

INTELLIGIBLE TESTING

Melvin A. Breuer and Sandeep K. Gupta
Electrical Engineering-Systems Department
University of Southern California
Los Angeles, CA 90089-2562

This paper deals with a new methodology for testing VLSI circuits as well as the use of digital circuitry. Current trends indicate that for high performance nano-meter technology, chip yields may go below 60%, thus driving up chip costs. In addition, cores will be used to a large extent; thus many chips will contain circuitry that is unused in some applications. Classically, once a newly manufactured chip is tested it is declared to be either good (fault free) or bad. The bad ones are discarded. But some of these bad chips may perform quite acceptably in some applications, such as image and speech processing. We thus propose a test methodology that grades imperfect chips based on certain attributes, such as error rate, error significance and error accumulation. Built-in self-test methods exist to quantify these measures. Thus this new form of testing is *application oriented*. We refer to it as *intelligible testing*. In this paper we present the rationale for this form of testing and discuss several concepts that need to be developed to make use of defective chips in real systems.

I. Introduction

Two major trends in the field of VLSI technology are the increase in circuit density of chips and the increase use of commercial off-the-shelf (COTS) components and/or complex cores. Soon commercial chips consisting of over 10 million transistors will be in common use. Unfortunately the yield predictions for these nano-meter technology chips is predicted to go below 50%. This low value of yield will put a tremendous burden on testing if defect levels are to remain low, such as 75 parts per million. In addition, low yield translates into higher chip costs to the consumer. Along with increased circuit complexity comes increased design time and complexity. The use of cores is one way to increase design productivity and decrease design cost, time and errors. So what is wrong with this picture? On the one hand we have chip complexity going up because of design innovations and manufacturing refinements, while trends in test complexity, defect levels and yield move toward a possibly intolerable situation.

To address these issues, in this paper we propose two new paradigm shifts for the world of testing digital circuitry. The first is that some operational systems that may or may not be considered to be fault tolerant, can, in fact, tolerate faults, and this fact should be included in their test methodology. This will lead to *enhance system availability*. Secondly, by using chips that are not defect free but that produce acceptable results, the *effective yield* of manufacturing can be significantly *enhanced*. The proposed test methodologies can go by several names depending on which problem one focuses. These names include *intelligible testing*, *application oriented testing*, *salvage* and *yield enhancement*.

The presence of some hardware faults (defects) in a component (chip) of a digital system may not necessarily cause errors in its input-output behavior, or if errors do occur, they degrade its behavior in an acceptable manner. Such a fault is said to result in *acceptable degradation* in a system under three main scenarios.

1. Systems increasingly are being designed using commercial off-the-shelf (COTS) components, such as general-purpose and DSP processors, standard interface chips, and commodity memories. The emergence of intellectual property (IP) *cores* will further increase the use of off-the-shelf designs. While taking advantage of economies of scale, any system designed using COTS-chips/IP-cores typically uses only a subset of functionality

of most of its constituent components. Any fault in a component that only affects the part of its functionality that is not used by the system will cause no degradation.

2. Certain faults in a system that employs redundancy (e.g. storage and communication subsystems that use error correcting codes) may cause no degradation that is visible at the outputs of the system, provided that the fault only creates errors that can be masked by the redundancy. Such faults, however, do cause *invisible* degradation, i.e. a decrease in the system's capacity to sustain transient errors caused by external noise.

3. Some faults in some components, such as those that digitally process voice or images, may cause degradation at the outputs of the system that is virtually imperceptible to a human listener/viewer. In other cases the degradation may be obvious but acceptable.

Traditional test methodologies only differentiate between *perfect* chips and *imperfect* ones, despite the fact that many imperfect chips may have faults that only cause invisible or acceptable degradation.

In this paper we present a new test methodology, namely *intelligible testing*, that exploits the notion of acceptable degradation. We envision at least two types of systems to which intelligibility testing can be applied. In the first category are *critical systems* that require high reliability and either higher availability or high survivability. In the second category are *low-end systems*, where low-cost is the prime objective. For such systems the notion of intelligibility testing can help further decrease the cost by increasing the effective chip yield.

In the following, we begin with a more detailed description of the motivation behind intelligibility testing. Several examples and a brief discussion follow this on how to implement intelligible testing.

II. Rationale Behind Intelligibility Testing

The concepts proposed here originate from the following premises. The first two premises describe the important characteristics of the existing test methodologies. The next two describe important characteristics of faults in systems that are not taken into consideration by existing hardware testing and fault tolerance techniques. The fifth premise presents important classical criteria for critical systems, namely high reliability, safety, and availability, as well as for low-end systems, namely cost.

Premise 1: Only fault free circuits pass tests

Classical test techniques attempt to attain 100% fault coverage and thus ensure that a part is defect free.

Premise 2: Tests are not application oriented

The insertion of test hardware, such as built-in self-test (BIST), into a VLSI chip or core is almost always independent of the system into which it is eventually used. The tests are designed to achieve as high a fault coverage as possible. In addition, most manufacturing tests do not report any measure of the severity of error caused by a fault within a chip. For example, logic BIST tests only report whether or not a correct signature is obtained.

To a large degree this is a reasonable assumption but may unnecessarily decrease the availability of critical systems and increase the cost of low-end systems, as illustrated by the following three premises.

Premise 3: Some faults may not cause any errors

Many hardware faults may lead to no erroneous behavior at the system level. This can occur in a digital system due to at least three reasons: presence of unused functionality, design to tolerate external noise, and design to tolerate hardware faults.

Unused Functionality: A system is typically built by integrating components on a board (using chips), an MCM (using bare die), or a single chip (using cores). When a component is used in a system, typically only a subset of its functionality is utilized, unless the component is designed exclusively for use in the given system. For example, a microprocessor may be used in a system where some instruction, e.g. multiply, may not be used by the application. In such a case, any hardware fault in the multiplier will cause no errors at system outputs. Similar scenarios also arise in VLSI chips that are designed using IP cores from a library. In short, any unused part of a component can be viewed as being *redundant* in the context of the specific system. Any fault in a redundant portion of the system will usually cause no errors at system outputs.

Tolerance to Noise: Since many systems operate in harsh environments, their communication links and control subsystems are typically designed to function correctly in the presence of external noise. An error correcting code, or a combination of an error correcting/detecting code with the ability to request re-transmission, is often used to achieve reliable communication in such systems. Similarly, robust control design principles are employed to design control subsystems that can operate reliably in the presence of a pre-specified level of inaccuracy in the data obtained from input sensors.

Consider a part of a communication link, e.g., a switch, with a fault that, when excited, can cause a *local* single bit error in the data being transmitted. If the receiver can correct the error, then the fault will cause no error at the system outputs. Now consider a system that pre-processes data received from sensors for a robust control system where a hardware fault manifests itself as an occasional *local* single bit error at its output. If the numerical significance of the local error is small, then it may not cause any errors at the system outputs.

Two points should be noted about this class of faults. First, only those faults that cause local errors within the error correcting capability of the code, or the level of noise that the robust control system can tolerate, cause no errors at system outputs. Secondly, these faults do cause system degradation, despite the fact that no errors are visible at the system outputs. This *invisible* degradation may manifest itself in different ways. In general, any fault in a communication link will reduce the level of external noise that the system can tolerate.

Fault Tolerance: Fault tolerant techniques are often employed to enhance the reliability/availability of critical systems. The presence of such faults should not reduce the availability of the system in the short term, but the system should be repaired at the first convenient opportunity to avoid incorrect operation due to additional faults. The invisible degradation in this case is the system's inability to guarantee correct operation if an additional fault is sustained.

In short, a class of faults may only cause local errors that do not manifest as errors at the system outputs and their presence in the system can be easily tolerated. Nonetheless, the presence of such faults should be detected to evaluate invisible degradation and to make repair decisions in the long term.

Premise 4: System operation must be acceptable but not necessarily perfect

The class of faults discussed above can be further expanded to include additional faults that are characterized based on the following premise. Many digital systems --- critical as well as low-end --- carry out

computations where the results do not always have to be “exactly” *correct* at all *times*. Examples of such systems include signal and image processing, voice and image communication, and robust control systems.

As an example consider a personal video communicator. A fault in the least significant bit of the video RAM will only cause a small deviation in the intensity or color of the corresponding pixel on the monitor and may not even be discernable to the user. Similarly, an error of small significance in the digital signal processor (DSP) that processes the video data may also result in an *acceptable degradation* of the resulting video quality. It has come to our attention that RAMs with a few faulty locations are routinely used in digital answering machines since the subsequent speech processing circuitry filters out the effects of the faults. With increasing role of digitally communicating and processing audio and video communication in military as well as commercial systems, a large class of faults may fall into this category. It should be noted that the concept of acceptable degradation is not limited to audio and video subsystems. The concept is apropos for example to any system whose final output is “analog”, e.g., control systems that produce outputs used to set positions of mechanical actuators.

Hence, a large class of faults either causes no degradation or only causes acceptable degradation in the system outputs.

Premise 5(a): Need for high survivability for critical systems

A high level of availability is an important goal for a critical system. Of special importance is the availability of a system to the end of an ongoing mission, i.e., survivability, even at a somewhat degraded level.

Premise 5(b): Need for low cost for low-end systems

Low cost is one of the key objectives of many low-end systems, such as toys and other non-critical domestic appliances, which now collectively use a large proportion of digital chips. Of special importance is the low cost of incoming chips.

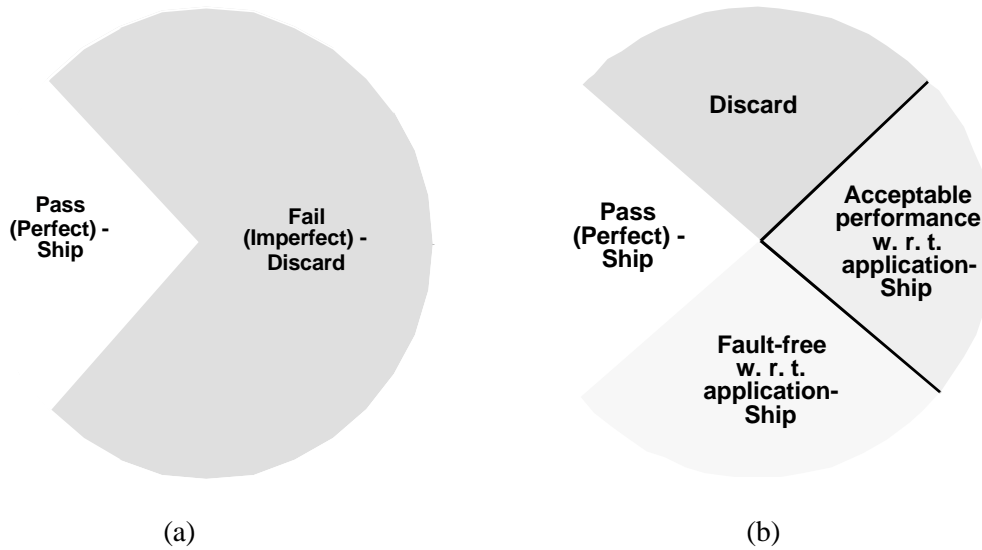
Hopefully it is clear that the concepts presented here arises out of incompatibility between the characteristics of classical test techniques and the above mentioned test objectives for many systems, both critical and low-end.

Consider critical systems. Since such systems are required to perform their tasks reliably, they are implemented using only the perfect chips among the chips manufactured. In addition, to ensure continued reliable operation, these systems typically use BIST circuitry to periodically test themselves to ensure the continued health of the system. Classical BIST techniques will declare a system faulty if the behavior of any of its constituent subsystem deviates from its good behavior in any way. In fact, as can be seen from Table 2, traditional BIST techniques will declare a system faulty even if a fault causes no system degradation. The classical techniques will also declare a system as BAD, where the behavior of the system is acceptable despite some errors caused by one or more faults. The declaration of a system with no or acceptable degradation as being BAD unnecessarily decreases the system's availability and survivability. This decrease may in turn jeopardize a mission, especially if only limited resources are available.

Note that the use of functional tests, developed at the system level for a specific system, can indeed be used to avoid the unnecessary decrease in system availability and survivability. However, such tests often take years to write, are usually incomplete, and are difficult to make application specific.

Next, consider chips manufactured for use in one or more low-end systems. The tests applied to each chip immediately after their manufacture also aims to identify the presence of any fault. Hence, ideally, only perfect chips pass these tests. However, many chips declared faulty by these tests may in fact cause either no

degradation or only acceptable degradation in a low-end system in which it may be used. By discarding such chips, existing test methodologies unnecessarily decrease chip yield and increase the cost of incoming chips for low-end systems. Such chips can and should be *salvaged* thereby increasing the *effective yield*. Figure 1 illustrates the benefits.



(a) (b)
Figure 1: Increase in effective yield due to intelligibility testing.

III. New Test Methodology

The key aspect of the intelligible testing is *application-oriented testing*. This includes error analysis at the hardware level, and error analysis and the system level. That is, the test methodology must be able to characterize errors that are caused by defects. In addition, the system designer must be able to characterize the types of errors that can be tolerated at the interface of certain blocks of circuitry.

Unlike the Go/NoGo result provided by classical test

techniques, intelligibility testing at the system level would provide an evaluation of the system's health using a scale that would include a perfect system, i.e., GOOD, as well as an inoperable system with unacceptable degradation (BAD), as its end-points. Between these endpoints would be a range of continuous/discrete points that would provide a measure of acceptability of the system.

Although the presence of a fault may not cause any deviation in a critical system's input-output behavior, it may still lead to system degradation. Such degradation is *invisible* at the system outputs, since it causes no errors at the outputs. Even a fault that causes only acceptable degradation in the input-output behavior may cause additional degradation that is invisible at the system outputs. The proposed methodology for test of critical systems must report such invisible degradation, since it is often a measure of the robustness of the system.

1. Criteria for Characterizing Errors

To motivate the discussion, consider a typical home TV set after five years of operation. Perhaps the sound quality has deteriorated, the tint is poor, and there is some picture roll at times. But the owner may find its performance acceptable! It is easy for the viewer to determine the degradation in quality and determine when the TV's performance is unacceptable. We hope to achieve the same type of criteria for system availability for digital systems, but we will *not* require the user to monitor the system performance to determine whether or not it is acceptable. *Instead this acceptance criteria will be precomputed via analysis of the system. In critical systems, the criteria will be built into the system itself as part of its system self-test and compensation/reconfiguration hardware. In low-end systems, the criteria will be used to develop appropriate tests and DFT for post-manufacture test of chips.* The question that first needs to be answered is what criteria can be used to characterize whether the observed erroneous behavior of a faulty circuit constitutes acceptable degradation.

Faults in circuits can cause errors at their outputs. We currently characterize these errors by three measures, namely *error rate* (R), *error accumulation* (A), and *error significance* (S). We call these RAS measures.

Error rate quantifies how often an error occurs. For example, consider a combinational circuit with 32 inputs and a single stuck-at fault that is detectable by 2^{16} test vectors. Then, assuming all inputs are equally likely, this circuit would have an error rate of 2^{-16} .

The *significance* of an error deals with the impact of the error on the usability (or accuracy) of an instantaneous result. For example, an error in a low order bit of a 32 bit arithmetic operator may not be too significant for some applications, while one in a high order bit may be serious.

Finally, depending on the memory retention or feedback structure of a design, the *accumulation* of errors might be quite harmful. Many signal and image processing applications consist of feed-forward pipelines, where memory retention is not an issue. In contrast, occasional errors of small significance due to a fault in an accumulator that is used as an integrator in a control system can accumulate over time to cause instability. Finite state machines are particularly susceptible to error accumulation.

For a given application, the acceptance criteria for each of the RAS measures must be specified. As a trivial example, high volume levels of noise at a low, but not high rate may be acceptable. These threshold values will form the basis for the development of the intelligible test methodology.

2. Built-in Self-Test for Intelligibility Testing

Testing can be carried out in several ways, such as by using functional tests, scan based tests along with DFT hardware, or built-in self-test. In this paper we only discuss the latter.

One way to determine the value of RAS parameters is to use a reconfigurable and enhanced BIST circuitry that includes classical BIST configurations for achieving high fault coverage. During test, classical self-test would first be used. If no faults (errors) are detected, then the subsystem testing is complete. On the other hand, if this self-test declares the circuit to be faulty, then it becomes necessary to determine whether the rate, accumulation, and the significance of the errors caused by the fault are within the acceptable levels specified for the subsystem, in the context of the specific system. This can be achieved in several ways.

Error rate can be determined by repeating the test several times at judiciously selected intervals of time, i.e. by varying test length. The test length must be varied to ensure that only faults that cause errors at a rate higher than the acceptable value result in erroneous signatures.

Error significance is related to which output lines contain errors and can be established by using a reconfigurable MISR that can perform signature analysis on various subsets of the output lines.

Error accumulation can be determined from data related to error rate and error persistence at outputs. It can also be determined by using arithmetic compressors, such as accumulator based compressors. Error persistence can be estimated by seeing if and when a circuit gets back into the correct state after being in an erroneous state.

Specifying Acceptability Criteria

To utilize this test methodology, it is necessary that system engineers specify the value of acceptable RAS errors. In general there could exist a family of surfaces in a 3-dimensional space, where each region corresponds to a degree of acceptability.

There are several ways that such acceptability surfaces could be constructed, but this topic is beyond the scope of this paper and thus will not be discussed here. But briefly, one could apply several different techniques including fuzzy logic, sensitivity analysis and control theory.

IV Case Studies on Error Rate

Figure 2 shows the distribution of fault coverage vs. test length for two ISCAS85 combinational benchmark circuits. The test sequence applied at the inputs to each of these circuits is composed of 2000 distinct random patterns. From Figure 2, we see that for c1355, the first 5% of the patterns detect about 90% of the faults, and for c1908, the first 1% of the patterns detects about 80% of the faults. Applying all the test patterns detects about 90% of the faults. The faults near the tail of the distribution are referred to as random pattern resistant faults. Here we have focused on random test patterns because we assume that they are representative of functional data.

We next illustrate that a considerable percentage of single stuck-at faults produce a significantly low error rate when functional patterns are processed through a circuit. Figure 3 shows the percentage of single stuck-at faults vs. the maximum fraction of output patterns that are in error, for circuit's c1355 and c1908. These results are based on applying 2000 unique patterns to each circuit. Clearly a large fraction of the faults cause a significantly low error rate. For example, for c1908, 40% of the single stuck-at faults produce an error rate of at most 2.8%. In fact, if one considers only random pattern resistant faults, then several of these faults could co-exist and the circuit might still have a low error rate.

Consider a feed-forward DSP circuit having blocks of logic C1, C2, C3, and C4 separated by registers R1, R2, and R3, where C_i feeds C_{i+1} through R_i. Now if C1 has a fault that causes an error rate of 1%, then the error rate at the output of C4 is at most 1%, and in general significantly less than this value. Thus it is clear that based on the location of a fault and the system architecture, very low error rates are possible and system availability can be greatly enhanced by employing intelligible testing.

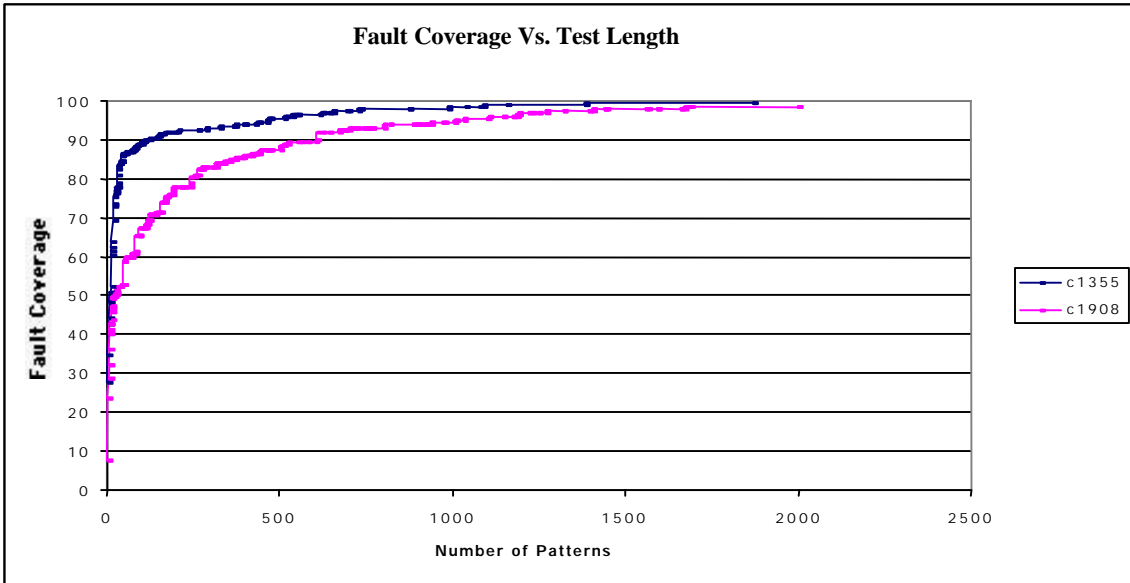


Figure 2

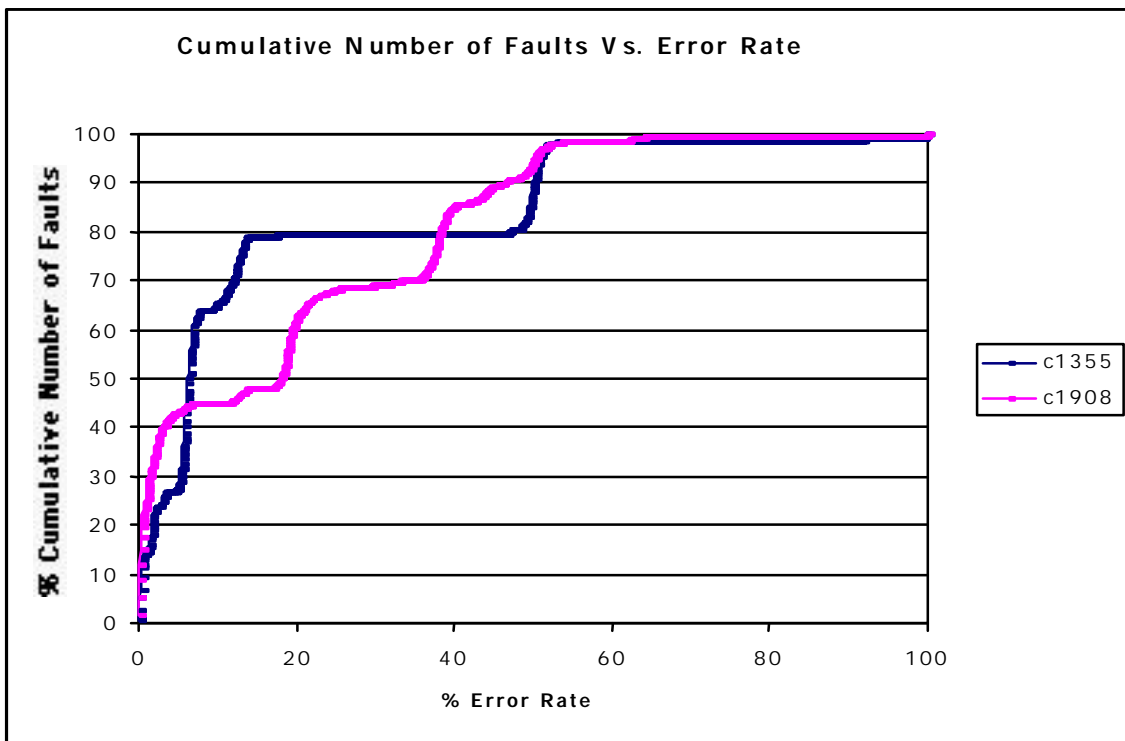


Figure 3

V Intelligibility Testing and Microprocessor

1. Existing Examples of Intelligibility Testing

Several past and current practices in microprocessor testing can be viewed as special cases of intelligibility testing.

1. Since cache memories constitute a large proportion of any modern microprocessor, a significant fraction of defective microprocessors have faulty caches. A large proportion of such faulty chips may be salvaged if provision is made to tolerate a certain class of defects, e.g., defects that affect a single cache row [1, 2].

Changes need to be made to the design, test and diagnostic procedures, and manufacturing, whenever defect tolerance methods are used to improve cache yield. Firstly, extra rows (and/or columns, as appropriate) must be added to the cache design. Secondly, cache tests that are capable of determining the severity of the faults, measured in, say, the number of faulty rows, are required. Furthermore, the test results must be further analyzed to identify the faulty row. Finally, an extra manufacturing step may be used where a focussed ion beam (FIB) is employed to disconnect the faulty row and connect the spare row.

Clearly, the benefits in the form of increased yield must exceed the extra costs in order to justify these extra steps.

2. In the past, a certain microprocessor test methodology was designed to determine whether or not the faults in a defective chip were confined to its floating point unit (FPU). Chips for which this was found to be the case were then sold at lower prices as versions without FPU.

Again a modification of the test methodology and associated costs were incurred, but the additional revenues more than offset these costs.

2. New Possibilities for Intelligibility Testing

The idea of intelligibility testing may be applied in several novel ways to microprocessors. Some of the types of faults that may be tolerated include (a) some other types of faults in caches and ROMs, (b) faults in register files that affect access to one or two registers, (c) faults in instruction decoder logic that only affect some rarely used instructions, and so on. Once again, the test and diagnostic procedures would need to be changed to identify whether or not the defects only cause faults that can be tolerated.

In the case of microprocessors, the defects may be tolerated in (a) changes to design and manufacturing process (as illustrated above in the example of defect-tolerant cache designs), (b) changes to the design of control units, such as memory management units or register file decoders, which may be configured in alternate modes to tolerate the defect, (c) changes to the microcode (if applicable), (d) changes to the operating system, (e) changes to the compilers. As an example, given a fraction of chips that do not handle a set of instructions I, a compiler could be modified so that code run on such chips did not employ any instructions in I.

3. Methodology to be Developed

Even though a range of compensation mechanisms are at a microprocessor manufacturer's disposal, many types of defects that significantly alter the core functionality of the processor may not be altered.

Even among defects that cause faults with small severity, the application of intelligibility necessitates the development of the following key components of a methodology.

1. Availability of a mechanism to compensate for the type of faults to be tolerated. The cost of design time, die area, and performance penalties need to be quantified.
2. Development of test and diagnostic methods that can not only identify which chips have defects that can be tolerated, but also provide information (such as fault location) that would help configure the compensation mechanism. Again the increase in costs of test-development and test application need to be quantified.
3. The benefits in terms of increased revenue provided by the sale of salvaged chips must be quantified and compared with the above costs to determine whether or not intelligibility testing should be applied to the particular types of defects and faults.

VI Other Applications

In this section we briefly describe several applications where less than perfect hardware has or can be utilized.

- For several years, suppliers of answering machines have used DRAMs that contained up to a few bad memory cells, since these chips produced acceptable performance and are significantly cheaper than perfect DRAMs.
- Neural networks in some respects have built-in fault tolerant capability and are ideal candidates for intelligible testing. The fact that in such systems, parameters (weights) are determined via a learning process inherently allows for some degree of erroneous operation.
- Feedback control systems can be characterized by several measures, such as speed of response, accuracy, relative stability and sensitivity. Internal errors in a digital controller due to defects can often effect the value of these measures in a minor way.
- The fields of digital signal and image processing as well as communication theory contain numerous applications where digital circuits can beneficially employ intelligible tests.

VII Summary and Conclusions

In this paper we have introduced the concept of intelligible testing, which, unlike most other forms of testing, is application oriented. Our work is motivated in part by the fact that yields may become quite low and the use of cores and COTS implies that a large fraction of a circuit's logic may be "functionally" redundant with respect to an application. In addition, for many applications, low levels of erroneous responses may be acceptable. We have presented three measures for characterizing erroneous responses, namely error rate, accumulation and significance. We have shown that for some circuits, error rate can be quite low. In addition we have alluded to ways of estimating the values of these

measures using BIST circuitry. We believe that by employing the concepts presented, (1) effective chip yield can be significantly enhanced leading to a reduction in costs to the consumer as well as increased revenues to the manufacturer - thus a win-win situation; (2) system availability/survivability is enhanced; and (3) system maintenance costs can be greatly reduced.

We see this area of technology as lying between classical testing, where only perfect chips are assumed to be used in systems, and fault tolerant computing where errors due to faults occur. Here, such errors are dealt with by such techniques as masking, error recovery and damage control [3].

References

- [1] D. Bhavsar and J. Edmondson, "Testability strategy of the alpha AXP 21164 microprocessor", *Int'l Test Conf.*, pp. 50-59, 1994.
- [2] D.K. Pradhan, editor, Fault-tolerant computing: Theory, and techniques, Vol. II, *Prentice Hall*, 1986.
- [3] T. Anderson and P.A. Lee, Fault tolerance; principles and practice, *Prentice Hall*, 1981.