD representing the structure function as the *structural BDD* for the circuit. In a completely specified circuit, the values assigned to the primary inputs completely specify the necessary value of all other net variables. Therefore, for each assignment of input variables in a completely specified circuit, exactly one path from the root of the BDD to a 1-terminal (a *1-path*) exists in its structural BDD.

2.1. Representing unknown structures

In many cases complete information regarding the overall functionality of the circuit is unavailable. We refer to any component whose functionality is not fully specified as a *blackbox* structure. Furthermore, we refer to the outputs of such a structure as *blackbox net variables.*

Let the behavior of an output $b \in \mathcal{N}$ of a blackbox component i with inputs $\mathbf{x_i}$ be defined by the partial function $f(\mathbf{x_i})$. Then the **characteristic function for blackbox output** b is defined to be:

$$\mathcal{X}^C_{i,b}(\mathbf{x_i}, \mathbf{b}) = \begin{cases} 0, & \text{if } f(\mathbf{x_i}) \neq \mathbf{b} \\ 1, & \text{if } f(\mathbf{x_i}) = \mathbf{b} \\ 1, & \text{if } f(\mathbf{x_i}) = \textit{undefined} \end{cases} \quad (3)$$

By defining the characteristic function of a partially specified component in this way, we can state the following property of the circuit's structure function.

Theorem 1 *Let $\mathcal{X}^S$ be the structure function for a combinational circuit A as defined in Equation 2 where $\mathcal{X}^C$ may represent a characteristic function for any structure, including structures specified by a partial function, as defined in Equation 3. Then $\mathcal{X}^S(\mathcal{N}) = 0$ only if the assignment of Boolean values to net variables $\mathcal{N}$ is never observable in the functioning circuit A [3].*

In a correctly functioning combinational circuit, for each assignment of Boolean values to primary inputs, there is exactly one corresponding assignment of non-primary input net variables which is observable in A. A structure function which includes the characteristic function of a blackbox output may only partially specify the functionality of the combinational circuit it represents. In such a structure function an assignment of net variables for which $\mathcal{X}^S(\mathcal{N}) = 1$ does not contradict any known relationship, but is not guaranteed to be observable in the functioning circuit.

Figure 1(a) presents a partial specification for `simplecircuit`, representing a portion of the implementation as a blackbox. Note in the BDD (Figure 1(c)) representing the partial specification's structure function (Figure 1(b)) that there is not always a unique 1-path for every input variable assignment. Thus, this structure function only partially specifies the functionality of the circuit. The BDD representation of the structural function allows us to efficiently identify input conditions under which output variables are sensitized to the value of BB (M_4).

Theorem 2 *Let G be a BDD representation of a structural function with a blackbox net variable b. Consider each 1-path in G. If the node corresponding to the net variable b does not appear on the 1-path, then b is a* don't care *under the variable assignment described by the 1-path. If such a node does appear, than at least one net variable farther from the root in the BDD's variable ordering is sensitized to b's value [3].*

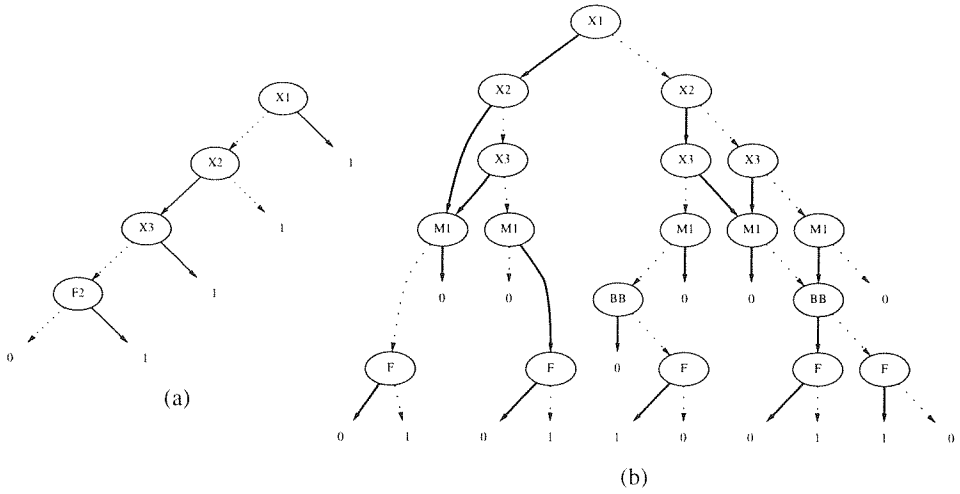


Figure 2. Using additional Boolean relationships. (a) The BDD representing the constraint characteristic function of the relationship $X_1' \cdot X_2 \cdot X_3' \rightarrow F_2$; (b) The BDD representing the structure function of `simplecircuit` after including the relationship provided by the relationship in (a).

2.2. Utilizing additional relationships

Any Boolean relationship between net variables can be introduced into the structure function to more completely specify the circuit's behavior.

For any Boolean relationship $\mathcal{R}(\mathbf{x}, \mathbf{y})$ between net variables $\mathbf{x}, \mathbf{y} \in \mathcal{N}$, a **constraint characteristic function** is defined as:

$$\mathcal{X}^R(\mathcal{N}) = 0 \text{ iff } \mathbf{x}\overline{\mathcal{R}}\mathbf{y} \tag{4}$$

Applying a constraint characteristic function to the structure function for a circuit introduces additional knowledge while maintaining the validity of Theorems 1 and 2. The introduction of constraint functions may allow the behavior of a partially specified circuit to be fully specified.

A great deal of information regarding a design undergoing reengineering is often available. If the available information describes a Boolean relationship between net variables, then its corresponding constraint characteristic function can be determined (regardless of the level of design at which the information was originally presented) and used to further specify the behavior of the blackbox (Figure 2).

Information from any level of design can be used to help determine the behavior of an unspecified blackbox structure. When a characteristic function for any relationship is applied to the structural function for the circuit, previously unspecified behavior of blackbox components become specified under certain conditions [3]. After constructing a structure function and introducing the all known relationships between internal variables as constraint characteristic functions, three cases present themselves.

In the first case, the important functionality of the blackbox can be extracted from the BDD representation of the structure function. The exact functionality of the blackbox is not necessarily defined; but as the functionality of the circuit is fully defined, the unknown specifications of the blackbox are *don't cares*. Thus the blackbox is specifiable to within *don't care* conditions, which is sufficient for reengineering.

The second case occurs when no value of a blackbox output can satisfy all constraints. This situation is characterized by the existence of an assignment of values to primary inputs for which no 1-path exists. If this situation occurs, we can identify the conflict but must resolve the situation externally.

In the final case, the overall circuit functionality is not fully specified, but no conflicts exist. In this case, additional knowledge is required to allow complete design recovery. Using the BDD representation we can extract the set of necessary relationships which must be determined in order to complete the specification. If these relationships can be determined (perhaps through the use of a working model or simulation of the circuit's behavior), the blackbox component can be specified to within *don't care* conditions without resorting to exhaustive testing.

3. Implementation and results

The structural BDD-based approach has been applied to a number of benchmark combinational circuits. Preliminary results are summarized in Table 1. The "Shared BDD Size" column presents the size of the BDD(s) representing the functionality of the circuit, represented by an efficient complemented-edge, shared-BDD implementation [8]. For each circuit, we test several blackbox scenarios including cases in which the subcircuit contains multiple blackboxes and multiple outputs. For each scenario, we present the size of the structural BDD and the CPU time (CPU seconds on a SPARCstation 20) necessary to construct the BDD and apply available additional information (in this case, the relationships found in each circuit's ATPG test vector set).

Since net variables which represent non-essential circuit structures are reduced, once it is no longer necessary to reference the variable, the size of the structural BDD is relatively independent

Circuit Name	Num. Inputs	Num. Outputs	Approx. Gates	Shared BDD Size	Unknown Gates	Structural BDD Size	Unresolved Vectors	CPU Time
alu4	14	8	681	1453	0	1663	0	4.6
					2	1774	0	7.61
					4	1779	7	8.8
f51m	8	8	43	73	0	765	0	17.1
					5	3057	0	20.9
					10	3898	12	20.9
pm1	16	13	39	42	0	1101	0	45.4
					4	1980	0	60.3
					8	3605	0	78.9
t481	16	1	2072	202	0	204	0	294.78
					10	441	4	353.1
					50	5061	68	1738.3
z4ml	7	4	20	47	0	80	0	3.8
					3	173	0	4.5
					10	215	10	5.3

Table 1. Preliminary results. Structural BDD sizes and run times for the discovery of functionality of partially specified components removed from benchmark circuits.

of the number of gates in the circuit. The limiting factors on its size are the number of primary inputs and outputs, as well as the number of essential internal structures represented. For a moderate number of internal structures, the structural BDD representation should be no more than an order of magnitude larger than its traditional BDD representation. Although the representation of the circuit's structure function is significantly larger than the traditional BDD representation, this complexity is the necessary cost of providing a framework for the representation of partial information.

The number of "Unresolved Vectors" column indicates the number of input/output relationships which must be determined in order to specify the behavior of the blackboxes to within don't care conditions and to therefore specify the functionality of the overall circuit. Note that the number of unresolved vectors is significantly influenced by the selection of the unknown gates. The number of blackbox structures, the numbers of inputs and outputs of each structure, and the importance of each unknown structure's role in the overall circuit functionality are all factors which affect the difficultly of completely determining the complete functionality of the circuit.

4. Conclusion

We have shown a formal approach to recovering the design of components in a partially specified combinational design. Unlike traditional BDD representations of circuit function, our approach is capable of *representing partial specifications* of the circuit's external or internal functionality. This approach uses characteristic functions to represent relevant Boolean relationships among net variables, where the relationships can come from any level of the design process.

We have presented techniques which allow for the deduction of the functionality of unspecified circuit components. When complete deduction is not possible, our representation allows for the enumeration of unknown relationships which may allow complete recovery if a means for acquiring these relationships exists.

Since no existing techniques provide an effective solution to this problem, we provide preliminary results to demonstrate the feasibility of this technique. This approach has been successfully applied to more complex circuits under a variety of partial knowledge scenarios, including cases in which the unspecified subcircuit contains multiple blackbox structures and multiple outputs. When combined with existing semantic matching techniques [4], this approach allows the recovery of behavioral level design for some incompletely described devices.

As this problem is inherently intractable, this approach must fail for problems of a certain size or complexity. Future goals of this research include a more detailed exploration of information available at various levels of design and appropriate encoding techniques, expanding the representation and deduction techniques to include sequential circuits, and the analysis of feasible problem size under various BDD variants.

References

[1] J. V. Breen. Reverse engineering at DMEA [Defense MicroElectronics Activity]. Presentation at the 1998 Reverse Engineering Workshop, January 5 – 8 1998.

[2] R. E. Bryant. Symbolic analysis methods for masks, circuits, and systems. In *Proceedings of the IEEE International Conference on Computer Design*, pages 6–8, Oct. 1993.

[3] T. Doom. *Design Recovery for Combinational Logic Exploiting Boolean Relationships*. PhD thesis, Department of Computer Science and Engineering, Michigan State University, Aug. 1998.

[4] T. Doom, J. White, A. Wojcik, and G. Chisholm. Identifying high-level components in combinational circuits. In *Proceedings of the 1998 Great Lakes Symposium on VLSI*, pages 313–318, Lafayette, LA, Feb. 1998.

[5] M. A. Dukes. Generating VHDL models from inadequately-documented integrated circuits. In *Proceedings of the Conference on Advances in Modeling and Simulation*, pages 165–171, April 1994.

[6] K. Keutzer. The need for formal methods for integrated circuit design. In *Proceedings of the First International Conference on Formal Methods in Computer-Aided Design*, volume 1166, pages 1–18. Springer Lecture Notes in Computer Science, November 1996.

[7] M. G. Rekoff. On reverse engineering. *IEEE Trans. on Systems, Man, and Cybernertics*, SMC-15(2):244–252, March/April 1985.

[8] F. Somenzi. CUDD: Colorado University Decision Diagram package. [URL: http://www.bessie.colorado.edu/~fabio/CUDD], 1997. Release 2.1.2.