Test methodologies and design automation for IBM ASICs

by P. S. Gillis
T. S. Guzowski
B. L. Keller
R. H. Kerr

IBM manufactures a very large number of different application-specific integrated circuit (ASIC) chips each year. Although these chips are designed by many different customers having various levels of test experience and all having tight deadlines, IBM ASICs have a reputation for their high quality. This quality is due in large part to the heavy focus on design for test (DFT) and the use of design automation to help ensure that customers' chips can be manufactured, tested, and diagnosed with minimal engineering effort. Prospective customers of IBM ASIC technologies find an explicit set of DFT methodologies to follow which provide a relatively painless, almost push-button approach to the generation of high-quality, "sign-off" test vectors for their chips. This paper discusses the DFT methodologies used for IBM ASICs and the design automation support that enables designers to be so productive with these methodologies. Data are given for several recently processed chips, some designed outside IBM.

Introduction

There are many challenges to overcome in order to be successful in the manufacturing test of application-specific integrated circuits (ASICs). An ASIC manufacturer deals with many different designs, and the volumes of individual designs are often quite low, especially when compared with high-profile chips such as microprocessors. The customers are varied and use a variety of design styles and tools, yet tests must be developed quickly to meet the time-to-market requirements.

High quality is a requirement for all chips, because the cost of allowing a defective chip to escape manufacturing test is very high. ASICs today may contain more than one million gates, and a single manufacturing defect could cause an entire system to fail. When combining one or more ASICs with other components in a system, it can be very difficult to detect and locate the bad component. Therefore, the most economical time to find defective chips is during chip manufacturing test.

Since there are so many different ASIC designs being processed at any one time, it is not economically feasible to assign a large engineering staff to create the tests for each ASIC. It is imperative that the test patterns be generated automatically. The test patterns must obtain a very high level of fault and defect coverage and should have a very high probability of success when applied at the tester. It is also important to have automated diagnostic tools to help quickly determine the cause of any tester fails. To achieve these goals, it is necessary for certain design-for-test (DFT) features to be built into the chip design.

0018-8646/96/$5.00 © 1996 IBM

Chip designers need to be able to easily incorporate the DFT features that are part of the IBM design methodology. Support is available from an IBM design center, but the tools and automated process must be used by the customers directly or there will be a large impact to the designers' productivity. The tools must work together, support a natural flow through the design methodology, and support the requirements within the design methodology for DFT and sign-off functions.

For many years IBM (internal use only) test automation tools were provided for and executed on IBM mainframe systems. More recently, a UNIX®-based set of test design automation tools was developed that implements many new features not available on the older system. The new system, called TestBench™, is the basis for the test-structure verification, test-pattern generation, and fault-grading functions described in this paper. TestBench has been commercially available since early 1994.

More recently, the integration of the collection of tools used for DFT insertion and ASIC sign-off has been undergoing significant changes. Starting from totally separate tools and a methodology that required considerable interaction with an IBM design center (for any ASIC designer not fully familiar with the IBM ASIC test methodology), the tools have now been placed in the hands of the ASIC designer, with software to guide the designer through the methodology.

In the IBM Microelectronics Division, we believe our test methodology for ASICs meets all of the various challenges. The methodology is based on level-sensitive scan design (LSSD) [1], plus boundary scan and special test controls; software aids are used to insert LSSD and other DFT features into the design, to check the design, and to generate and fault-grade test patterns. This test methodology, combined into a user package for an entire sign-off process, makes the tools, technology, and design flow easier for the designers to use and understand.

The three main sections of this paper follow, with the first describing the different kinds of testing done on IBM ASICs and summarizing the test flow. The next section describes the sign-off process used for IBM ASIC test and how we make it work. The third section describes how TestBench is used within our ASIC test methodology.

ASIC chip testing techniques used within IBM

IBM Microelectronics' ASIC test methodology has been developed to ensure economical, high-quality testing for high-density, high-performance ASICs. A range of tests, described below, are applied within this methodology.

- *Reduced-pin-count testing*

Our ASIC test strategy is based on using LSSD boundary scan to permit the use of relatively low-cost testers containing fewer full-function pin channels (ac pins) than the number of signal I/O (input/output) pins on the ASIC devices being tested [2]. This allows us to use older 256-pin testers and our internally developed 64-pin tester [3] to test ASICs with a variety of signal counts, up to 2000. We call the method "reduced-pin-count testing"; it can save substantially on tester costs, since the cost of logic testers is nearly directly proportional to the number of tester pins, and ever-increasing signal counts can lead to underutilization of existing investments in older testers.

Traditionally, boundary-scan capability is added to a chip to aid in testing modules or boards at higher levels of packaging. We take advantage of the boundary-scan design approach to allow testing of the internal circuitry of the chip using the boundary-scan latches to supply or receive values for chip I/O pins that are not contacted by the tester. All logic enclosed by the scan boundary can be tested using only the relatively small subset of signal I/O pins (no more than 64) required to perform LSSD clocking and scanning operations. These LSSD test-function I/Os are constrained to occupy only a fixed subset of physical chip I/O locations, thus permitting a standard probe set to be used.

At final test of the packaged ASICs, we test all signal I/Os either by the use of cheap parametric tester channels or by using the limited ac channels contacted in groups or banks. The boundary-scan latches are used to supply and observe values in an external configuration. Our test-generation software supports generation of the appropriate patterns that make use of the boundary latches and, when required, allow the patterns to be applied to banks of I/O pins.

IBM's reduced-pin-count testing requires LSSD boundary scan; IEEE 1149.1 boundary scan is not required. LSSD boundary-scan requirements are less rigorous and do not require as much circuit area as 1149.1 boundary scan. LSSD-compatible versions of 1149.1 boundary scan can also serve as LSSD boundary scan. Our ASIC customers can choose how best to implement LSSD boundary scan on their designs. Checking tools ensure that the design is sufficient for reduced-pin-count testing.

- *Weighted random-pattern (WRP) tests*

IBM has pioneered the approach of applying weighted random patterns to chips as an effective way to improve quality and reduce test data volume at the same time [4]. Pseudorandom pattern generators (PRPGs) are incorporated into tester hardware to produce a variable distribution of logical 1s and 0s for each test-pattern input bit. This method selectively biases, or weights, the test-pattern inputs to a greater probability of 0 or 1, as needed. Each scannable latch and each chip input receives its own weight. By applying patterns with a variety of weights, high test coverages can be achieved (as high as with stored patterns), since random-pattern-resistant

faults can be tested in a reasonable number of patterns. Furthermore, WRP testing also involves compressing the outputs into signature registers. The test data then consist of weight sets and signatures.

This approach has been very successful [3, 5]. IBM designed a tester which incorporates the weighting and compressing hardware, and that tester is one of the primary testers used by IBM Microelectronics for ASICs.

One potential drawback to WRP testing is that it can result in far more patterns being applied than with stored-pattern tests. While this may be good for detecting unmodeled defects, it adds considerably to the test time. To address this exposure, WRP, as applied by IBM testers, allows the tester to skip over WRP tests that do not detect any new faults, resulting in test vector counts and test application times very near to the stored-pattern approach. To allow for this skipping of WRP tests, the tester has a "shadow" register for each PRPG that is connected to each pin. Each cycle of a WRP test includes the following sequence of operations:

1. Restore PRPGs from shadow registers.
2. Cycle PRPGs once.
3. Save PRPGs in shadow registers.

To skip a cycle, the second step is repeated for each cycle desired to be skipped. By skipping over the PRPG states that result in unproductive tests, we avoid the time associated with loading those tests into the shift registers—by far the main factor in tester time.

The weights for programming the tester are generated by basically the same automatic test-pattern generation (ATPG) engine that is used for creating stored-pattern tests. Test patterns for many individual faults are combined into a "weight set" that should generate patterns to detect those faults and (it is hoped) many other faults as well. The weight generation scheme employs many heuristics to attempt to minimize both the number of weight sets generated and the number of WRP cycles required from all of the weight sets [6].

Because of the signature collection used for WRP testing, circuits that cannot be prevented from generating unpredictable responses are not compatible with WRP testing. The following are examples of circuits that can produce unpredictable responses:

- Uninitialized RAMs.
- Unterminated nets in a high-impedance state (Z).
- Outputs from "black-box" macros.

WRP tests can be either static or delay tests. Delay testing of most IBM ASICs is done with WRP tests. However, most IBM ASICs are tested in a static mode, since such tests are simpler to generate and apply.

- *Embedded memory tests*

A majority of all very large-scale integrated (VLSI) chips produced within the last few years have contained one or more embedded memories: random-access memory (RAM) or read-only memory (ROM). RAMs and ROMs require a significant number of patterns to fully test these devices, and these patterns are generally provided by the memory designer. In previous generations of IBM ASICs, use of a RAM or ROM required that it be possible to establish one-to-one correspondence between chip I/O pins and the memory I/O pins; test patterns for the memory were then mapped out to the chip I/Os. Reduced-pin-count testing runs counter to this test method, since many embedded memories have more than 64 pins. Also, the logical and physical design effort required to provide the correspondence for all memory I/O pins is considerable.

The DFT method used on current IBM ASICs is to provide a self-test engine in the RAM or ROM library element. This self-test is operated by setting an initialization state and then pulsing one or more clocks repeatedly. The patterns necessary to test the memory are generated by the self-test engine, and the results are stored in latches either as a pass/fail bit or a signature. The latches are then scanned out and the results observed. Provisions are also made for diagnosing defects once they have been detected. This scheme provides a fast and thorough test of the memory that uses very few signal I/Os. The self-test engine is parameterized and compiled along with the memory into the desired size and configuration.

For very small RAMs, the size of the self-test engine can be larger than for the memory. To avoid this problem, small RAMs can be compiled into an array of scannable latches which are then tested along with the rest of the scan logic. The scannable RAMs can also be configured with multiple read and write ports.

Embedded memory tests can be either static or delay tests; some self-test engines are run at or close to the system clock speed.

- *I/O wrap*

IBM Microelectronics 0.5-μm and smaller ASICs use the reduced-pin-count test concept to allow wafer testing of the die by probing only a small subset (64) of the signal I/Os. In order to be able to apply some tests to all of the I/Os, the noncontacted signal I/Os are made into bidirectional pins if they are not already functionally bidirectional. Then the boundary-scan external configuration can drive a value through the output buffer, and the input buffer can "see" it and latch it into a boundary-scan latch. This tests the chip inputs and outputs except for checking whether the drivers and receivers are actually connected to the I/O pad and whether they can

drive and receive valid voltage levels and current-carrying capabilities (these are tested later).

• *Input and output tests*

Chip input buffers must be tested to ensure that they operate correctly for their specified voltage range. Thus, each input receiver must be tested to ensure that it can distinguish between the lowest voltage that should be interpreted as a logical "1," and the highest voltage that should be interpreted as a logical "0." These tests are called receiver tests.

Similarly, chip output buffers must be tested to ensure that they can drive or drain current with various capacitive loads applied while maintaining the requisite voltage levels. Also, three-state output buffers must be tested to verify that they can achieve a high-impedance state. These tests are called driver tests; they ensure that the output buffer meets its voltage specifications.

The driver and receiver tests are automatically generated via ATPG. The IBM test-pattern-generation software has a special fault definition used to describe driver and receiver faults so that ATPG can create meaningful tests for the inputs and outputs. A driver fault, for example, must be observed at the primary output (or bidirectional) pin to which it is attached; it is not allowed to be observed at an internal scan latch or at any other I/O pin. The receiver faults force the driving I/O pin to be used to excite the fault (requiring that any bidirectional drivers be inhibited). Receiver faults are allowed to be detected at any valid observation point; the boundary-scan latch associated with the I/O is typically used. If there are multiple drivers attached to the same I/O net, each driver has its own set of faults associated with it to ensure that each driver is fully tested for drive strength. Similarly, if there are multiple receivers attached to an I/O net, each such receiver has faults which must be tested independently of the other receivers. The driver and receiver tests are static stored-pattern tests only.

• *Stored-pattern ATPG tests*

Stored-pattern tests are used for a number of different applications in IBM ASIC testing. Chip internal testing can be done with stored patterns or with WRP. Stored patterns can be applied on any logic tester, unlike WRP, thus allowing some testing to use older existing testers. Stored patterns can be used to test chips that do not conform to the boundary-scan signature-based testing requirements of the WRP testers. Also, stored patterns are used for the driver and receiver tests. Stored-pattern tests can be either static or delay tests, though they are mostly used as static tests.

By the use of full-scan LSSD in the chip designs, ATPG can achieve a very high (99.5%+) stuck-fault test coverage. In addition, stored-pattern tests produced by the TestBench ATPG system are very tightly compacted so that more tests can fit into the tester buffers. However, the pattern count required for high coverage on especially large (e.g., 700000 gates) chips or chips that are difficult to test can exceed even large stored-pattern tester buffers. When the generated test patterns exceed the capacity of the tester despite considerable effort to compact them, a decision must be made as to whether to use multiple tester buffer loads (at substantial additional expense), or to truncate the pattern set so that it will fit on the tester. TestBench provides a means to sort the test vectors to place the less effective vectors at the end of the set. This makes it easy for manufacturing to decide whether to truncate and where, since each group of test patterns includes an indication of the test coverage obtained after applying all patterns up to that point in the pattern set.

• I_{DDQ} *(quiescent current test)*

Checking for current leakage can be executed to very low current specifications on static CMOS circuits. It is necessary to inhibit any current-drawing paths while I_{DDQ} test measurements are being taken. IBM has been doing limited (fewer than ten patterns) I_{DDQ} testing for many years to screen out chips that use excessive current even though they may not exhibit other symptoms of being defective. Usually the I_{DDQ} vectors are generated algorithmically, using the LSSD shift registers and chip inputs to condition the circuits to a few different states. The pattern generation is done by manufacturing software, which uses information about the LSSD operation of the chip. The signals that are needed to inhibit current draw (as can happen with resistive pull-ups or differential receivers) are specially identified so that their states will be applied correctly. Part of the chip design and checking software ensures that these special signals are properly connected inside the chip.

For the future, automatically generated I_{DDQ} tests are being considered. To keep manufacturing test costs low, only a few such patterns can be applied, since the time required to let the chip settle into a quiescent state is relatively long. However, many studies [7–9] have shown that a small number of patterns can achieve a very high defect coverage level.

• *Burn-in*

Because some types of defects cause semiconductor chips to fail early in their expected life, IBM ASIC customers are offered the choice of a high reliability grade. To weed out these defects, the high-reliability chips are generally "burned in" (patterns are run through the chip at an elevated voltage and temperature). This simulates the early life of the chip and brings out defects that can shorten chip life. By applying test patterns that achieve high stuck-fault coverage in burn-in conditions, a very high

percentage of nodes in the circuit are forced to both logic 0 and logic 1. By observing the outputs during burn-in (known as *in situ* burn-in), soft defects that occur only at higher temperatures can also be detected. The standard burn-in procedure for IBM ASICs is to use weighted random patterns and embedded-memory self-test at elevated voltage and temperature conditions. By using weighted random patterns, we feel that most logic in the chip will have significant amounts of switching activity. A number of packaged chips are mounted on a burn-in board and the outputs of one chip are monitored, rotating through all the chips for the duration of burn-in. The chips then undergo a final test at normal conditions after burn-in.

- *Delay test*

Delay testing is not actually a separate test in IBM. The WRP or stored-pattern (logic or embedded-memory) tests can be generated in such a manner as to target delay defects [10]. Software is also available to use the wiring delay data (e.g., from a standard delay file) to generate pin timings for the tests under the constraints of the tester timing specifications. Occasionally, the tests are run to tighter timing constraints than the functional operation would require.

The test software can usually obtain 90–98% coverage of transition faults in addition to providing pin timings. Our experience has shown that applying delay tests with tight timing constraints can be very beneficial for improving the perceived quality of the chips. However, there is definitely an additional cost associated with creating and using delay tests:

- There are more tests, so the tester buffer capacity may be exceeded.
- The generated pin timing data sometimes have to be modified at the tester because the delay data were not accurate enough.

Delay test is an optional test feature, available to our ASIC customers.

- *Logic built-in self-test*

Logic BIST (LBIST) is not currently a prime ASIC chip testing technique in IBM, although it is used more entensively in processor chips. Weighted random patterns can achieve better fault coverage in much less tester time than LBIST takes. However, LBIST support elements are available in the technology cell libraries to allow designing an LBIST feature into a chip. Test tools can be used to compute expected signatures and fault coverage for the chip [11]. Running LBIST tests as an additional chip test can be accommodated for an additional charge; however, LBIST alone cannot attain the fault coverage usually needed to ensure a high quality for the chips that pass.

- *ASIC test methodology summary*

As described above, several different types of tests are applied to ASIC chips, some of them during wafer test. Since wafer test on the newer IBM ASIC technologies is done by probing only 64 signal I/Os, not all tests can be applied at this point. The wafer tests are used as a screen, so that the chips put into module packages have a high probability of being good. The final package test must provide the highest practical coverage of the potential chip defects.

The IBM ASIC chips are tested on stored-pattern testers at the wafer level, using only 64 or fewer signal I/Os. The tests applied are

1. I_{DDQ} tests to check both for leakage due to process problems and for random defects that can cause high static current.
2. Stored-pattern internal tests to test the bulk of the logic internal to the boundary-scan latches. These tests are typically truncated to achieve 70–90% fault coverage to reduce wafer test time or, for very large designs, to allow fitting the tests into the tester buffers. Full internal test is completed after the chips are packaged.
3. RAM and ROM BIST to test any embedded RAMs and ROMs.
4. I/O wrap patterns to test the basic function of the off-chip input and output buffers (receivers and drivers).

The chips that pass these tests are diced and mounted on module packages, and are then tested again. Most ASIC chips are tested on the IBM WRP tester, though some package types are tested on stored-pattern testers. The tests applied at the WRP tester are

1. I_{DDQ} tests.
2. WRP internal tests to test the bulk of the logic internal to the boundary-scan latches (to a high stuck-fault coverage).
3. RAM and ROM BIST to test any embedded RAMs and ROMs.
4. Driver and receiver tests to fully test the chip input and output buffers.

If the package is tested on a stored-pattern tester, stored-pattern internal tests are substituted for the WRP internal tests. If the chips are to undergo burn-in, it is done on the packaged modules. Full module test is repeated after burn-in. If the chips are to be delay-tested, this is done at the package test as part of the internal logic test.

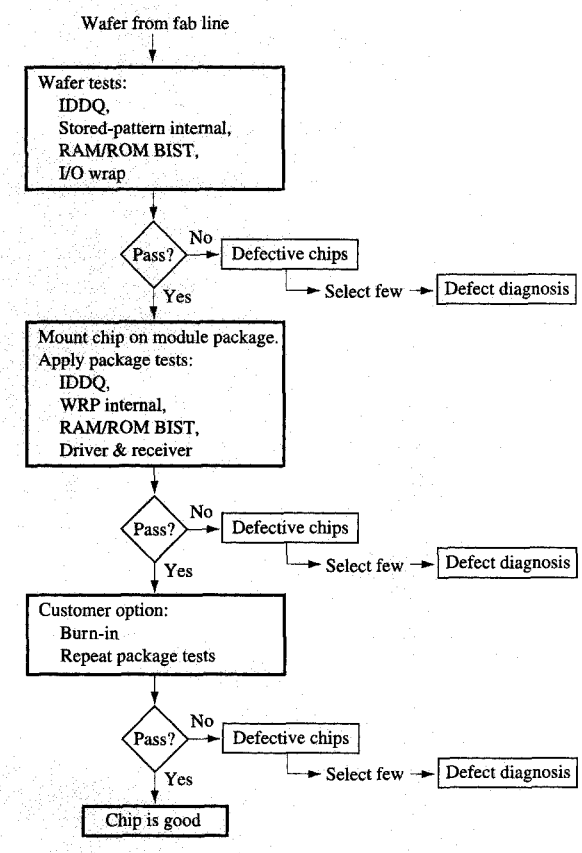

Figure 1

Flow of a typical ASIC through manufacturing test. Diagnosis is performed in low-yield or common-fail conditions.

There is no standard flow for applying logic BIST to an ASIC; it could be applied at either wafer test or package test. If applied at package test, it must be supplemented by other logic patterns (WRP or stored patterns) to attain the requisite high fault coverage.

A typical ASIC test flow during manufacturing testing is depicted in **Figure 1**.

IBM ASIC sign-off process

- *Need for a sign-off process*

One way to help ensure quality in a design is to enforce the use of a design methodology which has been successful. For the IBM ASIC business, this is an LSSD methodology. Furthermore, it is a methodology utilizing static timing analysis and test structure verification as sign-offs for design transfer from customer to manufacturing.

Many customers of IBM ASICs are not familiar with LSSD. Success in this type of customer environment relies upon four elements:

1. Documentation of the methodology.
2. A user framework to support and audit the methodology.
3. Integrated tools to support the methodology.
4. Methodology experts to assist customers in the process.

The better integrated these four elements are, the easier the design process will be for the customers, the faster the design and manufacture cycle will go, and the greater the number of designs that can be processed.

- *History of the IBM ASIC sign-off process*

The beginning

IBM has been moving to an environment that incrementally integrates and automates methodology documentation, framework, tools, and expert support. Coming from a history of internal supply and demand where the technology and methodology were well understood, IBM has had to learn how to make these elements available, understandable, and of increasing business advantage to customers less familiar with them. IBM ASICs initially supported a series of point tools strung together with the strong support of design centers (**Figure 2**).

As Figure 2 shows, the initial support provided to customers of IBM ASICs incorporated both IBM tools (boxes with "I") and non-IBM tools. The IBM-supplied tools were those created by the IBM design center (ClockPro for clock planning) and by the IBM Electronic Design Automation group [TestBench Test Structure Verification (TSV) for test validation, EinsTimer™ for timing analysis, CMOS Checks for technology checking, and BooleDozer™ for test insertion, clock synthesis, and chip finishing]. While there were some variations to this flow, this was the most common approach.

The use of these tools occurred in two phases (with loops within each phase). The first phase, completed by the ASIC designers, ensured that the design satisfied test verification, static timing analysis, and technology usage requirements necessary to hand off the design to the IBM design center. Once the design had been verified with the TSV, EinsTimer, and CMOS Checks tools, it moved to the second phase, which took place at the IBM design center. Here, the netlist entered into an exclusive IBM design tool methodology. These tools did additional test insertion and mapping, clock tree construction, and chip finishing. Again, test verification, timing analysis, and technology checking had to be run to ensure that the design was manufacturable.

This design flow relied heavily on two things: proactive participation of the IBM design center and reprocessing of designs within the IBM design center. The participation of the design center was considerable

because of the need to educate customers on design styles, tools, technologies, and methodologies. The design center had to tie the methodology to the tools, since the tools themselves were not directly tied to the methodology. The reprocessing of the designs came about for the same reasons: The design center had to ensure that everything was done and run correctly because the methodology and tools were new to the customers and unconnected to the specific methodology.

The process resulted in many successful designs. However, customers felt frustrated that they did not have access to the best tools for the job. There was also a lot of pressure, and some natural limits, on what the IBM design center could do and the number of customers and designs that could be handled at one time.

The next step

The first incremental improvement did little to strengthen the links between the methodology and tools or the ability to audit the use of the tools; however, it did immediately improve the customers' interaction with the tools. BooleDozer, a tool which had been used only in the design center, was made available for direct use by the ASIC designers. This enabled customers to understand and use the same tool used by the design center for IBM-compliant test insertion and design modification. BooleDozer's programmable interface, which was a great advantage to the design center, was now available to IBM ASIC customers. They could now synthesize, change, and tailor their designs much faster and in the same way as the IBM design center.

The additional advantage this provided was that the design center could ship to the customers specific synthesis transforms which were the same ones that they themselves used. BooleDozer was heavily requested by the IBM ASIC customers, and the design center was eager to be able to supply it, since it meant that both customer and design center would be using the same tool on the same design. With this, designs flowed more easily from customer to design center, fixes went more easily from design center to customer, the design-to-manufacture cycle was reduced, and the design center throughput was increased, all resulting in increased customer satisfaction.

A need for function

Getting BooleDozer into the hands of customers helped considerably, but more function was needed. Many unique synthesis transforms were being done by the design center on a customer-by-customer basis. This greatly increased the time it took to process a large number of customers, because the transforms were so specific to each individual customer that reuse of the transforms for additional customers was too difficult. A generalized package of synthesis and test insertion functions was

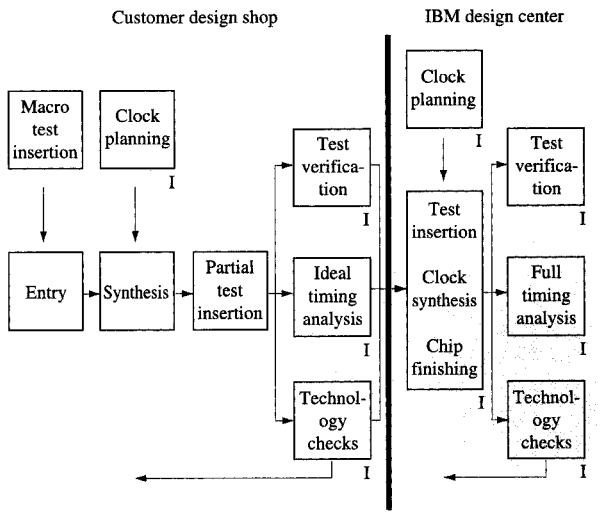

Figure 2

IBM ASICs initial handling of customers' designs.

needed to handle generic cases so that customers could develop to those standards. The ability to customize would still be required, because one size does not fit all.

The design center produced a list of requirements for design-for-test synthesis (DFTS) based on past customer experiences. That list was used to evaluate what a total customer/design center solution would be. Even after supplying BooleDozer to customers, and once the DFTS needs were filled, the design center would still have to tie together the tools and the methodology in order to improve customer understanding and throughput.

The DFTS requirements were addressed by a group within the IBM EDA organization which was already working with the IBM design center. As it turned out, IBM EDA already had many of the DFTS functions completed and many more already in-plan for near-term completion. IBM EDA had already been developing and assisting customers with a DFTS package but had not linked up with ASICs. Once the connection was made, collaboration on a total solution began.

- *Making the methodology work*

ASIC sign-off kit formulation

The need to present a cohesive collection of tools tied to a methodology became a major work item for IBM ASICs and IBM EDA. Once it was clear that the IBM DFTS product [12] could supply all of the DFT synthesis functions required by the IBM ASIC design center, the ASIC sign-off kit (ASOK) was conceived.

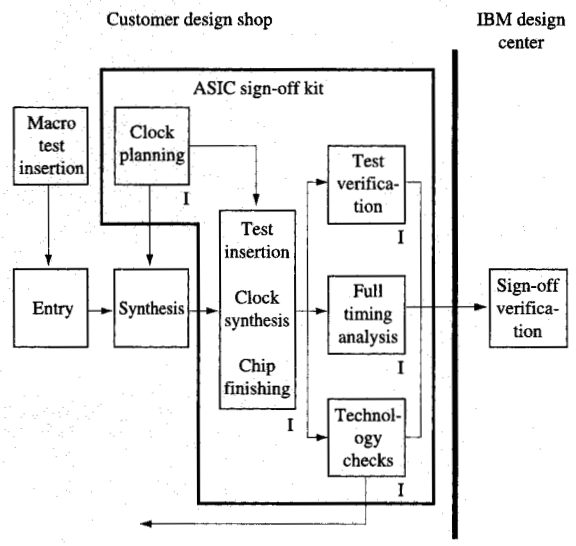

Figure 3

IBM ASIC sign-off kit (ASOK).

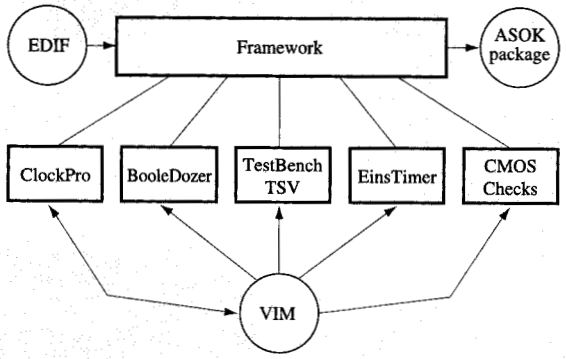

Figure 4

IBM ASOK conceptual architecture.

The ASIC design center knew the goal it wanted to achieve: moving the bulk of the actual design work to those who knew the design the best—the customers (see **Figure 3**). This would dramatically reduce the amount of "retooling" done by the design center. "Retooling" is both the rerunning of the same tools the customer has already run (Figure 2) and the amount of repetitive tool customization done for each customer.

Moving the design work to the customer required moving the methodology and tools to the customer. This triggered many actions:

- The tools would have to be more understandable, since much more of the running of the tools would be done by the customers.
- The methodology would have to be well documented, and preferably used to drive the tools, so that the customers could not easily go astray with a set of tools that could handle many different types of design styles and options.
- The design would have to be auditable to ensure that the customers used all of the correct tool options and followed the methodology in the prescribed manner.
- More documentation on the methodology and the fit of the tools within that methodology would be needed, since this would be completely new to most customers.

The concept behind the ASOK would have to be something that could tie tools to methodology and make those tools look more cohesive, as well as something that would be auditable so that the design center's rerunning of the tools could be minimized. The conceptual architecture, as shown in **Figure 4**, became the driving force behind how IBM ASIC would go from what it had (a strong but overworked design center staff and unconnected point tools) to what it needed (a design center that acted as consultants, and tools that were linked and methodology-based).

In order to address these needs and to assist customers quickly, a multistep approach was adopted. This allowed function to be rolled out to the design center and to the customer incrementally rather than waiting for the ultimate package to be complete. The steps were defined as follows:

- *Step 1: Cosmetic cohesiveness (base integration)* This is simply the ability to launch, from a single consistent interface, all of the point tools supplied to customers in the tool kit.
- *Step 2: Partial integration* This adds a documented methodology linked to the single tool interface which could be used to assist the customer in selecting which tool to run next and how to use it.
- *Step 3: Euphoric integration (final integration)* This step gives ASOK the ability to control the launching of the tools and provides the ability to audit the running of the tools and methodology.

ASOK Step 1

The conceptual view of Step 1 is represented in **Figure 5**. The intent of this step is to quickly assemble a system that loosely couples the point tools and forms the basis of a

larger integrated solution. The output of Step 1 is for use only in the IBM ASIC design center, so that users can refine their detailed requirements for a more cohesive total solution.

The reality that came from the conceptual view is represented in **Figure 6**. As the figure shows, the resulting ASOK menu contains many parameter specification areas and submenus. These are the direct result of the input from the design center. In fact, as with many applications, further changes are expected as more users are exposed to it.

An important piece of this menu was the "Methodologies" section, which is designed into the Step 1 menu as documentation for the methodology to follow. In Step 2, this becomes the section which helps users to understand what tools to run when and how.

ASOK Step 2

The "partial integration" step was the first step intended for use by customers of IBM ASICs. It first linked a documented methodology with the tools to be used in each step of that methodology. The intent was to have a single launching point for the tools (the Step 1 menu) and an on-line methodology that could be used to teach the user the order in which to use them. Additionally, the methodology, actually an outline of steps to run, contains information on how to run the tools.

Many of the tools in the ASOK have multiple steps to run once the user is within the tool. The methodology is intended to be used as an on-line guide both for getting into the tool and for using the tool once inside. The user can read the methodology to understand which tool to launch (from the "Tools" pull-down). The user can also have the methodology window up, once the tool is started, to explain what to do within the tool. This allows the user to read the methodology explanation of steps, set the appropriate options, and run the tools.

Although there is no way in the ASOK Step 2 implementation to ensure that the users run the tools in the order they are supposed to, or in the way they are supposed to, Step 2 makes the link between point tools and design methodology.

- *Future improvements*

ASOK Step 3

Step 3, euphoric integration, while not yet available, is the next logical step in the ASOK solution. It will be a strong linking of the methodology to the controlling of the processing. Instead of just displaying information on what tools to run when and how, the on-line methodology guide will enable the user to launch tools directly without subsequent pull-down menus.

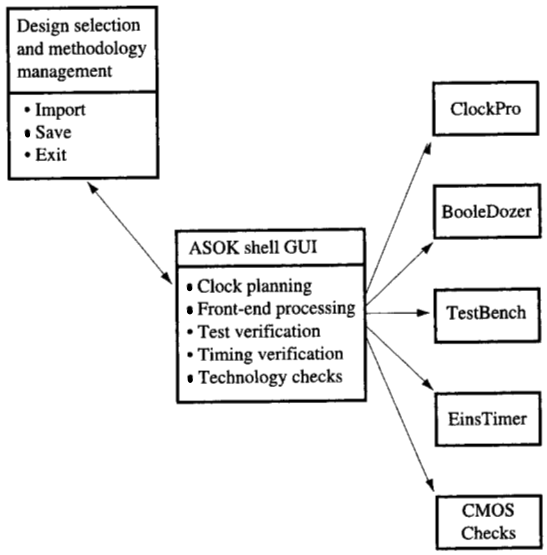

Figure 5

IBM ASOK conceptual user view.

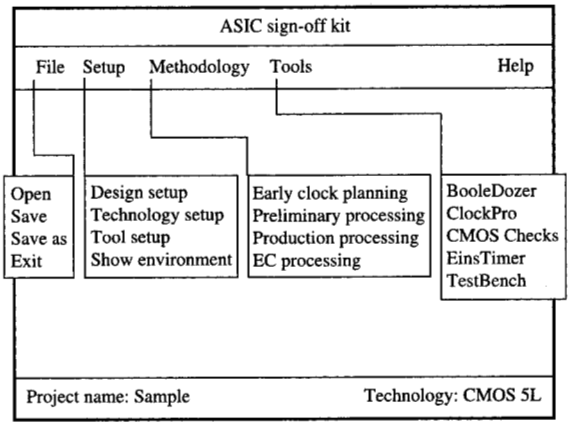

Figure 6

IBM ASOK main user screen.

In Step 3 the methodology will be active; that is, the user will be able to click on action lines in the methodology and launch tools. The user will be able to click on documentation lines and obtain extended help documentation. The user will also be able to see what steps in the methodology have been completed, are

available to be run, and cannot yet be run at this time. This will be done primarily through automatic methodology control. The user will also be able to restart at a particular step, resetting the methodology to that point. In short, the methodology will be the control and the audit for ensuring that the tools are run in the proper order.

Step 3 will still not be able to run tasks within a tool, but it will be able to initiate the tool. The user will no longer have to read what to do and then take a separate action to start the tool (e.g., move to the "Tools" pull-down to start the tool). The flexibility will also still exist to deviate from the methodology at any time and launch tools from the "Tools" pull-down. This will permit a user to pursue alternative processing solutions to specific customer needs.

Test tool requirements

Because of the wide range of test techniques that must be supported, the test automation tools must support a broad range of test types. The tools must keep track of the different test types and understand which manufacturing tester can handle each type of test. The test patterns for each type of test must be clearly identified and kept separate. The test coverage for each type of test must be accounted for and accumulated with coverage for other tests where appropriate.

It is also important to conceal as much of the complexity as possible from the logic designers. Much of the complexity can be hidden by using DFT synthesis tools, such as the DFT insertion products available from IBM EDA. This can quickly convert a nonscan design to full (or partial) scan, and can insert IEEE 1149.1 boundary scan [13] and/or IBM boundary-scan structures into a chip design. This can remove much of the manual effort from the DFT for a chip and can shorten a design cycle by many days.

- *Ensuring high-quality fault models*

In order to be confident of the quality of chips shipped from the factory, IBM manufacturing strongly encourages the use of test vectors that achieve a total of at least 99.5% static-fault coverage. The fault model currently used for static faults is a gate-level (pin-fault) model. Although many companies choose to fault only the pins on the boundary of the cell library members, IBM feels strongly that by assigning faults to the gates inside each cell library member, the generated patterns will target a higher percentage of potential defects. In addition, since some of the defects to which certain technology cells may be susceptible are not well modeled by the pin stuck-at fault model, these technology cells have faults added to them to ensure that the ATPG tools will generate patterns to excite the otherwise unmodeled defects. The cell libraries take advantage of the TestBench product to add (user- or technology-specified) "pattern" faults to the fault model [14].

A pattern fault is simply a fault that is defined by the pattern(s) that are required to excite the fault and how and where the effect of the fault first appears. For example, a multiplexor (MUX) cell may be susceptible to a particular pattern that is not guaranteed to be created by a test generator targeting pin stuck-at faults. By adding pattern faults to the MUX definition, it is possible to force a test generator to create the patterns for which the MUX is susceptible. In the extreme, it is also possible to add pattern faults to a cell definition that would force an exhaustive set of patterns to be generated for the cell. TestBench, by default, uses pattern faults to model defects for certain primitive functions (LATCH, XOR) that could easily contain defects not well modeled by pin stuck-at faults.

Many studies have shown that high stuck-fault coverage does not necessarily imply high coverage of potential defects [15]. Thus, being able to target known defect mechanisms that do not manifest themselves as pin stuck-at faults is essential to ensuring that the generated test vectors are the best possible.

TestBench computes separate test coverages for the following different classes of fault models:

- Primary input faults.
- Primary output faults.
- Driver faults.
- Receiver faults.
- Static faults.
- Dynamic faults.
- I_{DDQ} faults.

- *Test modes*

The TestBench product permits many different (test) modes of operation to be defined for the circuit being processed. Each test mode represents a particular configuration or setup in which to apply patterns to the circuit. When a test mode is defined, the pins that perform some type of test function (clocks, scan pins, etc.) are identified to the tool, usually via properties on the I/O pins. Other information is also supplied to indicate the target tester (and its capabilities), the type of tests desired, and the type of faults to be targeted by this test mode.

A test mode may be defined for boundary-scan internal or external testing. Test modes can have "fixed-value" latches that are set to a specific state to enable the particular mode of operation for the circuit. This avoids having to make all mode-select signals primary input pins and works in a manner similar to the IEEE 1149.1 instruction register.

Once a test mode has been defined by the user, the tool attempts to discover the test structures that are enabled in that mode. Once the test structures have been understood, these structures (shift registers, for example) can be listed by the user to verify that the logic is configured correctly. Any errors in the test structures can be analyzed interactively, with the TestBench circuit tracing and analysis features.

The various test types described in the section on test methodologies can be handled by using several different test modes. Manufacturing test can then pick from among the different test modes for the test patterns to apply at each stage of product testing. This allows a manufacturing location to pick different styles of test data (e.g., WRP and stored pattern) for different stages of test, such as die test and packaged chip test. The test data and fault detection credit accounting must be tracked very carefully. If three test modes are used for final test of a packaged chip, the total fault coverage for these modes (and not others) must be accurately accumulated. TestBench supports keeping track of different groupings of test modes in order to compute a total fault coverage for the group and to allow faults detected in one test mode to be either retargeted or considered already detected in a different test mode.

- *Test data types*

TestBench currently supports the following types of test data:

- Stored-pattern
  - Static or dynamic logic tests.
  - Static or dynamic embedded macro tests.
  - Shift register flush and scan tests.
  - Parametric tests.
  - I/O wrap tests.
  - Interconnect tests (for multichip modules or boards).
- Signature-based
  - Static or dynamic WRP logic tests.
  - Static or dynamic logic BIST.
  - Static or dynamic embedded macro BIST.

- *Support for low-pin-count testers*

As described in the section on test methodologies, IBM ASICs are tested on testers that usually have fewer full-function logic test pins than the product has signal I/O pins. To support this concept, TestBench is able to generate patterns using only the test control pins and boundary-scan latches. TestBench is also able to generate patterns to test the full set of chip I/O, using the boundary-scan latches. These will be separate test modes.

The tester often has a large number (512) of slow parametric units that can be used to test the chip external boundary logic. For those cases in which the number of chip pins is greater than the number of parametric units available on the tester, a bank switching of parametric units is assumed. Bank switching implies that not all pins can be contacted at the same time, which imposes additional restrictions on the test-generation software to ensure that all pins being stimulated or measured during a test are contacted by the tester for the duration of the test. These options are fully supported by TestBench.

- *Diagnostics*

Part of the lower costs associated with the highly structured DFT approach taken by IBM ASICs is achieved by having highly automated and accurate diagnostics to call out the net in the chip most likely to be the cause of the failure(s). Scan-based diagnostics work extremely well, except for those chips that fail so completely that even the shift registers do not function correctly. The WRP tests can be diagnosed by logging the scan and output data for a specific failing signature interval (typically 256 cycles) and using software simulation to compute the expected data to find the miscomparing output pins and latches [16]. Alternatively, the WRP (and logic BIST) cycles can be expanded into a set of equivalent stored-pattern vectors, allowing normal stored-pattern diagnostic approaches to be used.

Many years ago, IBM manufacturing used a fault dictionary approach to diagnostics. This worked reasonably well with small circuits, but became unreliable and impractical with large, dense chips. Now we use software simulation to perform diagnostics against the tester fail data.

- *Sign-off for test vectors*

The IBM ASIC development group goes through a rigorous qualification cycle for each new technology cell library to ensure that the test tool (TestBench) simulator(s) predict correct values and the test generators generate tests for all faults being modeled. While edge-triggered scan and LSSD both may achieve a desired high test coverage by making latches scannable, the level-sensitive aspect of LSSD allows the whole question of accurate delay models to be avoided as long as the DFT guidelines are followed. The TestBench product provides a comprehensive set of audits to allow the manufacturing site that receives the test data to determine what types of DFT violations exist for the circuit. Resimulation of the test patterns by manufacturing using a "golden simulator" is not required and, indeed, is not performed. If there are no audit failures, manufacturing can be very confident that the test vectors will work the first time. If there are any audit failures, the test data may be suspect, and engineering resource may be needed to investigate zero-yield conditions; although in most cases the test data will

Table 1 Data for some recently processed ASIC chips.

Chip name	*No. of gates*	*No. of latches*	*No. of RAMs*	*No. of faults*	*No. of test modes*	*No. of vectors*	*Fault coverage (%)*
IBM1	650,000	65,000	19	1,500,000	5	8900	99.85
IBM2	140,000	23,800	60	490,000	7	5600	99.75
IBM3	150,000	25,200	4	661,000	7	5841	99.59
IBM4	100,000	16,300	2	455,500	6	4400	99.90
extern1	330,000	54,200	0	1,475,000	5	3274	99.64
extern2	640,000	49,500	38	1,950,000	4	4803	99.89

simply be pessimistic (by assuming unknown responses), resulting in lower fault coverage. Audit check failures usually require an explanation from the customer.

The audited information includes

- Cell library version used.
- Modeling errors discovered during circuit import.
- DFT guideline violations.
- Fault coverage.
- Test types (to ensure that all required types are included).
- Test data audits (e.g., three-state conflicts exist in test data).

Currently, most external customers are required only to verify that their circuit does not violate the DFT guidelines. IBM generates the test data automatically once the design netlist is delivered for processing. For these customers, the "sign-off" is actually against the logic design, not the test vectors.

Data for a few recently processed chips

The IBM Microelectronics Division manufactures many different ASIC chips during any given year. Recently, external customers have been able to obtain ASICs built by IBM. Many of these are high-performance chips for various functions, such as MPEG2 encoders and decoders, graphics accelerators, and main processors for large systems. In **Table 1,** we provide data for a small sample of some chips that were processed during 1995. The fault coverage reported in the table accounts for only the (static) logic faults and does not include coverage for faults within any embedded memories. The actual fault coverage will be higher since the embedded-memory BIST achieves 100% coverage.

Conclusions and future directions

The test techniques used in testing IBM ASIC chips are very comprehensive. The TestBench tool usually attains close to 100% fault coverage using ATPG when the DFT guidelines are followed. The test vectors have a very high probability of working at the tester (if the audits indicate that no problems were found). IBM manufacturing can quickly check the audit data to determine how confident they can be in the quality of the test patterns. The result is a very high quality level in the ASICs shipped to the customer and a quick bring-up of the manufacturing test.

The ASOK sign-off process minimizes the impact to the customer's design cycle. DFT synthesis products make insertion of the test features relatively simple. Checking tools ensure that all of the manufacturing test requirements have been met. An on-line methodology guide helps the designer run the various tools as appropriate to the methodology. The test sign-off of a manufacturable ASIC is done on the design rather than the test patterns themselves, and this sign-off is a smooth one because the checking tools are first run directly by the ASIC designers. Automatic test-pattern generation can then create test patterns in all of the test modes required by manufacturing.

Future improvements will allow the DFT insertion and checking tools to be run from one common interface as part of ASOK Step 3. This will simplify running the tools and will ensure that the tools are run in the proper order.

The IBM ASIC test methodology has been run successfully on many IBM and non-IBM designs. The combination of test techniques, DFT insertion tools, checking tools, and test-generation tools can make the whole test experience a low impact to the design cycle with only moderate effort from the designers.

Future enhancements may include the use of LBIST for early wafer screen testing, the addition of a limited set of path delay tests, and enhancements to better support embedded "core" macros.

Acknowledgments

The authors acknowledge the contributions of many IBM colleagues in the Burlington and Endicott facilities to the development of our ASIC test methodology, the sign-off process, and the TestBench tool. We would also like to thank L. Milano, B. Deskin, and J. LeBlanc for providing data on some chips.

TestBench, BooleDozer, and EinsTimer are trademarks of International Business Machines Corporation.

UNIX is a registered trademark of UNIX Systems Laboratories, Inc. (now Novell, Inc.).

References

1. E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," *Proceedings of the 14th Design Automation Conference*, 1977, pp. 462–468.
2. R. W. Bassett, M. E. Turner, J. H. Panner, P. S. Gillis, S. F. Oakland, and D. W. Stout, "Boundary-Scan Design Principles for Efficient LSSD ASIC Testing," *IBM J. Res. Develop.* **34**, No. 2/3, 339–354 (1990).
3. R. W. Bassett, B. J. Butkus, S. L. Dingle, M. R. Faucher, P. S. Gillis, J. H. Panner, J. G. Petrovick, and D. L. Wheater, "Low Cost Testing of High Density Logic Components," *Proceedings of the 1989 IEEE International Test Conference*, pp. 550–557.
4. J. A. Waicukauski, E. Lindbloom, E. B. Eichelberger, and O. P. Forlenza, "A Method for Generating Weighted Random Patterns," *IBM J. Res. Develop.* **33**, 149–161 (1989).
5. J. H. Panner, R. P. Abato, R. W. Bassett, K. M. Carrig, P. S. Gillis, D. J. Hathaway, and T. W. Sehr, "A 300K-Circuit ASIC Logic Family CAD System," *IEEE J. Solid-State Circuits* **26**, No. 3, 300–309 (1991).
6. R. Kapur, S. Patil, T. J. Snethen, and T. W. Williams, "Design of an Efficient Weighted Random Pattern Generation System," *Proceedings of the 1994 IEEE International Test Conference*, pp. 491–509.
7. L. K. Horning, J. M. Soden, R. R. Fritzemeier, and C. F. Hawkins, "Measurements of Quiescent Power Supply Current for CMOS ICs in Production Testing," *Proceedings of the 1987 IEEE International Test Conference*, pp. 300–309.
8. R. Perry, "I_{DDQ} Testing in CMOS Digital ASIC's—Putting It All Together," *Proceedings of the 1992 IEEE International Test Conference*, pp. 151–157.
9. H. Ahuja, D. Arriens, B. Schneller, V. Verma, and W. Whitman, "Intel386™ EX Embedded Processor I_{DDQ} Testing," *Proceedings of the 1995 IEEE International Test Conference*, pp. 902–909.
10. B. Könemann, J. Barlow, P. Chang, R. Gabrielson, C. Goertz, B. Keller, K. McCauley, J. Tischer, V. Iyengar, B. Rosen, and T. Williams, "Delay Test: The Next Frontier for LSSD Test Systems," *Proceedings of the 1992 IEEE International Test Conference*, pp. 578–587.
11. B. L. Keller and T. J. Snethen, "Built-In Self-Test Support in the IBM Engineering Design System," *IBM J. Res. Develop.* **34**, 406–415 (1990).
12. IBM Electronic Design Automation, "LogicBench: BooleDozer DFT Synthesis User's Guide, Second Edition," IBM Corporation, Dept. 40J, 1580 Route 52, Hopewell Junction, NY, 1995.
13. IEEE Standard 1149.1, "IEEE Standard Test Access Port and Boundary-Scan Architecture," Institute of Electrical and Electronics Engineers, 1990.
14. IBM Electronic Design Automation, "TestBench: Library Data Reference, Fourth Edition," IBM Corporation, Dept. V33, 1701 North Street, Endicott, NY, 1995, pp. 151–170.
15. S. C. Ma, P. Franco, and E. J. McCluskey, "An Experimental Chip to Evaluate Test Techniques Experiment Results," *Proceedings of the 1995 IEEE International Test Conference*, pp. 663–672.
16. J. A. Waicukauski, V. P. Gupta, and S. T. Patel, "Diagnosis of BIST Failures by PPSFP Simulation," *Proceedings of the 1987 IEEE International Test Conference*, pp. 480–484.

Received November 7, 1995; accepted for publication February 22, 1996

Pamela S. Gillis *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (pgillis@vnet.ibm.com).* Ms. Gillis graduated from the University of California at Los Angeles in 1971 with a B.A. in physics and astronomy. She received an M.S. in physics in 1973 and an M.A. in mathematics in 1974, both from the University of Colorado at Boulder. From 1974 to 1982, she worked at TRW Inc., in McLean, Virginia, doing engineering analysis studies in defense and energy conservation and leading several projects. In 1982, Ms. Gillis joined IBM at Essex Junction. From 1982 to the present, she has worked in the areas of test generation, testability analysis, and ASIC test methodology. She is the leader of an ASIC test methodology team. Ms. Gillis is currently an advisory engineer in the ASIC products organization. She is a member of the Institute of Electrical and Electronics Engineers and a senior member of the Society of Woman Engineers.

Tom S. Guzowski *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (Guzowski at BTV).* Mr. Guzowski graduated from the Indiana University of Pennsylvania in 1977 with a B.S. in physics. In 1978, Mr. Guzowski joined IBM at Essex Junction. From 1978 to 1985, he worked on test-generation methodology and pattern-compaction algorithms for IBM chip designs. In 1985, Mr. Guzowski transferred to the IBM Storage Products Division, Tucson, Arizona, where he developed a hierarchical design and synthesis methodology for disk and tape controller products. In 1992, Mr. Guzowski returned to the Essex Junction laboratory and worked on the creation of a suite of front-end design automation services for the OEM ASIC market. Mr. Guzowski is currently a senior engineer and a team leader in the ASIC design center.

Brion L. Keller *IBM Microelectronics Division, Endicott facility, Endicott, New York 13760 (KELLERBL at ENDVM5).* Mr. Keller received a B.S. in computer science and chemistry from Pennsylvania State University in 1979. He joined IBM in 1979 and developed a software paging and database access method for use with very large design automation systems. Mr. Keller is currently a senior programmer in the Test Design Automation group in the IBM Microelectronics Division. He is the lead architect for the TestBench ATPG design automation system, which is commercially available from IBM. He has authored several technical disclosures and papers in the area of testing digital circuits. He received an Outstanding Technical Achievement Award for his technical leadership in the development of the IBM self-test design automation system. Mr. Keller is a member of the IEEE Computer Society.

Randal H. Kerr *IBM Microelectronics Division, Endicott facility, Endicott, New York 13760 (RHKERR at ENDVM5).* Mr. Kerr received a B.S. in computer science and a B.S. in political science from Potsdam State University in 1984. He joined IBM in Endicott in 1984 and developed an automated software testing system and a software educational authoring system. In 1988 Mr. Kerr transferred to the IBM Marketing and Services Division, Philadelphia, PA. There he worked

in competitive hardware and software sales and support. In 1991 Mr. Kerr returned to IBM in Endicott, where he was involved in third-party vendor negotiations in support of the System/390® products. Mr. Kerr is currently a senior engineering manager for the test-generation, simulation, and design-for-test synthesis software which is part of the IBM TestBench ATPG product.