An Overview of Advanced Failure Analysis Techniques for Pentium® and Pentium® Pro Microprocessors

Yeoh Eng Hong, Intel Penang Microprocessor Failure Analysis Department, Malaysia
Lim Seong Leong, Intel Penang Microprocessor Failure Analysis Department, Malaysia
Wong Yik Choong, Intel Penang Microprocessor Failure Analysis Department, Malaysia
Lock Choon Hou, Intel Penang Microprocessor Failure Analysis Department, Malaysia
Mahmud Adnan, Intel Penang Microprocessor Failure Analysis Department, Malaysia

Index words: failure analysis, DFT, DFFA, e-beam, DPM, RTL

Abstract

Failure analysis (FA) is one of the key competencies in Intel. It enables very rapid achievement of world class manufacturing standards, resulting in excellent microprocessor time-to-market performance. This paper discusses the evolution of FA techniques from one generation of microprocessors to another.

According to Moore's law, transistor count doubles as transistor dimensions are reduced in half every 18 months, allowing for more complex microprocessor architecture designs. For example the Intel486DX™ microprocessor had 1.2 million transistors while the Pentium® microprocessor contains 3.1 million transistors. With rapid technological advances such as more complex microprocessor architecture, an increasing number of interconnect layers, and flip-chip packaging technology for products like the Pentium® and Pentium® II microprocessors, conventional FA techniques, in use since the Intel386DX™ processor generation, are no longer effective. These conventional techniques require in-depth knowledge of the processor's architecture, and they involve exhaustive e-beam probing work, which typically results in very long FA throughput times. Other traditional defect localization techniques, such as emission microscopy from the frontside of the die, are also becoming less successful due to the increased number of metal interconnect layers that obscure local circuitry.

This paper provides insight into FA techniques that have been adopted at Intel. It discusses the evolution of software fault isolation techniques based on Design For Testability (DFT) features, and other special FA techniques. In this paper, we will discuss these techniques and show how they are effectively used to produce fast FA support turnaround for both silicon debug and manufacturing. We will also review their technical merits and return on investment, as well as the cost of each technique to Intel. The main focus of this paper is electrical fault isolation techniques, as opposed to physical defect localization techniques such as liquid crystal analysis and emission microscopy.

In the context of this paper, Fault Localization and Fault Isolation (FI) are synonymous. These are defined as the task of electrically isolating the location of a defect in logical space. Another approach, termed Defect Localization, refers to the task of isolating the location of a defect in physical space.

Introduction

Failure analysis plays a very important role in the semiconductor industry in enabling timely product time-to-market and world-class manufacturing standards (greater than 95% manufacturing yields, lower than 100 defects per million, or DPM). At Intel, the situation is even more compelling: quick failure analysis turnaround is necessary to support Intel's steep product ramp and high-volume manufacturing, where annual microprocessor production volume is in the range of tens of millions of units.

However, the increasing complexity of microprocessors in the form of more complex architecture designs, shrinking transistor feature sizes, and new packaging technologies have significantly increased the failure analysis challenges on Intel's Pentium and Pentium Pro family of microprocessors. A roadmap recently published by the Semiconductor Industry Association (SIA) predicts that by the year 2000, microprocessor transistor counts will exceed 21 million transistors. The same industry roadmap also predicts that by that time, microprocessors will utilize

full flip-chip technology, instead of current wirebond assembly and packaging technology. Traditional fault isolation techniques using intensive e-beam probing and assembly code minimization as well as die frontside defect localization with liquid crystals or emission microscopy are no longer sufficient even for today's complex microprocessors like the Pentium and Pentium Pro microprocessors.

Newer FA techniques based on design-for-testability (DFT) and design-for-failure-analysis (DFFA) features have proven to be highly successful for the Pentium and Pentium Pro microprocessors, as evidenced by high analysis success rates (>90%) and short analysis throughput time. Without these new FA techniques that pinpoint the exact failing location, detection of sub-micron defects such as silicon dislocation, as shown in Figure 1 below, is very difficult if not impossible.

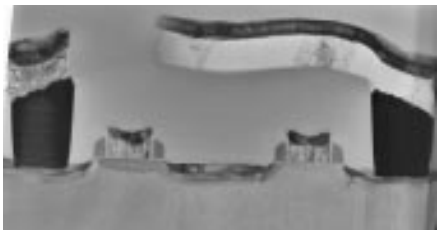

Figure 1: A TEM micrograph of silicon dislocation

This paper provides insight into the various FA techniques based on the DFT and DFFA features that have been successfully developed for the Pentium and Pentium Pro microprocessors. Two other advanced FA techniques and tools will also be discussed. These two techniques are the low-cost personal computer (PC) system-level tester solution and the processor cartridge-level (SECC) FA technique.

Failure Analysis Overview

Figure 2 illustrates a typical FA flow used at Intel. The first step is to verify the failure on a functional tester in the failure analysis lab to ensure that the failure's electrical signature observed on the production tester can be duplicated on the lab's functional tester. Fault localization or defect localization is the next step. Its function is to narrow down the fault to a small block of circuitry. In this case, a small block of circuitry could just be a via, a transistor, a gate, or even a functional unit block (FUB). Reverse engineering, or physical FA as it is better known, is the next step. Individual interconnect layers are selectively removed either mechanically or chemically and examined for defects. Occasionally, defect characterization is introduced to gain more insight into the defect's behavior over time and under stress. This is crucial in order to develop the best possible defect screen to maintain a high quality product. A case is considered closed when a root cause is determined and corrective action is put in place. A corrective action could be a minor change such as the addition of new test screens, or it could also involve major re-engineering via process and design fixes.

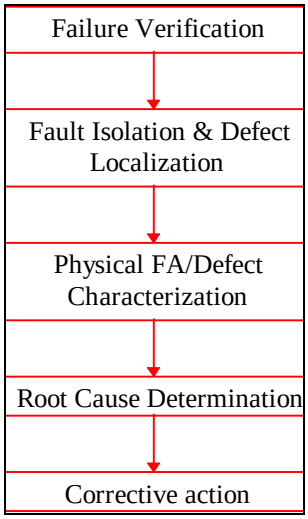

Figure 2: A typical failure analysis flow

From the flow in Figure 2, it is clear that the key to determining the overall FA success rate actually lies in the fault localization and defect localization steps. These steps provide the spatial information on where the defect is, and they control the rest of the steps.

Fault Localization

As the name implies, fault localization is the technique of localizing a fault in a failing circuit functionally and logically. Traditionally, this is a two-step process that involves coarse-level isolation using assembly code minimization followed by fine-level isolation using e-beam probing. With increasing device architecture complexity, coarse-level isolation using assembly code minimization becomes very time consuming and ineffective. Figure 3 below depicts the evolution of the two-step fault localization flows for different generations of Intel's microprocessors.

DFT and DFFA features built into newer microprocessors to accelerate the silicon debug and production testing processes can also be employed as coarse-level fault-localization tools. The use of DFT and DFFA features as FA tools in conjunction with e-beam probing is proven to be very successful on the Intel486DX and Pentium processors where the internal signals of interest are still accessible for e-beam probing. However, the increasing number of interconnect layers (Intel's Pentium® 90/100 MHz processor has four metal layers, and the Pentium® II

333 MHz processor has five metal layers) and changed packaging technologies from frontside wirebond technology to full flip-chip technology makes the e-beam probing process even more challenging. The use of Register Transfer Level (RTL) simulation as a fine-level isolation tool addresses this challenge.

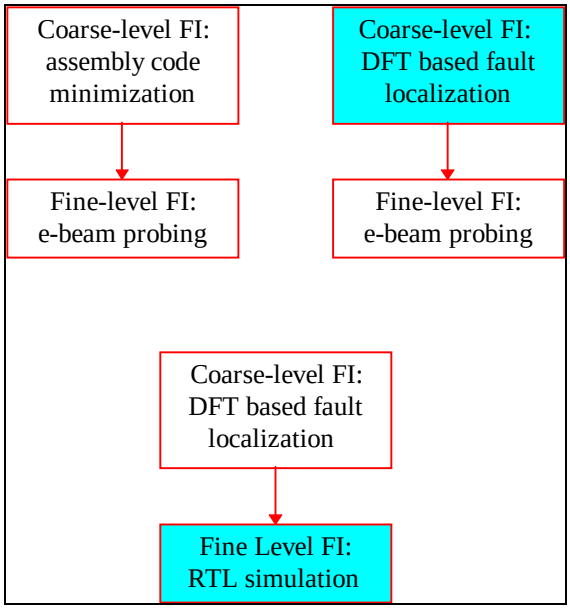

Figure 3: An overview of various fault-isolation approaches for the Intel386DX™ processor (top left), the Intel486DX™ and Pentium® processors (top right), and the Pentium® Pro and Pentium II® processors (bottom)

DFT-Based Fault Isolation Techniques

The following sections elaborate on the new fault localization techniques adopted from DFT features currently available on the Pentium and Pentium Pro microprocessors.

Micropatching

Microcodes are integrated instructions that dictate the processor's internal operation. In traditional architecture design, access to built-in microcodes is restricted. However, an ability to create new microcodes and control their flow would be very useful for fault isolation. With such a tool, the failure analyst would be able to create customized microcode subroutines and control microcode flow. This capability is now realized on the Pentium Pro microprocessor family through the introduction of the Micropatching DFT feature. This capability has caused significant breakthroughs in debugging processors with microcode failures. For example, with micropatching, it is now possible to systematically perform fault isolation on failures in the processor's reset subroutine, which is implemented in microcode.

The Micropatching DFT feature consists of two key elements: the microcode patch RAM and several pairs of Match and Destination registers. The microcode patch RAM stores externally programmed microcodes. (From here on, programmed microcodes that reside in the microcode RAM will be called *external* microcodes and built-in microcodes that reside in the microcode ROM will be called *internal* microcodes.)

The Match and Destination registers are used for controlling the microcode flow. Whenever a microcode address in Microcode Instruction Pointer (UIP) matches the content of a Match register, the UIP will be reloaded with a new address from the Destination register.

In order to control the microcode flow, the Match registers are loaded with the UIP that the user intends to jump from. Similarly, the Destination register must be set to the UIP that the user wishes to jump to.

Besides its usefulness in electrical fault isolation, the Micropatching FA tool can also be used in conjunction with liquid crystal analysis and emission microscopy. Without Micropatching, the processor may need to execute many other instructions in the test pattern completely unrelated to the failure before reaching the instruction that causes the failure. By then, the CPU would have generated a lot of heat and made the liquid crystal analysis less sensitive to the heat generated by the failing instruction (i.e., the heat generated by the defective circuitry). With Micropatching, the UIP for the reset subroutine can be set in the Match register to point to the failing UIP, thereby bypassing the reset subroutine altogether. This minimizes the generation of surrounding heat that could result in genuine hot spots, caused by the defect, to be hidden.

Array Dump

Memory arrays are important elements in a processor. Memory arrays are usually used for storing data, control signals, processor status, etc. Visibility into these arrays helps a failure analyst understand the internal operation of the device thereby speeding up the fault isolation process and reducing e-beam probing time. It is almost impossible to analyze a dynamic execution machine such as the Pentium Pro processor without knowing the contents of the arrays. With the Array Dump tool, the contents of many important arrays can be dumped out to produce snapshots of the arrays at any core clock. In order to save data analysis time and avoid human error, a post-processing program was developed to compare the acquired data with that from RTL simulation. Mismatches are automatically highlighted.

The Array Dump FA tool is developed by applying two new DFT features in the Pentium Pro processor family. The first DFT feature, called Micro Breakpoint, is one of the built-in debug trigger-response mechanisms that allows the processor to execute certain tasks in response to preprogrammed trigger events. An external debug breakpoint pin is set up to trigger a processor "micro-breakpoint" event when the pin is asserted. In response to this trigger event, the processor executes a preprogrammed debug command called "All Array Freeze" which is the second key DFT feature that makes Array Dump possible. The "All Array Freeze" feature causes all major memory arrays in the device to freeze their normal execution. The Array Dump feature is capable of acquiring array data at any core cycle relative to a processor's external bus cycle.

Scanout

Accessibility to internal control and datapath signals also greatly enhances FA capability. Scanout is the FA tool developed at Intel for monitoring internal control signals. Scanout has already been successfully and regularly used in Pentium microprocessor failure analysis. One of the Pentium processor's external pins is used to trigger a Test Access Port (TAP) instruction that causes the data from selected control and datapath signals to be latched into Scanout data buffers. The data captured and stored in these buffers are then shifted out serially through the TAP controller's Test Data Output (TDO) pin.

The Scanout FA tool's implementation on the Pentium Pro microprocessor is triggered differently from the Pentium processor's method. This is necessitated by the introduction of odd bus-to-core clock ratios. To overcome this problem, an innovative approach that uses the Micro Breakpoint DFT feature was developed similar to the one in the Array Dump tool. This Micro Breakpoint trigger-response mechanism makes it possible to capture Scanout data on every core cycle of the processor's execution.

A total of around 2000 internal nodes are available for observation. This coverage is much wider than what was available on the original Pentium processor (about 200 Scanout nodes) and newer versions of the Pentium processor family (about 400 Scanout nodes). A program was also written to compare the Scanout data with RTL simulation to aid in interpreting the data. The tool automatically highlights Scanout mismatches.

Memory DAT and LYA Mode

Direct Access Test (DAT) is a special test mode that is specifically intended for manufacturing use. Newer implementations of the Pentium Pro processors contain DAT capability to enhance the testability of most of the major core memory structures. Once DAT mode is enabled, the processor will behave like a pipelined SRAM. It accepts DAT commands every bus clock and returns data for a DAT memory read command to the external output bus a few clocks later.

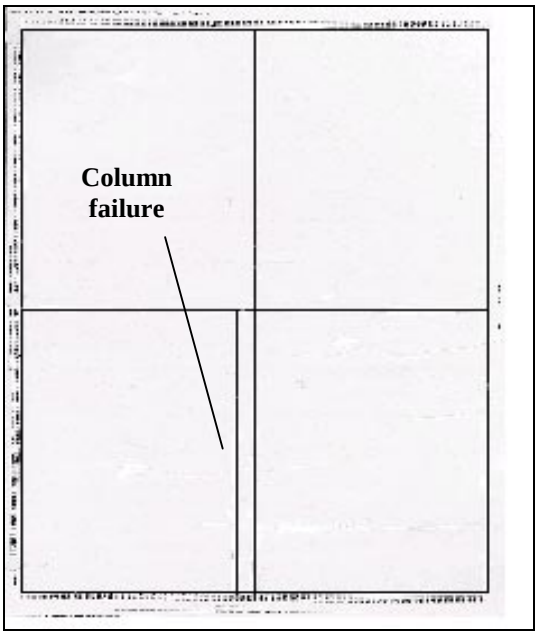

Figure 4: Example of a cache raster bitmap display showing a cache column failure

In DAT mode, the processor only understands DAT instructions. These instructions incorporate the memory array's address information, input data, and array access commands to specify a particular cache operation. The information is directly fed to the array under test. This direct access bypasses the decoding circuitry used in normal operation. Thus, it reduces the amount of effort required to determine whether a cache failure is caused by a problem in the memory array itself or a problem in the supporting circuitry such as the address decoders.

Because only a subset of the external pins are needed in DAT mode, it is very easy to develop a small but effective cache-testing pattern to raster the entire cache line within one memory array. Also, the entire set of advanced cache-testing algorithms used in production testing can be easily written into a small test pattern. If a failure occurs in production, the rastering program can easily provide the failing set and the way and bit information in a bitmap form for failure analysis purposes. Figure 4 above shows an example of the cache rastering program's bitmap display. (In this figure, bits (columns) are arranged horizontally, and cache lines (rows) are arranged vertically.)

However, although DAT mode testing provides the failing set and way and bit information, it provides little information about the failure mechanism. In the latest version of the Pentium Pro processor, a Low Yield Analysis (LYA) mode was added as an enhancement to DAT mode for the largest cache structure.

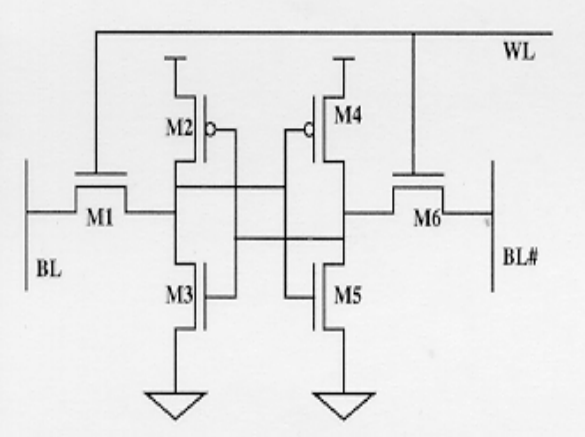

Figure 5: Schematic diagram of a 6-transistor SRAM cell for LYA mode illustration

When LYA mode is enabled, the bitline (BL) and bitline# (BL#) signals of a memory cell are connected to external pins. By having this direct access through the external pins, DC measurements on the memory cell can be carried out easily. LYA mode testing produces cell transistor current readings, as well as cell trip points and other DC measurements. From these LYA "signatures" acquired from a failing device, the failure can be quickly categorized into a few different failure mechanisms, such as open transistor, missing contact, or shorted transistor. With this information, the defect location can be more accurately located, and the physical FA can focus on that location. This results in shorter analysis throughput time and improved success rates in handling cache failures.

Figure 5 above shows a 6-transistor memory cell. When LYA mode is enabled, the BL and BL# signals of this memory cell are directly accessible from the external pins. Enabling the word line select (WL) signal activates this memory cell. In this case, both transistors M1 and M6 are turned on to allow various DC measurements to be performed on this memory cell.

PBIST

Programmable Built-In Self-Test (PBIST) is a memory DFT feature that incorporates all the required test systems into the chip itself. The test systems implemented on-chip are as follows:

- algorithmic address generator
- algorithmic data generator
- program storage unit
- loop control mechanisms

PBIST was originally adopted by large memory chips that have high pin counts and operate at high frequencies, thereby exceeding the capability of production testers. The purpose of PBIST is to avoid developing and buying more sophisticated and very expensive testers.

The interface between PBIST, which is internal to the processor, and the external tester environment is through the standard TAP controller pins. Algorithms and controls are fed into the chip through the TAP controller's Test Data Input (TDI) pin. The final result of the PBIST test is read out through the TDO pin.

PBIST supports the entire algorithmic memory testing requirements imposed by the production testing methodology. In order to support all of the required test algorithms, PBIST must have the capability to store the required programs locally in the device. It must also be able to perform different address generation schemes, different test data pattern generation, looping schemes, and data comparisons.

The program storage structure is used to store the test algorithm. The algorithm includes the types of operations to be performed (e.g., memory read, memory write), the types of address generation modes, the data to be written into and read out from the memory array, and the types of looping schemes.

The address generator is responsible for generating the memory address where the next data are read from or written into. Correct address generation is very important because the physical mapping of the array is always different from its logical mapping. In order to achieve the required test coverage, the address generator needs to be able to generate addresses in different fashions in order to accommodate different kinds of addressing flows such as March-C, Galloping patterns, Address Complements, Fast X, and Fast Y.

The data generator plays a very similar role to the address generator. In order to get the inverse data for each physically adjacent cell, data has to be generated based on the logical-to-physical mapping of the memory array and the data background that is required, such as checkerboard, reverse checkerboard, column stripe, row stripe, and diagonal.

The loop control system is the major sequencing logic in PBIST that allows testing of the entire memory array using only a few program steps. Without the looping control mechanism, test programs of thousands of lines need to be written in order to test an entire array. With

PBIST, testing of the large on-chip memory arrays becomes a lot simpler. Also, PBIST can be used for Test During Burn-In (TDBI) where the processor is tested in the burn-in oven. Other cache-testing methods cannot be used because only the TAP controller pins and a few other control pins are active in the burn-in environment while the rest of the pins are tristated.

RTL Simulation For FA

The motivation for doing fault isolation based on RTL simulation is driven by the need for detailed information about the internal workings of the processor. This information is readily available from the RTL model. Furthermore, as described in the previous sections, the implementation of more sophisticated DFT and debug features in the Pentium, Pentium Pro, and Pentium II generations of microprocessors have helped to promote this technique by providing better observability of the internal signals, thus resulting in less dependency on e-beam probing.

In previous generations of Intel's microprocessors, RTL simulation was used in the failure analysis flow primarily for test pattern generation, test modification, and signal tracing. But the fault isolation process largely depended on extensive e-beam waveform probing work to trace the failure from an architectural starting point, moving upstream until the signals acquired at the inputs of a logic block were correct, while the output was faulty.

Because of the lack of observability of the internal nodes, e-beam probing became the necessary means to gathering the detailed internal signal information of a microprocessor. However, relying on e-beam probing alone has become increasingly difficult or even futile on current generations of microprocessors as more metal interconnect layers are used (obscuring most local signals that run only in Metal1 and Metal2 layers). Additionally, while products are beginning to move into C4 packaging, next-generation waveform probers that allow probing of internal signals through the backside of the die are still not widely available.

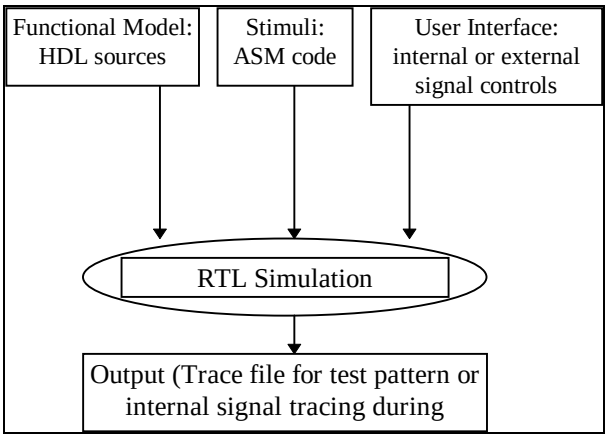

Figure 6: A high-level block diagram of the RTL model simulation environment

Advances and improvements in today's RTL simulation tools used for Intel microprocessors result in a better and more efficient environment for performing fault isolation than that of the previous generation of RTL simulation tools. More sophisticated capabilities and user-friendly features have been added to the RTL simulation environment, such as more extensive node coverage; the capability to use application programming interface (API) programs; an interactive simulation mode; and reliable simulation state, save, and restore functions. Employing other design tools such as schematic viewers, layout databases, and circuit-level simulation to complement the RTL simulation analysis further helps in the fault isolation process. Figure 6 provides an overview of the key components of the RTL model simulation environment.

The simulation tool's API enables application programs to access signal information from the full chip RTL model of the microprocessor during the model's execution. The application program runs as a separate process and uses routines provided in the API library to obtain signal information from the model. The primary purpose of the API is to isolate the model from user events, enabling the designer (as well as failure analyst) to quickly modify application programs without having to relink the model. The advantage of such an API is its reusability. It is platform independent and can run with several models. For the average user, the API is easier to support than a user event, and it is simpler to write and debug.

A very useful RTL simulation API is the *p6watch* tool, which enables visualization of large amounts of RTL signal values by displaying them in a human format. For example, instead of displaying a set of control signals for a finite state machine (FSM) in binary or hexadecimal format, the user can use p6watch to convert the signals to strings that represent the individual states of the FSM. Another example would be to use p6watch to display the

field contents of an array in terms of their decoded or functional meanings.

In this method of performing fault isolation, the failure location is deduced based on extensive analysis of the full-chip RTL model. The failure analyst is also required to have an understanding of the processor's microarchitecture to aid in interpreting the HDL information and the RTL simulation results, and comparing these data with the behavior observed in the failing device. This fault isolation method is very similar to RTL debugging during the design phase.

The RTL simulator provides crucial information on the expected behavior of the processor. By running the tests that cause the DUT to fail using the RTL simulator, the failure analyst can observe the processor's pipe stages and see how instructions and data are being transacted and propagated. It is also useful to run passing tests to compare the differences in control and data handling. Among the things to look for at the high level is the flow of data along the datapath blocks and the control signals that arbitrate the machine pipeline. The gathered information will help to determine the logic block that is most likely to have caused the failure.

When the logic block has been isolated, more detailed RTL signal analysis is performed within that logic block. By isolating the failure to a small logic block, the analyst can generate more exhaustive and more focused tests for that block of logic. These new focused tests are then run on the failing device to see how it responds to each test case. For example, if a failure is isolated to a multi-ported memory block, then the focused tests are targeted at all of the available ports to see if there is any specific dependency of the failure on a particular port. Other test cases would involve testing the address generation logic.

Moreover, when running a test it is possible to introduce a "defect" into the model and simulate its effects in terms of the processor's response to do a "what-if" type of analyses.

E-beam-Less Fault Isolation

E-beam-less fault isolation can produce results faster and reduce analysis by focusing on test data collection, data analysis, and comparison with simulation results. The cost of the analysis is reduced by avoiding the e-beam probing step in the fault isolation process. E-beam probing work involves many hours of sample preparation time (to perform depassivation and prepare probe holes using the FIB) as well as many more hours of waveform acquisition time. Avoiding the e-beam probing step can save a significant amount of analysis time. However, in order to skip e-beam probing work, an alternative method is required to collect valuable information from the failed device. This alternative approach is described in the following paragraphs in this section.

First of all, the failing signature needs to be identified in order to understand why the unit is failing from the standpoint of the processor architecture. Then, DFT tools are used to collect internal node information to further understand the failure mode. Later, RTL simulation is performed to verify the hypothesis made based on the collected data and to identify the failing node. As RTL simulation is usually good enough to anticipate the processor's internal operation and build a hypothesis about the failure location, probing is not necessary. When compared to the traditional approach where probing is needed to confirm the failing node, which takes an average of 31 days, this e-beam-less approach takes an average of only 15 days throughput time. This new approach has been shown to produce a very accurate estimation of the defect location with greater than 95% success rate on the Pentium and Pentium Pro microprocessors. The high success rate can be attributed to the avoidance of the high-risk steps involved in e-beam probing, especially during sample preparation. This method readily and efficiently supports Intel's virtual factory concept where fault isolation work is easily shared among all participating factories. By using the FA tools, data collection can be done at the site where the failing device is located. These data are then sent to the site where the product FI expertise resides for in-depth analysis. The predicted defect location is then sent back to the original site for physical FA.

Advanced FI Techniques

The next sections of this paper present the advanced fault isolation techniques used on the Pentium and Pentium Pro microprocessor families. Specifically, these techniques involve the PC FA tool and the SECC FA tool, both of which were developed to address issues that will be discussed in the following sections, and to overcome the challenges and roadblocks described in the introduction.

PC-based FA Tester Platform

With the increasing complexity of Intel's microprocessor designs, failure analysis TPT would be longer without the adoption of DFT and DFFA features in fault isolation techniques. However, most of the DFT-based fault isolation tools and techniques have been developed for use on expensive functional testers. To reduce the dependency on these expensive lab testers, a new approach has been developed based on the personal computer (PC) platform. The key elements of the PC FA tester platform are shown in Figure 7.

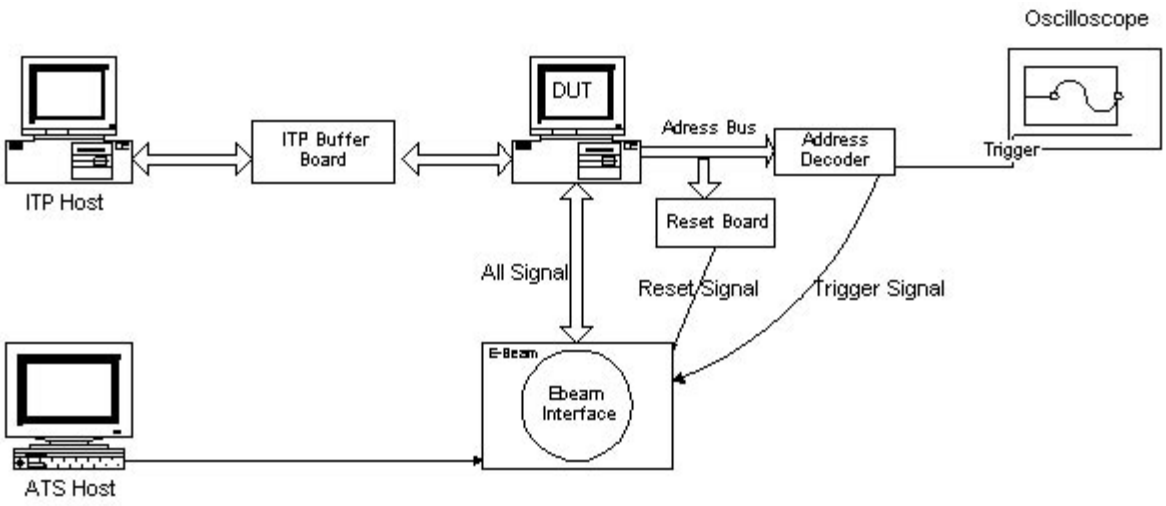

Figure 7: A simplistic block diagram of the PC-based FA platform configuration

The In-Target Probe (ITP) tool has been widely used for system-level debug and validation. On this test platform, a host PC is used to control a target PC where the device under test (DUT) resides. The host PC is used to control the DUT and to upload the test program that the DUT will execute. The ITP is developed around the processor's *Probe Mode* debug feature and is able to access all programmer-visible and non-visible registers in the DUT. This gives the ITP the necessary control mechanisms to halt the DUT, modify its internal state, and resume normal program execution at any chosen instruction boundary.

In this new approach, the ITP is used as the controller of the target PC where the failing pattern can be uploaded to the target PC through the TAP controller interface. The host-target PC configuration acts like a tester where it drives the necessary data to the input pins of the microprocessor. However, the output pins of the microprocessor are not strobed to compare with expected data. Rather, the output data is monitored by a hardware board that sends out a signal when certain conditions are met. This signal can be used as a trigger signal, such as the e-beam trigger signal, to enable hooking up the DUT to the e-beam, although this has only been proven experimentally.

In order to do e-beam probing, the DUT needs to execute the test program in a loop. To achieve this, another hardware board is used to monitor the activities of the DUT. When a preset condition is met, the hardware board sends out a signal that is used to reset the DUT. Once reset, the DUT restarts execution from the normal boot-up address and reruns the entire test pattern.

FA tools based on the processor's DFT features have also been implemented on the PC test platform where the host PC is used as an interface to the DUT. Commands to invoke the DFT features are fed into the DUT through the TAP controller interface on the ITP board. The results of the test are dumped to the host PC through the same TAP controller interface.

FA tools that employ the DFT features such as Scanout, Array Dump, and cache rastering have been developed for the PC test platform. To use tools such as Scanout and Array Dump, the DUT is initially halted. The required register for stopping the DUT from execution is programmed through the ITP. The trigger condition for the breakpoint is set, and the desired processor response to the breakpoint is also programmed. Next, the original architectural state is restored, and the DUT is allowed to resume execution from the point where it was halted. When the preprogrammed breakpoint condition is met, the DUT will stop, and as a result of the response to the breakpoint, the data are shifted out through the ITP to the host PC. These data provide information on the internal state of the DUT to enable further fault isolation.

To perform cache rastering on the DUT, a data background for the processor's internal cache is preset by writing into each memory location. Examples of the patterns typically used for cache rastering include checkerboard and reverse checkerboard patterns. Next, the data in the memory array are read out and compared with the data that were originally written. If there is any discrepancy between the acquired and expected data, the failing memory cell is identified. The PC is a perfect platform for data-retention type of memory testing because all commands and data are shifted serially through the TAP controller. This method takes longer to

perform cache rastering as compared to lab functional testers, but it is cheaper, and it frees up the utilization time of the lab testers.

By providing capabilities similar to a lab tester such as providing stimuli to drive the DUT, enabling interface to the e-beam, and supporting various FA tools, the PC test platform can be used for functional FA work, a task which had been performed predominantly by lab testers. The complete test platform is significantly cheaper and more compact than the traditional workstation and tester combination.

A few FA cases have been successfully resolved using this PC test platform. E-beam waveform of a clock signal has been acquired indicating the feasibility of using the PC to replace a tester.

SECC Form-Factor Testing

The concept of processor *bus fractions* was introduced to make it possible to increase the processor's core clock frequency while at the same time maintaining the external bus clock frequency at a lower speed. To support this idea, a frequency multiplier is designed into the processor to multiply the external bus clock to produce a higher frequency clock in the processor core. The term "1:2 bus fraction" refers to a clock configuration in which the internal processor frequency is twice as fast as the speed of the external bus. For example, for an external processor bus clock of 66 MHz, the internal clock frequency is 133 MHz. In this situation, there are two clock domains: one running at 66 MHz and another running at 133 MHz. Extra care is required when developing test programs for products that support bus fractions: input and output data that cross the clock boundaries must be correctly aligned. The Pentium processor only has one bus clock domain and one core clock domain. These two different clock domains are isolated from each other by the processor's input and output buffers. In this case, data alignment is handled by the processor itself, thus reducing the amount of complexity associated with aligning the data in the test pattern.

The Pentium II processor introduces a new bus fraction concept, which increases the complexity of the test pattern development and testing. The Pentium II processor implements two separate sets of bus frequencies: a frontside and a backside. The frontside bus is connected to external components such as chipsets and other peripheral interface devices. Alternatively, the backside bus is a local bus for the Pentium II processor that is connected only to the level-2 (L2) memory. These two buses run at different clock speeds. The backside bus frequency is always equal to or faster than the frontside bus frequency. Together with the internal core clock, there are a total of three different clock domains in this implementation. In this case, the processor's bus fraction configuration is described in terms of all three clock domains relative to each other. The term "1:4:2 bus fraction" refers to a processor with a backside bus running 2x faster than the frontside bus, and a core frequency that is 4x faster than the frontside bus or 2x faster than the backside bus. The data alignment between the frontside bus and the core logic is handled by the frontside input and output buffers. Meanwhile, the data alignment between the backside bus and the core logic is handled by the backside input and output buffers. The user, or the test program developer, is responsible for managing the data alignment between the frontside and backside buses.

For a round numbered (or even) bus fraction configuration between the frontside bus and the backside bus, the data alignment can be easily handled by the test program. This is illustrated in Figures 8 and 9 below.

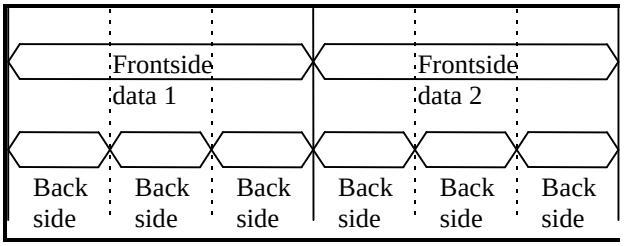

Figure 8: A processor's frontside and backside data switching and alignment for a 1:3 (frontside-to-backside) bus fraction

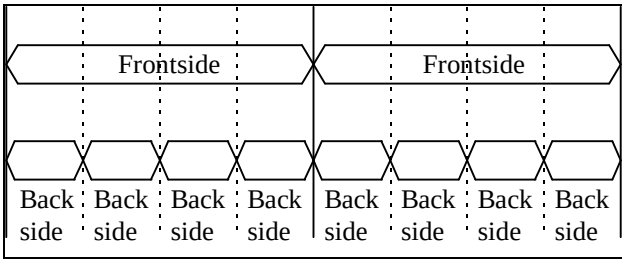

Figure 9: A processor's frontside and backside data switching and alignment for a 1:4 (frontside-to-backside) bus fraction

In the examples shown in both of these figures, frontside data switches at the same time as backside data. This boundary can be used as the alignment point for both the frontside and backside data. However, in a fractional (or odd) bus fraction configuration, data switching and data alignment become more complicated. An example of a fractional frontside-to-backside bus fraction is a 4:7 (frontside to backside) configuration.

In the 4:7 example shown in Figure 10 below, when the frontside data FD1 switches to FD2, the data are not aligned with backside data switching. The same situation

occurs during the transitions from FD2 to FD3 and FD3 to FD4.

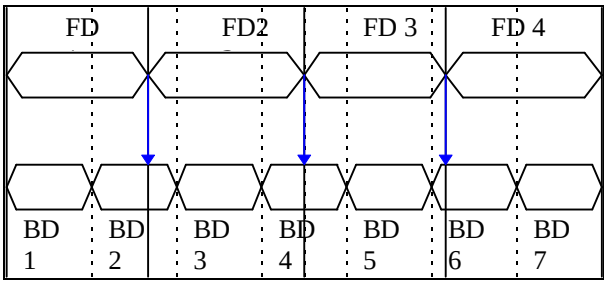

Figure 10: The die frontside and backside data switching and alignment for 4:7 (frontside-to-backside) bus fraction

The functional tester needs to simultaneously drive input data to the processor and strobe for the output data on both the frontside and the backside bus. Advanced production testers are capable of handling these situations because they offer very flexible timing capabilities where the rising and failing edges of the data can be placed anywhere within a tester period. This allows the edge placement of the first cycle to be different from the edge placement of the second cycle. Unfortunately, the failure analysis test platform employed by the FA labs does not support advanced flexible timing capabilities. Therefore, functional testing of a Pentium II processor in a lab environment becomes very difficult.

The fractional bus testing problem is addressed by mounting the Pentium II processor onto a Single Edge Contact Cartridge (SECC) card. The backside bus, which used to be accessible externally, now only communicates with the on-board L2 cache. Only the frontside bus is accessible to the external tester through the edge connector.

In the SECC card form factor testing, the FA tester only needs to handle the data transactions occurring on the frontside bus; the backside bus data transactions are transparent to the test platform. Hence, the complex *frontside:core:backside* bus fraction is reduced to a simple *frontside:core* bus fraction. This setup enables the normal FA functional tester to execute test vectors that use complex bus fractions.

To support SECC testing on a normal FA tester, a pattern conversion tool is needed to convert a pattern from component format into SECC format. This includes extracting the frontside bus data and realigning it into SECC format. This concept has been proven to be very successful in analyzing the latest Pentium II microprocessors.

Future Directions In Failure Analysis Techniques

The advent of more sophisticated DFT and debug features, coupled with faster RTL simulators and better computer-aided fault debugging tools increases the usage of the fault isolation methods described in this paper. The trend towards more extensive use of computer based analysis tools is continuing. With the introduction of new DFT features which provide high degrees of controllability in addition to observability in next generation Intel processors, the concept of performing computer-aided fault diagnosis at the logic or RTL level is taken to the next abstraction level of the design, where fault diagnosis is done at the gate level using intelligent computer-aided fault diagnostic tools. By having both observability and controllability of many important internal signals, the concept of virtual probing will be realized.

Conclusion

As device complexity and interconnect layers increase, new fault isolation techniques need to be continuously developed in order to maintain a high analysis success rate and short throughput time. As discussed in this paper, performing FA using traditional methods has been shown to lower FA success rates. By quickly isolating the defect location and identifying the failure mechanism that is causing low manufacturing yields or abnormal failure rates, problems can be fixed in a short amount of time to improve manufacturing yields. In addition, the corrective actions developed as a result of these FA techniques are documented and proliferated to newer products in the form of design rules. This will prevent the problems from recurring, which in turn enables even a steeper volume ramp. The innovative FA techniques developed for the Pentium and Pentium Pro processors have been one of the key elements in the continued exceptional performance of Intel's product time-to-market and high-volume manufacturing.

References

[1] Yeoh Eng Hong, Martin Tay, "The Application of Novel Failure Analysis Techniques for Advanced Multi-Layered CMOS Devices," *International Test Conference*, 1997.

[2] Adrian Carbine, Derek Feltham, "Pentium ® Pro Processor Design for Test and Debug," *International Test Conference*, 1997.

Authors' Biographies

Yeoh Eng Hong graduated from Monash University in 1992. He joined Intel as a microprocessor failure analysis

engineer. He is now the microprocessor product FA manager in Intel Penang. His main technical interests are developing new FA/FI techniques and studying new failure mechanisms. His e-mail address is Yeoh_Eng_Hong@ccm.ipn.intel.com.

Seong Leong, Lim graduated from University Science Malaysia in 1992. He joined Intel as a microprocessor failure analysis engineer. He is now leading the Pentium Pro microprocessor Fault Isolation group in Intel Penang. His main technical interests are microprocessor architecture and fault isolation. His e-mail address is Seong_Leong_Lim@ccm.ipn.intel.com.

Wong Yik Choong graduated from the University of Malaya in 1994. He joined Intel as a microprocessor failure analysis engineer. He is now the lead Pentium II processor FA engineer in the Penang Microprocessor FA department. His main technical interests are computer architecture and networking, and software development. His e-mail address is Yik_Choong_Wong@ccm.ipn.intel.com

Lock Choon Hou graduated from the University of Malaya in 1996. He joined Intel as a microprocessor product failure analysis engineer. His e-mail address is Choon_Hou_Lock@ccm.sc.intel.com

Mahmud Adnan received a BSEE from Bradley University, Peoria, Illinois in 1990. He joined Intel Penang in 1991 as a microprocessor failure analysis engineer. He is currently working on Merced™ processor's DFT design validation and Merced debug and FA tools development. His e-mail address is Mahmud_Adnan@ccm.sc.intel.com.