

Reverse Engineering of VLSI Chips: A Roadmap

Tarek Sobh, M. Khaled Elleithy and Sarosh Patel

Department of Computer Science and Engineering, University of Bridgeport,
 Bridgeport, CT 06604

Abstract: The reverse engineering process for 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 study, we present an overview of the process of reverse engineering VLSI chips. The study outlines the steps involved in the process of reverse engineering chips as well as the different techniques used to extract the functionality of these chips. Furthermore, the paper presents two case studies for reverse engineering VLSI chips.

Key words: Architectures, models, VLSI, multilevel structures, geometric properties, benchmark examples

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 and Cross, 1990). In computer hardware, designers have used reverse engineering to extract gate-level models from transistor circuits (Madisetli *et al.*, 1999).

Madisetli *et al.* (1999) introduced the rationale for reengineering legacy embedded systems (Madisetli *et al.*, 1999). 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 from (Chikofsky and Cross, 1990) shows the relationship between requirements, design and implementation and where forward engineering and reverse engineering fit. Chikofsky and Cross defined the following terms (Chikofsky and Cross, 1990):

- 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.

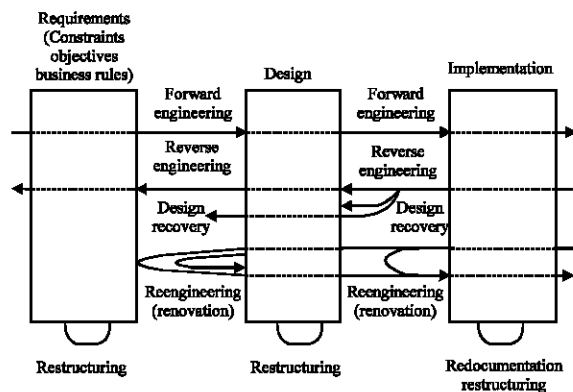


Fig. 1: Relationship between terms (Chikofsky and Cross, 1990)

- 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.

This study provides a literature survey and presents the most up-to-date reported research in the area of reverse engineering chips. The first case is the reverse engineering for the ISCAS-85 benchmark. The second case is the reverse engineering for the AWACS Radar System by the Air Force which is a project the Air Force awarded to 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.

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 study 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, 1999) the authors define and relate six terms: forward engineering, reverse engineering, re-documentation, 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 and promise higher performance at a 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 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 that 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 formalize 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 the use of 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 (Maller *et al.*). 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 the 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, cost-effectively.

Reverse engineering has been heralded as one of the most promising technologies to combat this legacy systems problem. Muller *et al.* (1996) 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 (Muller *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 *et al.*, 1999) about 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 sub circuit structure within the layout. These syntactic techniques fail to identify functionally equivalent sub circuits, which are differently implemented, optimized, or otherwise obfuscated. In (Doom *et al.*, 1998) a mechanism for identifying functionally equivalent sub circuits 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 ensure Air Force mission readiness and increase nagging obsolescence (Stogdill, 1999). EPOI is developing management and 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 key areas of study:

- Parts Obsolescence Management and Re-engineering Tools,
- The Application of Commercially Manufactured Electronics (ACME) and
- 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.

Reverse engineering techniques: Hayes and Hansen have defined the following techniques for the reverse engineering of hardware (Hayes *et al.*, 1999).

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 sub circuit 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 function: With a few structural clues to a sub circuit'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 function: 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.

The reverse engineering process: Chisholm *et al.* (1999) They suggested the following outline for the reverse-engineering process (Tummala and Madisetti, 1999).

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.

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.

Geometric description: Next, geometric data is extracted 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 the knowledge about the implementation technology to provide recognition of geometric entities.

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.

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 but boost a device's output without affecting the logic.

Module level description: In this step, a module-level description is to be derived from the gate-level netlist.

Register transfer and behavioral descriptions: Subsequent abstraction of the module-level description produces a register-transfer-level description. Further abstraction results in a behavioral description. At present, however, these last two levels in the reverse-engineering hierarchy are beyond typical technological capabilities.

Case study: The ISCAS 85 benchmark: The techniques presented were used in reverse engineering the ISCAS-85 benchmark circuits in (Hays *et al.*, 1999). We present the most complex circuit of this benchmark, which is a 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 given in Fig. 2 contains a 34 bit adder (M5-Fig. 3), a 34 bit magnitude comparator (M8-Fig. 4) using another 34 bit adder and a parity checker (M9-Fig. 5). 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.

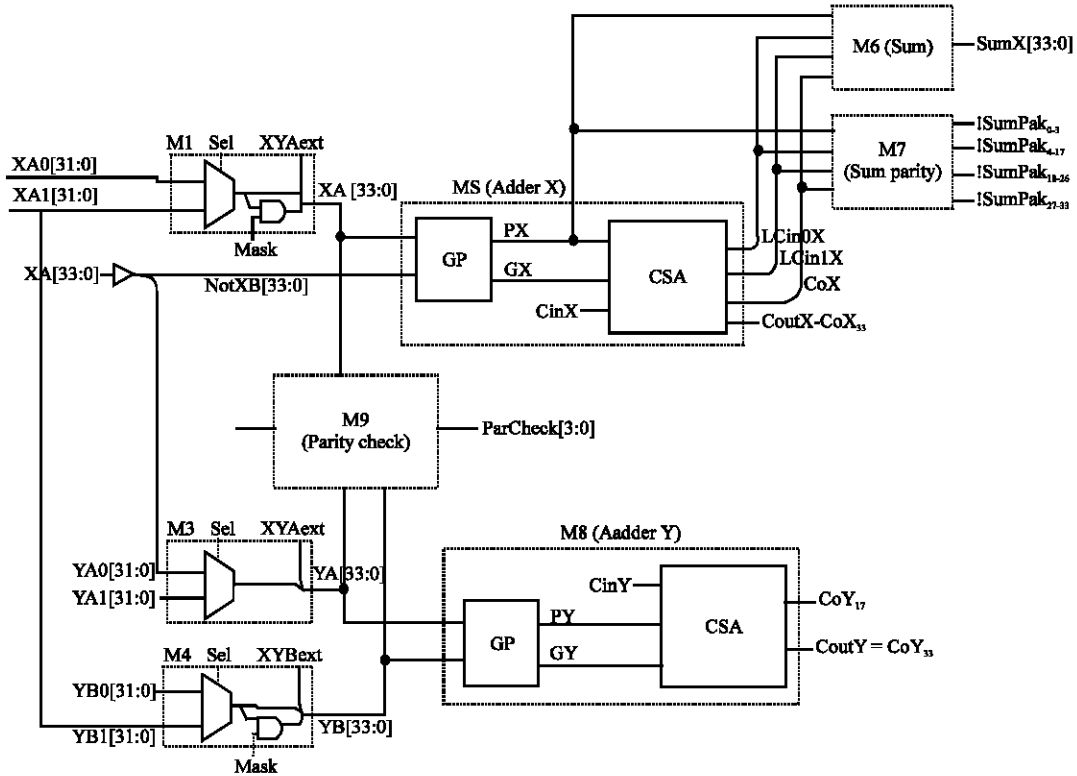
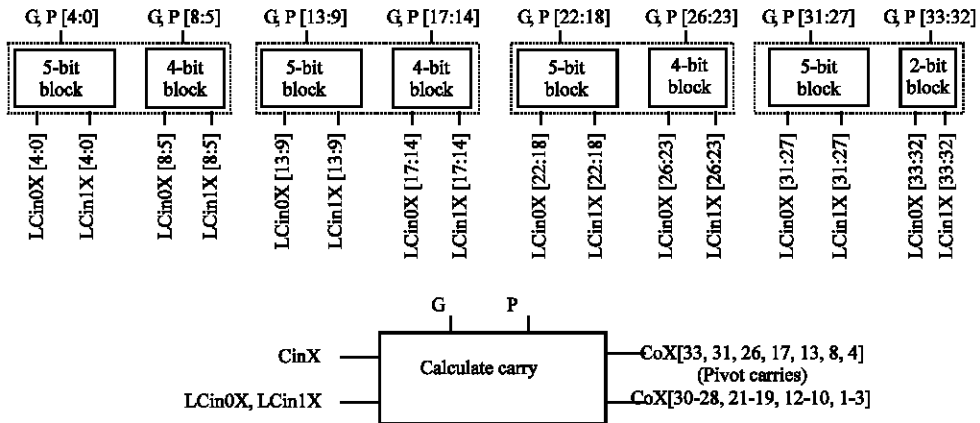


Fig. 2: ISCAS-85 benchmark circuit



Note: Carry[i] = carry from bit position i to (i+1)

Model M6 (SUM)

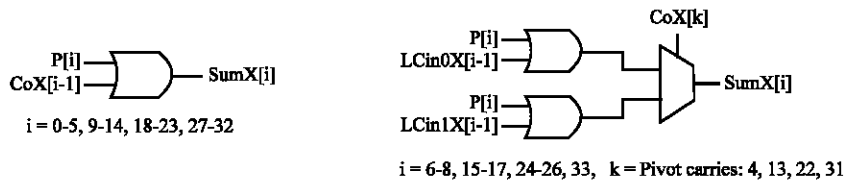


Fig. 3: MS/CSA and sum modules of the benchmark circuit

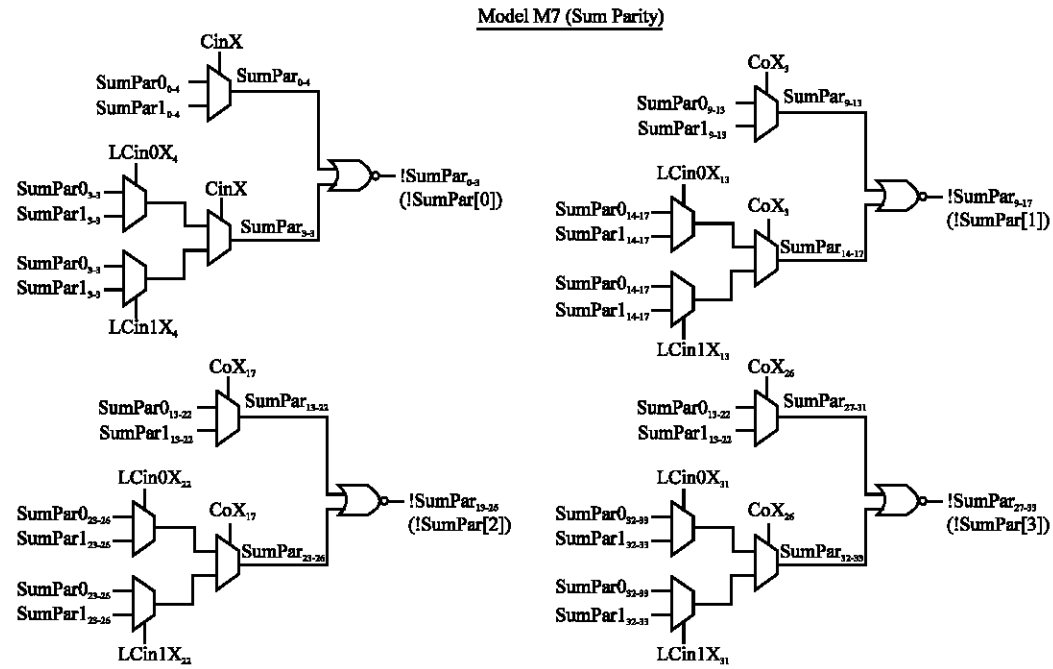


Fig. 4: Circuit of M7 module of the benchmark circuit

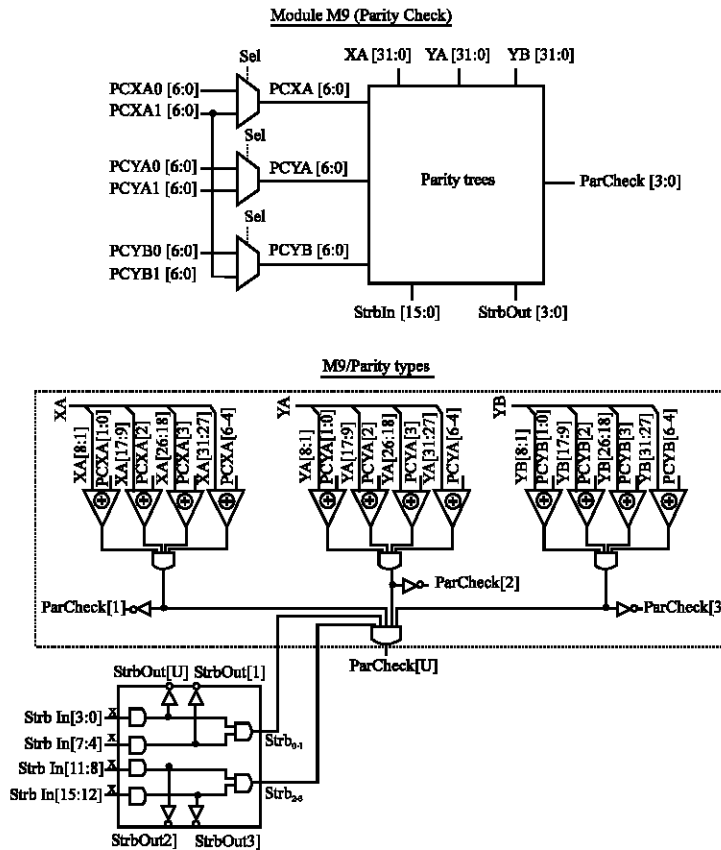


Fig. 5: Circuit of M9 module of benchmark

Models used

- 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)

Detailed bus definitions

- XA:

- $XA [21:0] = XA0 [21:0]$ if Sel = 0, $XA1 [21:0]$ if Sel = 1
- $XA[31:22] = XA0[21:0]. Mask$ if Sel = 0, $XA1[21:0]. Mask$ if Sel = 1
- $XA[32] = XA[33] = XYAext$

- NotXB:

- $NotXB[0] = ! XB[0]$ if Sel = 1, logic 0 if Sel = 0
- $NotXB[31:1] = ! XB[31:1]$
- $NotXB[33:32] = ! (XB[33:32]. XYBext)$

- YA:

- $YA[0] = logic 1$ if Sel = 0, $YA1[0]$ if Sel = 1
- $YA[31:1] = NotXB[31:1]$ if Sel = 0, $YA1[31:1]$ if Sel = 1
- $YA[32] = YA[33] = XYAext.$

- YB:

- $YB[21:0] = YB0[21:0]$ if Sel = 0, $XA1[21:0]$ if Sel = 1
- $YB[31:22] = YB0[21:0]. Mask$ if Sel = 0, $XA1[21:0]. Mask$ if Sel = 1
- $YB[33:32] = YB0[33:32] + ! XYBext$

- XBbuf[33:0] = XB[33:0]

- PCYA0buf[3:0] = { PCYA0[6], PCYA0[3], PCYA0[2], PCYA0[0] }

Table 1a and b shows the detailed inputs and outputs and the corresponding netlist numbers.

Evaluation: The reverse engineering process reported in (Hayes *et al.*, 1999) starts with gate level representation towards higher levels representations. This process is different from starting from a physical chip and extracting transistor information then synthesizing gate level information.

A circuit of 3512 gates is a very small circuit compared to complex chips that contain millions of transistors.

Table 1a: Inputs and corresponding netlist numbers

Input	Netlist No.
XA0[31:0]	213, 214, 215, 216, 209, 153, 154, 155, 156, 157, 158, 159, 160, 151, 219, 220, 221, 222, 223, 224, 225, 226, 217, 231, 232, 233, 234, 235, 236, 237, 238, logic 0
XA1[31:0]	10 *{logic 1}, 135, 144, 138, 147, 66, 50, 32, 35, 47, 121, 94, 97, 118, 100, 124, 127, 130, 103, 23, 26, 29, 41
XB[33:0]	1496, 1492, 1486, 1480, 106, 1469, 1462, 2256, 2253, 2247, 2239, 2236, 2230, 2224, 2218, 2211, 4437, 4432, 4427, 4420, 4415, 4410, 4405, 4400, 4394, 3749, 3743, 3737, 3729, 3723, 3717, 3711, 3705, 3701
YA1[31:0]	88, 112, 87, 111, 113, 110, 109, 86, 63, 64, 85, 84, 83, 65, 62, 61, 60, 79, 80, 81, 59, 78, 77, 56, 55, 54, 53, 73, 75, 76, 74, 70
YB0[33:0]	2204, 1455, 166, 167, 168, 169, logic 1, 173, 174, 175, 176, 177, 178, 179, 180, 171, 189, 190, 191, 192, 193, 194, 195, 196, 187, 200, 201, 202, 203, 204, 205, 206, 207, logic 0
!Sel	18
CinX, CinY	4526, 89
Mask=!(Mask1+!Mask2)	112, 9
Mask1, Mask2	
XYAext, XYBext	38, 4528
PCXA0[6:0]	logic 1, 211, 212, 161, 227, 239, 229
PCXA1[6:0]	3*{logic 1}, 141, 115, 44, 41
PCYA0[6:0]	1459, 1496, 1492, 2208, 4398, 3701, 3698,
PCYA1[6:0]	114, 2204, 1455, 82, 58, 70, 69
PCYB0[6:0]	170, 164, 165, 181, 197, 208, 198
StrbIn[15:0]	199, 188, 172, 162, 186, 185, 182, 183, 230, 218, 152, 210, 240, 228, 184, 150
MiscIn[7:0]	57, 5, 133, 134, 1197, 15, 163, 1

Table 1b: Outputs and corresponding netlist numbers

Output	Netlist No.
SumX[33:0]	469, 471, 327, 330, 333, 336, 324, 298, 301, 304, 307, 310, 313, 316, 319, 295, 347, 350, 353, 356, 359, 362, 365, 368, 344, 376, 379, 382, 385, 388, 391, 394, 397, 373
!SumPar[3:0]	338, 321, 370, 399
CoutX1, CoutX2	(270, 246) *, (273, 276) *
CoutY1, CoutY2	(258, 264) *, 249
CoutY_17	252
ParCheck[3:0]	416, 414, 412, 418
XBbuf[33:0]	440, 438, 442, 444, 446, 448, 436, 480, 482, 484, 486, 488, 490, 492, 494, 478, 524, 526, 528, 530, 532, 534, 536, 538, 522, 544, 546, 548, 550, 552, 554, 556, 558, 542
StrbOut	410, 408, 406, 404
PCYA0buf[3:0]	450, 496, 540, 560
MiscOut[5:0]	402, 289, 292, 279, 278, 2

* (a,b): a,b are identical outputs

A real life example of reverse engineering: The redesign of the awacs radar 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

Table 2: Cost analysis

Item	Cost
Saving per card	\$470,000
Saving for 33-AWACS fleet	\$15,100,000
Cost for redesigning each board using current technology	\$250,000
Cost for redesign the 17 boards	\$4,250,000
Cost of reverse engineering the board	\$1,000,000
Saving per system	\$3,250,000

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. Table 2 shows the cost items in the reverse engineering of the board.

The results of this proof-of-concept project serves 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 in (Stogdill, 1999).

CONCLUSIONS

In this study we present an overview of the reverse engineering process of VLSI chips. We discuss the steps involved in this process. Two case studies are reported. In the first study, we examine the reverse engineering process of ISCAS-85 benchmark.

In the second study, we examine the reverse engineering of the AWACS radar system.

Reverse Engineering the ISCAS-85 benchmark starts with a gate level representation and advances in steps from lower levels to higher levels of representation. This process is different from starting with a physical chip, extracting transistor information then synthesizing gate level information. Furthermore, the most complex circuit used has 3512 gates, which is a very small circuit compared with current chips that contain millions of transistors.

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.

Although Reverse engineering has experienced increased attention since the mid-1990s in the United States Department of Defense (DoD) as well as the commercial arena, most reported approaches in literature are ad-hoc. The future of reverse engineering will certainly include automated methods of reverse engineering using systems to measure, manufacture and test components. Finally, from a systems point of view, a reverse engineered component will be more reliable than the older design which will increase system reliability, shorten design time and improves output productivity.

REFERENCES

- Chikofsky, E.J. and J.H. Cross, 1999. Reverse engineering and design recovery: A Taxonomy. *IEEE Software*, pp: 13-17.
- Chisholm, G., S.T. Eckmann, C.M. Lain and R.L. Veroff, 1999. Understanding Intedrated Circuits, *IEEE Design and Test of Computers*, pp: 24-34.
- Madiseti, V.K., Y.K. Jung, M.H. Khan, J. Kim, J. and T. Finnessy, 1999. Reengineering legacy embedded systems. *IEEE Design and Test of Computers*, pp: 38-47.
- Tummala, R.R. and V.J. Madiseti, 1999. System on chip or system on package. *IEEE Design and Test of Computers*, pp: 48-56.
- Jarzabek, S. and I. Woon, 1997. Towards a Precise Description of Reverse Engineering Methods and Tools. 1st Euromicro Working Conference on Software Maintenance and Reengineering, Singapore.
- Muller, H.A., J.H. Jahnke, D.B. Smith, M.A. Storey, S.R. Tilley and K. Wong, *Reverse Engineering: A Roadmap*.

- Mueller-Glaser, K.D., G. Lehmann and B. Wunder, 1996. Basic Concepts for an HDL Reverse Engineering Tool-Set. International Conference on Computer-Aided Design (ICCAD '96), Germany.
- Hayes, J.P., M.C. Hansen and H. Yalcin, 1999. Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering. IEEE Design and Test of Computers, pp: 72-80.
- Doom, T., J. White, A. Wojcik and G. Chisholm, 1998. Identifying High-Level Components in Combinational Circuits. Great Lakes Symposium on VLSI 98, Michigan, pp: 313.
- Prinetto, P., R. Vietti, E.M. Rudnick, F. Corno and A. Ellis, 1998. Fast Sequential Circuit Test Generation Using High-Level and Gate-Level Techniques. Design Automation and Test in Europe, pp: 570.
- Stogdill, R.C., 1999. Dealing with Obsolete Parts. IEEE Design and Test of Computers, pp: 17-25.