◈ Application Papers

Fault-Tolerant Analysis of Redundancy Techniques in VLSI Design Environment

Cho, Jai Rip*

Abstract

The advent of very large scale integration(VLSI) has had a tremendous impact on the design of fault-tolerant circuits and systems. The increasing density, decreasing power consumption, and decreasing costs of integrated circuits, due in part to VLSI, have made it possible and practical to implement the redundancy approaches used in fault-tolerant computing.

The purpose of this paper is to study the many aspects of designing fault-tolerant systems in a VLSI environment. First, we expound upon the opportunities and problems presented by VLSI technology. Second, we consider in detail the importance of design mistakes, common-mode failures, and transient faults in VLSI. Finally, we examine the techniques available to implement redundancy using VLSI and the problems associated with these techniques.

1. INTRODUCTION

In fact, today's technology often allows multiple computers to be a single Integrated Circuits (ICs) so that fault detection or fault tolerance is attained within the IC itself. IC simply implies that more than one component is placed on a single piece of semiconductor material, and all of the components and associated wiring are an integral part of the IC and cannot be physically separated [8]. An IC is in contrast to discrete devices, where each component is packaged separately and all wiring is external to the component itself. An IC, however, might contain a number of transistors, diodes, and other elements on a single piece of semiconductor material and placed in a single package. The actual semiconductor material in an IC is called a chip or die and can have an area of several hundred square mils. The chip is usually placed in ceramic or plastic packages that contain metal pins to provide electrical connections.

The number of components placed on an IC is representative of the level of integration [3]. When IC technology was first introduced only one or two gates could be placed on a single chip. Today, however, it is possible to place more than 100,000 gates on a single

*Dept, of Industrial Engineering, Kyung-Hee University, Korea

chip. Small-Scale Integration(SSI) refers to an IC that contains fewer than 10 logic gates or, equivalently, fewer than 30 transistors. Medium-Scale Integration(MSI) refers to ICs containing between 10 and 100 logic gates (30 to 300 transistors). Large-Scale Integration(LSI) implies that an IC contains 100 to 10,000 logic gates (300 to 30,000 transistors). Finally, Very-Large-Scale Integration (VLSI) represents ICs containing more than 10,000 logic gates (more than 30,000 transistors).

The precise number of logic gates used in the definitions of SSI, MSI, LSI, and VLSI are not universally accepted. However, it does appear that most people accept the 10,000 gate level as the beginning of VLSI because ICs with fewer than 10,000 gates can be manufactured with the same technology, whereas those containing more than 10,000 gates typically require new technology.

There are a number of advantages to using ICs rather than discrete devices, and many of the advantages are extremely important in the design of fault-tolerant circuits and systems. Consequently, the three most important advantages of using IC technology in the design of fault-tolerant systems are : (1) Reduced power consumption (2) Increased reliability (3) Decreased size and weight.

First, when the majority of the connections in a system are made on a single chip, the parasitic capacitances are reduced, and the power required to drive signals throughout the system is also reduced. Consequently, over all power consumption is minimized. Second, placing components on a single chip reduces the probability of loose connections, broken wires, and poor solder joints, therefore decreasing the overall failure rate. Finally, integration simply decreases size and, as a result, physical weight and volume. Many of the critical applications that require fault tolerance also mandate minimum weight and volume.

2. FAILURE MODES IN VLSI TECHNOLOGY

As might be expected, it would be extremely difficult to list all the possible causes of failure in an integrated circuit simply because the list would be extremely long. In addition, the list of faults would undoubtedly be different for bipolar devices than for metal-oxide semiconductor(MOS) ones. Finally, we would expect that changes in technology would produce subsequent changes in the distribution of faults among the various categories ; for example, the common faults in SSI technology may be insignificant in LSI and VLSI technology.

In this paper, we want to discuss the common categories of faults that occur in integrated circuits and investigate the changes that we have seen in the distribution of faults as we have progressed from SSI to MSI and to even higher levels of integration. Note that the list of faults is in no way exhaustive ; it is intended to illustrate the types of faults that can occur.

At the highest level, fault in ICs can be divided into two primary categories ; (1) those resulting from the manufacturing process and (2) those resulting from wear-out or other phenomena in the field [9]. During the manufacturing process, scratches, dust, or other foreign particles often corrupt a device.

Likewise, the improper packaging of the device or the bonding of the leads to the actual semiconductor material can result in a failed device. While a device is in operation, electromagnetic fields can wear down an insulator and cause a short to occur.

Similarly, moisture can become trapped in the IC and produce metal corrosion that leads to broken lines or degraded conductivity of a particular device.

We will consider seven categories when examining fault distributions ; metal systems, diffusion, foreign material, oxide, package and bonding, chip mounting, and misapplication.

3. DISTRIBUTION OF FAULTS IN VLSI TECHNOLOGY

Now that we have identified some of the major causes of faults in ICs, we will examine the percentage of total faults that is the results of each individual cause. We will examine this fault distribution as a function of the level of integration (SSI, MSI, and so on). Unfortunately, very little data is available on advanced VLSI devices because such devices are just now beginning to be manufactured and used in large quantities. However, by examining the data for SSI, MSI, and LSI, we can see a number of trends that clearly indicate the dominant sources of faults in higher levels of integration.

Table 1 shows the fault distribution percentages for bipolar ICs as a function of the technology [9].

First, the percentage of faults resulting from misapplication has been steadily decreasing as a function of the integration level. In SSI technology, misapplication accounted for 35% of the faults, whereas in LSI technology misapplication accounts for slightly more than 5% of the faults. Several factors can explain the marked decrease in the percentage of misapplications.

(Table 1) Fault Distribution Percentages for bipolar ICs

Failure Mode	SSI(%)	MSI(%)	LSI(%)
Metal Systems	9.5	17.5	27.0
Diffusion	8.0	12.0	24.5
Foreign Material	4.0	11.0	12.0
oxide	17.5	20.0	13.5
Package & Bonding	13.5	7.0	4.0
Chip Mounting	5.5	3.0	1.5
Misapplication	35.0	16.0	5.5
Miscellaneous	7.0	13.5	12.0

Second interesting feature apparent in Table 1 is the significant decrease in the percentage of faults due to packaging and bonding. Over 13% of all faults in SSI technology were the result of packaging or bonding problems, but the same problems cause only about 4% of the faults in LSI technology. The decrease in packaging and bonding problems indicates an improvement in the quality of the packaging and bonding techniques and perhaps an increase in the faults that are occurring on the IC itself.

Finally, note in Table 1 the significant increase in the percentage of faults associated with the actual semiconductor material of the IC. Suppose we consider metalization,

diffusion, foreign material and oxide problems as internal faults associated with the semiconductor material itself. Also, suppose we consider packaging, bonding, mounting, and misapplication as faults resulting from external factors. Table 1 shows that, for SSI technology, internal factors account for approximately 39% of all faults, whereas external factors account for approximately 54%. However, in LSI technology, the internal factors account for over 75% of the faults, whereas the external factors account for approximately 11% of the faults. Consequently, we have seen a significant change in the primary sources of faults as we have gone from SSI to LSI technology

Table 2 shows the fault distribution for LSI MOS devices [9]. Although the failure modes are categorized a little differently in Table 2. it is still easy to see that the dominant cause of faults is due to the internal factors rather than the external factors. In fact, oxide problems account for approximately 33% of all faults in MOS LSI devices. Faults due to the package, bonding, and mounting account for only 10% of the total faults in MOS devices.

(Table 2) Fault distribution percentages for LSI MOS ICs

Failure mode	Percentage of faults LSI MOS technology(%)
Oxide	33
Electrical overstress	15
Electrical	13
Metallization and particles	3
Package and wire	5
Bond and chip mount	5
Photolithographic	5
Other	21

As a results, the data for ICs indicates that more and more faults are resulting from factors unrelated to the packaging, mounting, bonding, or wiring of the IC. Several important points result from knowledge of the change in the fault distribution. First, in the days of SSI and MSI, it did not make sense to incorporate redundancy into the ICs because the most common way for the device to fail was in the bonding or the packaging.

Consequently, internal redundancy would have a very limited impact (if any) on the overall reliability of the device or the system. In addition, the SSI technology did not provide the capability to include any redundant devices on the chip, primarily because of space limitations. In LSI and subsequently VLSI, the packaging, bonding, and wiring are becoming less of a factor, so it is possible to improve the reliability of an IC by incorporating redundancy into the IC itself. Also, the use of LSI and VLSI technology provides the capability to incorporate redundant circuitry in many applications.

4. OPPORTUNITIES AND PROBLEMS PRESENTED BY VLSI

In general, VLSI provides one simple capability : the ability to put more circuitry in a smaller, more reliable, and, in many cases, less expensive package. For the designer of fault-tolerant circuits and systems, the ability to have more circuitry implies that many of the approaches that were previously not cost effective can now be used. For example,

duplicated processors can now be placed on a single chip, whereas previously they required multiple boards or even multiple cabinets. Likewise, triple modular redundancy, quad redundancy, or even higher levels of redundancy can now be practical because of the size, power, and cost savings attributable to VLSI technology.

The basic problem associated with VLSI is that the failure modes are now different. We have already seen, for example, that as the level of integration has increased from SSI to LSI, the common faults have moved from the pins and the package to the semiconductor material. In addition, the increased complexity of the design has increased the probability of design mistakes. Finally, the lower operating voltages of integrated circuits (many VLSI devices use less than 5.0 volts) has decreased the noise margin of the devices and increased the frequency of transient faults.

In this paper, we consider common-mode failures, design mistakes, and the increasing occurrence of transient faults in VLSI circuits.

In general, a common-mode failure occurs when two or more identical modules are affected by faults in exactly the same way at exactly the same time [12]. The problems with such failures are numerous. For example, if two modules in a triple modular redundancy system experience a common-mode failure, the two faulty modules could force the output of the majority voter to become erroneous, even though a majority of the modules (the two faulty ones) would agree. Similarly, a common-mode failure in a duplication with comparison scheme would go undetected because the duplicated modules would produce identical, erroneous results.

A VLSI design environment increases the probability of common-mode failures for several reasons. First, VLSI devices are extremely complex ; therefore, the possibility of design mistakes is increased significantly. Many modern ICs contain hundreds of thousands of logic gates and often require the efforts of tens of designers to complete. The expectation that the design is never going to contain latent design mistakes is often unrealistic.

Second, identical modules can be located very close to one another on a single chip or wafer. Consequently, stuck-type faults can easily affect both modules. For example, it is very reasonable that two lines, one from each of two identical modules, could become physically stuck at the same logical value : the result could be a common-mode failure.

Finally, the small feature sizes and resulting low operating voltages of VLSI devices are increasing the likelihood that external disturbances will impact the operation of the device. For example, an IC can be subjected to radiation or lightning that equally affects all identical modules contained on the IC.

In general, there are four primary causes faults : (1) implementation (2) design mistakes (3) external disturbances (4) random component defects. The fault tolerance approaches considered thus far have ignored the category of design mistakes and have assumed that such causes of faults are handled via fault avoidance techniques. When designs are relatively simple, fault avoidance is easy to accomplish and can be extremely effective in preventing design mistakes. However, with VLSI technology, designs are no longer simple. Single ICs can have hundreds of thousands of gates, and there can be hundreds of ICs within a given system. Clearly, the likelihood of design mistakes is increased when

designing in a VLSI environment.

The problem of design mistakes in VLSI is similar, in many respects, to the problem of latent bugs in software. Because of the large number of paths through many software routines, it is not practical, from cost and time viewpoints, to exhaustively verify the correctness of each possible path. Yet, the failure to exhaustively verify the correctness of the routines creates the possibility that design mistakes go undetected until certain operating conditions occur in the field, at which point it may be too late to correctly handle the problem.

In the design of VLSI - based systems, the design tools (both hardware and software) we use are often comparable in complexity to the system being designed. Consequently, we not only have to worry about design mistakes occurring because of the limitations of the human designers, but we must also be concerned with design mistakes that are a result of problems in the design tools.

5. REDUNDANCY TECHNIQUES IN A VLSI DESIGN ENVIRONMENT

We will consider the following four primary categories dealing with the use of redundancy in a VLSI environment : (1) duplication with complementary logic (2) self-checking circuits (3) reconfigurable arrays (4) yield enhancement.

The use of complementary logic is a technique developed to overcome the problem of common-mode failures.

Complementary logic has been applied in a number of different situations and has proven to be very effective [10]. Self-checking circuits were developed initially to solve the "Who checks the checker? " problem. For example, in duplication with comparison, the comparator is a weak link whose failure can result in the system either erroneously indicating that a fault has occurred or ignoring the occurrence of a legitimate fault. Reconfigurable arrays are becoming more popular as the ability to put large numbers of processors on a single chip or wafer continues to increase. Redundancy is often added to processing arrays to achieve real-time fault tolerance or to improve the probability of initially obtaining an operational array, Finally, redundancy has been used in many cases to improve the yield of VLSI devices by placing spare elements on the IC ; memories are excellent examples of where such techniques have been employed.

(1) Duplication with complementary logic

The problems with duplication with comparison are : (1) the comparator is subject to failure, and (2) the approach relies on the assumption that only one of the duplicated modules will fail at a given time. As we have already seen, the possibility of common-mode failure implies that we cannot safely assume that only one of the two modules will fail. Consequently, we need to modify the design of the duplication with comparison scheme to ensure that the effect of common-mode failures is minimized. Several approaches can be used to help alleviate the problem of common-mode failure in duplication. One technique focuses on minimizing the possibility of identical design mistakes appearing in both modules by requiring that two separate design teams independently develop the two modules[1]. The hope is that any design mistakes that occur will not be

identical because the designers have been working independently. The problem with such an approach is that the expense is often intolerable. The company must provide twice as many designers and design resources. Also, the procedure does not address other causes of common-mode failures such as those occurring during fabrication, packaging, or as a result of external disturbances during normal operation. A second approach relies completely on fault avoidance techniques. Designs are checked and double checked, and the production process is closely monitored to attempt to ensure that the potential causes of common-mode failures are eliminated. The difficulty here is that fault avoidance techniques are seldom 100% effective, and once again there is no consideration for problems that arise during the normal operation of the circuit or system.

(2) Self-checking logic

The concept of self-checking logic has increased in popularity because of the traditional "checking the checker" problem. In many designs that use coding schemes or duplication with comparison, it is necessary to compare the outputs of two modules or to verify that the output is a valid code word. The basic problem with such techniques, as we have seen, is the reliance of the approaches on the correct operation of comparators or code checkers. If the code checker fails, for example, the system can indicate that an error exists when in fact one does not, or the system can fail to detect a legitimate error that occurs. In many applications either condition is unacceptable. One possible solution is to design comparators and code checkers that are capable of detecting their own faults. Consequently, the concept of self-checking logic has been developed.

In general, a circuit is said to be self-checking if it has the ability to automatically detect the existence of a fault without the need for any externally applied stimulus [6]. In other words, a self-checking circuit determines if it contains a fault during the normal course of its operations. Self-checking logic is typically designed using coding techniques similar to those discussed under information redundancy. The basic idea is to design a circuit that, when fault free and presented a valid input code word, will produce the correct output code word. If a fault exists, however, the circuit should produce an invalid output code word so that the existence of the fault can be detected. To formalize the concept of self-checking logic, we will define fault secure, self-testing, and totally self-checking. In each definition, note that we are considering circuits designed to accept code words on their input lines and produce code words on their output lines. A circuit is said to be fault secure if any single fault within the circuit results in that circuit either producing the correct code word or producing a noncode word, for any valid input code word[6]. In other words, a circuit is fault secure if the fault either has no effect on the output or the output is affected such that it becomes an invalid code word. A circuit would not be fault secure, for example, if a fault resulted in the output becoming incorrect but still a valid code word. A circuit is said to be self-testing if there exists at least one valid input code word that will produce an invalid output code word when a single fault is present in the circuit[6]. In other words, a circuit is self-testing if each single fault is detectable since a fault that resulted in valid output code words for each possible input code word would be undetectable.

Finally, a circuit is said to be totally self-checking if it is both fault secure and

self-testing[6]. The fault secure property guarantees that the circuit will either produce the correct code word output or an invalid code word output when any single fault occurs. The self-testing property guarantees that there is at least one input code word that will produce an invalid code word output from the circuit when a fault is present. In summary, a circuit is totally self-checking if all single faults are detectable by at least one valid code word input, and when a given input combination does not detect the fault, the output is the correct code word output.

(3) Reconfigurable array structures

Perhaps one of the most promising areas of research in the fault-tolerant computing field is the design and analysis of array structures for highly parallel and high-speed processing. The goal of a parallel processing system is to exploit the fact that the individual operations required in a given calculation do not necessarily have to be performed sequentially. As a simplistic example, suppose that a system must sample eight temperatures, convert each temperature to degrees Celsius, and display each temperature on one of eight separate display units. One design approach for this simple problem would be sequential in nature and would use a single processor that samples the temperatures in sequence, performs the conversion of each temperature in sequence, and provides the results to the appropriate display unit, again in sequence. If, however, you wanted to sample the temperatures as often as possible, you might consider a structure whereby eight processors are used, and each temperature is sampled, processed, and displayed in parallel. Theoretically, the parallel approach would be capable of sampling each temperature eight times more frequently than the sequential technique. The advent of VLSI technology allows efficient and effective implementations of array structure for parallel computations. A single chip, for example, might contain hundreds or thousands of processing elements connected in a near-neighbor structure. Also, wafer-scale integration(WSI) might allow hundreds or thousands of chips to be interconnected[4]. The potential for such designs seems almost unlimited. Two fundamental problems, however, must be solved via fault tolerance techniques. First, a wafer or chip manufactured with thousands of processing elements will likely contain failed elements as soon as it comes off the production line. If the design depends on all the elements being operational to be useful, you may never obtain a useful device. So, the design must be performed such that the faulty elements within the array can be bypassed and the fault-free elements interconnected to achieve a functional array. Second, many applications will require that the array be capable of handling elements failures that occur during the normal operation of the array. In some applications, the array will be allowed to shut down to perform a reconfiguration, whereas in other cases the array must continue its normal processing during the reconfiguration proces. Three specific types of reconfiguration can be identified: (1) fabrication-time reconfiguration that is performed immediately after manufacturing to produce an operational processing array, (2) compile-time reconfiguration that is performed before each use of the array, but not while the array is performing its normal operations, and (3) real-time reconfiguration that is performed while the array is in operation and continues to provide uninterrupted performance of its normal operations[5]. The most difficult reconfiguration to perform is real-time reconfiguration. and the easiest may very well be fabrication-time

reconfiguration.

(4) Redundancy to enhance yield of VLSI circuits

An important application of redundancy techniques in a VLSI design environment is to improve the yield of integrated circuits. For our purposes, yield is defined as the number of fabricated devices that work correctly divided by the total number of fabricated devices [2]. For example, if a company fabricates 50,000 processor ICs and finds that 5,000 of those processors perform their functions correctly, the yield is 5,000 / 50,000, or 0.1. Expressed as a percentage, the yield in this example is 10%.

In many VLSI applications, it is not unusual to experience yields on the order of 10%, or perhaps even less. Consequently, the cost of manufacturing a circuit can become prohibitive since ten ICs must be produced to obtain one that works. A number of investigators have examined approaches that attempt to use redundant elements to replace failed ones so that a faulty circuit can be repaired and made usable [7]. For example, it is quite common in the semiconductor memory industry to include spare rows or columns of storage elements such that a faulty row or column can be replaced using the fabrication-time reconfiguration techniques described in the previous section for array-type structures. If used solely for yield improvement, the reconfiguration is performed immediately after fabrication and is irreversible. The purpose of this parts is to examine the improvement that can be obtained in circuit yield by using redundancy. The specific redundancy techniques that may be used for yield improvement include many of the approaches already considered in the previous parts. For example, the most common application of yield improvement techniques is memory ICs, and the most common redundancy technique is the addition of redundant rows or columns that can be used to replace faulty rows or columns.

Several yield models have been developed in the past, and the majority are based on a similar set of assumptions[2][11]. First, all failures on the IC are assumed to be the result of what are called spot defects. Spot defects are localized to a given area within an IC and are assumed to affect no more than one module within that IC. The specific size of the module is not important here; the important point is that it is assumed that the defect is not large enough to span more than one module. However, a module may have more than one spot defect. If we assume, for example, that sport defects affect no more than one module, we can use duplicate modules to provide redundancy, and a single defect cannot render both modules faulty. Spot defects are in contrast to area defects, which affect complete sections of a chip or wafer. Specifically, an area defect might result in the failure of several modules within the same chip or several chips on the same wafer.

The second major assumption in yield modeling is that any single spot defect will result in the chip being inoperative unless some type of redundancy is included. In other words, all defects are considered to be fatal defects in nonredundant chips. In practice, a defect might result in performance degradations that do not render the chip completely useless. The yield models, however, assume that any single defect in a nonredundant chip results in the chip being completely inoperative.

The final major assumption in yield modeling is that spot defects are randomly distributed, in a physical sense, throughout a chip and a wafer. In other words, the number of defects contained in any one area of a device is a random variable.

6. CONCLUSIONS

This paper has presented the technology methods to the extremely vital topic of designing fault-tolerant systems in a VLSI environment, using the advantages of VLSI and accounting for its disadvantages.

REFERENCES

[1] Avizienis,A.,(1982),"Design diversity-The challenge of the eighties," Proceedings of the 12th Annual International Symposium on Fault-Tolerant Computing, June 22-24, Santa Monica, Calif., pp.44-45.

[2] Bernard, J.,(1988), "The IC Yield Problem : A Tentative Analysis for MOS/SOS Circuits," IEEE Transactions on Electron Devices, Vol. ED-25, No. 6, pp. 939-944.

[3] Hodges, D. A. and Abraham, J. A.,(1988), Analysis and design of digital integrated circuits, McGraw Hill, Inc., New York.

[4] Hwang, K. H. and J. A. Abraham,(1984),"Algorithm-based fault tolerance for matrix operations, " IEEE Transactions on Computers, Vol.C-33, No.6, pp.518-528.

[5] Kung, S. Y., S. C. Lo, S. N. Jean and J. N. Hwang,(1987),"Wavefront array processors concept to implementation," Computers, Vol.20, No.7, pp.18-33.

[6] Lala, P. K., (1985), Fault-Tolerant and Fault - Testable Hardware Design, Prentice Hall International, London.

[7] Mangir, T. E. and Avizienis, A., (1982), " Fault - Tolerant Design for VLSI : Effect of Interconnect Requirements on Yield Improvement of VLSI Designs," IEEE Transactions on Computers, Vol. 31, No. 7, pp. 609-616.

[8] Muroga, S.,(1982), VLSI System Design, John Wiley and Sons, New York.

[9] Peattie, C. G., Adams, J. D., Carrell, S. J., George, T. D. and Valek, M. H.,(1984), "Elements of Semiconductor Device Reliability," Proceedings of the IEEE, Vol. 62, No. 2, pp. 149 - 168.

[10] Sedmak, R. M. and Liebergot, H. L.,(1980), "Fault Tolerance of a General Purpose Computer implemented by Very Large Scale Integration," IEEE Transactions on Computers, Vol. C-29, No. 6, pp. 492-500.

[11] Siewiorek, D. P. and R. S. Swarz,(1982), The Theory and Practice of Reliable System Design, Digital Press, Bedford, Mass..

[12] Tamir, Y. and Sequin, C. H.,(1984), "Reducing Common Mode Failures in Duplicate Modules," Proceedings of the 1984 IEEE International Conference on Computer Design, pp. 302-307.