ELSEVIER
IFAC
PUBLICATIONS
www.elsevier.com/locate/ifac

A FRAMEWORK FOR REVERSE ENGINEERING VLSI CHIPS

Khaled M. Elleithy and Tarek Sobh

Computer Science and Engineering Department
University of Bridgeport
Bridgeport, CT 06601
elleithy@bridgeport.edu , sobh@bridgeport.edu

Abstract: Reverse engineering process of VLSI chips is a complex operation that can cost from $10,000 for the simplest chips to hundreds of thousands of dollars for complex chips. In this paper we are presenting an overview of the process of reverse engineering of VLSI chips. The paper outlines the steps involved in the process of reverse engineering of chips as well as the different techniques used to extract the functionality of the chip. Also, the paper presents two case studies of reverse engineering of chips.

1. INTRODUCTION

Reverse engineering can be defined as the construction of a high-level functional representation of an implemented system to facilitate one's understanding of the system. The construction process is algorithmic and uses the strategy of generating descriptions at successively higher levels of abstraction. For ICs, each step consists of identifying sets of components that constitute an abstract function and then recasting the circuit description in terms of these abstractions.

Designers use reverse engineering to determine system's specifications, output functions, or other design characteristics from an existing implementation. This contrasts with the customary "forward" (specification to implementation) design process. Companies often reverse-engineer their competitors' products to discover how they are made or to evaluate their quality. In the software industry, for example, reverse engineering refers to updating, for reuse, programs whose specifications have been lost or inadequately documented as described by Chikofsky (1991). In computer hardware, designers have used reverse engineering to extract gate-level models from transistor circuits (Madisetti *et al.* 1999).

Madiseti *et al.* (1999) introduced rationale for reengineering legacy embedded systems. Legacy systems are hardware and/or software systems currently performing useful tasks but requiring reengineering or upgrading for various reasons. The most pressing reasons are parts obsolescence and system needs such as greater functionality, increased processing and interface scalability, better form (size, weight, power, volume), and decreased maintenance and life-cycle support costs. Another reason is the availability of superior algorithms, architectures, and technologies that meet or exceed the system's specifications, often at a lower cost.

Figure 1, shows the relationship between requirements, design, and implementation and where forward engineering and reverse engineering fit. Chickosfky and Cross (1991) defined the following terms:

- Requirements: specification of the problem being solved, including objectives, constraints and business rules
- Design: specification of the solution
- Implementation: coding, testing, and delivery of the operational system
- Forward engineering: is the traditional process of moving from high-level abstractions and logical, implementation-independent designs to the physical implementation of a system.
- Reverse engineering. Reverse engineering is the process of analyzing a system to identify the system components and their relationships and create representation of the system in another form or at a higher level of abstraction.

Section 2 of this paper provides a literature survey and presents the most up-to-date reported research in the area of reverse engineering. The fowling sections present two case studies. The first case is the Reverse Engineering of the ISCAS-85 benchmark. The second case was reverse engineering of the AWACS Rader System by the Air Force which is a project the Air Force awarded Northrop Grumman Corporation for a proof-of-concept project aimed at capturing the functionality of the E3 Airborne Warning and Control System (AWACS) radar system hardware in VHDL. The final section of the paper offers summary and conclusions.

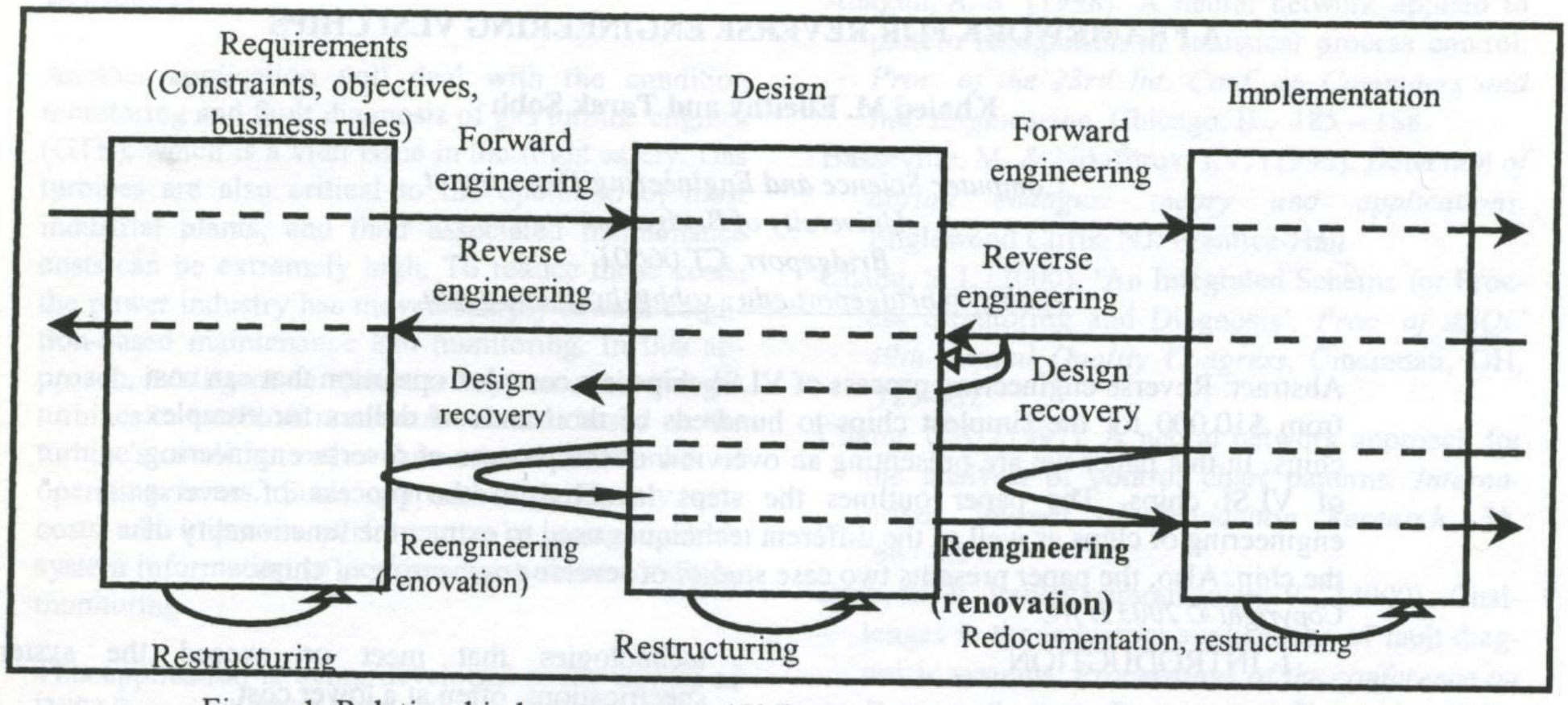

Figure 1: Relationship between terms (Chikofsky and Cross 1990).

2. REVERSE ENGINEERING OVERVIEW

Reverse engineering is the inverse of the design process (Chisholm *et al.* 1999). The design process begins with an abstract description of a target device and, via a succession of refinements, produces a design that can be implemented directly. Reverse engineering, on the other hand, begins with the disassembly of a manufactured device and culminates with an abstract description of the device's functionality. In the case of integrated circuits, the disassembly process consists of obtaining an image of the internal structure of a circuit and extracting a transistor-level netlist from the image. This description is then transformed to successively higher levels of abstraction until a suitably high-level description of the circuit's behavior is obtained.

The key to applying computer-aided software and hardware engineering to the maintenance and enhancement of existing systems lies in applying reverse-engineering approaches. However, there is considerable confusion over the terminology used in both technical and marketplace discussions. In (Chikofsky and Cross 1990) the authors define and relate six terms: forward engineering, reverse engineering, redocumentation, design recovery, restructuring, and reengineering. Their objective was not to create new terms but to rationalize the terms already in use. The resulting definitions apply to the underlying engineering processes, regardless of the degree of automation applied.

Electronics products of the future must be realized efficiently promising higher performance at lower cost within much shorter product design and upgrade cycles. ASIC foundries and EDA vendors see increasing VLSI integration capabilities as a promising new business opportunity through the System-on-Chip (SOC) paradigm that extends ASICs design from the component level to the system level. The systems integration community and electronics packaging design vendors see the systems market as an extension of their current business, the so-called Systems-on-Package (SOP) paradigm, and one that raises their role to new level of importance in the product supply chain linking electronics packaging directly to product specification, early design and ASIC design. In addition to political issues there exist technical, legal, and business challenges, both paradigms must overcome to find broad-based acceptance. In (Tummala and Madisetti 1999) the authors suggest that the Systems-on-Package (SOP) paradigm promises a higher return on investment (ROI) at a much lower risk for the electronics products design, well into the new millennium.

In (Jarzabek and Woon 1997) the authors start to formalizing what we already know about reverse engineering, and propose a framework for describing and evaluating reverse engineering methods and tools. First, they build design models for a source language and for the recovered design. Then, they describe what a given reverse engineering method or tool achieves as a formal mapping from the source language design model into the recovered design model. They show use object recovery scenarios to illustrate the presented concepts.

By the early 1990s the need for reengineering legacy systems was already acute, but recently the demand has increased significantly (Muller *et al.* 2000). Legacy hardware and software systems are defined as those that are currently performing useful tasks, but face possible interruption or termination of operation in the future due to a number of reasons (Madisetti *et al.* 1999). The "push" reasons include the need for increasing functionality, processing and interface scalability, better form (size, weight, power, volume) requirements, decreased maintenance and lifecycle support costs, and resilience to parts obsolescence. The "pull" reasons can include the availability of superior competing algorithms, architectures, and technologies meeting (or exceeding) the specifications of the legacy system, often at a lower cost. Legacy systems can be found everywhere in the military and commercial electronics area. Indeed, in commercial arena, electronics systems, such as PCs and cellular phones, are often obsolete in a matter of months, and increasing pressures of time-to-market has institutionalized re-engineering of products. In the military arena, the long lifetimes of deployed systems, decades in the case of radar systems, has made it inevitable that one is faced with the problem of legacy systems.

The demand by all business sectors to adapt their information systems to the web has created a tremendous need for methods, tools, and infrastructures to evolve and exploit existing applications efficiently and cost-effectively. Reverse engineering has been heralded as one of the most promising technologies to combat this legacy systems problem. Muller *et al.* (2000) present a roadmap for reverse engineering research for the first decade of the new millennium, building on the program comprehension theories of the 1980s and the reverse engineering technology of the 1990s.

Designer's productivity has become the key-factor of the development of electronic systems. An increasing application of design data reuse is widely recognized as a promising technique to master future design complexities. Since the intellectual property of a design is more and more kept in software-like hardware description languages (HDL), successful reuse depends on the availability of suitable HDL reverse engineering tools. In (Mueller-Glaser *et al.* 1996) new concepts for an integrated HDL reverse engineering tool-set are presented as well as an implemented evaluation prototype for VHDL designs. Starting from an arbitrary collection of HDL source code files, several graphical and textual views on the design description are automatically generated. The tool-set provides novel hypertext techniques, expressive graphical code representations, a user-defined level of abstraction, and interactive configuration mechanisms in order to facilitate the analysis, adoption and upgrade of existing HDL designs.

Digital designers normally proceed from behavioral specification to logic circuit; rarely do they need to go in the reverse direction. One such situation examined in (Hayes and Hansen 1999): recovering the high-level specifications of a popular set of benchmark logic circuits. The authors present their methodology and experience in reverse engineering the ISCAS-85 circuits. They also discuss a few of the practical uses of the resulting high-level benchmarks and make them available for other researchers to use.

The problem of finding meaningful sub-circuits in a logic layout appears in many contexts in computer-aided design. Existing techniques rely upon finding exact matching of subcircuit structure within the layout. These syntactic techniques fail to identify functionally equivalent subcircuits, which are differently implemented, optimized, or otherwise obfuscated. In (Doom *et al.* 1998) a mechanism for identifying functionally equivalent subcircuits that is capable of overcoming many of these limitations is presented. Such semantic matching is particularly useful in the field of design recovery.

In (Prinetto et al. 1998) a new approach for sequential circuit test generation is proposed that combines software testing based techniques at the high level with test enhancement techniques at the gate level. Several sequences are derived to ensure 100% coverage of all statements in a high-level VHDL description, or to maximize coverage of paths. The sequences are then enhanced at the gate level to maximize coverage of single stuck-at faults. High fault coverages have been achieved very quickly on several benchmark circuits using this approach.

As a real life example of reverse engineering the Air Force funded of the Electronic Parts Obsolescence Initiative (EPOI) to insure Air Force mission readiness and increase nagging obsolescence (Stogdill 199). EPOI is developing management & re-engineering tools for defense systems affected by parts obsolescence and reliability models for commercially manufactured electronics utilized in defense systems. This initiative currently consists of eight programs covering three k9ey areas of work: 1) Parts Obsolescence Management and Re-engineering Tools, 2) The Application of Commercially

Manufactured Electronics (ACME), and 3) Pilot Demonstration Programs. The initiative's main technology foci are mixed signal electronics, Application Specific Integrated Circuits (ASIC), Physics of Failure validation with commercial field return data, and standardized information exchange.

3. REVERSE ENGINEERING TECHNIQUES

Hayes and Hansen (1999) have defined the following techniques for reverse engineering of hardware:

- *Library modules*. Common components, such as multiplexers, decoders, adders, and CLA generators, are found in IC manufacturers' data books or cell libraries and in textbooks. The modules usually exist in variants due to differences in input size (fan-in or word length) and gate types.

- *Repeated modules*. Often a subcircuit whose logic function is not apparent occurs frequently, especially in data-path circuits where the same circuit slice repeats for different bits of input data.

- *Expected global structures*. After recognizing several modules, the reverse engineer can look for common structures, signals, or functions that use these modules.

- *Computed functions*. With a few structural clues to a subcircuit's role, we can compute its logic function in symbolic or binary (truth table) form, then relate it to known functions or to other circuit functions. This is feasible only for functions of typically no more than four or five signals.

- *Control functions*. We can often identify key control signals whose settings partition a complex function into simpler ones.

- *Bus structures*. The outputs of repeated modules often can be grouped into buses. Further circuit partitioning can result from noting where these common signals lead.

- *Common names*. When analyzing netlists, we sometimes find a shared name among several elements. We may not know what that name implies, but grouping the elements together temporarily can lead to further structural insights.

- *Black boxes*. If all else fails, we can encapsulate a circuit as a module of unknown function or black box. This step is unavoidable when dealing with low-level control circuits consisting of truly random logic.

4. THE REVERSE ENGINEERING PROCESS

Chishom, *et. al.* suggested the following outline for the reverse-engineering process.

A. Sample Preparation

The first step in reverse-engineering an integrated chip is to extract the chip's design layout. This involves removing the chip's overburden material either by chemical etching or mechanical slicing, which are both destructive. Removing the overburden is an extracting process that must adequately expose the underlying transistors and their interconnections without damaging them.

B. Image acquisition

The next step is to scan the sample. The scanning methodology used depends on the density of the transistors in the sample. For example, a state-of-the-art chip may require a scanning electron microscope (SEM) with a highly accurate stage. The SEM captures a series of high-resolution images or micrographs, which are assembled (via stitching or mosaicking) to form a complete image of the device. The image is stored as bitmap data.

C. Geometric Description

Next, we extract geometric data from the bitmapped image. The software used for this process converts the image into a geometric data stream format such as GDS-II. This process depends on knowledge about the implementation technology to provide recognition of geometric entities.

D. Transistor Netlist

This step transforms the geometric description into a transistor-level netlist via design rule checkers that examine the geometric data and recognize physical structures such as resistors and transistors.

E. Gate Level Netlist

This level consists of mapping transistor cells to gates. Typically, there are a limited number of mappings, suggesting that a pattern-matching

approach is well suited for automating this process. However, the automation approach must be capable of performing the mapping in the presence of elements that have no logical function—elements that boost a device's output without affecting the logic.

F. Module Level Description

In this step we need to derive a module-level description from the gate-level netlist.

G. Register Transfer and Behavioral Descriptions

Subsequent abstraction of the module-level description produces a register-transfer-level description. Still further abstraction results in a behavioral description. At present, however, these last two levels in the reverse-engineering hierarchy are beyond technological capabilities.

5. CASE STUDY: THE ISCAS 85 BENCHMARK

The techniques presented in section 2 have been used in reverse engineering the ISCAS-85 benchmark circuits in (Hayes and Hansen 1999). In this section we present the circuit for the most complex circuit of this benchmark, which is 34-bit adder and magnitude comparator with input parity checking. The number of gates for this circuit is 3512.

Statistics: 207 inputs; 108 outputs; 3512 gates

Function: 34-bit adder and magnitude comparator with input parity checking

This benchmark circuit contains a 34-bit adder (M5), a 34-bit magnitude comparator (M8) using another 34-bit adder, and a parity checker (M9). Each of the XA, YA, and YB buses is fed by a set of 2:1 multiplexers controlled by the Sel input. Bits 31-22 of XA and YB can be set to logic 0 with the Mask input. The two adders M5 and M8 are identical, and are of carry select type, as are those of c5315. They consist of alternating 4- and 5-bit blocks, with the last block being 2 bits. The comparator (M8) of this benchmark is similar to that of c2670. It performs the comparison YB>XB (if Sel=0) or YB>!YA1 (if Sel=1) by calculating YB+!XB (if Sel=0) or YB+!YA1 (if Sel=1) (Note: the input bus YA1 is assumed to be inverted). The comparator has an output (CoutY) for the whole 34-bit inputs as well as an output (CoutY_17) for the 17-bit portion of its inputs. Module M7 calculates the parity for the following four parts of the adder output SumX: SumX[8:0], SumX[17:9], SumX[26:18], SumX[33:27]. Module M9 appears to be a type of sanity checker that calculates the AND of the parities of all its inputs.

Models:

- I. Original ISCAS gate-level netlist
  - in ISCAS-89 format
  - in Verilog
- II. Verilog hierarchical netlist (functionally equivalent to I)
- III. Verilog flat netlist (flat version of II; functionally equivalent to I, but with minor structural differences)

Evaluation of Reverse Engineering of ISCAS 85 Benchmark:

1. The reverse engineering reported in (Hayes and Hansen 1999) starts with a gate level towards higher level. This is different from starting from a physical chip and extracting transistor information then synthesis gate level information.
2. A circuit of 3512 is a very small to circuit compared to complex chips today that contain millions of transistors which contains millions of transistors.

6. A REAL LIFE EXAMPLE OF REVERSE ENGINEERING: THE REDESIGN OF AWACS RADER SYSTEM BY THE AIR FORCE

In August 1997, the Air Force awarded Northrop Grumman Corporation a proof-of-concept project aimed at capturing the functionality of the E3 Airborne Warning and Control System (AWACS) radar system hardware in VHDL. The Air Force Research Laboratory Materials and Manufacturing Directorate and Northrop Grumman funded this effort jointly. The project evaluated the cost-effectiveness of describing the AWACS radar synchronizers' functions in VHDL code and using the VHDL model to redesign circuit card assemblies plagued by parts obsolescence.

During the AWACS' long life cycle, designers have developed several configurations of its AN/APY-1 and AN/APY-2 synchronizers. The current synchronizer is a two-level card cage that resides in the radar's analog cabinet. It consists of 29 circuit

card assemblies, of which 18 are unique styles and 17 contain a large number of obsolete components, making them unsupportable or irreparable.

Northrop Grumman successfully developed a process to capture the AWACS synchronizer functionality with VHDL code. Using the code, they needed less time than usual to redesign each assembly. Also, they could use the latest VHDL model of the hardware as a baseline when inserting new technology. Another advantage was that one VHDL design could replace multiple circuit card assemblies that could not be repaired and for which no spares were available. For approximately the same cost as replacing the single, failed circuit card assembly, a replacement containing the functionality of a whole group of assemblies could be inserted into the system. The smaller number of assemblies would cost less to procure and the new system would be more reliable.

Cost Analysis

Item	Cost
Saving per card	$470,000
Saving for 33-AWACS fleet	$15,100, 000
Cost for redesign each board using current technology	$250,000
Cost for redesign the 17 board in the system	$4,250,000
Cost of the Reverse Engineering of the board	**$1,000,000**
Saving per system	**$3,250,000**

The results of this proof-of-concept project will serve as a model for further reducing the number of circuit card assemblies in the AWACS radar. The process model Northrop Grumman used to develop the VHDL designs is applicable to all defense systems. More details of this example can be found at (Stogdill 1999).

SUMMARY AND CONCLUSIONS

In this paper we have presented an overview of the process of reverse engineering of chips. We have discussed the steps involved in this process. We have examined two case studies. In the first case we examined the reverse engineering process of ISCAS-85 benchmark. In the second case we examined the reverse engineering of the AWACS radar System.

Reverse Engineering of the ISCAS-85 benchmark starts with a gate level towards higher level. This is different from starting from a physical chip and extracting transistor information then synthesizing gate level information. Also, The most complex circuit used has 3512 gates, which is a very small to circuit.

Reverse engineering of the AWACS Rader System by the Air Force was a proof-of-concept project aimed at capturing the functionality of the E3 Airborne Warning and Control System (AWACS) radar system hardware in VHDL. The cost of reengineering the board was $1,000,000.

REFERENCES

Chikofsky, E. J., Cross II, J. H. (1990). Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, January 1990, 13-17.

Chisholm, G., Eckmann, S. T., Lain, C. M., Veroff, R. L.(1999) Understanding Integrated Circuits. *IEEE Design and Test of Computers,* April 1999, 24-34.

Cifuentes, C., Fitzgerald, A. (1999) Is Reverse Engineering Always Legal? *IT Professional,* March 1999, 42-48.

Doom, T., White, J., Wojcik, A., Chisholm, G. Identifying High-Level Components in Combinational Circuits. (1998) Great Lakes Symposium on VLSI 98, Michigan, February 1998, 313.

Hayes, J. P., Hansen, M. C., Yalcin, H. (1999) Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering. *IEEE Design and Test of Computers* , July 1999, 72-80.

Jarzabek, S., Woon, I. (1997). Towards a precise description of reverse engineering methods and tools. *1st Euromicro Working Conference on Software Maintenance and Reengineering,* Singapore, March 1997.

Tummala, R. R., Madisetti, V. J. (1999) System on Chip or System on Package. *IEEE Design and Test of Computers*, April 1999, 48-56.

Madisetti, V. K., Jung, Y. K., Khan, M. H., Kim, J., and Finnessy, T. (1999) Reengineering Legacy Embedded Systems. *IEEE Design and Test of Computers*, April 1999, pp. 38-47.

Mueller-Glaser, K. D., Lehmann, G., Wunder, B. (1996). Basic Concepts for an HDL Reverse Engineering Tool-Set.1996 International Conference on Computer-Aided Design (ICCAD '96), Germany, November 1996, 0134.

Muller, H. A., Jahnke, J. H., B. Smith, D. B. Storey, M. A. Tilley, S. R., and Wong, K. Reverse Engineering: A Roadmap.

Prinetto, P., Vietti, R., Rudnick, E. M., Corno, F., Ellis, A. (1998) Fast Sequential Circuit Test Generation Using High-Level and Gate-Level Techniques. *Design Automation and Test in Europe,* February 1998, pp. 570.

Stogdill, R. C. (1999) Dealing with Obsolete Parts. (1999) *IEEE Design and Test of Computers,* April 1999, pp. 17-25.