This is to certify that the

thesis entitled

REDESIGN PROCESS OF DIGITAL VLSI CIRCUITS WITH INCOMPLETE IMPLEMENTATION INFORMATION

presented by

Mohammad Athar Khalil

has been accepted towards fulfillment
of the requirements for

Master's degree in Electrical Eng

Major professor

Date 05/05/98

REDESIGN PROCESS OF DIGITAL VLSI CIRCUITS WITH INCOMPLETE IMPLEMENTATION INFORMATION

By

Mohammad Athar Khalil

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical Engineering

1998

ABSTRACT

REDESIGN PROCESS OF DIGITAL VLSI CIRCUITS WITH INCOMPLETE IMPLEMENTATION INFORMATION

By

Mohammad Athar Khalil

This thesis deals with the problem of redesigning digital VLSI circuits with incomplete implementation information. Given a digital circuit with incomplete implementation information, the developed redesign process recovers functionality of the missing parts in original design using test generation techniques. A circuit is redesignable if the transfer functions of the portion with incomplete implementation information can be derived. The derived transfer functions are then used to re-implement the missing portions. We do not intend to discover the exact circuit schematic and components that were present in the circuit originally implemented. Rather, the functions originally intended to be present will be identical. The developed redesign process is comprised of three steps: Redesignability check, Feasibility check and Re-implementation. A set of simple rules have been developed to quickly analyze whether redesigning the missing parts of a target circuit is cost effective or not. A number of circuit partitioning schemes have also been developed to decrease the computational complexity and to improve the quality of the redesign process. Several benchmark circuits have been tested and satisfactory results are obtained.

To my mother and father.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Chin-Long Wey, for all his assistance with my research and the development of this thesis. His guidance, care, patience and constant encouragement not only kept me motivated but also were crucial for keeping the research objectives in perspective. Thanks are also due to the members of my committee, Dr. James A. Resh and Dr. Anthony S. Wojcik, for all their help and support.

I am very grateful to my family for encouraging me to pursue higher studies, especially to my parents for the encouragement and support they have given me every day of my life. I also wish to thank all my friends for their moral support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Industry and military community increasingly rely on the use of "smart" systems for intelligent manufacturing control systems and weapon systems. The components that make these systems "smart" are the complex microelectronics devices that form their "brain." Microelectronics technologies are extremely dynamic and now become obsolete every 18 months. This makes microelectronics the main factor driving the "smart" systems obsolescence and defence missions degradation. The life of these "smart" systems can be generally extended by improving their reliability and maintainability using advanced technologies.

Replacement methods have been effectively used to resolve the microelectronics obsolescence problem to enhance maintainability. However, the methods are effective only if the digital designs are well documented. Unfortunately, the present methodologies perform poorly and many designs are undocumented [1]. Many existing designs have been developed without the assistance of a comprehensive CAD process, which means that the detailed information about the design at various intermediate levels is not available. The netlist, HDL (Hardware Description Language), or design data is not present, and system interfaces and functional requirements are not documented. As a result, one may either use the exact replacement parts from sources that were not on the original documentation, or

take a similar part that is not a direct replacement. Further, one may develop a re-engineering process for a form, fit, and function replacement based on initial specification.

Consider a process model of re-engineering shown in Figure 1.1 [2]. The process model is captured by two sectioned triangles. The higher levels are concepts and requirements, while the lower levels include designs and implementations. Forward engineering is the process of developing a system by moving from high level abstract specification to detailed, implementation-specific manifestations [3]. Conversely, reverse engineering is the process of analyzing a system in order to identify system components, component relationships, and intended behavior. In other words, reverse engineering is the process of constructing high level representation from lower level instantiations of an existing system.

Reverse engineering process for digital circuits verifies schematics and performance specification against actual hardware and provides a method of identifying internal structures down to a device level to determine an optimal design approach [1]. The individual devices are identified and characterized and a working schematic is developed through the use of various instruments and design tools. Reverse engineering has been an effective method that enhances the maintainability of the existing undocumented designs. However, manual reverse engineering methods won't work for complex devices. Hence, new methodologies are needed to deal with complex digital microcircuits.

Even though today's CAD tools provide a sophisticated design process from behavioral level description to detailed physical implementation, redesigning a circuit due to a minor change or technology change requires a design time equal to that of the entire circuit, or achieves a performance worse than the original one. The new methodologies can also be employed to reduce the redesign time of existing circuits for minor changes while still

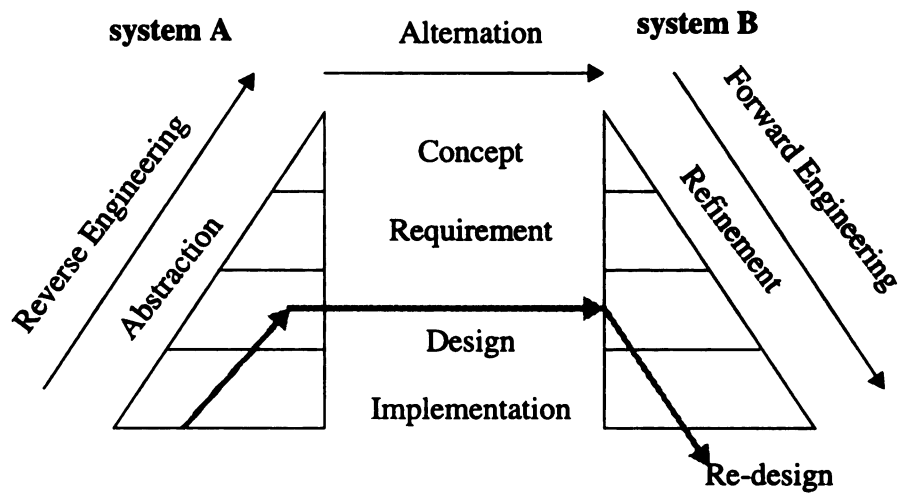


Figure 1.1: Process Model of Re-engineering.

maintaining or improving the original performance, that is, such techniques can be used for safe replacement of modules, blocks of logic gates, in the gate level design of entire circuit. Safe replacement refers to substitution of a module in the existing design with a redesigned module, in order to achieve area reduction and/or timing improvement of gate level design while preserving the overall functionality of the existing circuit.

This study deals with a challenging problem for enhancing maintainability of undocumented complex digital designs. Consider the flow describing the redesign process illustrated in Figure 1.1, the original implementation information in System A is either missing or incomplete. Redesign starts with only partial knowledge in the implementation level [4]. One of the most difficult aspects of redesign is the recognition of the functionality of existing implementation [5]. Two problems involving the recognition of the functionality can be identified:

1. The implementations of the missing or incomplete parts are given, but their functionality are unknown; and
2. The implementations of the missing or incomplete parts are unknown, but the functionality of the entire digital circuit is known.

The former problem is to recognize the functionality of each missing part from the possible design styles and/or known libraries. The later problem is to recover the functionality of the missing or incomplete parts and to re-implement these parts [6,7]. This study deals with the later problem and develops a redesign methodology to support undocumented complex digital designs using the die and test vectors as data sources.

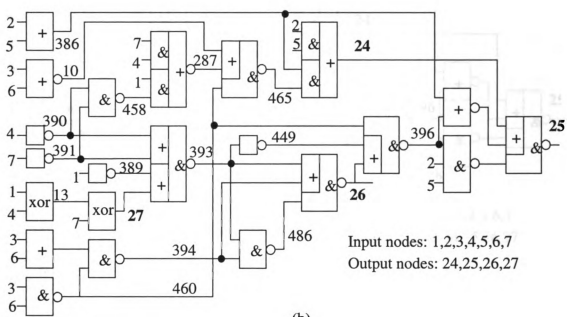
More specifically, given a digital VLSI circuit, the original implementation information is either missing or incomplete. Consider the example circuit, *z4ml* [8], which is a

three bit adder. Figure 1.2(a) shows the circuit netlist generated by *sis* [9], while Figure 1.2(b) illustrates the schematic circuit diagram. It is assumed that the implementation of Block B is missing or incomplete. The masked portion, of the netlist and schematic for example circuit I shown in Figure 1.3(a) and (b) respectively, represents the missing or incomplete implementation information. With the partial knowledge in the implementation and the functionality of the digital circuit, an efficient redesign process is developed to recognize the functionality of the missing/incomplete parts and to recover the original design from the existing implementation. A circuit is redesignable [4] if the transfer function, i.e., inputs/outputs relationship, of each missing part can be derived. Therefore, the missing parts can be re-implemented from the derived transfer functions. Note that we do not intend to discover the exact circuit schematic and components that were present in the circuit originally implemented. Rather, the functions originally intended to be present will be identical.

In this study, the redesign problem is resolved by using some test generation techniques. Chapter 2 will briefly review the test generation schemes used in this study. In Chapter 3, the redesign problem is stated formally and the problem is formulated with a system model. Chapter 4 describes the development of redesign process and also presents schemes to improve its quality. Chapter 5 presents the redesign process and also reports our experimental results. Finally, Chapter 6 gives the conclusions and presents some ideas for future research.

```
.model z4ml
.inputs 1 2 3 4 5 6 7
.outputs 24 25 26 27
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 0.07 0.07
.default_output_load 3.00
.gate or2 a=2 b=5 O=[386]
.gate nor2 a=3 b=6 O=[10]
.gate inv1 a=4 O=[390]
.gate inv1 a=7 O=[391]
.gate nand2 a=[390] b=[391] O=[458]
.gate aoi22 a=[458] b=1 c=4 d=7 O=[287]
.gate nand2 a=3 b=6 O=[460]
.gate oai21 a=[10] b=[287] c=[460] O=[465]
.gate ao22 a=[386] b=[465] c=2 d=5 O=24
.gate xor a=1 b=4 O=[13]
.gate xor a=[13] b=7 O=27
.gate inv1 a=1 O=[389]
.gate oai22 a=27 b=[389] c=[390] d=[391] O=[393]
.gate inv1 a=[393] O=[449]
.gate or2 a=3 b=6 O=[452]
.gate nand2 a=[452] b=[460] O=[394]
.gate nand2 a=[393] b=[394] O=[486]
.gate oai21 a=[393] b=[394] c=[486] O=26
.gate oai21 a=[449] b=26 c=[460] O=[396]
.gate nor2 a=[386] b=[396] O=[289]
.gate nand3 a=[396] b=2 c=5 O=[476]
.gate oai21 a=[289] b=24 c=[476] O=25
.end
```

(a)



(b)

Figure 1.2: Example Circuit, *z4ml*: (a) Netlist; and (b) Schematic.

```
.model z4ml
.inputs 1 2 3 4 5 6 7
.outputs 24 25 26 27
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 0.07 0.07
.default_output_load 3.00
.gate or2 a=2 b=5 O=[386]
.gate nor2 a=3 b=6 O=[10]
.gate inv1 a=4 O=[390]
.gate inv1 a=7 O=[391]
.gate nand2 a=[390] b=[391] O=[458]
.gate aoi22 a=[458] b=1 c=4 d=7 O=[287]
.gate nand2 a=3 b=6 O=[460]
.gate oai21 a=[10] b=[287] c=[460] O=[465]
.gate ao22 a=[386] b=[465] c=2 d=5 O=24
.gate xor a=1 b=4 O=[13]
.gate xor a=[13] b=7 O=27
.gate inv1 a=1 O=[389]
.gate oai22 a=27 b=[389] c=[390] d=[391] O=[393]
.gate inv1 a=[393] O=[449]
.gate or2 a=3 b=6 O=[452]
.gate nand2 a=[452] b=[460] O=[394]
.gate nand2 a=[393] b=[394] O=[486]
.gate oai21 a=[393] b=[394] c=[486] O=26
.gate nand3 a=[396] b=2 c=5 O=[476]
.gate oai21 a=[289] b=24 c=[476] O=25
.end
```

(a)

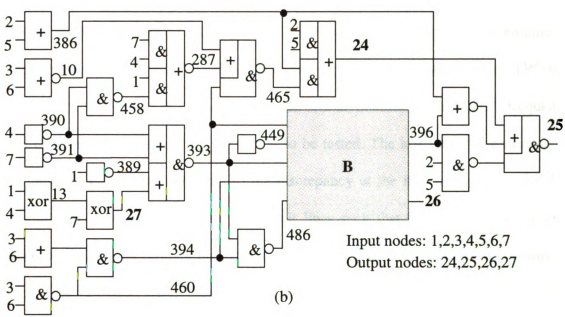


Figure 1.3: Example Circuit I, with Incomplete Implementation Information: (a) Netlist; and (b) Schematic.

Chapter 2

BACKGROUND

This chapter reviews the test generation methods that can be used to resolve the redesign problem. There are various algorithmic automatic test pattern generation systems (ATPG systems) currently in use for combinational logic circuits. Some ATPG systems generate algebraic equations for the circuit and then perform symbolic manipulation on these equations to generate all possible test patterns for a particular fault. Other ATPG systems find the test vectors in a topological, or structural, manner. Such ATPG's frequently use a data structure representing the circuit to be tested. The test vector is generated by assigning the values corresponding to the discrepancy at the line with a fault and then searching for consistent values for all circuit lines such that the discrepancy becomes observable at a circuit primary output [10]. A brief discussion of one algebraic manipulation method (Boolean Difference) and one structural search method (D-Algorithm) follows:

2.1 Boolean Difference

The basic principle involved in Boolean Difference is to derive two Boolean expressions, one of which represents the fault free behavior of the circuit and the other rep-

resents the logical behavior under an assumed single stuck at fault condition. These two equations are then exclusive-ORed; a fault is indicated if the result is 1 [11].

Let $F(X) = F(x_1, \ldots, x_n)$ be a logic function of n variables. If one of the inputs to the logic function, e.g. input x_i, is faulty, then the output would be $F(x_1, \ldots, \bar{x}_i, \ldots, x_n)$. The Boolean difference of $F(X)$ with respect to x_i is defined as:

$$\frac{d}{dx_i}F(X) = F(x_1, \ldots, x_i, \ldots, x_n) \oplus F(x_1, \ldots, \bar{x}_i, \ldots, x_n)$$

It can also be represented as [12]:

$$\frac{d}{dx_i}F(X) = F_i(0) \oplus F_i(1)$$

where

$$F_i(0) = F(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$$

$$F_i(1) = F(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

The function $\frac{d}{dx_i}F(X)$ is called the Boolean difference of $F(X)$ with respect to x_i. It can be seen that when $F(x_1, \ldots, x_i, \ldots, x_n) \neq F(x_1, \ldots, \bar{x}_i, \ldots, x_n)$, $\frac{d}{dx_i}F(X) = 1$ and that when $F(x_1, \ldots, x_i, \ldots, x_n) = F(x_1, \ldots, \bar{x}_i, \ldots, x_n)$, $\frac{d}{dx_i}F(X) = 0$. To detect a fault on x_i, it is required to find input combinations so that whenever x_i changes to $\bar{x}_i$, (due to a fault), $F(x_1, \ldots, x_i, \ldots, x_n)$ will be different from $F(x_1, \ldots, \bar{x}_i, \ldots, x_n)$. In other words the aim is to find input combinations for each fault occurring on x_i such that $\frac{d}{dx_i}F(X) = 1$.

Some useful properties of the Boolean Difference are as follows [13]:

$$\frac{d}{dx_i}\overline{F(X)} = \frac{d}{dx_i}F(X); \quad \overline{F(X)} \text{ denotes the complement of } F(X). \qquad (2.1)$$

$$\frac{d}{dx_i}F(X) = \frac{d}{d\bar{x}_i}F(X) \qquad (2.2)$$

$$\frac{d}{dx_i}\left(\frac{d}{dx_j}(F(X))\right) = \frac{d}{dx_j}\left(\frac{d}{dx_i}(F(X))\right) \qquad (2.3)$$

$$\frac{d}{dx_i}|F(X)G(X)| = F(X)\frac{d}{dx_i}(G(X)) \oplus G(X)\frac{d}{dx_i}(F(X)) \oplus \left(\frac{d}{dx_i}(F(X))\right)\left(\frac{d}{dx_i}G(X)\right) \tag{2.4}$$

$$\frac{d}{dx_i}|F(X) + G(X)| = \overline{F(X)}\frac{d}{dx_i}(G(X)) \oplus \overline{G(X)}\frac{d}{dx_i}(F(X)) \oplus \left(\frac{d}{dx_i}(F(X))\right)\left(\frac{d}{dx_i}G(X)\right) \tag{2.5}$$

$$\frac{d}{dx_i}|F(X) \oplus G(X)| = \frac{d}{dx_i}F(X) \oplus \frac{d}{dx_i}G(X) \tag{2.6}$$

A Boolean function $F(X)$ is said to be independent of x_i if and only if $F(X)$ is logically invariant under complementation of x_i, that is if $F(x_1, \ldots, x_i, \ldots, x_n) = F(x_1, \ldots, \bar{x}_i, \ldots, x_n)$.

This implies that a fault in x_i will not affect the final output $F(X)$ and $\frac{d}{dx_i}F(X) = 0$. Based on this, some additional properties can be added [11]:

$\frac{d}{dx_i}F(X) = 0$; if $F(X)$ is independent of x_i.

$\frac{d}{dx_i}F(X) = 1$; if $F(X)$ depends only on x_i.

$\frac{d}{dx_i}|F(X)G(X)| = F(X)\frac{d}{dx_i}(G(X))$; if $F(X)$ is independent of x_i.

$\frac{d}{dx_i}|F(X) + G(X)| = \overline{F(X)}\frac{d}{dx_i}(G(X))$; if $F(X)$ is independent of x_i.

The effect of two faults at the input of a logic circuit on its output can be analyzed by defining the double Boolean difference as follows [14]:

$$\frac{d^2}{dx_i dx_j}F(X) = \frac{d}{dx_i}\left(\frac{d}{dx_j}(F(X))\right) = \frac{d}{dx_j}\left(\frac{d}{dx_i}(F(X))\right)$$

$$\frac{d^2}{dx_i dx_j}F(X) = F(0,0) \oplus F(0,1) \oplus F(1,0) \oplus F(1,1)$$

Thus test generation for multiple stuck at faults can be generalized by using Multiple Boolean Differences.

$$\frac{d^p}{dx_i \ldots dx_{ip}}F(X) = \frac{d}{dx_i}\left(\frac{d^{(p-1)}}{dx_i \ldots dx_{ip-1}}F(X)\right)$$

The Boolean difference method generates all tests for every fault in a circuit. It is a complete algorithm and does not require any trial and error. However, the method is costly in terms of computation time and memory requirements. For large circuits, a great amount of algebraic manipulation may be required to derive test for a given fault [15]. This is the reason, that we have used efficient partitioning schemes, (discussed in chapter 4), to reduce the size of target circuit, so that usage of Boolean difference method for our problem becomes feasible.

Consider example circuit given below in Figure 2.1, which is a part of the circuit

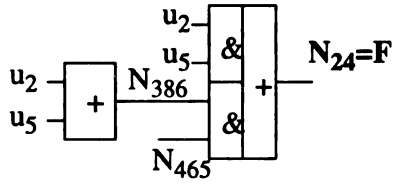


Figure 2.1: Example Circuit for Boolean Difference.

shown in Figure 1.2. Using Boolean difference for the node N_{465}, we can write

$$F = N_{24} = N_{465}(u_5 + u_2) + u_5 u_2$$

$$F_{465}(1) = u_5 u_2$$

$$F_{465}(1) = u_5 + u_2$$

$$\frac{dF}{dN_{465}} = F_{465}(0) \oplus F_{465}(1)$$

$$\frac{dF}{dN_{465}} = \bar{u}_2 u_5 + u_2 \bar{u}_5$$

The above expression shows that the node N_{465} will be sensitized to primary output N_{24} when inputs u_2 and u_5 are not equal. Considering that the node N_{465} is not accessible, the logic value of this node can be observed at the output node N_{24} by setting u_2 and u_5 to be complement of each other. The Boolean difference expression gives all possible sensitized paths for the node, that is, in this example the possible ways to sensitize N_{465} to N_{24} are to have either $\bar{u}_2 = u_5$ or $u_2 = \bar{u}_5$.

2.2 D-Algorithm

Before going into the details of D-Algorithm, the main concepts of structural test generation methods for stuck at faults are discussed briefly. There are three following fundamental operations involved in generating a test for a stuck at fault:

Fault Sensitization

It is the process of generating a discrepancy at the fault site.

Fault Propagation

It is the process of moving a discrepancy closer to a circuit output.

Line Justification

It is the process of assigning consistent values to all of the lines in the circuit that were not assigned values through fault sensitization or fault propagation.

The D-algorithm [16] uses sensitized paths to find a test vector for a fault if one exists. It has been specified formally, and is appropriate for computer implementation. The notations and certain new terms that will be used are first described before going in to the details of D-Algorithm [17].

D-Notation

To keep track of error propagation values must be considered in both the fault free circuit N, and the faulty circuit N_f defined by the target fault f. For this we define composite logic values of the form v/v_f, where v and v_f are the values of the same signal in N and N_f. The composite logic values that represent errors, 1/0 and 0/1, are denoted by symbols D and $\overline{D}$ respectively. This is a compact way of specifying how faults propagate through a circuit. D implies that in the good machine a 1 is to be found at the node holding D, whereas in the faulted machine a 0 is to be found at that node. $\overline{D}$ is defined analogously. The other two composite values, 0/0 and 1/1, are denoted by 0 and 1. Any logic operation between two composite values can be done by separately processing the fault free and faulty values, then

composing the results. For example, $\overline{D} + 0 = 0/1 + 0/0 = 0 + 0/1 + 0 = 0/1 = \overline{D}$. To these four binary composite values a fifth value (X) is added to denote an unspecified composite value, that is, any value in the set $\{0,1,D,\overline{D}\}$. It can be verified that D behaves consistently with the rules of the Boolean algebra.

Singular Cover

The Singular Cover of a logic gate is a compact representation of its truth table. For example, the truth table and singular cover of a 2-input NOR gate, with inputs a, b and output c, are given in Table 2.1 and Table 2.2 respectively.

Table 2.1: Truth Table, 2-input NOR gate

a	b	c
0	0	1
0	1	0
1	0	0
1	1	0

Table 2.2: Singular Cover, 2-input NOR gate

a	b	c
0	0	1
X	1	0
1	X	0

Each row of the singular cover is called a CUBE. The set of cubes, which contains 0 as the output value, is called the P0 set. The set of cubes containing 1 as the output value

is called the P1 set.

Propagation D-Cubes

The propagation D-Cubes of a gate are those which cause the output of a gate to depend solely on one or more of its inputs (usually one). This allows a fault on this input to be propagated through the gate. For example, the propagation D-cubes for a 2-input NOR gate are given in Table 2.3.

Table 2.3: Propagation D-Cubes, 2-input NOR gate

a	b	c
0	D	$\overline{D}$
D	0	$\overline{D}$
D	D	$\overline{D}$

Propagation D-cubes can be derived from the singular cover, or by inspection. To generate the propagation D-cubes, intersect region P0 of a gate's cover with region P1 according to the following algebraic rules:

$$0 \cap 0 = 0 \cap X = X \cap 0 = 0$$

$$1 \cap 1 = 1 \cap X = X \cap 1 = 1$$

$$X \cap X = X$$

$$1 \cap 0 = D$$

$$0 \cap 1 = \overline{D}$$

In general, it is possible to have up to $2^{(2N-1)}$ propagation D-cubes for an N-input gate, so normally only those cubes with a single D in the inputs are stored.

Primitive D-Cubes of Fault (P.D.C.F.)

A Primitive D-Cube of Fault for a fault in a circuit is used to specify the existence of a given fault. It is a set of inputs to the circuit which bring the fault to the circuit output. The P.D.C.F. for a fault are generated in the following manner:

1. Generate singular covers for the circuit in both its faulted and fault-free states.
2. Intersect the P0 cubes of the fault free cover with the F1 cubes of the faulted cover and intersect the P1 cubes with the F0 cubes. F1 and F0 play analogous roles in the faulted cover to P1 and P0 in the fault-free cover. Intersection is defined by intersecting each element of the cubes in the same manner as already defined for propagation D-cubes.

D-Intersection

The D-Intersection is the method used for building sensitized paths. It is a set of rules which show how D signals at the outputs of gates intersect with the propagation D-cubes of other gates, allowing a sensitized path to be constructed. Following are the set of rules, for the D-cube intersection:

Let $A = (a_1,a_2, \ldots,a_n)$ and $B = (b_1,b_2,\ldots,b_n)$ be D-cubes where a_i and $b_j \in \{0,1,D,\bar{D},X\}$ for $i,j = 1,2,\ldots,n$. The D-intersection, denoted by $A \cap B$ is given by:

1. $X \cap a_i = a_i$
2. If $a_i \neq X$ and $b_i \neq X$ then

$$a_i \cap b_i = a_i; \text{ if } b_i = a_i$$

$$= \varnothing; \text{ otherwise.}$$

Finally $A \cap B = \varnothing$, i.e. the empty cube, if for any i, $a_i \cap b_i = \varnothing$; otherwise

$A \cap B = a_i \cap b_i, \ldots, a_n \cap b_n$

The Full D-Algorithm

1. Choose a P.D.C.F. for the fault under consideration.
2. Sensitize all possible paths from the faulty gate to a primary output of the circuit. This is done by successively intersecting the P.D.C.F. of the fault with the propagation D-cubes of successor gates. The process is called the "D-Drive".
3. Justify the net assignments made during the D-drive by intersecting the singular covers of gates in the justification path with the expanding D-cube. This is called the "Consistency Operation".

The D-algorithm is applied in solution of our redesign problem with the following modification.

1. Choose D as the value for the unknown node in the circuit.
2. Sensitize paths from this node to a primary output of the circuit. This is using the propagation D-cubes of gates. The process is similar to the "D-Drive".
3. Justify the net assignments made during the D-drive by intersecting the singular covers of gates in the justification path with the expanding D-cube. This is the "Consistency Operation" of the D-Algorithm.

Consider the example circuit shown in Figure 2.2. Assume that the node 396 is not accessible and we want to find the logic value at this node. Value D is selected for this node.

This node can be observed at primary output 25. So a sensitized path is found first to this output. Using NAND3 gate for sensitizing 396 to 25, we set nodes 2 and 5 equal to 1, so that the output of the gate 476 is $\bar{D}$. Now consider NOR2 gate with output 289, setting 2 and 5 equal to 1 makes one of its inputs (386) equal to 1 whereas the other input (396) is D. So the output of this gate will be $\bar{D}$. So, in order to make D observable at the primary output node 25, the logic value needed at node 24 is 1. Since setting 2 and 5 equal to 1 forces the output of node 24 to 1, thus, the node 396 can be observed at node 25 with nodes 2 and 5 set equal to 1. The next step is to find the appropriate values of the inputs of the gates which have been assigned logic values for sensitizing the path. There must not be any conflict of logic value at a single node of the circuit. The final values for each node of the example circuit are shown in parenthesis in Figure 2.2. So, if we set inputs 2 and 5 equal to 1 the logic value at output 25 will be the same as that at node 396. Hence observing logic value at node 25, with the above condition satisfied the value at node 396 can be found without accessing it.

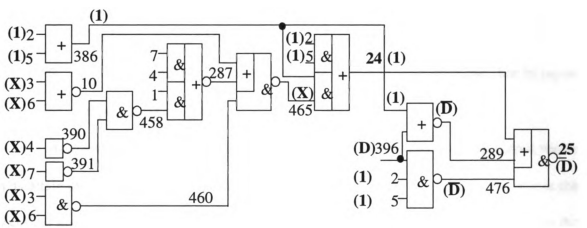


Figure 2.2: Example Circuit for D-Algorithm.

Chapter 3

PROBLEM STATEMENT

Basically, a circuit can be described by a system model, *Component Connection Model* (CCM) [18,19], as shown in Figure 3.1(a) where $\boldsymbol{a}$ and $\boldsymbol{b}$ are the *component input* and *output variables*, respectively, and $\boldsymbol{u}$ and $\boldsymbol{y}$ are the primary inputs and outputs, respectively. A digital circuit can also be described by CCM model, where each component may represent a block of logic gates. For the redesign problem, as illustrated in Figure 3.1(b), the target circuit is comprised of *Missing Parts* as indicated by shaded blocks, and the *Known Parts*. Without loss of generality, the blocks can be re-arranged as in Figure 3.1(c), where B-group contains all unknown blocks, while both A-group and C-group contain the remaining known blocks. For the redesign problem, the following assumptions are made:

1. The functionality of the target circuit is given; and
2. The functionality and internal structure of B-group are unknown, but its input/output nodes are given.

Assumption 1 implies that, for any input vector $\boldsymbol{u}$, the corresponding output vector $\boldsymbol{y}$ is attainable. Based on ***Assumption 2***, B-group is equivalent to a black-box, where the external nodes are known. One trivial solution is to apply all possible combinations to the inputs of B-group and probe the outputs. Here, it is assumed that the inputs and outputs of

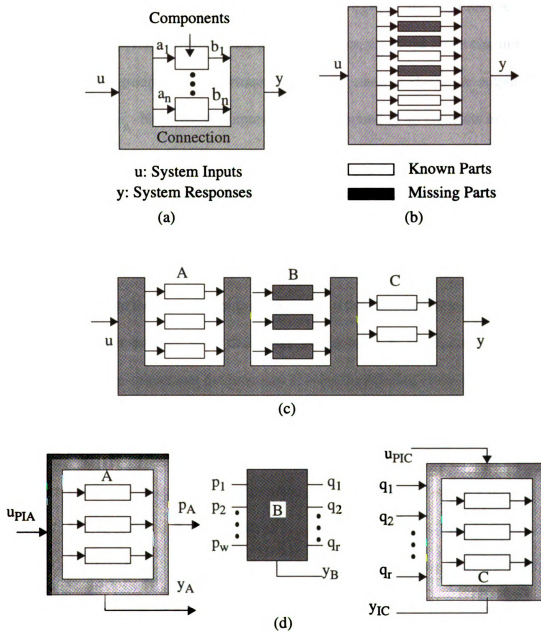


Figure 3.1: System Model: (a) CCM; and (b)-(d) Redesign Modeling.

B-group may not be all accessible except the primary inputs and outputs.

Based on the assumptions, the target circuit can be partitioned into three groups, as shown in Figure 3.1(d), where $p_B=\{p_1,p_2,...,p_w\}$ and $q_B=\{q_1, q_2,...,q_r\}$ are the inputs and outputs of B-group, respectively. Some primary outputs, y_B, of the target circuit may result from B-group. A-group takes the primary inputs, u_{PIA}, and produces the outputs, p_A, and primary outputs, y_A. Note that the inputs p_B are comprised of all elements in p_A, possibly some elements in y_A, and some primary inputs. On the other hand, the inputs of C-group include all elements in q_B, possibly some of the primary inputs, u_{PIC}, and some primary outputs, y_{IC}, resulted from both A-group and B-group. The outputs of C-group are the primary outputs, y_C. The primary inputs to these three groups may have some in common. The major task is to find the inputs/outputs relationship of the *Missing Parts* in B-group. Therefore, the redesign process is to derive the functions F_j's, j=1,2,...,r, where $q_j=F_j(p_1, p_2,...,p_w)$, and to re-implement the functions F_j's for the *Missing Parts* in B-group.

For simplicity of presenting the material, the example circuit, *z4ml*, is employed, where its netlist generated by *sis* and schematic circuit diagram are shown in Figure 1.2(a) and (b) respectively. It is a three bit full adder. The circuit has 22 gates, 7 primary inputs (nodes 1, 2, 3, 4, 5, 6, and 7 are denoted as u_1, u_2, u_3, u_4, u_5, u_6, and u_7 respectively) and 4 primary outputs (24, 25, 26, and 27, denoted as y_1, y_2, y_3, and y_4 respectively). The masked portion, of the netlist and schematic for example circuit I shown in Figure 1.3(a) and (b) respectively, represents the missing or incomplete implementation information.

The redesign process first starts with partitioning the target circuit into three groups, where B-group includes all unknown blocks, while A-group and C-group contain the remaining known blocks. In this implementation, A-group and C-group are partitioned in

such a way that C-group contains those gates which takes q_j's as their inputs, and their fan-in and fan-out gates, while the remaining known blocks are included in A-group. Figure 3.2(a) shows that B-group for example circuit I has p_B={393,394,449,460,486}, q_B={396}, and y_B= {26}. For C-group, the gates associated with q_B, or node 396, as shown in Figure 3.2(b), are the NOR2 gate "289" (denoted as the gate with the output node 289), and the NAND3 gate "476". The gate "289" has a fan-in gate, OR gate "386", and a fan-out gate, OAI21 gate "25". Therefore, C-group has u_{PIC}={2,5}, y_{IC}={24}, and y_C={25}. Furthermore, A-group includes the remaining gates and it has u_{PIA}={1,2,3,4,5,6,7}, p_A={393,394,449,460,486} and y_A={24,27}.

After partitioning the target circuit into three groups, the next step is to check if the outputs, $q_B=\{q_1,q_2,...,q_r\}$, of B-group are all *observable* from the primary outputs y_C. In Figure 3.1(d), the primary output vector y_C is a function of u_{PIC}, q_B, and y_{IC}, i.e.,

$$y_C = G_C(u_{PIC}; \{q_1,q_2,...,q_r\}; y_{IC}) \tag{3.1}$$

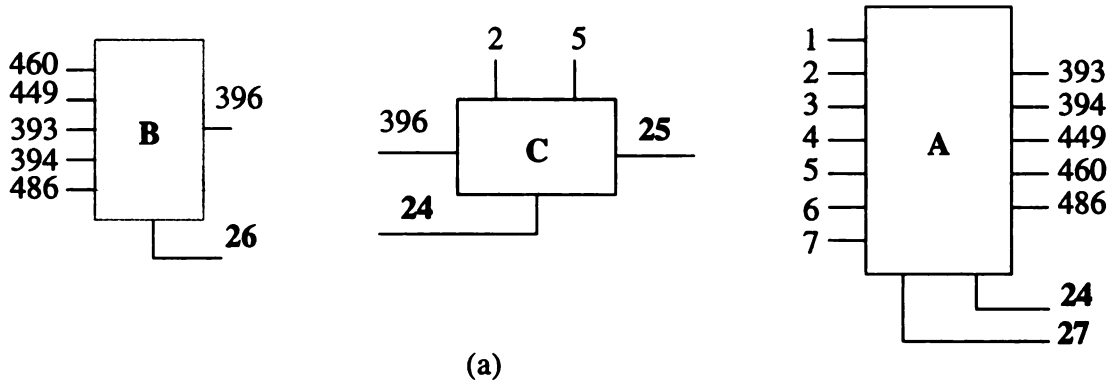
For example circuit I, in Figure 3.2(b) we have,

$$N_{25} = G_C(\{2,5\}; \{396\}; \{24\}) = \bar{N}_{24}[(u_2+u_5)+N_{396}] + u_2u_5N_{396}. \tag{3.2}$$

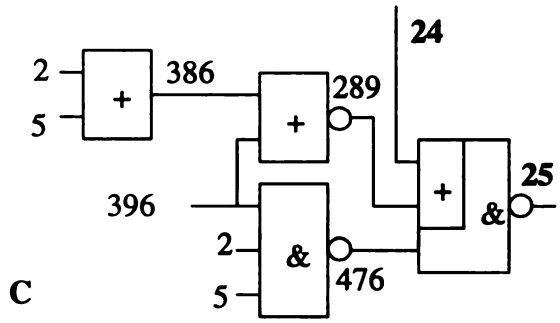
The observability of an input q_i of C-group, i.e., an output of B-group, can be checked if there exists an input vector which sensitizes q_i to any primary outputs in y_C. Otherwise, the outputs of B-group are not observable. For example, in Figure 3.2(a), {396} is observable from the primary output {25}. By (3.2), one can generate a set of input vectors as follows:

$$N_{25}=N_{396} \text{ if } [(u_2=u_5=0) \,\&\, N_{24}=0] \text{ or } [(u_2=u_5=1)] \tag{3.3}$$

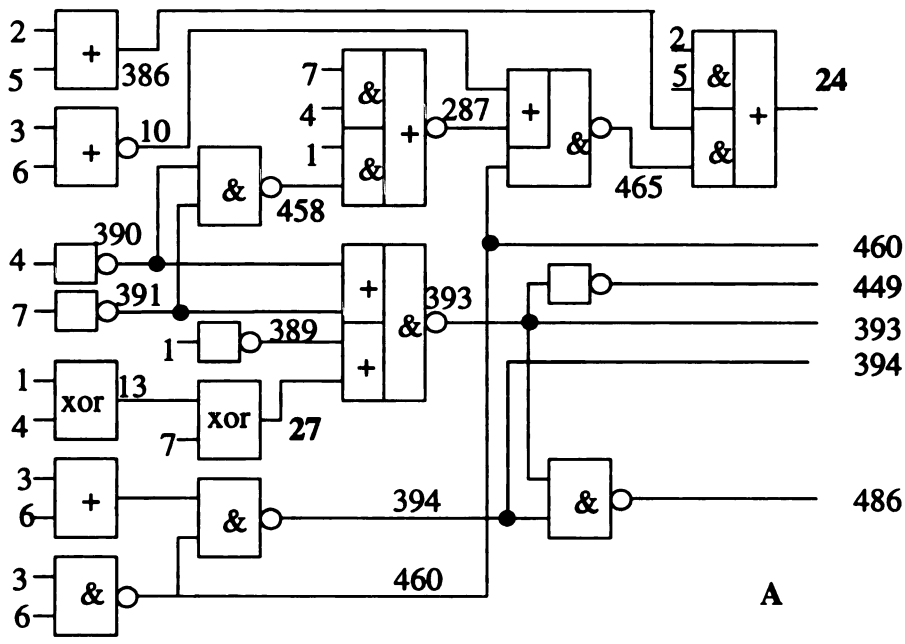
Since the gates in A-group are known and thus the outputs, p_A, of A-group can be



(a)



(b)



(c)

Figure 3.2: Circuit Partitioning Scheme: (a) Example Circuit I Partitioning; (b) and (c) Schematic of the Example Circuit I Partitions.

resulted from the primary inputs in u_{PIA}. Note that the inputs $p_B=\{p_1,p_2,...,p_w\}$ of B-group include all elements in p_A, and may include some elements in y_A and some primary inputs. Therefore, when an input vector is applied to A-group, the corresponding output vector including p_A and y_A results. This implies that an input vector, p_B^a, of B-group results when the input vector u^a is applied to A-group. For simplicity, we refer that p_B^a is *reachable* by u^a. An input vector of B-group is *unreachable* if it can not be resulted from primary inputs. The following property concludes:

Property 3.1

The outputs, q_1, q_2,..., q_r, and y_B, of B-group are "don't cares" if the corresponding input vector is unreachable from primary inputs.

Consider a reachable input vector p_B^a of B-group. Let $q_B^a=(q_1^a,q_2^a,...,q_r^a)$ denote the corresponding output vector of B-group for p_B^a. By (3.1), q_i is observable if there exists an input vector of C-group which can sensitize q_i along with a sensitized path to primary output(s), where the input vector of C-group is comprised of $\{u_{PIC}; \{q_1,q_2,...,q_r\}; y_{IC}\}$. By ***Assumption 1***, primary outputs, y_C, can be obtained from the primary inputs which can reach p_B^a and observe q_i. Thus, q_i^a is determined by derived primary outputs and known primary inputs. On the other hand, if there exist no such sensitized input vectors, then q_i is not observable, resulting q_i^a is a "don't care", i.e., q_i^a ="x". Therefore, the following property results:

Property 3.2

Let $q_B^a=(q_1^a,q_2^a,...,q_r^a)$ denote the corresponding output vector of B-group for a reachable input vector p_B^a. If q_i is not observable, then q_i^a is a "don't care."

Let R_B be the collection of primary input vectors, which can reach to p_B^a, an input vector of B-group. Suppose that there exists an input vector in R_B such that

$$y_C = G_C(u_{PIC}; \{q_1,q_2,...,q_r\}; y_{IC}) = G_C(u_{PIC}; \{q_i\}; y_{IC}) \quad (3.4)$$

This means that q_i is observable and y_C is function of the inputs u_{PIC} and y_{IC} and the only unknown parameter q_i. By ***Assumption 1***, the primary outputs, y_C, can be obtained from the primary inputs which can reach p_B^a and observe q_i. Thus, the only unknown parameter q_i in (3.4) can be resolved from the simulated y_C and the primary input vectors. Thus, the following property concludes:

Property 3.3

Let $q_B^a=(q_1^a,q_2^a,...,q_r^a)$ denote the corresponding output vector of B-group for a reachable input vector p_B^a. If q_i is observable and the sensitized input vector is independent of $q_1, q_2, ..., q_{i-1}, q_{i+1}, .., q_r$, then q_i^a can be determined by the derived primary outputs and the known primary inputs.

Consider the case that there exists an input vector in R_B such that

$$y_C = G_C(u_{PIC}; \{q_1,q_2,...,q_r\}; y_{IC}) = G_C(u_{PIC}; \{q_i,q_j\}; y_{IC}); \text{ where } q_i \neq q_j. \quad (3.5)$$

Since y_C can be obtained from the primary input vectors in R_B, q_i can be observable and q_i^a is determined by y_C, u_{PIC}, and y_{IC}, if q_j has been pre-determined. However, in general, both q_i and q_j are unknowns unless both q_i and q_j in (3.5) can be determined indepen-

dently. Therefore, q_i^a is said to be *undefined* and denoted as q_i^a="U".

Property 3.4

Let $q_B^a=(q_1^a,q_2^a,...,q_r^a)$ denote the corresponding output vector of B-group for a reachable input vector p_B^a. If q_i is observable and the sensitized input vector depends on $q_1, q_2, ..., q_{i-1}, q_{i+1}, ..., q_r$, where there exists at least one of q_j's which cannot be pre-determined, then q_i^a is undefined.

Consider an input vector $u^0=(u_1,u_2,u_3,u_4,u_5,u_6,u_7)=(0,0,0,0,0,0,0)$. When it is applied to the target circuit, the corresponding output $y=(N_{24},N_{25},N_{26},N_{27})=(0,0,0,0)$. Here u^0 satisfies the condition in (3.3), and thus, by Property 3.3, $N_{396}=N_{25}=0$. When the input vector is applied to A-group, we obtain (393,394,449,460,486)=(0,1,1,1,1). This implies that, when the inputs (0,1,1,1,1) is applied to B-group, the output $N_{396}=0$ and $N_{26}=0$.

It can be easily verified that (393,394,449,460,486)=(0,0,0,0,0) is unreachable, by Property 3.1, both N_{396} and N_{26} are don't cares. Table 3.1 shows all input combinations of (393,394,449,460,486) and their corresponding outputs (396,26), where all possible 2^7=128 primary input combinations are simulated and the resultant inputs and outputs of B-group are tabulated. Results show that only six combinations are reachable. Therefore, the Boolean expressions of B-group can be derived from Table 1 as follows:

$$N_{396} = \overline{N}_{394}N_{393} + \overline{N}_{460}$$

$$N_{26} = \overline{N}_{393}\overline{N}_{394} + N_{393}N_{394}$$

Table 3.1: Simulation Results of Example Circuit I

393	394	449	460	486	396	26
0	0	0	0	0	X	X
0	0	0	0	1	X	X
0	0	0	1	0	X	X
0	0	0	1	1	X	X
0	0	1	0	0	X	X
0	0	1	0	1	X	X
0	0	1	1	0	X	X
0	0	1	1	1	0	1
0	1	0	0	0	X	X
0	1	0	0	1	X	X
0	1	0	1	0	X	X
0	1	0	1	1	X	X
0	1	1	0	0	X	X
0	1	1	0	1	1	0
0	1	1	1	0	X	X
0	1	1	1	1	0	0
1	0	0	0	0	X	X
1	0	0	0	1	X	X
1	0	0	1	0	X	X
1	0	0	1	1	1	0
1	0	1	0	0	X	X
1	0	1	0	1	X	X
1	0	1	1	0	X	X
1	0	1	1	1	X	X
1	1	0	0	0	1	1
1	1	0	0	1	X	X
1	1	0	1	0	0	1
1	1	0	1	1	X	X
1	1	1	0	0	X	X
1	1	1	0	1	X	X
1	1	1	1	0	X	X
1	1	1	1	1	X	X

Apparently, the developed B-group may not have exactly the same topological structure as the original one. However, it can be easily verified that the circuit with the developed B-group has the same functionality as the original one. In fact, it is not necessary for the developed B-group to have the same functionality as the original B-group as long as the functionality of the circuit with the developed B-group and the original circuit are the same. This leads to a way of identifying redundant nodes and gates. For example, the above expressions require only three inputs, i.e., 393,394,460. Thus, nodes 486 and 449 are redundant, so are the corresponding NAND2 and INV1 gates.

Discussion:

Properties 3.1 and 3.2 determine don't care outputs of B-group, while Property 3.3 defines the corresponding output values. If the inputs/outputs of B-group can be completely determined by Properties 3.1, 3.2, and 3.3, as in the example circuit I shown in Figure 3.2, then the target circuit is redesignable. By ***Assumption 1***, the functionality of the target circuit is given. This implies that the target circuit is always redesignable. The worst case is to redesign the target circuit based on the given functionality. We suppose that this redesign solution is costly. Therefore, by the term "redesignable" we mean that the circuit can be redesigned at a reasonably low cost. In other words, the redesign solution can be found from the partitioned groups. On the other hand, the term "unredesignable" implies that the redesign solution cannot be generated from the current partitioned groups.

The redesign problem can be solved using the digital test generation techniques such as Boolean difference, path sensitization, backtracking methods [11]. More specifically, the primary input vectors of A-group that reach an input vector of B-group are called

controllable primary input vectors, for short, and the reachable input vectors of B-group are referred to as *reachable input vectors*. In this implementation, all reachable and unreachable input vectors must be derived. For those reachable input vectors, by Property 3.2, we must check if their corresponding outputs of B-group are observable from the controllable primary input vectors. Thus, the redesign problem involves checking the observability of B-group and deriving its reachable and unreachable input vectors. Therefore, one may use exhaustive simulation, Boolean difference, path sensitization, or backtracking method to generate all controllable primary input vectors and all reachable input vectors. Which method is more effective depends upon the number of inputs/outputs in each group and the computation complexity for performing those methods. Hence, in order to simplify the redesign process, it has been developed in various steps, so that the decision whether the target circuit is "redesignable" or not is made as early as possible and with the minimum amount of computation. The different steps of redesign process are discussed in the next chapter.

Chapter 4

DEVELOPMENT

This chapter presents the details of the developed redesign process. First section describes our initial development. Various steps involved in the redesign process are explained using examples. The second section gives the improvements made to different steps in order to improve the overall quality of the redesign process.

4.1 Redesign Methodologies

In this development, the redesign process is comprised of the following three major steps:

1. Redesignability Check
2. Feasibility Check
3. Re-implementation

The first step checks if the circuit is redesignable. If so, the second step determines the cost of finding the functionality of the missing parts. In case it is cost effective to extract the functionality of missing parts, the cost associated with the re-implementation of missing parts is estimated. If it is feasible to re-implement the missing parts the third step extracts the functionality and re-implement them.

4.1.1 Redesignability Check

Consider a reachable input vector p_B^a of B-group and its corresponding output vector $q_B^a=(q_1^a,q_2^a,...,q_r^a)$. By Property 3.3, an output value q_i^a of B-group can be determined from the derived primary outputs and the known primary inputs if the output q_i is observable and the sensitized input vector is independent of q_1, q_2, ..., q_{i-1}, q_{i+1}, ..., q_r. Otherwise, by Property 3.4, q_i^a is undefined. Consider C-group in Figure 3.1(d), the inputs of C-group include all elements in q_B, and possibly some primary inputs, u_{PIC}, and some primary outputs, y_{IC}, that resulted from both A-group and B-group. The outputs of C-group are the primary outputs, y_C. Without loss of generality, say $y_C=\{y_{C1},y_{C2},...,y_{Cm}\}$.

Property 4.1

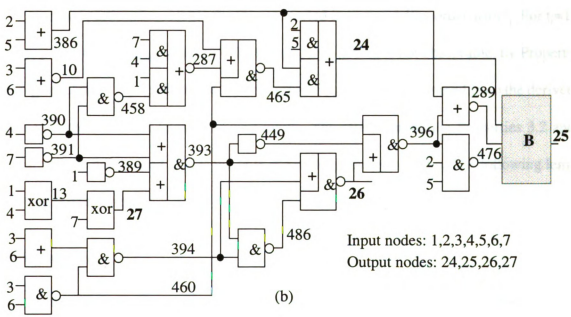
The target circuit is redesignable if $y_C = \varnothing$ or $r = 0$.

Consider the circuit in Figure 1.2, with B-group now including the OAI21 gate with output node 25, as shown in Figure 4.1(a) and (b). Let the circuit be denoted as example circuit II. Then the B-group, as illustrated in Figure 4.1(c), has the input nodes p_B = {24,289,476} and the output nodes y_B = {25} and y_C = $\varnothing$. By Property 4.1, this circuit is redesignable.

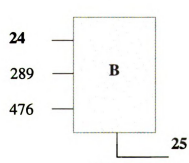
To simplify the redesignability check process, C-group is partitioned into v blocks where $v \leq m$. Two different partitioning schemes are developed: One partitions the C-group based on the input q_j, referred to as *CI-Partitioning Scheme*, while the other is based on the output y_C, referred to as *CO-Partitioning Scheme*. These two schemes are discussed below in detail:

```
.model z4ml
.inputs 1 2 3 4 5 6 7
.outputs 24 25 26 27
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 0.07 0.07
.default_output_load 3.00
.gate or2 a=2 b=5 O=[386]
.gate nor2 a=3 b=6 O=[10]
.gate inv1 a=4 O=[390]
.gate inv1 a=7 O=[391]
.gate nand2 a=[390] b=[391] O=[458]
.gate aoi22 a=[458] b=1 c=4 d=7 O=[287]
.gate nand2 a=3 b=6 O=[460]
.gate oai21 a=[10] b=[287] c=[460] O=[465]
.gate ao22 a=[386] b=[465] c=2 d=5 O=24
.gate xor a=1 b=4 O=[13]
.gate xor a=[13] b=7 O=27
.gate inv1 a=1 O=[389]
.gate oai22 a=27 b=[389] c=[390] d=[391] O=[393]
.gate inv1 a=[393] O=[449]
.gate or2 a=3 b=6 O=[452]
.gate nand2 a=[452] b=[460] O=[394]
.gate nand2 a=[393] b=[394] O=[486]
.gate oai21 a=[393] b=[394] c=[486] O=26
.gate oai21 a=[449] b=26 c=[460] O=[396]
.gate nor2 a=[386] b=[396] O=[289]
.gate nand3 a=[396] b=2 c=5 O=[476]
.end
```

(a)



(b)



(c)

Figure 4.1: Example circuit II, with Incomplete Implementation Information: (a) netlist; (b) schematic; and (c) B-group.

CI-Partitioning Scheme

In this scheme, C-group is partitioned into v blocks, as illustrated in Figure 4.2(a). Each block C_i, as shown in Figure 4.2(b), may contain s_i primary inputs, say $u_{PICi}=\{u_{PICij}, j=1,2,...,s_i\} \subseteq u_{PIC}$, where $s_i \geq 0$; z_i primary outputs produced from A-group and B-group, $y_{ICi}=\{y_{ICij}, j=1,2,...,z_i\}$, where $z_i \geq 0$; and t_i outputs of q_B, $Q_{Ci}=\{q_{Cij}, j=1,2,...,t_i\} \subseteq q_B=\{q_1,q_2,..., q_r\}$, where $t_i \geq 1$ and $t_i \leq r$. $y_{Ci} \subseteq y_C$, where n_{yCi} is the number of primary outputs in y_C and $n_{yCi} \geq 1$.

After partitioning, all blocks C_j's are sorted in an ascending order with t_j. For $t_i=1$, i.e., $y_{Ci}=G_{Ci}(u_{Ci};\{q_1,q_2,...,q_r\};y_{ICi})=G_{Ci}(u_{Ci};\{q_i\};y_{ICi})$, if q_i is not observable, by Property 3.2, q_i^a is a "don't care"; otherwise, by Property 3.3, q_i^a can be determined from the derived primary outputs and the known primary inputs. Therefore, combining Properties 3.2 and 3.3, q_i^a can be determined if q_i is the only unknown parameter in Q_{Ci}. The following lemmas and property result:

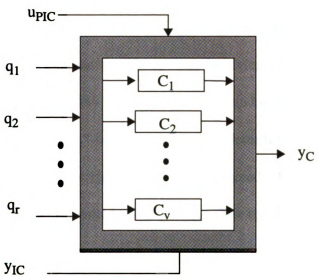
Lemma 4.1

(a) If q_i is the only unknown parameter in Q_{Ci}, then q_i^a can be determined.

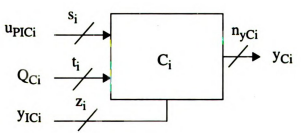
(b) If $Q_{Ci}=\{q_i\}$, i.e., $t_i=1$, then q_i^a can be determined.

Property 4.2

Let $C=\{C_1,C_2,..., C_m\}$ and $q_B=\{q_1,q_2,...,q_r\}$. If m = r, the target circuit is redesignable.



(a)



(b)

Figure 4.2: CI-Partitioning: (a) Blocks; (b) Block C_i.

Proof

m = r implies that $t_i = 1$ for all blocks C_i's. By Lemma 4.1(b), q_i^a can be determined and thus the circuit is redesignable.

Consider a block C_j with $t_j=2$, and let $Q_{Cj}=\{q_{Cj1},q_{Cj2}\}=\{q_{j1},q_k\}$. If $q_k \in Q_{Ck}$ and $t_k=1$, i.e., $Q_{Ck}=\{q_k\}$, by Lemma 4.1(b), q_k^a can be determined. Thus, q_{j1} is the only unknown parameter q_{j1} in Q_{Cj}. By Lemma 4.1(a), q_j^a can be determined. Lemma 4.1 can be generalized as follows. Let t_i^* be the number of undetermined q_i^a's. Initially, $t_i^*=t_i$. If $Q_{Ci}=\{q_i\}$, i.e., $t_i=1$, by Lemma 4.1(b), q_i^a can be determined. Once q_i^a is determined, t_j^* is updated for all j, i.e., t_j^* decrements by 1 if $q_i \in Q_{Cj}$. The updating process is summarized in Algorithm 1, and Property 4.3 results:

Property 4.3

Let t_i^* be the number of undetermined q_i^a's in Q_{Ci} with the updating process in Algorithm 1, if $t_i^*=1$ for all i, then the target circuit is redesignable.

To describe the above redesignability check, the target circuit in Figure 1.2 is again considered, but B-group is changed to include the OAI21 gate with output node 465, the INV1 gate with output node 449, and the AND2 gate with the output node 486. Thus, the B-group for this example circuit III, as illustrated in Figure 4.3(a), has the input nodes {10,287,460,393,394} and the output nodes {465,449,486}. As shown in Figure 4.3(b), C-group includes the inputs $u_{PIC}=\{u_1,u_2,u_3,u_4,u_5,u_6,u_7\}$, $Q_C=\{465,449,486\}$, and $y_{IC}=\{27\}$, and the outputs $y_C=\{24,25,26\}$. C-group is partitioned into three sub-groups, C_1, C_2, and C_3, as shown in Figure 4.3(c), with the schematic circuit diagrams in Figure 4.3(d), each has $t_i=1$. Therefore, by Property 4.2, the circuit is redesignable.

Algorithm 1

Step 0: Let $C=\{C_1,C_2, .., C_m\}$ and $q_B=\{q_1,q_2,...,q_r\}$;

Initialize $TS[i] = t_i$, for i=1,2,...,r.

Step 1: (Updating)

1.1: count := 0;

1.2: FOR j=1 TO r

IF TS[j]=1 THEN {count++; param[count]=j;}

1.3: IF (count = 0) GOTO Step 2;

FOR k=1 TO count

TS[param[j]]--;

GOTO Step 1.1;

Step 2: $t_i^* = TS[i]$, for i=1,2,...,r.

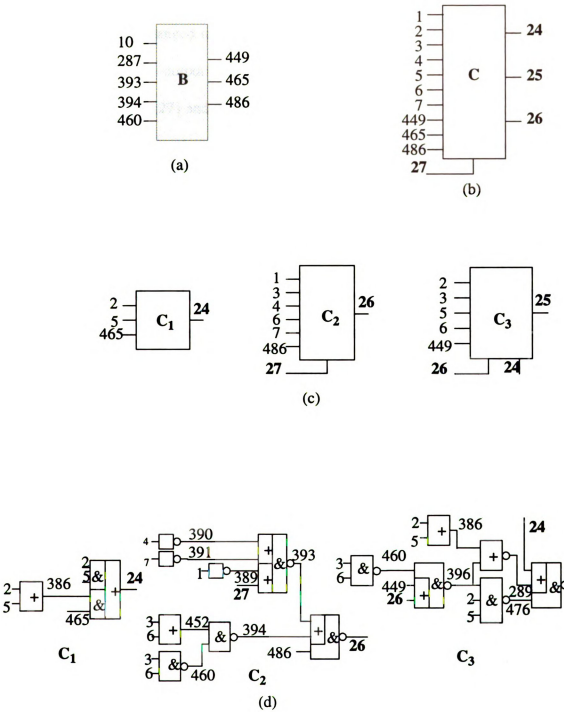


Figure 4.3: Example Circuit III: (a) B-group; (b) C-group; (c) Partitioned Blocks C_1,C_2,C_3 and (d) Circuit Schematics for C_1, C_2, C_3.

Consider another example circuit IV obtained from the target circuit in Figure 1.2. Here the B-group is changed to include the AOI22 gate with output node 287 and the OAI22 gate with the output node 393. Therefore, B-group has the input nodes {1,4,7,389,390,391,458,27} and the output nodes {287,393} as shown in Figure 4.4(a). C-group, illustrated in Figure 4.4(b), includes the inputs $u_{PIC}=\{u_2,u_3,u_5,u_6\}$, $Q_C=\{287,393\}$, and the outputs $y_C=\{24,25,26\}$. Using CI-partitioning, C-group is partitioned into two sub-groups, C_1, and C_2, as shown in Figure 4.4(c). Since each block has $t_i=1$, therefore, by Property 4.2, the circuit is redesignable.

Property 4.3 shows that the circuit is redesignable if $t_i^*=1$ for all i, after performing the updating process in Algorithm 1. On the other hand, if $t_i^* > 1$ for some i, the circuit is unredesignable with the present circuit partitioning scheme. Since a circuit may be partitioned in many different ways, the circuit is unredesignable in the present partitioning scheme, but it may be redesignable with other partitioning schemes. As mentioned previously, all circuits are redesignable if the redesign cost is acceptable. With the present circuit partitioning scheme, the following unredesignability checks result:

Property 4.4

Let t_i^* be the number of undetermined q_i^a's in Q_{Ci} with the updating process in Algorithm1, if there exists at least one $t_i^*>1$, then the target circuit is unredesignable.

Property 4.5

Let $C=\{C_1,C_2,...,C_m\}$ and $q_B=\{q_1,q_2,...,q_r\}$. If $m < r$, the target circuit is unredesignable.

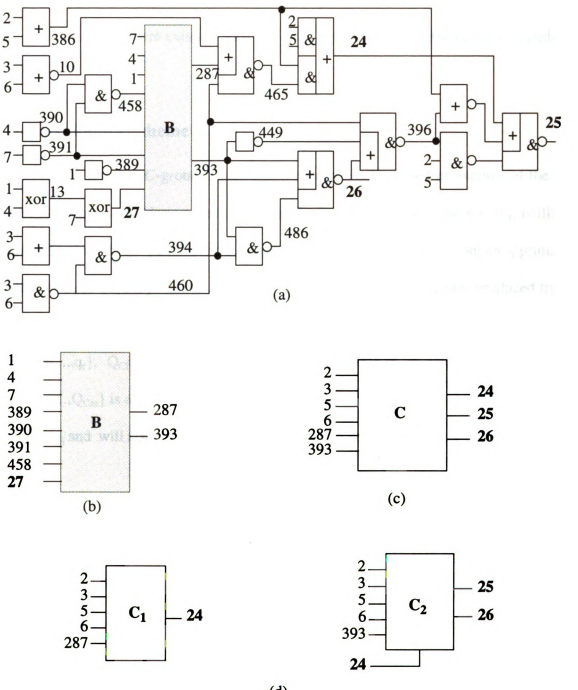


Figure 4.4: (a) Example Circuit IV: (b) B-group; (c) C-group; and (d) Partitioned Blocks C_1,C_2.

Proof

If m < r, then there exists at least one ti*>1, by Property 4.4, the circuit is unredesignable.

CO-Partitioning Scheme

In this scheme, C-group is partitioned into m blocks, based on outputs of the C-group, y_C. Again considering the C-group in Figure 3.1(d), the CO-partitioning is illustrated in Figure 4.5(a). Each block C_i, as shown in Figure 4.5(b), may contain s_i primary inputs, say $u_{PICi}=\{u_{PICij}, j=1,2,...,s_i\} \subseteq u_{PIC}$, where $s_i \geq 0$; z_i primary outputs produced from A-group and B-group, $y_{ICi}=\{y_{ICij}, j=1,2,...,z_i\}$, where $z_i \geq 0$, and t_i outputs of $q_B=\{q_1,q_2,...,q_r\}$, $Q_{Ci}=\{q_{Cij}, j=1,2,...,t_i\} \subseteq q_B$, where $0 \leq t_i \leq r$. Another set $Q_C*=\{Q_{C1},Q_{C2},...,Q_{Cm}\}$ is also defined that is used to keep information about the redesignability of each Q_{Ci} and will also be used later in section 4.2 to improve redesignability of a target circuit.

To describe CO-partitioning scheme, the target circuit in Figure 1.2 is again considered, but B-group is changed to include different gates, so that various cases can be considered that are possible during partitioning. As already mentioned, the first step is to partition C-group into m blocks based on its outputs. The partitioned m blocks are then divided into following three categories:

(a) Blocks with t_i=0

Consider the target circuit in Figure 1.2, but for this example circuit V, B-group now includes the OAI21 gate with output node 465, and the AND2 gate with output node

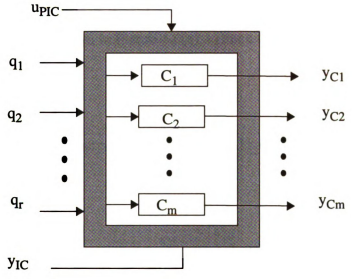


Figure 4.5: CO-Partitioning: (a) Blocks; (b) Block C_i.

486. Thus, Figure 4.6(a) shows that the B-group has the input nodes p_B = {10,287,460,393,394} and the output nodes q_B = {465,486}. As shown in Figure 4.6(b), C-group includes the inputs u_{PIC}={u_1,u_2,u_3,u_4,u_5,u_6,u_7}, Q_C={465,486}, and y_{IC}={27}, and the outputs y_C={24,25,26}. According to the partitioning scheme, C-group is partitioned into 3 blocks, C_1, C_2, and C_3, as shown in Figure 4.6(c). Block C_3 has t_3= 0.

(b) Blocks with t_i=1

Blocks C_1, and C_2 of Figure 4.6(c) have t_1= 1 and t_2= 1.

(c) Blocks with t_i>1

Again consider the target circuit in Figure 1.2. The B-group now includes the AOI22 gate with output node 287, the OAI22 gate with output node 393, the NAND2 gate with output node 458, the INV1 gate with output node 390, the INV1 gate with output node 391, and the NOR2 gate with output node 10. Figure 4.7(a) shows that the B-group has input nodes {u_1,u_3,u_4,u_6,u_7,27} and the output nodes {10,287,393}. As shown in Figure 4.7(b), C-group includes the inputs u_{PIC}={u_2,u_3,u_5,u_6}, Q_C={10,287,393}, and y_{IC}={27}, and the outputs y_C={24,25,26}. C-group is then partitioned into 3 blocks, C_1, C_2, and C_3, as shown in Figure 4.7(c). Block C_1 has t_1=2.

In the first step of this scheme of redesignability check, the blocks with t_i=0 are considered. Since such blocks do not provide useful information for the redesign process, therefore these are eliminated from Q_C* resulting in Q_C*={Q_{C1},Q_{C2},...,Q_{Cm^*}} where m*≤ m.

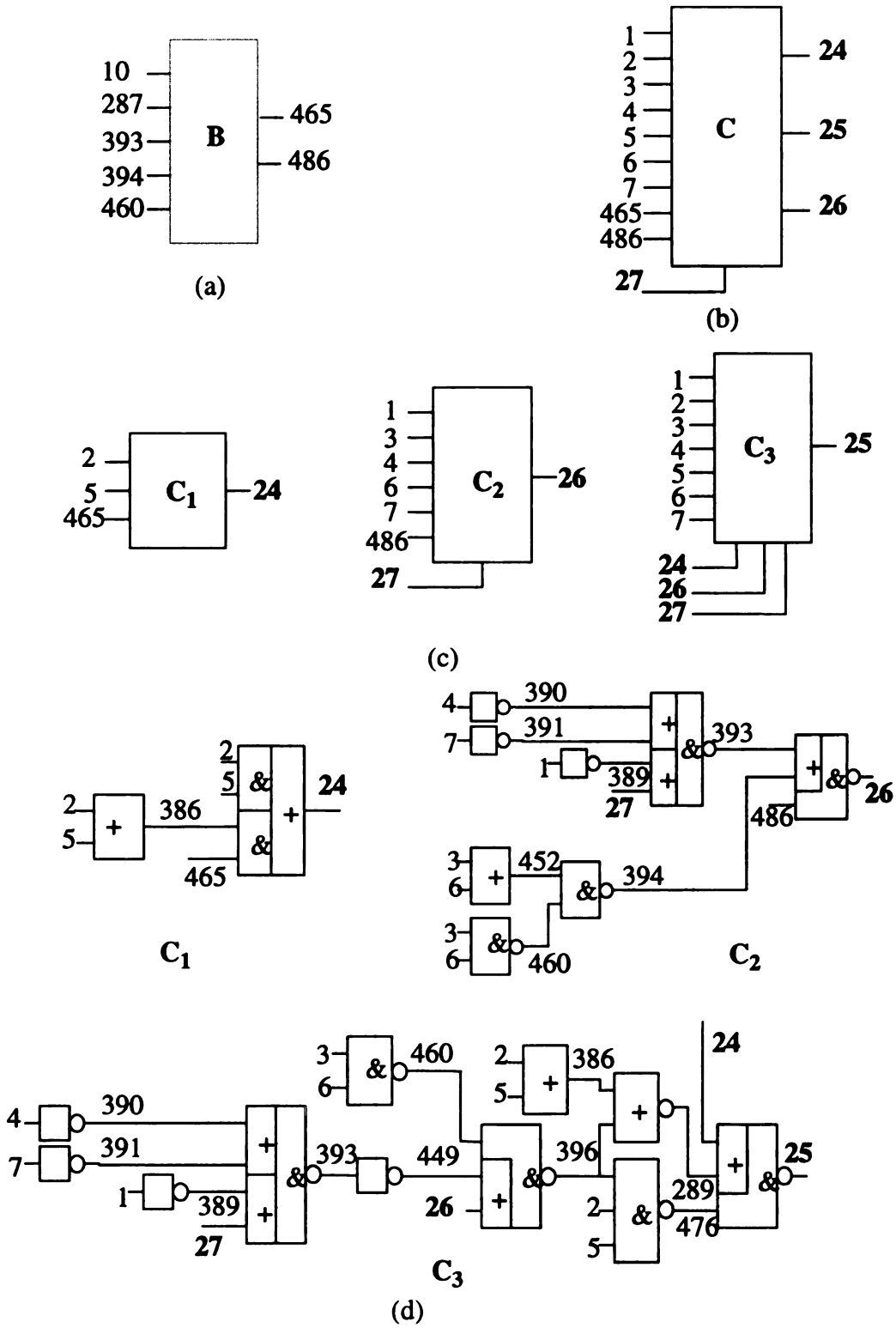


Figure 4.6: Example Circuit V: (a) B-group; (b) C-group; (c) Partitioned Blocks C_1,C_2,C_3 and (d) Circuit Schematics for C_1, C_2, and C_3.

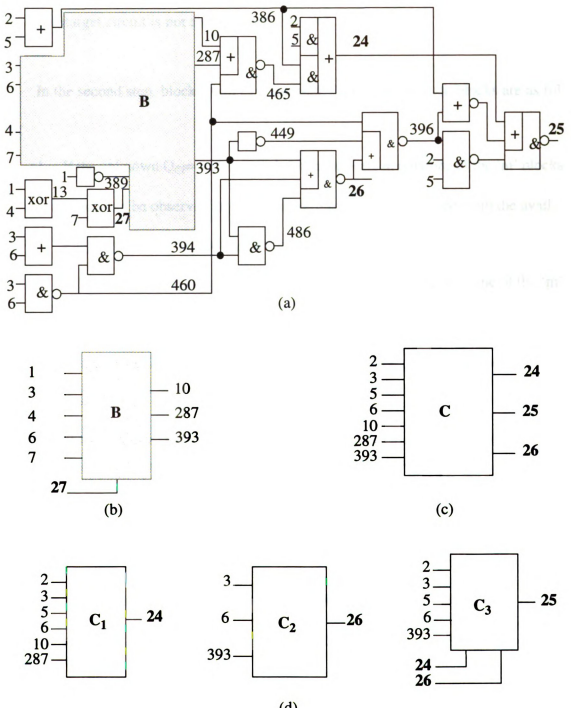


Figure 4.7: (a) Example Circuit VI: (b) B-group; (c) C-group; and (d) CO-Partitioned Blocks C_1,C_2,C_3.

Property 4.6

The target circuit is not redesignable if $r > m^*$.

In the second step, blocks with $t_i=1$ are used. The rules for such blocks are as follows:

1. If the unknown $Q_{Ci}=\{q_x\}$, where $1 \leq x \leq r$, appears in only one of the 'm' blocks then it can be observed at y_{Ci} and therefore can be determined from the available information. Hence Q_{Ci} is eliminated from Q_C^*.
2. If the unknown $Q_{Ci}=\{q_x\}$, where $1 \leq x \leq r$, appears in more than one of the 'm' blocks and each block Q_{Cj} containing q_x has $t_j = 1$, then q_x can be observed at y_{Ci} and y_{Cj}'s. This implies that the unknown can be determined and the corresponding Q_{Ci} and Q_{Cj}'s are removed from Q_C^*.
3. If the unknown $Q_{Ci}=\{q_x\}$, where $1 \leq x \leq r$, appears in more than one of the 'm' blocks and Q_{Cj}'s containing q_x has $t_j>1$, then all sensitized paths U_{xi} for $q_x \in Q_{Ci}$ and U_{xj} for $q_x \in Q_{Cj}$ are found. If $U_{xi} \supseteq U_{xj}$, this means that q_x can be determined from block Q_{Ci} by observing it at y_{Ci}. Hence q_x becomes a known for Q_{Cj} and therefore t_j is decremented for corresponding Q_{Cj}'s and Q_{Ci} is eliminated from Q_C^*.

The above two steps of redesignability check are performed successively and the

circuit is redesignable if $Q_C^* = \varnothing$ is obtained.

Property 4.7

The target circuit is not redesignable with the available information if $Q_C^* \neq \varnothing$ after applying the redesignability check.

Corollary 4.7.1:

The target circuit is not redesignable with the available information if all the partitioned C_i where $1 \leq i \leq m$ have $t_i > 1$.

This redesignability check is summarized in Algorithm 2.

Discussion:

The two schemes presented above provide simple rules for redesignability check and determine whether the circuit is unredesignable in the present partitioning scheme. So after partitioning the target circuit, the redesignability check will be performed. If the circuit passes the check, the next step will be the feasibility check. However, if the circuit fails the check alternative partitioning schemes will be tried as discussed later in this chapter.

4.1.2 Feasibility Check

Once a target circuit passes the redesignability check, this implies that the outputs of B-group, $q_B=(q_1,q_2,...,q_r)$, are observable. In feasibility check, we determine whether it is cost effective to determine the inputs/outputs relationship of B-group and then to re-implement it. In this part of the redesign process, first reachability of the inputs of B-group, $p_B = (p_1,p_2,...,p_w)$, is determined using A-group. In the example circuit I, given in Chapter

Algorithm 2

Step 0: Set $Q_C^* = \{Q_{C1}, Q_{C2}, \ldots, Q_{Cm}\}$

Step 1: Set count = 0 (Used for # of Q_{Ci} with ti > 1)

Select Q_{Ci} with $t_i = 0$;

$Q_C^* = Q_C^* - \{ Q_{Ci} \mid t_i = 0\}$

m = m - # of $\{ Q_{Ci} \mid t_i = 0\}$

Step 2: If r > m, go to Step 6

Step 3: Set i = 1

3.1: If $t_i \neq 1$ go to Step 5

3.2: If $Q_{Ci} \cap Q_{Cj} = \emptyset$, $\forall j \neq i$, go to Step 4

3.3: If $t_j = 1$, for j's with $Q_{Ci} \cap Q_{Cj} \neq \emptyset$, go to Step 4

3.4: If $\{ U_{xj} \mid$ for j's with $Q_{Ci} \cap Q_{Cj} \neq \emptyset$ & tj > 1$\} \notin U_{xi}$, go to Step 5

Step 4: r = r – 1

$t_j = t_j$ -1 for j with Qci ∩ Qcj ≠ ∅

If r = 0, circuit is redesignable [STOP] else go to Step 1

Step 5: count = count + 1

If count ≠ m, i = i + 1, go to Step 3.1

Step 6: The circuit is not redesignable [STOP]

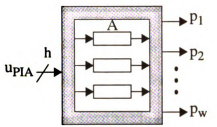
3, all reachable input vectors of B-group were derived using exhaustive simulations, i.e., simulate the circuit in A-group with all possible combinations of the primary inputs. As the number of inputs and outputs of A-group increase computational complexity becomes a critical issue. Therefore, a cost function has been defined associated with functionality extraction based on the number of inputs and outputs of A-group. The redesign process is continued only if it is cost effective, otherwise we say that it is not feasible to redesign the circuit.

To reduce the computational complexity, a simple circuit partitioning scheme is proposed in this implementation. As illustrated in Figure 3.1(d), A-group takes some primary inputs u_{PIA} and produces the outputs $\{p_1,p_2,...,p_w\}$ as the inputs of B-group and some primary outputs y_A. If we assume that the number of primary inputs in u_{PIA} is k, then 2^k logic simulations on A-group are needed. Thus the number of simulations can be reduced if A-group can be partitioned into many smaller blocks.

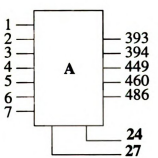
The partitioning of A-group is performed in two steps. In the first step A-group is partitioned to contain only those gates which contribute to the inputs of B-group $\{p_1,p_2,...,p_w\}$, as shown in Figure 4.8(a). The number of primary inputs in A-group will now be 'h' where $h \leq k$.

Consider the A-group, shown in Figure 4.8(b), for the example circuit I discussed in Chapter 3. The A-group can be partitioned as described above. Figure 4.8(c) shows that A-group will contain only {393,394,449,460,486} as outputs, which will result in $u_{PIA}=\{u_1,u_3,u_4,u_6,u_7\}$. The number of simulations has decreased from 2^7(=128) to 2^5(=32), resulting in significant reduction of computational complexity.

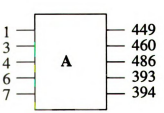
In the second step, let A-group, obtained after first step, be partitioned into g blocks,



(a)



(b)



(c)

Figure 4.8: A-group Partitioning: (a) First Step;
Example Circuit I: (b) Before Partitioning;
and (c) After Partitioning.

denoted by $A_1,A_2,...,A_g$. Each block A_i, as shown in Figure 4.9(a), may take α_i primary inputs, say $u_{Ai}=\{u_{Aij}, j=1,2,..., \alpha_i\}\in u_{PIA}$, where $1 \leq \alpha_i \leq h$, and produce some outputs $p_{Ai}\in \{p_1,p_2,...,p_w\}$. If $u_{Ai}\cap u_{Aj}=\varnothing$, for any i and j, then $h=\alpha_1+\alpha_2+...+\alpha_g$. Thus the number of simulations required is reduced from 2^h to $(2^{\alpha_1}+2^{\alpha_2}+...+2^{\alpha_g})$. Practically, however, we may partition the A-group into many smaller groups such that $(\alpha_1+\alpha_2+...+\alpha_g)$ is minimum.

Consider the A-group, shown in Figure 4.9(b), for the example circuit III given in Figure 4.2. It takes the inputs $u_{PIA}=\{u_1,u_3,u_4,u_6,u_7\}$ and produces the outputs $p_A=\{10,287,460,393,394\}$. According to the partitioning scheme, A-group is partitioned into two blocks, A_1 and A_2 as shown in Figure 4.9(c) and (d) respectively, where $u_{A1}=\{u_3,u_6\}$ and $u_{A2}=\{u_1,u_4,u_7\}$. The number of simulations is reduced from $2^5(=32)$ to $2^3+2^2(=12)$. Thus, the computational complexity has reduced considerably.

A number $n_{uPIA}(MAX)$ has been defined that determines the maximum allowable number of inputs in any of the partitioned block of A-group. The choice of $n_{uPIA}(MAX)$ is variable depending upon the available computing facilities and will be decided by the designers. If the number of inputs in a single A-group partition exceeds $n_{uPIA}(MAX)$, we say that the solution is not cost effective. However, it should be noted here that we can also apply backtracking technique used in ATPG's to find reachable vectors of the B-group. Since simulation is faster compared to backtracking process, the criterion used here assumes backtracking to be fifty percent efficient compared to the simulation. Hence, back-tracking can also make the solution cost effective in a case where the number of inputs of a partitioned block exceed $n_{uPIA}(MAX)$, but the number of outputs of that block is less than $n_{uPIA}(MAX)/2$. Let n_{uAi} be the number of inputs and $n_{(pAi + yAi)}$ be the number of outputs of each block A_i, then we have the following property:

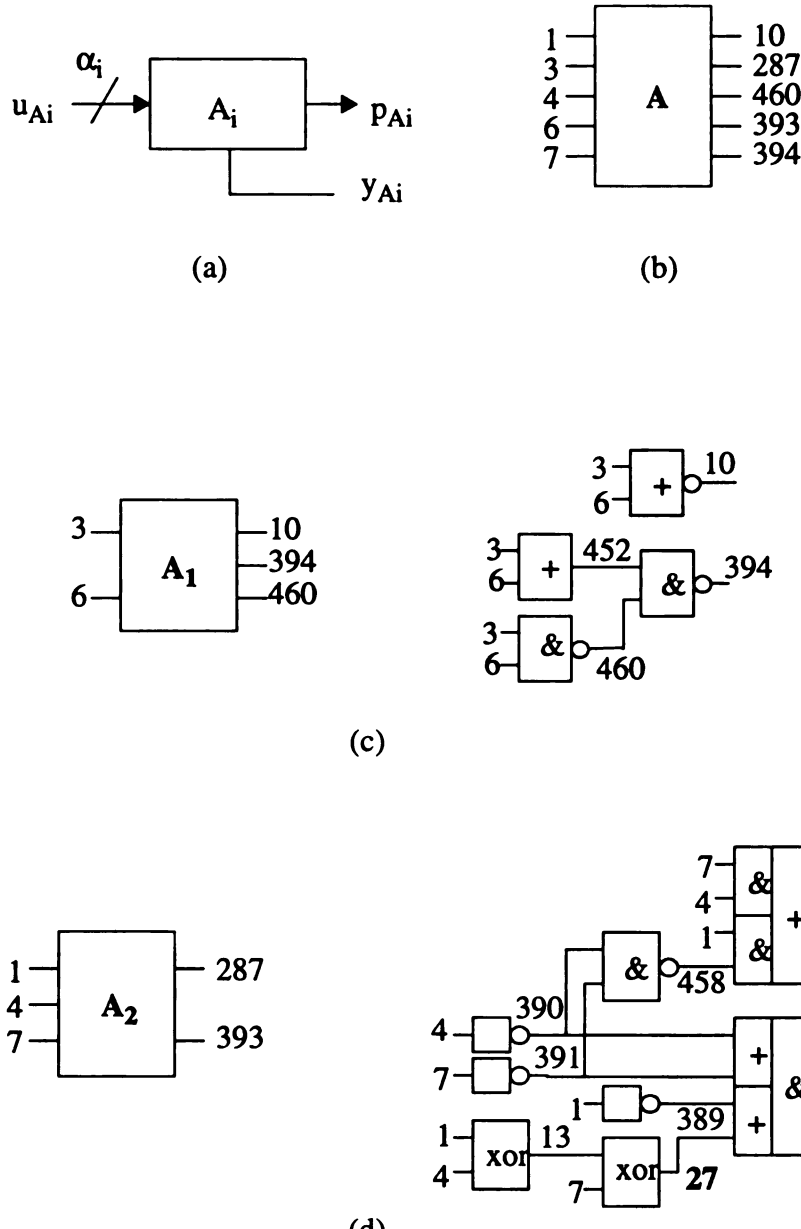


Figure 4.9: A-group Partitioning: (a) Second Step, Block A_i;
Example Circuit III: (b) A-group; (c) Block A_1; and (d) Block A_2.

Property 4.8

The redesign solution is not cost effective if there exists a single partitioned block A_i with $n_{uAi} > n_{uPIA}(MAX)$ and $n_{(pAi + yAi)} > (n_{uPIA}(MAX)/2)$.

After it is determined that redesign solution is cost effective, the next step is to find the reachable input vectors of B-group. In order to efficiently find the reachable vectors both simulation and backtracking are employed. Consider the two blocks, A_1 and A_2 as shown in Figure 4.9(c) and (d) respectively, we have already seen that partitioning has reduced number of simulations from 32 to 12. Since the number of outputs in block A_2 are 2, we shall require 2^2 backtrackings. So, by using simulation for block A1 and backtracking for block A2, the reachable input vectors can be found using 4 simulations and 4 backtrackings.

The choice of using either simulation or backtracking for a partitioned block of A-group depends on the number of its inputs and outputs. As discussed earlier, simulation is faster compared to backtracking process, the criterion used specifies that if the number of outputs in block A_i is less than half the number of its inputs, then use of backtracking is more efficient compared to simulation. So for each partitioned block of A-group, either simulation or backtracking is employed which ever is efficient in that particular case.

Algorithm 3 summarizes the procedure for finding reachable input vectors of B-group.

After finding the reachable input vectors, feasibility check determines the cost of re-implementing the circuit. The critical issue involved in the re-implementation of missing

Algorithm 3

Step 1: Partition A-group to contain only $\{p_1,p_2,...,p_w\}$ as outputs

Step 2: Partition A-group obtained in Step 1 into many smaller groups such that $(\alpha_1+\alpha_2+...+\alpha_g)$ is minimum

Step 3: IF $n_{uPIAi} \leq n_{uPIA}(MAX)$ for all i THEN go to Step 4

3.1: IF $n_{(pAi+yAi)} > (n_{uPIA}(MAX)/2)$ for i with $n_{uPIAi} > n_{uPIA}(MAX)$ THEN go to Step 6

Step 4: FOR i=1 TO g

IF $n_{uPIAi} \leq 2*n_{(pAi+yAi)}$ simulate partitioned block

ELSE backtrack partitioned block

Step 5: Find reachable input vectors $p_B=(p_1,p_2,...,p_w)$ [STOP]

Step 6: Redesign solution is not cost effective [STOP]

Algorithm 4

Step 1: If $Q_{Ci} \cap Q_{Cj}=\varnothing$, $\forall j \neq i$, where $Q_{Ci} \cap Q_{Cj} \in Q_C^*$, $Q_{Ci}'=Q_{Ci}$, else merge blocks i and j for those j's where $Q_{Ci} \cap Q_{Cj} \neq \varnothing$ to get Q_{Ci}'.

Step 2: Rearrange blocks of C-group based on the Q_{Ci}'s obtained in Step 1

Step 3: Partition each block C_i', so that it contains only one element in each Q_{Ci}'

Step 4: Modify the B-group and C-group accordingly, the circuit now becomes redesignable

parts is the number of reachable input vectors of B-group. Whether one can efficiently handle implementation of a large circuit primarily depends on the available design tools and computing resources. Another factor of consideration will be the necessity of the task. If there are no alternatives one will have to do it. So keeping these points in mind, the designers will decide whether it is feasible to re-implement the missing parts or not. A number $INPUT\_VECTOR_{MAX}$ has been defined that determines the maximum number of reachable input vectors in order to have the re-implementation feasible. If the reachable input vectors of B-group exceed $INPUT\_VECTOR_{MAX}$, we say that re-implementation is not feasible. The following property concludes:

Property 4.9

If the reachable input vectors of B-group exceed $INPUT\_VECTOR_{MAX}$, then re-implementation of the target circuit is not feasible.

Discussion

In the feasibility check, we have introduced partitioning scheme for A-group and the use of backtracking technique in order to efficiently determine the reachable input vectors of B-group. Feasibility check first determines whether finding the reachable input vectors is cost effective. If so it determines the reachable vectors and checks the feasibility of re-implementing the circuit. This check halts the redesign process if finding reachable vectors is not cost effective or re-implementation is not feasible. However, if the target circuit passes the feasibility check, we proceed to the next step of re-implementation.

4.1.3 Re-implementation

Once a circuit passes the redesignability and feasibility check, it is determined that the outputs of B-group are observable and functionality extraction along with re-implementation is cost effective. We can find the relation between inputs and outputs of B-group using different approaches. Here, two such approaches are presented that can be used to derive the transfer function of B-group. The selection of approach depends on the number of inputs and outputs of A, B and C-groups of the target circuit. After deriving the transfer function, the B-group can be re-implemented using appropriate minimization and synthesis tools. This implementation uses VLSI design tools, *espresso* and *sis*. The two approaches to determine the transfer function of B-group are summarized below:

Approach 1

In this approach, first A-group is solved to find all reachable combinations of inputs of B-group. For each reachable combination, the outputs of B-group are then sensitized to be observed at the primary outputs. This approach is suitable for circuits in which the function for the primary output vectors y_C is complex. The following steps are followed:

1. Simulate/Backtrack A-group, to find the reachable combinations of $p_B = \{p_1,p_2,...,p_w\}$. Find all the primary input vectors that generate each reachable input of B-group.
2. Find sensitized path for each $q_i \in q_B=\{q_1,q_2,...,q_r\}$, in C-group for each reachable combination of $p_B = \{p_1,p_2,...,p_w\}$.
3. Redesign B-group.

Consider example circuit III shown in Figure 4.3. We shall use this approach to re-implement the missing parts. The first step gives us only 6 reachable vectors out of the 32 possible combinations of the inputs of B-group. So the outputs of B-group for these 26 combinations will be don't care i.e. 'X'. In the second step, using the modified D-algorithm discussed in Chapter 2, we find out that B-group output node 465 is observable for all the 6 reachable input vectors while node 486 and 449 are observable only for 5 and 2 reachable input vectors respectively. The results obtained are given in the form of a truth table for B-group in Table 4.1. Using this truth table the B-group is re-implemented and the results are as follows:

$$465 = \overline{(460(10 + 287))}$$

$$486 = \overline{(393)(394)}$$

$$449 = \overline{393}$$

The netlist, generated by *sis*, for this redesigned B-group is exactly the same as that of the original B-group and replacing the missing block with this netlist results in the original entire circuit without missing information.

Approach 2

In this approach, first C-group is solved to find all sensitized paths for outputs of B-group. A-group is then simulated/backtracked under the conditions obtained for sensitized paths. This approach is suitable for circuits in which the function for the primary output vector y_C is simple and the number of inputs and outputs of A-group are large compared to the number of outputs of B-group. The steps involved are outlined below:

1. Find all sensitized paths for each $q_i \in q_B = \{q_1, q_2, ..., q_r\}$, in C-group.

Table 4.1: Truth Table for B-group of Example Circuit III

10	287	460	393	394	449	465	486
0	0	0	0	0	X	X	X
0	0	0	0	1	X	X	X
0	0	0	1	0	X	X	X
0	0	0	1	1	X	1	0
0	0	1	0	0	X	X	X
0	0	1	0	1	X	X	X
0	0	1	1	0	0	1	1
0	0	1	1	1	X	X	X
0	1	0	0	0	X	X	X
0	1	0	0	1	X	1	1
0	1	0	1	0	X	X	X
0	1	0	1	1	X	X	X
0	1	1	0	0	X	0	X
0	1	1	0	1	X	X	X
0	1	1	1	0	X	X	X
0	1	1	1	1	X	X	X
1	0	0	0	0	X	X	X
1	0	0	0	1	X	X	X
1	0	0	1	0	X	X	X
1	0	0	1	1	X	X	X
1	0	1	0	0	X	X	X
1	0	1	0	1	X	X	X
1	0	1	1	0	X	X	X
1	0	1	1	1	X	0	0
1	1	0	0	0	X	X	X
1	1	0	0	1	X	X	X
1	1	0	1	0	X	X	X
1	1	0	1	1	X	X	X
1	1	1	0	0	X	X	X
1	1	1	0	1	1	0	1
1	1	1	1	0	X	X	X
1	1	1	1	1	X	X	X

2. Simulate/Backtrack A-group for $u \in u_{PIA}$, which satisfy the condition of sensitized paths, found in 1.

3. Redesign B-group.

Consider example circuit I given in Figure 1.3. We shall use the above approach to re-implement the missing parts. The first step is to find all sensitized paths for outputs of B-group. We shall use Boolean difference for this purpose. Considering the C-group of the circuit and using the expression describing N_{24}, we can write

$$F = N_{25} = \bar{u}_3\bar{u}_6 (N_{396} (\bar{u}_5 + \bar{u}_2) + u_2 \bar{u}_5 + \bar{u}_2 u_5) + N_{396} (\bar{u}_2 \bar{u}_5 + u_2 u_5)$$
$$+ (\bar{u}_6 + \bar{u}_3) (N_{396} (\bar{u}_5 + \bar{u}_2) + u_2 \bar{u}_5 + \bar{u}_2 u_5) (\bar{u}_7 (\bar{u}_4 + \bar{u}_1) + \bar{u}_1 \bar{u}_4)$$

$$F_{396}(0) = \bar{u}_3\bar{u}_6 (u_2 \bar{u}_5 + \bar{u}_2 u_5) + (\bar{u}_6 + \bar{u}_3) (u_2 \bar{u}_5 + \bar{u}_2 u_5) (\bar{u}_7 (\bar{u}_4 + \bar{u}_1) + \bar{u}_1 \bar{u}_4)$$

$$F_{396}(1) = \bar{u}_3\bar{u}_6 (\bar{u}_5 + \bar{u}_2) + (\bar{u}_6 + \bar{u}_3) (\bar{u}_5 + \bar{u}_2) (\bar{u}_7 (\bar{u}_4 + \bar{u}_1) + \bar{u}_1 \bar{u}_4) + \bar{u}_2 \bar{u}_5 + u_2 u_5$$

$$dF/dN_{396} = F_{396}(0) \oplus F_{396}(1)$$
$$= \bar{u}_2 \bar{u}_5 + u_2 u_5$$

The above result shows that node 396 is observable at primary output node 25 when $u_2 = u_5$. After this step, the reachable input vectors are determined by simulating A-group for those primary input vectors for which $u_2 = u_5$. The results obtained are the same as already given in Table 3.1. The re-implemented B-group is given below:

$$N_{396} = \bar{N}_{394}N_{393} + \bar{N}_{460}$$

$$N_{26} = \bar{N}_{393}\bar{N}_{394} + N_{393}N_{394}$$

The netlist, generated by *sis*, for this redesigned B-group is not the same as that of the original B-group but replacing the missing block with this netlist results in a circuit that

is functionally equivalent to the original entire circuit in all respects.

Discussion

The approaches given above require finding sensitized paths which can be accomplished by using the digital test generation techniques such as Boolean difference, path sensitization, backtracking methods [11]. For approach 1, we have used D-algorithm with modification as already described in Chapter 2. The problem of finding all possible sensitized paths is handled using Boolean difference. The partitioning of C-group into smaller blocks makes it effective to use Boolean difference for finding all possible sensitized paths. In our implementation we are using *sis* and *espresso* to find the boolean difference.

Therefore, one may use exhaustive simulation, Boolean difference, path sensitization, or backtracking method to generate all controllable primary input vectors and all reachable input vectors. Which method is more effective depends upon the number of inputs/outputs in each group and the computation complexity for performing these methods.

4.2 Improvement

In this section, various improvements made to the initial development are described. These improvements are related to the circuits which either fail the redesignability check or the feasibility check. In order to handle these problems an augmented partitioning scheme for B-group is proposed. The B-group is augmented to B'-group, as shown in Figure 4.10(a). B'-group may cover some gates in A-group, and/or some gates in C-group. The augmented partitioning is classified as *BO-Augmentation* when gates from

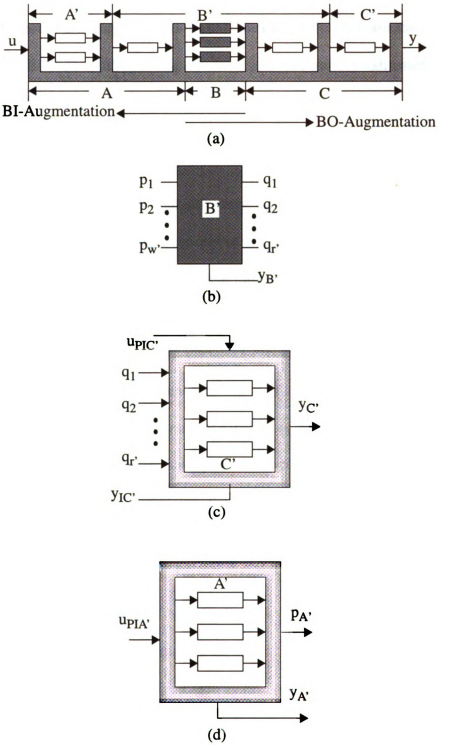


Figure 4.10: Augmented Partitioning: (a) Concept; (b) Augmented B-group; (c) BO-Augmented C-group; and (d) BI-Augmented A-group.

C-group are included in B-group and as *BI-Augmentation* when gates from A-group are included in B-group. BO-augmentation is used when a target circuit fails the redesignability check while BI-augmentation is employed to make functionality extraction of B-group cost effective. The two partitioning schemes are discussed below:

4.2.1 BO-Augmentation

This partitioning scheme is used when target circuit fails the redesignability check, discussed in section 4.1.1. The basic concept of this partitioning is to reduce the number of q_i's in the circuit, so that each q_i can be determined independent of other q_j's for all $j \neq i$. This is accomplished by finding the points of minimum connectivity in C-group. In case of failure of the redesignability check, the set Q_C* contains Q_{Ci}'s that cannot be determined from the current partitioning, on termination of Algorithm 2. In other words, the unknowns that can be resolved under current partitioning have been eliminated from Q_C*. So in order to make the circuit redesignable, we need to focus our attention to Q_{Ci}'s that are contained in Q_C* when the redesignability check halts. Thus instead of dealing with all Q_{Ci}'s, we work only with those which are essential to make the circuit redesignable.

Once the Q_{Ci}'s have been found as described above, we consider only those blocks from the partitioned 'm' blocks that contain these Q_{Ci}'s as their input. These blocks are regrouped such that $Q_{Ci'}$ and $Q_{Cj'}$ are disjoint for all $i \neq j$. By such partitioning we have confined the unknowns, q_i's, that depend on each other in separate blocks. The next step is to partition each individual block $C_{i'}$, so that it contains only one unknown. In other words, we find the node in the group where all the unknowns in $Q_{Ci'}$ converge. While doing this the worst case would be to go up to the primary outputs, which supports the fact that with

Assumption 1 taken in our problem, every circuit is redesignable. Since now each block contains only one unknown it can be redesigned but in order to make it redesignable, we have not only modified the B-group, as shown in Figure 4.10(b), but the C-group has also been altered, as illustrated in Figure 4.10(c). The number of outputs in $q_{B'}$ will be less than that in q_B while the number of primary outputs in $y_{B'}$ will be greater than or equal to that in y_B.

The above partitioning scheme to improve redesignability properties of the target circuit can be best described by an example. Consider the circuit given in Figure 1.2 with B-group for example circuit VI including the AOI22 gate with output node 287, the OAI22 gate with output node 393, the NAND2 gate with output node 458, the INV1 gate with output node 390, the INV1 gate with output node 391, and the NOR2 gate with output node 10. The CO-partitioning for C-group of this circuit into 3 blocks, C_1, C_2, and C_3, is shown in Figure 4.7(c). We can see that for block C_1, we have $u_{PIC1} = \{u_2,u_3,u_5,u_6\}$, $Q_{C1}=\{10,287\}$, $y_{IC1}=\emptyset$ and $y_{C1}=\{24\}$, for block C_2, $u_{PIC2} = \{u_2,u_3,u_5,u_6\}$, $Q_{C2}=\{393\}$, $y_{IC2}=\{24,26\}$ and $y_{C2}=\{25\}$ and similarly for block C_3, $u_{PIC3}= \{u_3,u_6\}$, $Q_{C3}=\{393\}$, $y_{IC3}=\emptyset$ and $y_{C3}=\{26\}$. This circuit fails the redesignability check because $Q_{C1}=\{10,287\}$ can not be resolved with the current partitioning of the circuit. Algorithm *2* for the redesignability check will terminate with $Q_C^*=\{Q_{C1}\}$. Since the unknowns 10 and 287 are contained only in Q_{C1}, we do not need to merge any other block with this. The regrouping will result in a block identical to the block C_1. The block C_1, as shown in Figure 4.11(a) contains $u_{PIC'}=\{u_2,u_3,u_5,u_6\}$, $Q_{C1}=\{10,287\}$, $y_{IC1}=\emptyset$ and $y_{C1}=\{24\}$. Then we partition block to get $C_{1'}$ such that $Q_{C1'}$ contains only one element. It can be seen that the two unknowns 10 and 287 converge at OAI21 gate with output node 465. By making $Q_{C1'}=\{465\}$, as illustrated in Fig-

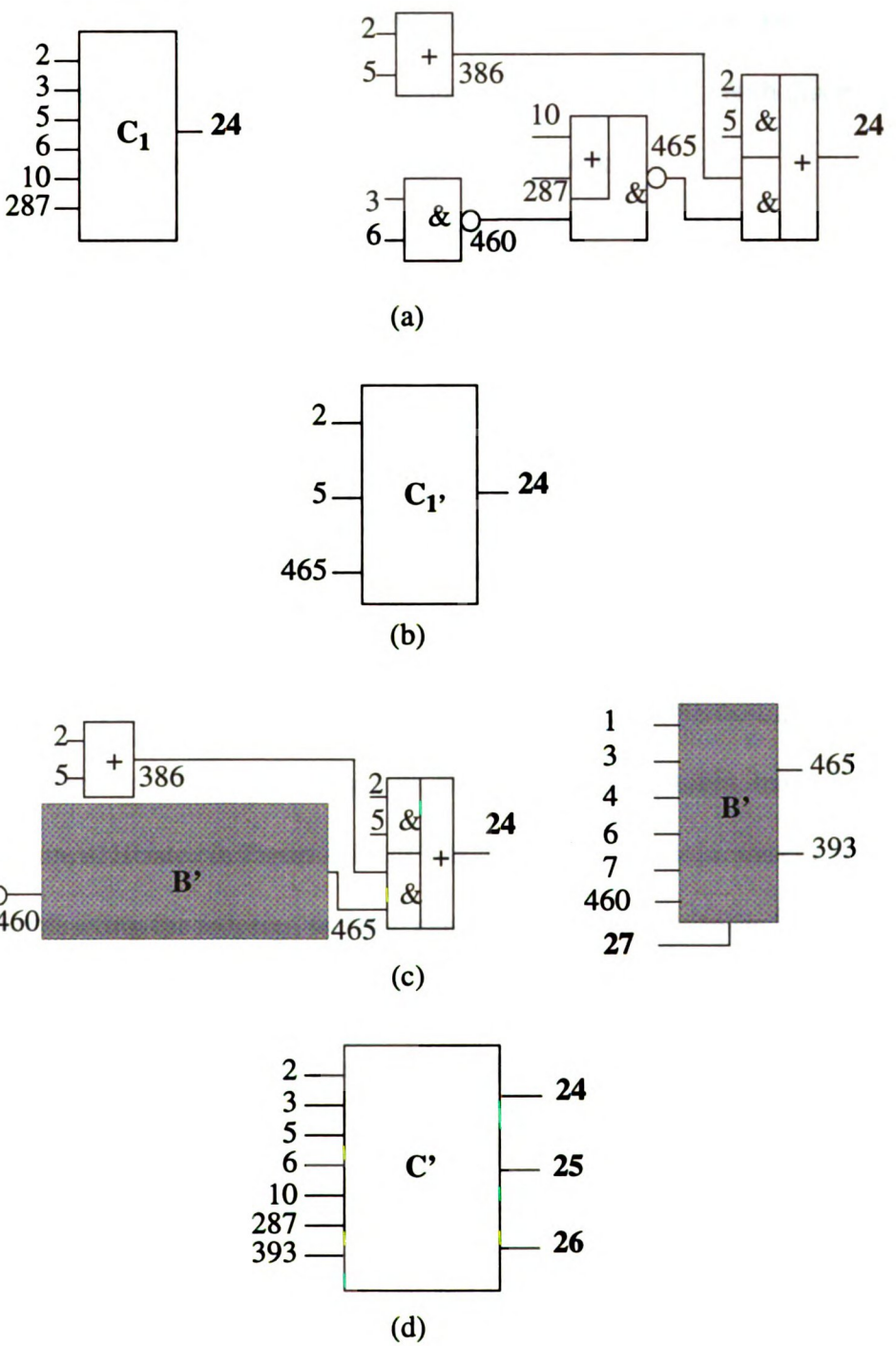


Figure 4.11: BO-Augmentation: Example Circuit VI; (a) Block C_1; (b) Block C_1'; (b) Augmented B'-group; and (c) C'-group.

ure 4.11(b), we have met the condition of single unknown q_i in each block C_i'. Hence the circuit is redesignable with the new B'-group and C'-group which are shown in Figure 4.11(c) and (d) respectively. B'-group now contains $q_{B'}$ ={465,393}. With this modified, B'-group, the target circuit will pass the redesignability check.

BO-augmentation is summarized in Algorithm 4.

4.2.2 BI-Augmentation

This partitioning scheme is used to make functionality extraction of the target circuit cost effective. The basic goal of this partitioning is to reduce the computational complexity of finding the reachable input vectors of B-group. Computational complexity generally, but not absolutely, increases with the number of inputs and outputs of B-group. Thus, if we augment B-group, as illustrated in Figure 4.12(a), for the example circuit I in Figure 1.3. It shows that the original B-group, shown in Figure 4.12(b), has 5 inputs and now the B'-group, illustrated in Figure 4.12(c), has only 3 inputs. Hence, the computational complexity for deriving the redesign solution can be reduced significantly. The number of inputs $p_{B'}$ will be less than that in p_B. While altering the B-group, this partitioning also modifies the A-group, as shown in Figure 4.10(d). The A'-group for example circuit I is illustrated in Figure 4.12(d).

Discussion

The augmented circuit concept not only makes the redesignability check and functionality extraction simpler, but also reduces the circuit size. More specifically, when a practical large circuit is considered, since the majority parts are known and only a small

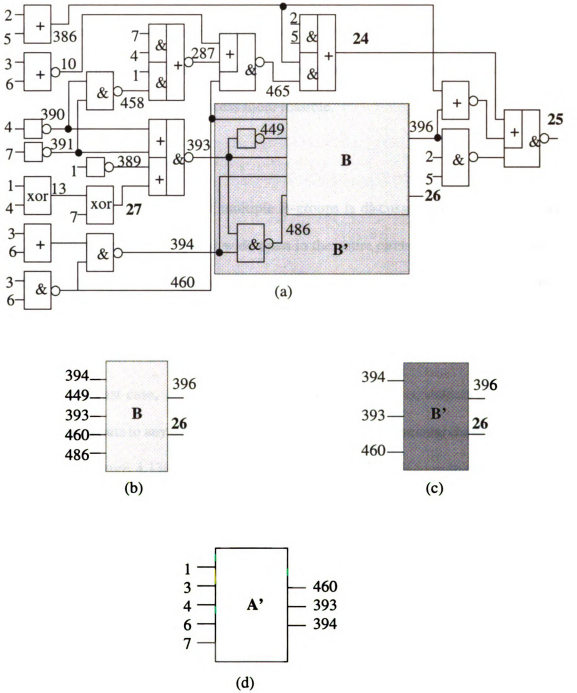


Figure 4.12: BI-Augmentation: (a) Example Circuit I; (b) Original B-group; (c) Augmented B'-group; and (d) A'-group.

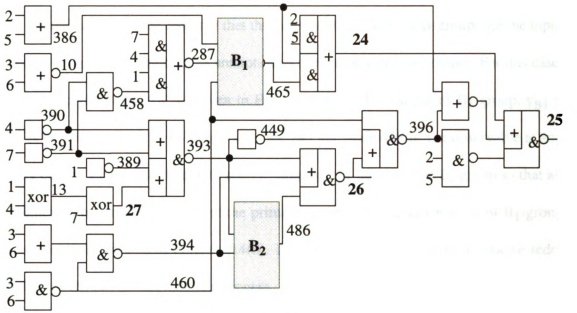
portions of the circuit is missing, we may apply the circuit augmentation to find the functionality of a relatively smaller circuit compared to the entire circuit which still includes the missing parts. Thus, the circuit complexity can be reduced and this makes the proposed partitioning schemes for redesign process more feasible.

4.2.3 Multiple B-Groups

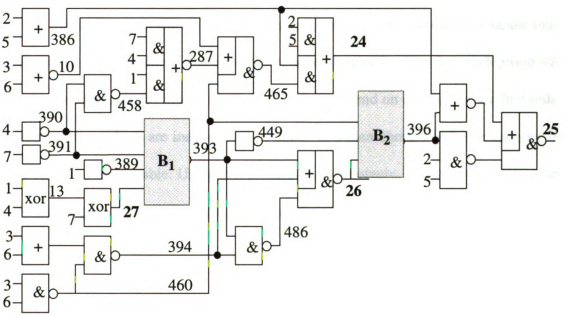
In this section, the case of multiple B-groups is discussed, that is, the parts with missing implementation are spread wide apart in the entire circuit. Again the assumption made is that the input and output nodes of each B-group are known. Let the number of B-groups in the circuit be 'p', so that we have B_1, B_2, ...,B_p. In case of multiple B-groups, we can have three possible cases.

In the first case, the groups B_i are independent, that is, outputs of none of the groups are the inputs to any other group directly or indirectly. Consider the example circuit VII shown in Figure 4.13(a) for such a case. The output of B_1-group, $y_{B1} = \{465\}$ is observable at primary output node 24 whereas output of B_2-group, $y_{B2} = \{486\}$ is observable at primary output node 26. The reachable input vectors of each group can be found independently. So both the groups B_1 and B_2 can be redesigned using our developed redesign process.

The second case is when the outputs of one or more B-groups are the input of other B-groups, but such outputs are observable at one or more primary outputs. Consider example circuit VIII as illustrated in Figure 4.13(b). The output of B_1-group, $y_{B1} = \{393\}$ is in the input path to B_2-group via node 449 but node 393 is observable at primary output node 26. So, in this case we shall have to redesign B_1-group first using our developed process.



(a)



(b)

Figure 4.13: Multiple B-Groups: (a) Example Circuit VII; (b) Example Circuit VIII.

After the re-implementation of B_1-group is introduced in the circuit B_2-group can be redesigned with the developed redesign process.

In the third case, we consider that the outputs of one or more B-groups are the input of other B-groups, but such outputs are not observable at any primary output. For this case, consider example circuit IX, as given in Figure 4.14(a). The output of B_1-group, y_{B1} = {458} is in the input path to B_2-group via node 287 and node 458 is not observable at any primary output. In order to deal with such a situation, we augment the B_i-group so that all the unknowns become observable at the primary outputs. The augmentation of B_1-group to $B_{1'}$-group is illustrated in Figure 4.14(b). The augmented $B_{1'}$-group can now be redesigned using the developed redesign process.

Discussion

In this section, it is shown that the developed redesign process can also handle multiple B-groups. If the multiple B-groups do not depend on each other, then each group will be resolved independently. When multiple B-groups depend on each other, we first redesign those groups which are independent of others and then check whether remaining B-groups become redesignable. Using this procedure successively, if we are unable to redesign certain B-groups then we augment so that a redesignable B-group is obtained.

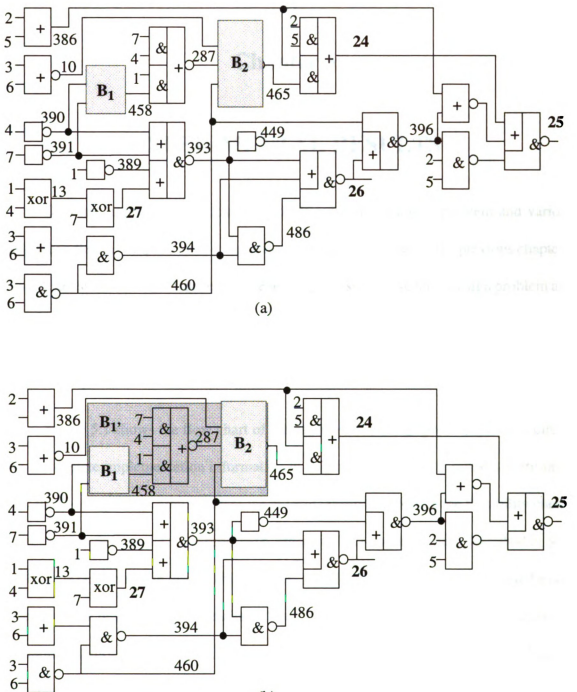


Figure 4.14: Multiple B-Groups: Example Circuit IX (a) B_1 and B_2 groups ; and (b) Augmented B_1'-group.

Chapter 5

EXPERIMENTAL RESULTS

Different approaches that can be used to solve the redesign problem and various ways to handle the computational complexity have been described in the previous chapters. This chapter summarizes the developed redesign process to solve the redesign problem and presents the experimental results of some benchmark circuits.

5.1 Redesign Process

Figure 5.1 shows the flow chart of the developed redesign process. Given a circuit with incomplete implementation information in B-group, the circuit is initially partitioned to get the C-group. If the circuit passes the redesignability check then the feasibility check, as discussed in Section 4.1.1., is processed, otherwise BO-augmentation, described in Section 4.2.1, is performed. In the feasibility check, initially A-group is considered and the cost associated with determining the reachable input vectors of B-group, in terms of number of simulations and/or backtrackings, is evaluated. In case it is not cost effective to find the reachable input vectors of B-group with initial partitioning as described in Section 4.1.2, BI-augmentation, discussed in Section 4.2.2, is applied to make redesign solution cost effective. The criterion used in the redesign process is the one given in Property 4.8. The maximum allowable number of inputs in any partitioned block of A-group, $n_{uPIA}(MAX)$ =

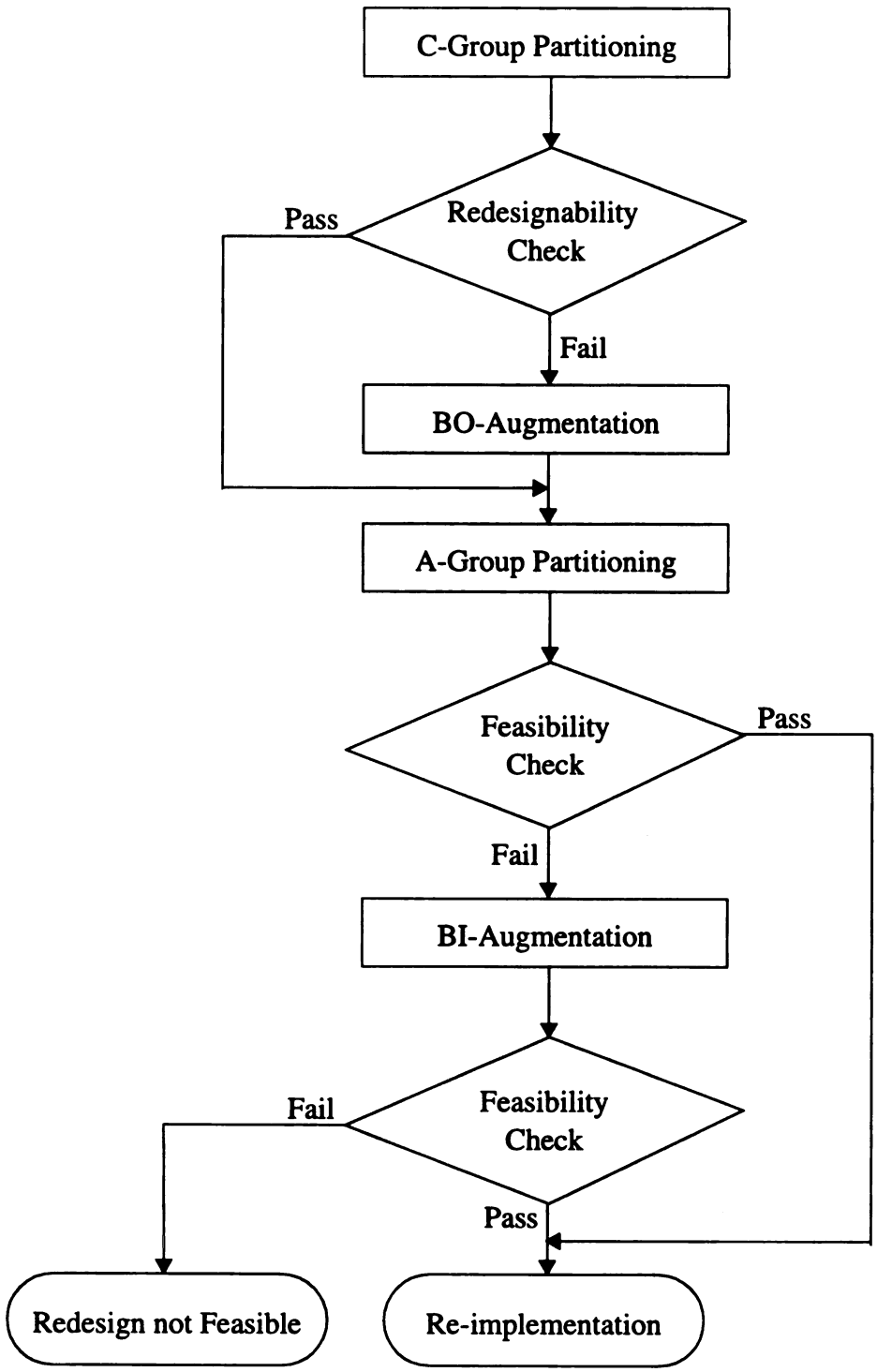


Figure 5.1: Redesign Process

10, is assumed for these benchmark circuits. That is, if there exists a single partitioned block of A-group which has number of inputs and outputs more than 10 and 5 respectively, it is not cost effective to find the reachable vectors. If the reachable input vectors of B-group can be found in a cost effective manner, we then proceed to check the feasibility of re-implementation. The criterion for feasible re-implementation is based on Property 4.9. The maximum number of reachable input vectors, $INPUT\_VECTOR_{MAX} = 2^{12}$, is assumed, i.e., if the number of reachable input vectors of B-group exceed the number 4096, the re-implementation is not feasible with the available resources. When a circuit passes both the redesignability and feasibility check, the input/output relationship is determined using the approaches given in Chapter 4. After deriving the transfer function, the missing parts are re-implemented using existing VLSI design tools, where *espresso* and *sis* are used in this implementation.

5.2 Results

In order to demonstrate the effectiveness of the developed redesign process, a set of benchmark circuits has been tested. The benchmark circuits were generated using *sis* with the script file in Figure 5.2(a). for MCNC benchmark circuits in the path /sis/ex/comb/mcnc91/mlex/. More specifically, this implementation first read the blif file of a circuit, where Figure 5.2(b) shows the blif file of the benchmark circuit "cm138a". The circuit is then optimized using only one cycle of the suggested script "script.rugged" in *sis*. However, better results may be obtained by repeating the script file for several time. Here, the cell library "scmos.genlib" is employed. The circuit is mapped using this library to obtain the netlist, as shown in Figure 5.2(c). Table 5.1 lists the number of primary inputs, the number

```
sis> read_blif filename.blif
sis> source script.rugged
sis> read_library scmos.genlib
sis> map -W
sis> write_blif -n filename.gate
```

(a)

```
.model CM138
.inputs a b c d e f
.outputs g h i j k l m n
.names a b c j0 g
1--- 1
-1-- 1
--1- 1
---1 1
.names a b c j0 h
0--- 1
-1-- 1
--1- 1
---1 1
.names a b c j0 i
1--- 1
-0-- 1
--1- 1
---1 1
.names a b c j0 j
0--- 1
-0-- 1
--1- 1
---1 1
.names a b c j0 k
1--- 1
-1-- 1
--0- 1
---1 1
.names a b c j0 l
0--- 1
-1-- 1
--0- 1
---1 1
.names a b c j0 m
1--- 1
-0-- 1
--0- 1
---1 1
.names a b c j0 n
0--- 1
-0-- 1
--0- 1
---1 1
.names f e d j0
1-- 1
-1- 1
--0 1
.end
```

(b)

```
.model CM138
.inputs a b c d e f
.outputs g h i j k l m n
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 0.07 0.07
.default_output_load 3.00
.default_max_input_load 999.00
.gate nor2 a=e b=f O=[326]
.gate nand2 a=d b=[326] O=[330]
.gate or2 a=a b=[330] O=[331]
.gate or3 a=[331] b=b c=c O=g
.gate inv1 a=a O=[327]
.gate or2 a=[327] b=[330] O=[335]
.gate or3 a=[335] b=b c=c O=h
.gate inv1 a=b O=[332]
.gate or3 a=[331] b=c c=[332] O=i
.gate or3 a=[335] b=c c=[332] O=j
.gate inv1 a=c O=[333]
.gate or3 a=[331] b=b c=[333] O=k
.gate or3 a=[335] b=b c=[333] O=l
.gate or3 a=[331] b=[332] c=[333] O=m
.gate or3 a=[335] b=[332] c=[333] O=n
.end
```

(c)

Figure 5.2: Benchmark Circuit: (a) Script File; (b) cm138a.blif; and (c) cm138a.gate.

Table 5.1: Benchmark Circuits

Circuit	No. of Primary Inputs	No. of Primary Outputs	No. of Gates
alu2	10	6	202
apex6	135	99	417
b1	3	4	5
C17	5	2	3
cm138a	6	8	15
cm42a	4	10	17
cm82a	5	3	11
cmb	16	4	28
count	35	16	80
decod	5	16	30
example2	85	66	174
i3	132	6	46
i5	133	66	66
i7	199	67	407
majority	5	1	8

of primary outputs, and the gate count for each benchmark circuit generated from the above procedure. For example, the resultant benchmark circuit for "cm138a" has 6 primary inputs (a, b, c, d, e, f), 8 primary outputs (g, h, i, j, k, l, m, n) and 15 gates (1 NOR2, 1 NAND2, 2 OR2, 8 OR3, 3 INV1).

Based on the resultant benchmark circuits in Table 5.1, a set of benchmark circuits for redesign process are generated and listed in Table 5.2. The table shows the number of inputs and outputs in each group. For example, consider the unknown B-group of the benchmark circuit "cm138a", as shown in Figure 5.3(a). The partitioned C-group and A-group are illustrated in Figures 5.3(b) and 5.3(c), respectively. The inputs and outputs of each group are listed in Figure 5.3(d). For B-group, the number of inputs n_{pB}=2, the number of outputs excluding the primary outputs n_{qB}=1, the number of primary outputs n_{yB}=0, and the number of gates $\#_B$=1. The numbers are listed in the Columns 2 to 4 in Table 5.2. Similarly, for C-group, n_{uC}=2, n_{yIC}=0, n_{yC}=4, and $\#_C$=6; and for A-group, n_{uA}=4, n_{pA}=2, n_{yA}=0, and $\#_A$=3. The detailed input and output nodes of the partitioned groups for each benchmark circuit in Table 5.2 are given in APPENDIX A.

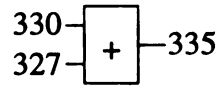
Based on the redesign process illustrated in Figure 5.1, Table 5.3 summarizes the experimental results for all benchmark circuits in Table 5.1. Consider the benchmark circuit "cm138a", as shown in Figure 5.3. The redesignability check was first performed on the C-group shown in Figure 5.3(b). Since the unknown q_B = {[335]} can be observed at the primary outputs of the circuit, so the circuit passed the redesignability check without BO-augmentation, i.e., BO-augmentation is not required to pass the test. It is followed by the feasibility check. Since the number of inputs of A-group, n_{uPIA}=4, is less than n_{uPIA}(MAX)=10, finding reachable vectors is cost effective and thus the circuit passes the

Table 5.2: Partitioning Parameters for Benchmark Circuits

Circuit	n_{pB}	n_{qB}	n_{yB}	$\#_B$	n_{uC}	n_{yIC}	n_{yC}	$\#_C$	n_{uA}	n_{pA}	n_{yA}	$\#_A$
alu2	6	1	0	2	10	2	3	199	5	6	0	12
apex6	11	2	2	4	1	0	2	2	19	8	0	36
b1	4	1	0	1	0	1	1	1	2	1	1	2
C17	4	0	1	1	0	0	0	0	2	1	0	1
cm138a	2	1	0	1	2	0	4	6	4	2	0	3
cm42a	2	2	0	2	2	0	4	7	2	2	0	2
cm82a	4	1	0	1	2	0	2	6	3	2	0	2
cmb	4	2	0	2	13	0	2	8	0	0	0	0
count	13	3	0	8	3	0	3	5	14	7	0	13
decod	3	4	0	5	3	0	16	24	1	1	0	1
example2	16	5	1	6	41	1	16	77	8	11	0	13
i3	16	1	0	5	16	0	1	6	0	0	0	0
i5	26	0	10	10	0	0	0	0	19	0	6	9
i7	52	25	0	27	28	0	25	54	53	25	0	82
majority	3	1	0	1	5	0	1	4	3	3	0	3

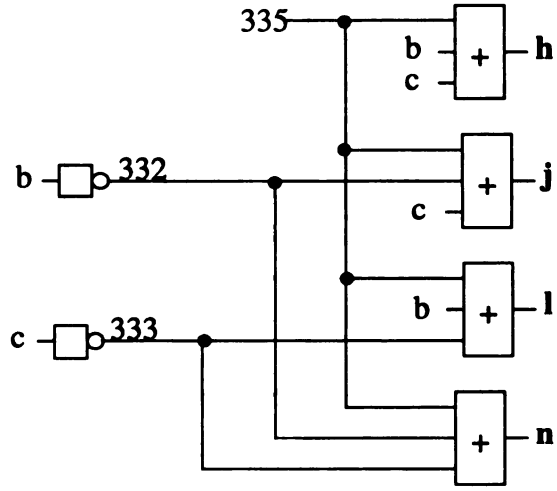
n_{pB} : Number of inputs in B-group.
n_{qB} : Number of outputs in B-group excluding those outputs that are primary outputs.
n_{yB} : Number of primary outputs which are the outputs of B-group.
$\#_B$: Number of gates in B-group.
n_{uC} : Number of primary inputs in C-group.
n_{yIC} : Number of primary outputs which act as an input to C-group.
n_{yC} : Number of primary outputs in C-group.
$\#_C$: Number of gates in C-group.
n_{uA} : Number of primary inputs in A-group.
n_{pA} : Number of outputs of A-group which are not a primary output.
n_{yA} : Number of primary outputs in A-group.
$\#_A$: Number of gates in A-group.

```
.model CM138B
.inputs [327] [330]
.outputs [335]
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 0.07 0.07
.default_output_load 3.00
.gate or2 a=[327] b=[330] O=[335]
.end
```



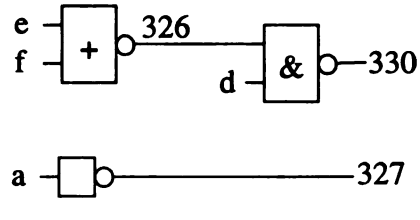
(a)

```
.model CM138C
.inputs [335]  c  b
.outputs  n  l  j  h
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 0.07 0.07
.default_output_load 3.00
.gate or3 a=[335] b=b c=c O=h
.gate or3 a=[335] b=c c=[332] O=j
.gate or3 a=[335] b=b c=[333] O=l
.gate or3 a=[335] b=[332] c=[333] O=n
.gate inv1 a=b O=[332]
.gate inv1 a=c O=[333]
.end
```



(b)

```
.model CM138A
.inputs  d  e  f  a
.outputs [327] [330]
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 0.07 0.07
.default_output_load 3.00
.gate inv1 a=a O=[327]
.gate nand2 a=d b=[326] O=[330]
.gate nor2 a=e b=f O=[326]
.end
```



(c)

p_B = {[327], [330]}; q_B = {[335]}; $y_B = \varnothing$
u_{PIC} = {b, c}; $y_{IC} = \varnothing$; y_C = {h, j, l, n}
u_{PIA} = {a, d, e, f}; p_A = {[327], [330]}; $y_A = \varnothing$

(d)

Figure 5.3: Circuit cm138a Partitioning: (a) B-Group; (b) C-Group; (c) A-Group. and (d) Inputs and Outputs of Each Group.

Table 5.3: Experimental Results of Benchmark Circuits

Circuit	**Redesignability Check**		**Feasibility Check**	
	Without BO-Augmentation	With BO-Augmentation	Functionality Extraction	Re-implementation
alu2	Fail	Pass	Pass	Pass
apex6	Pass	N/R	Pass	Pass
b1	Pass	N/R	Pass	Pass
C17	Pass	N/R	Pass	Pass
cm138a	Pass	N/R	Pass	Pass
cm42a	Pass	N/R	Pass	Pass
cm82a	Pass	N/R	Pass	Pass
cmb	Pass	N/R	Pass	Pass
count	Pass	N/R	Pass	Pass
decod	Pass	N/R	Pass	Pass
example2	Pass	N/R	Pass	Pass
i3	Pass	N/R	Pass	Pass
i5	Pass	N/R	Pass	Pass
i7	Pass	N/R	Pass	Fail
majority	Pass	N/R	Pass	Pass

N/R : Not Required.

feasibility check without the need of BI-augmentation. Finally, the feasibility of implementation is checked after finding the reachable input vectors. The number of reachable input vectors, in this case is 4 which is less than $INPUT\_VECTOR_{MAX}$=4096, so re-implementation is feasible. The transfer function was determined and the re-implemented B-group was identical to the unknown B-group. Therefore, the circuit passes all the tests listed in Table 5.3.

For the circuit "alu2", the B-group is comprised of two NOR gates with the output nodes [465] and [466], respectively. When the redesignability check is applied to C-group, both unknowns [465] and [466] cannot be observed without the involvement of each other using either CI-partitioning or CO-partitioning schemes discussed in Section 4.1.1. Thus, the redesignability check failed and BO-augmentation was performed. This augmentation resulted in B-group including the NOR2 gate [993] from C-group in addition to the NOR3 gate [465] and NOR3 gate [466]. With this BO-augmentation the circuit passes the redesignability check. The feasibility check was processed next, since the number of inputs of A-group, n_{uA}=5, does not exceed n_{uPIA}(MAX)=10, finding reachable vectors is cost effective and we proceed to check the feasibility of implementation after finding the reachable input vectors. The total number of reachable input vectors for B-group is 21 which is less than $INPUT\_VECTOR_{MAX}$=4096, so re-implementation is feasible and the circuit has passed the feasibility check. The transfer function for B-group was determined and the circuit can be re-implemented.

For the circuit "apex6", the circuit passes the redesignability check as shown in Table 5.3. For the feasibility check, as shown in Table 5.2 and in Appendix A, the number of inputs and outputs of A-group are 19 and 8, respectively, and they exceed

$n_{uPIA}(MAX)=10$ and $n_{uPIA}(MAX)/2=5$, respectively. Therefore, as discussed in Section 4.1.2, the A-group was partitioned into two blocks A_1 and A_2 where the number of inputs and outputs of A_1 were 9 and 5, respectively, and those of A_2 were 10 and 3, respectively. Thus, this partitioned A-group passes the criterion of Property 4.8 and hence finding reachable vectors is cost effective. The circuit passed the feasibility check and the redesigned B-group can be re-implemented.

For the circuit "count", it is similar to the case of "apex6",where the number of inputs and outputs of A-group exceed $n_{uPIA}(MAX)$ and $n_{uPIA}(MAX)/2$, respectively. Therefore, the A-group was partitioned into two blocks A_1 and A_2, where the number of inputs and outputs of block A_1 were 13 and 6, respectively. Here, block A_2 passed the test but block A_1 failed the criterion of Property 4.8. So, BI-augmentation was performed resulting in inclusion of gate [669] in B-group. With this BI-augmentation the number of outputs of block A_1 reduced from 6 to $5=n_{uPIA}(MAX)/2$, meeting the criterion of Property 4.8 and hence finding reachable vectors is cost effective. The circuit passed the feasibility check and the redesigned B-group can be re-implemented.

For the circuit "cmb", the B-group is chosen as all its inputs are the primary inputs of the original circuit, where no A-group is included. Results show that the C-group passes the redesignability check. Since the total number of reachable input vectors for B-group is 16 which is less than $INPUT\_VECTOR_{MAX}$, re-implementation is feasible.

For the circuit "i5", the B-group is chosen as all its outputs are the primary outputs of the original circuit, where no C-group is included. The circuit passes all tests and can be re-implemented.

For the circuit "i7". Redesignability check performed on the C-group obtained for

this circuit was successful. In the feasibility check, the first criterion of cost effectiveness of finding the reachable input vectors was satisfied but the number of reachable input vectors was 2^{52}, which is greater than $\text{INPUT\_VECTOR}_{MAX}$, so re-implementation is not feasible according to Property 4.9. Hence, the circuit has failed the feasibility check, as indicated in Table 5.3, and the redesign process is halted.

Chapter 6

CONCLUSIONS

This thesis describes a new problem of redesigning digital VLSI circuits with incomplete implementation information. Given a digital circuit with incomplete implementation information, the proposed redesign process is to recover the original design from the partial known implementation information. In this study, the developed redesign process is comprised of three steps: redesignability check, feasibility check and re-implementation. Computational complexity is of major concern in this problem. A number of properties have been developed to determine whether the target circuit is redesignable or not, without involving excessive computation. As mentioned earlier, based on the assumptions, all target circuits are redesignable. However, some redesign solutions may end-up with redesigning the entire circuit which is costly. Therefore, by "redesignable" in this study it is meant that the circuit can be redesigned at a reasonably low cost. The cost factor is considered at the second stage of the redesign process, that is the feasibility check. In this check, the cost associated with determination of reachable input vectors of B-group is first considered. If it is cost effective to find the reachable vectors, feasibility of re-implementation of the missing parts is evaluated. The decision whether to re-implement the missing parts or not is made taking various cost factors into account which include factors like complexity of the

transfer function of missing parts and the need of the task in hand. A number of efficient partitioning schemes for the target circuit have also been developed in order to make the redesign process cost effective both in terms of time and money. In our developed redesign process, the system with incomplete implementation information is redesigned using concepts of test generation techniques. This redesign process was used to redesign the missing parts of various benchmark circuits and satisfactory results were obtained.

The developed redesign process can be applied to efficiently deal with microelectronics obsolence problem. It can be very helpful in configuration management by generating documentation for undocumented designs. The process can also be effective in cases where safe replacement of some functional blocks in an existing implementation of the circuit is required. Safe replacement may be needed due to reasons such as performance improvement, obsolete parts. This developed redesign process can also be extended to be used for correction of single design errors .

This study also outlined a number of interesting research topics for future research:

1. Based on the proposed partitioning schemes, how to efficiently and effectively generate the reachable input vectors of B-group.
2. How to efficiently and effectively determine the observability of the outputs of B-group.
3. How to efficiently and effectively find the minimum number of primary input vectors that generate all reachable input vectors of B-group while making the corresponding outputs of B-group observable.
4. Develop new partitioning schemes which can efficiently and effectively deter-

mine unredesignability within a constrained cost defined by the designers.

5. How to take into account the timing constraints, such as wire and propagation delays, hold times, of the original implementation of missing parts.
6. How to determine whether the missing parts included some redundant elements or not. Redundant elements might have been used for the purpose of hazard removal.
7. How to efficiently and effectively apply this technique for sequential circuits with missing or incomplete implementation.

APPENDIX A

APPENDIX A

BENCHMARK CIRCUITS PARTITIONING DETAILS

alu2

p_B = {[1083], [1080], [1107], [1628], [1079], [1090]}; q_B = {[465], [466]}; $y_B = \varnothing$
u_{PIC} = {a, b, c, d, e, f, g, h, i, j}; y_{IC} = {m, n}; y_C = {k, l, o}
u_{PIA} = {e, f, g, h, j}; p_A = {[1083], [1080], [1107], [1628], [1079], [1090]}; $y_A = \varnothing$

apex6

p_B = {AXZ0, AXZ1, XZ161, [116], [431], [597], [2187], [3791], [2325], [2210], [3773]}; q_B = {[428], [429]}; y_B = {XZ323_P, XZ161_P}
u_{PIC} = {RYZ}; $y_{IC} = \varnothing$; y_C = {AXZ0_P, AXZ1_P}
u_{PIA} = {CBT2, PSYNC, ICLR, TXMESS_N, A, B, QPR0, QPR1, QPR2, QPR3, QPR4, AXZ0, XZ320, XZ321, XZ322, XZ323, XZ324, XZ160_N, XZ161}; p_A = {[2210], [597], [116], [431], [2187], [3791], [2325], [3773]}; $y_A = \varnothing$

b1

p_B = {b, c, g, [293]}; q_B = {[273]}; $y_B = \varnothing$
$u_{PIC} = \varnothing$; y_{IC} = {e}; y_C = {f}
u_{PIA} = {a, c}; p_A = {[293]}; y_A = {g}

C17

p_B = {1GAT(0), 2GAT(1), 3GAT(2), [307]}; y_B = {22GAT(10)}
$u_{PIC} = \varnothing$; $y_{IC} = \varnothing$; $y_C = \varnothing$;
u_{PIA} = {2GAT(1), 3GAT(2)}; p_A = {[307]}; $y_A = \varnothing$

cm138a

p_B = {[327], [330]}; q_B = {[335]}; $y_B = \varnothing$
u_{PIC} = {b, c}; $y_{IC} = \varnothing$; y_C = {h, j, l, n}
u_{PIA} = {a, d, e, f}; p_A = {[327], [330]}; $y_A = \varnothing$

cm42a

p_B = {[337], [338]}; q_B = {[339], [397]}; $y_B = \emptyset$
u_{PIC} = {a, d}; $y_{IC} = \emptyset$; y_C = {e, f, m, n}
u_{PIA} = {b, c}; p_A = {[337], [338]}; $y_A = \emptyset$

cm82a

p_B = {a, c, [326], [350]}; q_B = {[327]}; $y_B = \emptyset$
u_{PIC} = {d, e}; $y_{IC} = \emptyset$; y_C = {g, h}
u_{PIA} = { b, c}; p_A = {[326], [350]}; $y_A = \emptyset$

cmb

p_B = {j, k, l, m}; q_B = {[296], [397]}; $y_B = \emptyset$
u_{PIC} = {a, b, c, d, e, f, g, h, i, j, n, o, p}; $y_{IC} = \emptyset$; y_C = {q, t}
$u_{PIA} = \emptyset$; $p_A = \emptyset$; $y_A = \emptyset$

count

p_B = {e, g, q, a0, d0, f0, [663], [660], [654], [651], [669], [666], [633]}; q_B = {[303], [664], [670]}; $y_B = \emptyset$
u_{PIC} = {j, q, s}; $y_{IC} = \emptyset$; y_C = {q0, t0, v0}
u_{PIA} = {q, r, u, v, w, x, y, z, a0, b0, c0, d0, e0, f0}; p_A = {[663], [660], [654], [651], [669], [666], [633]}; $y_A = \emptyset$

decod

p_B = {a, e, [348]}; q_B = {[351], [352], [357], [358]}; $y_B = \emptyset$
u_{PIC} = {b, c, d}; $y_{IC} = \emptyset$; y_C = {f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u}
u_{PIA} = {d}; p_A = {[348]}; $y_A = \emptyset$

example2

p_B = {b, b1, a2, c2, r0, [989], [1000], [1042], [994], [988], [1039], [1756], [1130], [78], [993], [985]}; q_B = {[356], [382], [1081], [1772], [1780]}; y_B = {m2}
u_{PIC} = {b, f, g, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, p0, r0, s0, b1, q1, r1, s1, t1, u1, v1, w1, x1, y1, a2, b2, c2, d2, e2, f2, g2, h2}; y_{IC} = {m2}; y_C = {g3, d4, e4, f4, g4, h4, i4, j4, k4, o4, p4, r4, s4, t4, u4, v4}
u_{PIA} = {b, f, p0, a2, b2, c2, e2, h2}; p_A = {[989], [1000], [1042], [994], [988], [1039], [1756], [1130], [78], [993], [985]}; $y_A = \emptyset$

i3

p_B = {V56(26),V28(26),V56(27),V28(27),V120(0),V88(0),V120(1),V88(1),V120(2), V88(2),V120(3),V88(3),V120(4),V88(4),V120(5),V88(5)}; q_B = {[439]}
u_{PIC} = {V56(18),V28(18),V56(19),V28(19),V56(20),V28(20),V56(21),V28(21), V56(22),V28(22),V56(23),V28(23),V56(24),V28(24),V56(25),V28(25)}; y_{IC} = ∅;
y_C={V138(1)}
u_{PIA} = ∅; p_A = ∅; y_A = ∅

i5

p_B = {V183(15), V76(14), V64(14), V76(11), V64(11), V183(12), V118(2), V115(2), V199(14), V100(13), V88(13), V199(8), V124(1), V121(1), V100(7), V88(7), V199(11), V100(10), V88(10), V183(0), V132(1), V128(1), V112(3), V109(3), V52(15), V40(15)}
q_B =∅ ; y_B = {V183(14), V183(11), V199(13), V167(0), V167(12), V183(8), V167(15), V199(4), V199(10), V199(7)}
u_{PIC} = ∅; y_{IC} = ∅; y_C = ∅
u_{PIA} = {V64(15), V76(15), V88(11), V100(11), V88(14), V100(14), V88(15), V133(0), V100(15), V115(3), V118(3), V121(2), V124(2), V121(3), V124(3), V128(2), V132(2), V128(3), V132(3)}; y_A={V183(15), V183(12), V199(14), V199(8), V199(11), V183(0)}

i7

p_B = {V199(1), V199(0), V160(16), [1941], V160(24), [1893], V160(23), [1899], V160(22), [1905], V160(21), [1911], V160(20), [1917], V160(26), [1881], V160(19), [1923], V160(18), [1929], V160(13), [1959], V160(17), [1935], V160(11), [1971], V160(10), [1977], V160(25), [1887], V160(15), [1947], V160(8), [1989], V160(27), [1875], V160(14), [1953], V160(5), [2007], V160(12), [1965], V160(4), [2013], V160(3), [2019], V160(9), [1983], V160(6), [2001], V160(7), [1995]}
q_B = {[1132], [1260], [1244], [1228], [1212], [1196], [1292], [1180], [1164], [1084], [1148], [1052], [1036], [1276], [1116], [1004], [1308], [1100], [956], [1068], [940], [924], [1020], [972], [988]}; y_B = ∅
u_{PIC} = {V199(1), V199(0), V199(4), V192(27), V192(26), V192(25), V192(24), V192(23), V192(22), V192(21), V192(20), V192(19), V192(18), V192(17), V192(16), V192(15), V192(14), V192(13), V192(12), V192(11), V192(10), V192(9), V192(8), V192(7), V192(6), V192(5), V192(4), V192(3)}; y_{IC} = ∅;
y_C = {V259(31), V259(30), V259(29), V259(28), V259(27), V259(26), V259(25), V259(24), V259(23), V259(22), V259(21), V259(20), V259(19), V259(18), V259(17), V259(16), V259(15), V259(14), V259(13), V259(12), V259(11), V259(10), V259(9), V259(8), V259(7)}
u_{PIA} = {V199(1) V199(0) V128(27) V199(4) V128(26) V128(25) V128(24) V128(23) V128(22) V128(21) V128(20) V128(19) V128(18) V128(17) V128(16) V128(15) V128(14) V128(13) V128(12) V128(11) V128(10) V128(9) V128(8) V128(7) V128(6)

V128(5) V128(4) V128(3) V192(27) V192(26) V192(25) V192(24) V192(23) V192(22) V192(21) V192(20) V192(19) V192(18) V192(17) V192(16) V192(15) V192(14) V192(13) V192(12) V192(11) V192(10) V192(9) V192(8) V192(7) V192(6) V192(5) V192(4) V192(3)}

p_A = {[1941] [1893] [1899] [1905] [1911] [1917] [1881] [1923] [1929] [1959] [1935] [1971] [1977] [1887] [1947] [1989] [1875] [1953] [2007] [1965] [2013] [2019] [1983] [2001] [1995]}; $y_A = \varnothing$

majority

p_B = {[299], [300], [330]}; q_B = {[322]}; $y_B = \varnothing$

u_{PIC} = {a, b, c, d, e}; $y_{IC} = \varnothing$; y_C = {f}

u_{PIA} = {a, b, c}; p_A = {[299], [300], [330]}; $y_A = \varnothing$

REFERENCES

REFERENCES

[1] Defense Microelectronics Activity [Online] Available http://www.dmea.osd.mil, January 3, 1998.

[2] E.J. Byrne, "A Conceptual Foundation for Software Re-engineering," Proc. IEEE Conf. on Software Maintenance, pp. 226-235, 1992.

[3] E.J. Chikofsky and J.H. Cross II, "Reverse Engineering and Design Recovery: A Taxonomy," IEEE Software, Vol. 7, pp.13-17, January 1990.

[4] C.L. Wey, "Development of Redesign Process for Digital VLSI System," Proc. of 40th Midwest Symp. on Circuits and Systems, pp. 1001-1004, August 1997.

[5] A. Wojcik, C.L. Wey, T. Doom, and J. Samarziya, "An approach to the redesign of digital circuits from partial information," Tech. Rep. MSUCPS:TR97-47, Department of Computer Science, Michigan State University, Dec, 1997.[Online] Available http://web.cps.msu.edu/TR/MSUCPS:TR97-47.

[6] C.L. Wey and M.A. Khalil, "Redesignability Analysis of Digital VLSI Circuits with Incomplete Implementation Information," to appear in Proc. of IEEE International Symp. on Circuits and Systems, May 1998.

[7] M.A. Khalil and C.L. Wey, "Using Test Generation Techniques for Redesigning Digital VLSI Circuits with Incomplete Implementation Information," Proc. International Conference on Chip Technology, Hsinchu, Taiwan, pp. 146-152, April 1998.

[8] MCNC Library from the 1989 International Work-shop on Logic Synthesis, [Online] Available http://www.cbl.ncsu.edu/pub/Benchmark_dirs/LGSynth89/DOCUMENTATION/ doc.txt.

[9] Sentovich et al., "SIS: A system for Sequential Circuit synthesis," Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720, May, 1992.

[10] M. Abramovici, M.A. Breuer, and A.D. Friedman, Digital Systems testing and Testable Design, Computer Science Press, 1990.

[11] P.K. Lala, Fault-tolerant and Fault-Testable Hardware Design, Prentice-Hall International, London, 1985.

[12] H. Fujiwara, Logic Testing and Design for Testability, 1985.

[13] S.B. Akers, Jr., "On tthe theory of Boolean functions," Journal of the society for Industrial and Applied Mathematics, vol. 7, pp. 487-498, 1959.

[14] Ku, C.T., G. M. Masson, "The Boolean difference and Multiple Fault Analysis," IEEE transactions on computers, vol. c-24(1): pp. 62-71.

[15] E.J. McCluskey, Logic Design Principles, Prentice Hall, Englewood Cliffs, NJ, 1986.

[16] J.P. Roth, "Diagnosis of automata failures: A calculus and a method," IBM Journal of Research and Development, pp. 278-291, July, 1966.

[17] B. Strunz, C. Flanagan, and T. Hall, Design for Testability in Digital ASICS, [Online] Available http://www.ul.ie/~strunz/dft_notes/course.html, January 23, 1995.

[18] C.L. Wey, "UUT Modeling for Digital Test - A Self-Test Approach," Proceedings of the IEEE Fourth Annual Phoenix Conference on Computers and Communication, pp. 312-316, March 1985.

[19] C.L. Wey, "A Searching Approach Self-testing Algorithm for Analog Fault Diagnosis," in *Testing and Diagnosis of Analog Circuits and Systems*, edited by R.-W. Liu, Chapter 6, pp.147-186, North-Holland, 1991.