UNDERSTANDING NTF COMPONENTS FROM THE FIELD

Scott Davidson

Sun Microsystems, Inc.   Sunnyvale, CA

Abstract

No Trouble Found (NTF) parts are the bane of the test engineer's existence, since it is difficult to determine the cause of a fail for a part that won't. A companion paper [1] discusses parts that are NTF after failing in the board or system factory. This paper discusses the issue of parts that fail in the field. We classify field NTFs, describe a method of determining the defect coverage of a component under board and system test, and describe how data mining techniques can be used to assist in explaining the causes of field NTFs.

1. Introduction

No Trouble Found components (NTFs) are the bane of the test engineer's existence. When a component is returned from the field or from a system customer as having failed, we wish to determine the cause of the problem, so we can improve our test program or our processes to ensure that the failure does not happen again. This is difficult to do if the part refuses to fail on retest. This problem is compounded by very high NTF rates, commonly above 50%. Thus, we have the situation where many parts fail for no apparent reason.

Field Replaceable Units (FRUs,) such as boards, are retested after being returned, and can be NTF. When a component of a FRU such as a microprocessor is indicted as being the cause of a failure, it is often returned to the vendor or system manufacturer and retested. In this paper we are concerned only with IC NTFs.

ICs can be indicted when FRUs fail during board or system manufacturing (in-line failures) or when FRUs fail at a customer site (field failures.) A companion paper [1] analyzed the in-line failure case. This paper concentrates on the issue of failures in the field.

Why should we care about NTFs? First, NTFs represent an opportunity to improve test coverage. A test written that causes a formerly NTF part to fail can be incorporated in the original manufacturing test, reducing test escapes and increasing quality. Second, NTFs offer an opportunity to improve field repair and diagnosis processes. As will be seen below, some percentage of NTF parts are actually good. If we can keep from indicting and returning good parts, we will reduce the number of spares needed and, more importantly, correctly diagnose the real cause of a problem faster.

Third, NTF rates can be used as a measure of product quality. [1] describes how NTF rates can be used as an early indicator of component reliability problems. Comparing measured NTF rates against expected rates can tell us if something in our test and repair processes is amiss. Finally, many engineers have preconceived ideas of what the NTF rate should be. If these ideas are not realistic, a poor allocation of resources might be made to deal with NTFs, and some warning signs of more important problems might be missed.

The goal of this paper is to examine the causes of field NTFs, quantify field NTF rates, use these rates to measure the quality of board and system test with regard to components, and recommend actions to take when certain situations arise. The context of this work is Sun Microsystems Sparc™ microprocessors.

Very little if anything has been written about NTFs in the literature. A search on the IEEE Electronic Library on NTF revealed only a panel statement by the author of this paper. A search through 25 years of ITC proceedings yielded nothing more. In-line NTFs are associated with test escapes and thus test coverage (see [2] for a good overview) but little has been written about NTFs themselves.

The structure of the rest of this paper is as follows. Section two defines NTFs and reviews our testing and return processes, both for field and in-line fails. Section three briefly summarizes findings on in-line fails from [1]. Section four analyzes the types of field NTFs we see, and provides causes for each. It also provides techniques to determine the root cause of each type of NTF, and recommends countermeasures where appropriate. Section four also shows how NTF rates can be used to measure component test coverage at the board and system level. Section five describes how this model can be used as a process monitor, and shows how data mining of product data can be used to help determine the cause of a failure. Section six concludes and offers suggestions for future work.

0-7803-9039-3/$20.00 © 2005 IEEE

2. NTF Background

2.1 Definitions

A *no trouble found (NTF)* component has the following characteristics:

- It has passed a certain test step in the component lifecycle.
- It has failed a subsequent test, or has been indicted as the cause of failure of a higher level part of which it is a component.
- It passes the first test step again

One example of an NTF is a microprocessor which passes all levels of IC test, is indicted as causing a failure during system test, and upon return to the supplier, passes the IC test again. Another is a field replaceable unit (FRU) which passes board test and is shipped to the field, and is later diagnosed as being faulty, but which passes board and system test at a repair depot.

We can further split IC NTFs depending on the level of retest performed. A *Tester NTF (TNTF)* passes when retested on an IC tester. It is often useful to try to reproduce the failure as seen by the customer by placing the indicted component in a system platform and rerunning a system test. A *System NTF (SNTF)* part is one which does not fail this verification step. It is possible for a component to be any of the four combinations of TNTF, SNTF, verified fail (the complement of SNTF) and a retest fail (the complement of TNTF).

2.2 The Sun Component Return Process

Different processes are used for the return of processors from manufacturing and the field. Figure 1 shows our flow for in-line (system manufacturing) and field returns. Processors diagnosed as having failed in-line are sent to our silicon vendor where they are retested. Parts that fail are held there, and parts that pass the retest are shipped to us, and are inserted into a system in our lab, and system tests run. Parts that pass are SNTFs, and parts that fail are TNTFs.

Parts that are indicted as causing a field fail are sent directly to us, where they are retested on our ATE. Parts passing are retested in a system (called reverification). If they fail, the parts are TNTF, and if they pass they are SNTF.

Information about the returned parts, including origin system, fail symptoms, lot codes, date codes, wafer locations, etc., are kept in a database. In addition, we can query Sun databases that give additional information about the FRU's history in the field. We have built tools to combine this information to give us a complete picture of the history of a failing part and the environment in which it failed.

All returned parts are kept, and we try to write additional tests to improve coverage and cause the NTF part to fail. The more information we can provide about failure symptoms, the more successful this process is. We will discuss ways of providing clues to the root cause of a failure below.

3. In-line NTFs

In this section we summarize results from [1]. Components that are indicted as causing failures during board and system manufacturing are sent back and retested. We have found that a large majority of confirmed failures are NTF. This should be expected. Parts that retest NTF represent test escapes, which naturally do not fail when tested with the same test program that passed them a short time before. Of more concern are parts that do fail retest. These must have failed between shipment from the fab and test on a board. Some will fail due to damage, but this should be a small percentage of returns. The rest fail due to early life reliability issues. If the NTF rate on in-line fails goes down, this indicates that the reliability failure rate is going up, which can mean a serious process problem that will result in an elevated rate of field failures. This is true even if the overall DPM rate as measured in the system factory remains unchanged. We observed just this behavior when we experienced a process quality problem. Thus, contrary to the intuition of many test engineers, it is better to see 100% NTFs for a given DPM rate than 100% retest failures. Reducing DPM rates by improving test coverage is always advantageous, but for an acceptable DPM level a high NTF rate is nothing to be concerned about. We are now tracking NTF percentages for in-line returns as a process quality monitor.

There has been much work done on predicting DPM rates given yields and defect coverage [2, 3, 4]. A major problem with these techniques is the need to measure defect coverage. Fault coverage as reported by ATPG tools is not accurate as a defect coverage measure even for circuits which are logic only, and the addition of memories, I/O and mixed signal blocks makes the problem worse. In [1] we showed that given a measured DPM rate, and using only NTFs, not retest fails, we can run the Seth-Agrawal formula backwards to get the defect coverage for a device with a given yield. This defect coverage is often much higher than stuck-at fault coverage, which should not be surprising since it includes the contribution of delay tests, functional tests, and memory tests which provide high defect coverage for a large part of the silicon.

Given a coverage baseline, we can use this method to compute the coverage improvement for new tests, or to detect yield fluctuations for incoming parts. It can also detect shifts in defect distributions, which might lead to a higher proportion of defects migrating to regions of low test coverage.

We also presented data in [1] showing that during a period in which devices are failing in the field due to process issues, there is a higher rate of retest fails than normal, and a nearly unchanged number of NTFs.

In-line failure is the easiest case, since the environment in which the part fails is well known, good record-keeping is done, and the short interval between fab and failure means that there should be relatively few reliability and thus retest fails. Field failures are more uncertain. We will investigate them in the next section.

4. Field Return NTFs

When a field failure has been diagnosed as having been caused by a hardware problem, a set of diagnostic programs can be run, under Solaris, to pinpoint the cause of the failure. One or more FRUs are removed and returned to a repair vendor. There, the failure is diagnosed to a component, the FRU repaired, and the FRU returned to the field.

Because of the number of microprocessors, ASICs, and memories on our boards, it is understood that not all diagnoses are correct. In most cases if a microprocessor is indicted as being the cause, the processor is moved to another location on the board and the test rerun to see if the failure follows the component.

Failing processors are shipped back to us, along with information on failure symptoms, the FRU and system it came from, and pointers to information on the field repairs made. With this information we can access several databases giving the history of the FRU, when it was manufactured, its test history, and its field history. Information on the processor itself gives us further information about when and where it was fabbed. This wealth of data has been very useful in finding subtle problems at the FRU and component levels.

As mentioned above, there are three results from retesting a component returned from a repair vendor:

- the component fails retest.
- the component passes retest, but when inserted into a system causes the system to fail.
- the component passes retest, and a system with the component inserted passes also.

The first of these cases we call a component fail, the second a TNTF, and the third an SNTF. We will analyze these in more detail.

4.1. Component Fails

This is the simplest case. if the retest was done using the same test program as the part was originally tested with, we can be sure that the part failed either because damage or reliability issues. Standard failure analysis techniques can be used to find the cause of the failure; these are outside the scope of this paper. However we need to do some further analysis, by collecting and analyzing all our retest fails.

First, are the parts failing as we would expect? Microprocessors today are memory rich. The memories are dense and thus more prone to failure than logic. Therefore we would expect to see relatively more fails during memory test than logic test. If this is not happening, it might point to a specific reliability problem in a part of the design.

Second, we should collect and report on how long it took parts to fail. A large number of early life failures would indicate that insufficient stress is being applied to the part, or that there is an issue with the treatment of the part during board and system manufacturing, installation, or in the customer environment. A rise in failures during a certain interval might indicate a failure mechanism being activated after so many hours of use. We also track by board manufacturing dates and by fab lot codes, to see if certain manufacturing intervals produced more failures.

4.2. TNTFs from the Field

One might think that TNTFs represent test escapes: defective ICs that passed both IC and board/system test, but then failed in the field. However, by their definition, they fail a subsequent system retest. We can identify four causes for this.

1. The system test used for the retest is different from that used for the original factory test. If this is so, it is possible that the new test is a better screen. This can be confirmed by rerunning the part on a system using the old test. Another, similar, possibility is that the system on which the part is retested is different in some way from the system in which the part was originally inserted, and that this difference is enough to cause the defective part to fail.
2. The defect occurred in the field, but is not covered by the IC test, only the system test. This is possible but not likely, especially if the IC test has high defect coverage.

3. The returned part is tested under different conditions when it is being reverified and retested. The system retest and the ATE retest should be run under the same frequency, voltage and temperature conditions. For example, if a 2 Ghz part must pass a 2080 Mhz test when being tested on a device ATE, but in system test the margin is not four percent but six, it might be failing at 2090 Mhz in system but passing on the device tester.
4. The defect is mechanical, or in the interface between IC and system. If the package is warped, it might be the case that failures will occur in system at certain temperatures. These package issues are unlikely to be picked up by device test. The electrical and thermal environment of the system might also sensitize a weakness not found during device test. In certain cases shmooing the part might detect the issue, but this is too time consuming to do on any but critical returns. This just touches on potential causes of this situation; the main point is that silicon might not be the place to look for the problem.

4.3. SNTFs from the Field

This is by far the most complex of the cases to resolve. There are three possibilities:

1. The defect escaped both IC and board test, but was detected by a customer application.
2. The part does not have a hard defect, but is marginal, working correctly under test conditions but not in the field environment. This is usually not the result of single event upsets (SEUs) in our case. Though these happen, they are handled by the system, and it is unlikly an SEU would recur at just the right time during diagnosis to cause a processor to be indicted.
3. The IC is not defective at all, but was misdiagnosed.

If the first case is true, we have a major problem: two test holes that let defective parts out into the field. It is imperative that we find the root cause of the failure, improve coverage for IC test, board/system test, or both, and assess the risk of this defect affecting more customers.

The second case also represents a critical problem. The part is not robust enough to be sufficiently reliable. While this is often a design issue, it means that we must modify the test program to test to more stringent conditions to ensure quality.

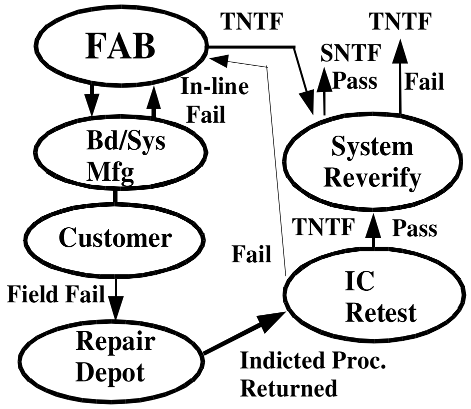

Figure 1 Sun Processor Return Process

In the third case, our effort in diagnosing the IC would be wasted, and would divert us from finding the real source of error. Much time and many resources might be spent in this wild goose chase.

Here is a case where traditional testing falters against the NTF problem. Testing a good chip does not provide the necessary answers.

Distinguishing among these possibilities is essential in improving quality. Assuming that all SNTF parts are good, and have been returned due to misdiagnosis can result in missing a serious test hole. Assuming that all are actually bad, and that test generation effort needs to be devoted to make them fail can cost a lot in test writing resources and in occupying expensive systems for long test cycles. We need a method to at least roughly distinguish parts likely to be bad from those likely to be good. How to distinguish among these cases is discussed in the next section.

4.4.Resolving SNTFs

When faced with an SNTF, the usual approach is to go into debug mode. This involves shmooing the suspect device, searching for potential test holes by examining coverage near the internal functional unit of the IC related to the system fail, and writing additional diagnostic system tests which, it is hoped, will cause the system to fail. Sometimes a similar strategy is employed for TNTFs, especially when the time to fail on the system is very long.

All of this requires a lot of resources, highly skilled diagnostic engineers, takes a lot of time, and might never find the cause of the problem. While the traditional approach should not be abandoned, we have found that a parallel approach, involving the examination of the history of the failing part and system, and similar parts and systems, to be highly useful.

The first and simplest technique is to collect the history of a part. A part whose replacement led to a successful repair is more likely to be faulty than a part which was one of several indicted components. One should check for repair policies which might lead to the return of good parts, such as removing all parts of the same type for an upgrade. At times a particular component may be on the top of the repair vendor's suspicion list, if there has been a history of the part causing problems. Even if the problem has been solved, the part might be removed first when no other cause of the failure is known. If this is noted on the repair record we can provisionally class these parts as good, and give them a lower priority for root cause analysis.

A second and more complex method is to find similarities between SNTF parts forming sub-populations of the installed base of parts. For instance, some repair vendors might return parts with much higher NTF rates than others. Certain parts might have higher failure and SNTF rates when used in one product versus another. Some parts may fail only when on a board with another part from a certain vendor.

These steps will not resolve all SNTFs, but will mark some part of the SNTF population as being less likely to be defective. Once this is done we can estimate expected SNTF rates.

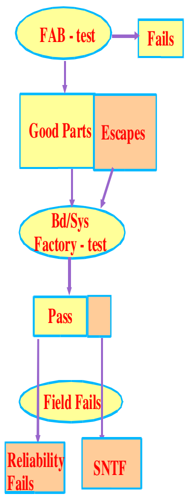

Figure 2. Sources of Field Failures

Figure 2 reviews the sources of field failures. After IC test in the fab, we can identify three classes of components: good, bad, and escape. (We group false positives with the bad for this discussion.) Bad parts are discarded, and the good and escape parts are shipped to the the board/system factory. Here, board and system tests are applied, which test the component also. Some percentage of escaped parts fail these tests; it is clear that the yield of a component at board/system test should be very much higher than the yield at the fab. Here yield is expressed as components shipped in products over components assembled into products.

Some good parts will fail due to reliability issues. Escapes from board and system test will, if and when they fail in the field, be SNTFs. We would be able to predict the contribution of escapes to the SNTF rate if we knew the defect coverage of the board/system test as applied to the processor by using one of the escape rate calculation techniques, such as [4]. We do not have this information, and obtaining it through fault simulation is impractical, due to the lack of a good defect model, the length of board and system tests, and the lack of an effective functional level fault simulator, or a fault simulator able to handle tens of millions of gates that does not assume combinational or near combinational designs. We can, however, guess some bounds for the defect coverage of a processor under a system test. The coverage is unlikely to be very low, below 50%, since much of the logic and memory of the processor is being exercised. We also do not expect it to be very high, since a system test does not target specific defects in the processor. Can this knowledge give us a clue about the escape rate?

Consider the following example.

Example: Consider a population of 1,000,000 components shipped from a fab. 2,000 of these parts fail in the board and system factory. The effective yield is thus 99.8%.

Over the life of the system, 1,000 components are indicted as being the cause of failures. When retested, 500 are SNTF, and another 500 are verified failures. Of the 500 verified failures, 450 fail retest, and 50 are TNTFs.

Assuming that we have determined the IC test to have 99% defect coverage using the techniques described in [1], we can expect that 5 of the 50 TNTFs are reliability fails, which escaped the retest (about 1% of the retest fails). That leaves 45 TNTFs probably caused by factors described in section 4.2. We know that these are not systems test escapes, since they have failed a systems test.

This leaves 500 SNTFs, which have possibly escaped both IC and systems test. The escape rate ranges from 500 (if all the parts are actually bad) down to zero, if all the parts are actually good. Neither of these extremes is likely. However, we can use these as bounds to compute potential defect coverages of the systems test.

We will use the Seth-Agrawal equation (equation 1) run backwards to compute these coverages. As shown in [1], we can use a mathematics package such as Mathematica [5] to solve for f1 given the yield Y, the number of defects per failing part n0, and the reject rate RR.

$$RR = \frac{(1-Y)(1-f_1)e^{-(n0-1)f_1}}{Y+(1-f_1)(1-Y)e^{-(n0-1)f_1}} \quad (1)$$

In the first case, Y = .998 and RR = 0.0005. Since the parts failing system test have gone through an extensive IC test, we can assume that each failing part has only one defect, so n0=1. Substituting these values, and running the equation backwards, we get a defect coverage of 74.99%.

If RR = 0, then f1 is 100%, which is extremely unlikely. If half the SNTF parts are bad, then RR = 0.000250, and f1 is 87.5%. Note that n0 being one makes RR linear in f1 given that all other parameters are held constant.

What the right value of f1 is cannot be precisely determined from this exercise, but system test writers can provide some idea of its effectiveness based of historical field return rates, effort put into the development of this part of the test, and its effectiveness in detecting IC test escapes. Since some percentage of SNTF components can be determined to be good by the studies described above, the predictions of this technique can be evaluated against the upper bound of bad SNTFs so determined.

While this information is interesting, it is unlikely to drive many changes in a systems test. Most of a systems test is and should be focused on testing system components that have never worked together before, not on retesting microprocessors and ASICs that have been previously tested. Therefore most system test writing effort goes into other areas. If an escape problem is found for an IC, it should be corrected at the fab, through a process improvement, more effective stress testing, and/or more effective manufacturing tests. The coverage of a processor or ASIC at system test can be used as a process monitor, though, which will be described in the next section.

5. Improving Quality by NTF Monitoring

Though the quantity and characteristics of field returns, especially NTF field returns, may seem chaotic, we have found that fluctuations in these are indicators of particular problems usually caused by a single issue. This assumes that the manufacturing and repair processes are under control: if not several problems may be encountered at once. In this section we describe several parameters that we monitor, and how changes in these indicate particular problems.

5.1 Time to Fail (TTF)

There seems to be a tendency to consider all field failures the same. As shown above, this is not true. Measuring time to failure is an important way of differentiating failure types.

Time to fail is measured from initial power on in a customer environment to the first indication of a failure that results in a processor or other component being indicted. If reliability wear-out were the only cause of component failure in the field, we could expect to see a smooth curve (figure 3a) of fails over time until we reach the end of product life. This indicates that we in the flat part of the reliability bathtub curve, and that our failure acceleration techniques, such as burn-in, are working.

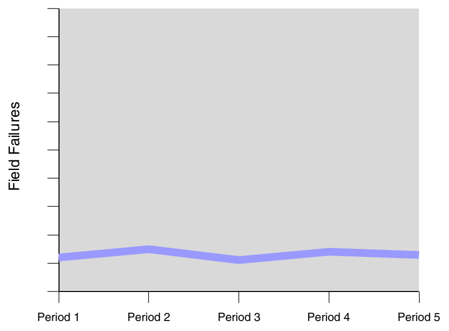

Figure 3a Fails over time (best case)

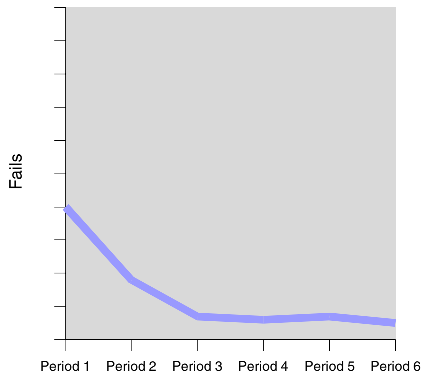

Figure 3b Fails over time, early life fail case.

More frequently we see a curve like that in figure 3b, with a relatively large number of early life failures. This would seem to indicate that our failure acceleration techniques are not working. This could lead to the addition of more, expensive, burn-in on parts.

Let us split the curve into NTF and non-NTF parts (figure 3c, based on real data). We show only the percentage fails during a period in each category. We include both TNTF and SNTF parts in the NTF category here. We see that the percentage of NTFs falls from about 50% to under 20% with increasing TTF. The cause for this behavior should be clear. The failures are either a result of test escapes, or of a peculiarity of the user environment not reproduced in the system reverification environment. Unless additional stress testing causes defects that escape the test to become gross enough to be detectable, additional stress will not improve quality.

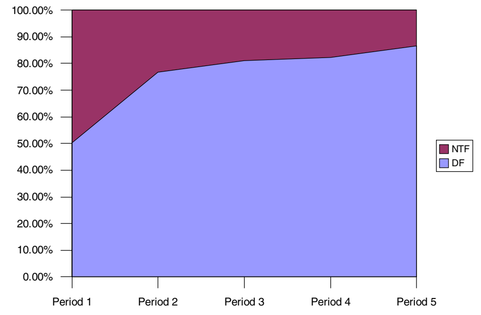

Figure 3c. **Detected fails vs. NTF fails**

If, on the other hand, a situation like that in figure 3d arises, there is a problem. Here the number of early life fails that are not NTF is high. In this case there is likely a reliability issue which should be immediately addressed. Often this situation should be detected before shipment to customers by monitoring in-line NTF rates, but if not, at least it can be detected before suspect parts are in the field for very long.

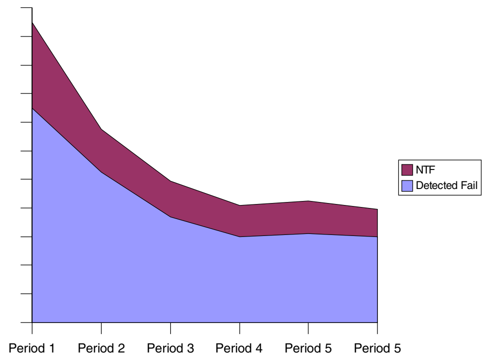

Figure 3d **Hypothetical fails over time for reliability issue**

5.2 NTF rates by Manufacture Date

Figure 4 shows data on NTF and non-NTF fails by manufacture date. (The data has been smoothed and the scale removed.) Here we see a background of NTFs, and a spike in retest fails. This was the result of a process issue. Plotting data this way can show problems by time even when the failures are spread over weeks or even months. This information can be used, if the problem is bad enough, for "surgical" purges, of only the material that is at risk. Besides the obvious cost advantage of a limited purge, there is a quality advantage. As Figure 3c indicates, new components are more at risk of failure than older ones, due to test escapes. A policy of removing components that are mature enough to be relatively stable from will increase the number of customer visible failures, and decrease quality instead of increasing it.

Figure 4 also supports our contention that most reliability failures will be detected by a high quality IC test. The number of NTF components during the time of the process issue hardly budged. Components that fail for reliability reasons fail retest.

5.3 NTF Sub-populations

Examining time to fail and manufacturing date information may be useful if the processor or ASIC is the source of the quality problem. If it is not, other techniques must be used to track down the actual culprit. The most useful we have found is to use data mining techniques to create sub-populations of failing parts.

Example: A microprocessor is used in a variety of products on several boards. The processor is being returned at a higher than expected rate. The vast majority of processor returns are SNTFs.

How can we determine the true cause of this problem? We might start by computing the returns per million rate for each product. To do this we must not only take into account the number of processors shipped in each product but their introduction dates, since a reliability issue with a long time to fail may not have shown up in a recently introduced part. If this does not provide us with a product that looks suspicious, we can move on to an analysis by board type and by board manufacturer. If a particular manufacturer is showing a higher rate of failures, their processes can be examined to ensure that they are still within agreed upon specifications.

Say none of these analyses show an obvious cause of the problem. At this point, if we have recorded system configuration information, we can do an analysis of fails by equipment location and neighboring components.

We might discover that a large portion of the fails occur when an ASIC within a certain serial number range is

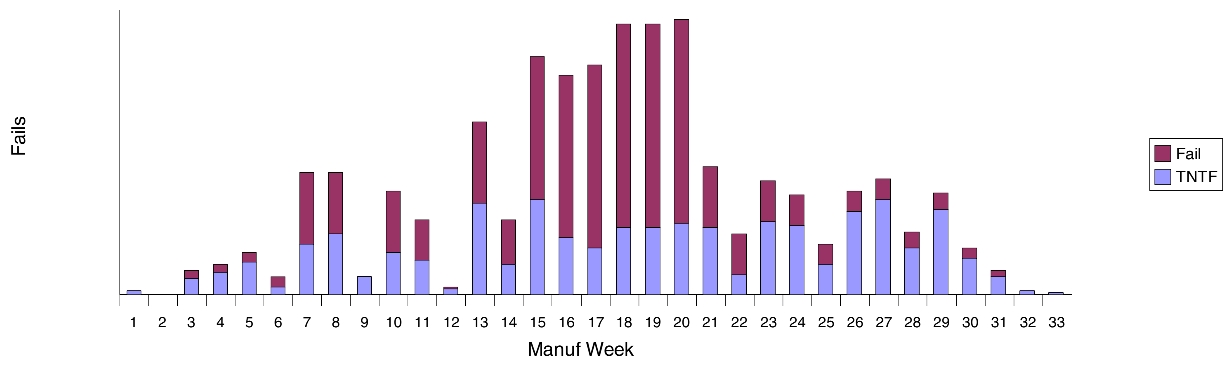

Figure 4. NTF (blue) vs. Fails (Red)

also on the board. At this point a large amount of further testing is required to root cause the problem. The ASIC might be at fault, the architecture of the board might be at fault, the processor I/O might not be quite fast enough, or the problem can be in the interaction of the processor and ASIC. These techniques will not provide us with the answer to the quality problem, but just a good place to look.

Traditional debug methods might never find this problem. Unless we are lucky enough to test the returned parts with an ASIC of the proper vintage in the proper location, the circumstances causing the failure might never be repeated.

In this example, the processor was indicted correctly. The board will likely fail on retest, and since all the ASICs are likely to be of a similar vintage, the failure should follow the processor when it is placed in another location. When the component is removed, the environment of the failure is lost, and unless it is fortuitously reproduced on the system used for reverification, the part will be SNTF.

Data mining techniques are not needed when the returned component is fails retest. They can also be used for non-component related failures, such as attempting to understand the causes of FRU NTFs.

5.4 Requirements for NTF Resolution

Not every product can make use of the techniques described in the last section. The requirements for such a capability are:

- Fab-to-field data gathering and data storage. Data must be recorded for items such as installation dates, failure causes, repair actions and system configuration. Data quality is important, because if a portion of data is missing the results of data mining might be incorrect or misleading. In addition we need a vertical slice through the system, including information on components, FRUs and systems.

- High percentage of failures are diagnosed and repaired. This implies that the systems for which this analysis is to be done are both mission critical (requiring high levels of availability) and expensive, making repairing instead of scrapping the FRUs cost effective. In a low cost system, independent repair vendors are likely to find it cheaper to replace and scrap a failing FRU. In this case board or system manufacturers will only be able to see gross failure information (from warranty requests or replacement part purchases) and will be less likely to make a correct diagnosis of a problem.
- Customer requirements justify cost. The expense of the diagnosis and data storage described in the last subsection will not make sense for inexpensive systems without stringent availability requirements. In addition, there must be enough volume for the startup costs of the databases and data acquisition processes needed to be amortized over a large number of systems.
- The data collection, storage, and access infrastructure exists. We have just touched on the data that needs to be stored for an effective information-based diagnostic system. Terabytes of data must be stored, a large number of accesses should be possible in a reasonable amount of time, and access to the system should be possible through an effective interface for those who are not database experts.

The beginnings of such a system already exists at Sun, and it has been used to find a problem that had resisted traditional debug and test writing.

6. Benefits and Conclusions

NTFs have been traditionally analyzed by examining the symptoms of failure at the next higher level and attempting to write tests to reproduce the conditions under which the component failed. This method is labor and equipment intensive, and often does not produce results. Therefore most of the high percentage of returns that are NTF never get resolved.

We showed in [1] that in-line NTFs represent escapes from IC test, and that if the quantities are within the range expected from our less than perfect tests they are less worrisome than returns which fail retest. We have shown here that the situation for field NTFs is not so sanguine. First, the cost of a field failure, and the urgency of determining the cause of the failure, are much greater than for a manufacturing fail. Second, while some SNTFs or TNTFs may be escapes from both IC and board test, some may be indicative of hard to reproduce but significant quality issues. Third, being able to partition field SNTFs into escapes and other issues would allow us to estimate the IC coverage of our system test, which would tell us if it needed to be improved. Even with the uncertainty we face today, however, we can still determine some bounds on coverage.

We provided some new methods of pinpointing the causes of SNTFs. Statistical diagnosis is not new, the ILIAD [6] system recommended repair actions based on the repairs that successfully corrected similar problems in the past. Determining if a part is actually bad based on sub-populations and the performance of like parts is somewhat similar to the Statistical Post-Processing method described in [7]. Collecting information across a customer base is far more challenging than collecting it in a fab, though.

Components that fail retest are treated as before, but those that pass retest should not be automatically classed as reliability problems without further study.

The benefits of an NTF-based field return approach are:

- preventing an unnecessary increase in stress testing due to field returns that are either escapes or not component reliability issues.
- faster debug time through a parallel data mining effort that can often tell debug engineers where to look for the cause of the problem.
- better information on NTFs by analysis along several dimensions. Process issues can be detected, and variations across vendors that result in high NTF rates can be found.
- splitting fail populations into NTFs and non-NTFs and separately analyzing them can eliminate noise that hinders the analysis of a true reliability problem, and can prevent a recall that can actually hurt quality of parts in the field instead of helping it.

Though our work on NTFs has allowed us to put in inexpensive early screens for reliability problems, and has helped partition SNTFs, this work is still preliminary. Analysis of SNTFs is ad hoc, and getting data on the history of large numbers of parts is still difficult. Analysis is still manual, and not understood well enough yet for a tool to be developed. Our analysis of system test coverage is still preliminary, and has been applied only to microprocessors, not ASICs.

Finally, the problem of FRU or board NTFs is greater than that of IC NTFs. It should be possible to apply some of these techniques to FRUs. There are many problems to be addressed before this can be done. For instance, the concept of yield is unclear for components that are repaired. Can the yield for a FRU be considered a function of yields at each assembly step? Certainly we would expect a board that passed all tests the first time to be more reliable than one which has been repaired several times before shipment, but how much more reliable it is needs to be quantified. While we are still far from a science of component NTFs, we are still further from a science of board NTFs.

6. Acknowledgments

Thanks to a number of people at Sun, including Dorthe Clarke, Kanti Bhabuthmal, Aaron Hale, Sheri Lloyd and Ivy Chan, who have collected data that I rummaged through, and Sridhar Vajapey who asked difficult questions that started me thinking about NTFs.

7. References

[1] S. Davidson, “Towards an Understanding of No Trouble Found Devices,” 2005 VLSI Test Symposium, pp. 147-152, May 2005.

[2] P. Maxwell and R. Aitken, “All Fault Coverages are Not Created Equal”, *IEEE Design & Test of Computers*, Mar. 1993, pp. 42-51.

[3] T. W. Williams and N. C. Brown, “Defect Level as a Function of Fault Coverage,” *IEEE Trans. Computers,* Vol. C-30, No. 12, Dec. 1981, pp. 987-988.

[4] V. D. Agrawal, S. C. Seth, and P. Agrawal, “Fault Coverage Requirements in Production Testing of LSI Circuits,” *IEEE J. Solid State Circuits,* Vol. SC-17, No. 1, Feb. 1982, pp. 57-61.

[5] Wolfram, S., *The Mathematica Book Online,* http://documents.wolfram.com/mathematica/

[6] Yau, C. W. "ILIAD: A Computer-Aided Diagnosis and Repair System," Proc. 1987 International Test Conference, Sept. 1987, pp. 890-898.

[7] Madge, R., M. Rehani, K. Cota, W. R. Daasch, "Statistical Post-Processing at Wafer Sort – An Alternative to Burn-in and a Manufacturable Solution to Test Limit Setting for Sub-micron Technologies," VLSI Test Symposium, pp. 69-74, April 2002.