DESIGN FOR TESTABILITY: TUNNELLING THROUGH THE TEST WALL

Peter C. Maxwell
Integrated Circuit Business Division
Hewlett-Packard Company
Palo Alto, California

Abstract

This paper discusses design for testability as a means of ensuring that high quality tests can be generated for an IC in reasonable time by avoiding the “over the wall” mentality. A discussion on fault models is followed by general benefits of DFT. Scan design is then described in some detail, together with general DFT guidelines, including representative design rules which must be followed. Some cost/benefit tradeoffs are considered, and the paper concludes by discussing various required components of a composite test suite.

1. Introduction

There are many areas which have to be taken into consideration when designing an integrated circuit. There is the obvious one of meeting the design specification, in terms of functionality, operating speed and marginality with respect to temperature, voltage and IC fabrication variances. Equally important are the economic constraints which have to be satisfied. Although these are complex, two which are appropriate to highlight here are time to market and product cost. It does little good to design an IC which takes so long to complete that the intended product is no longer appropriate in the market place. It does no more good if the IC is completed in a timely manner, but costs so much that the intended product is not competitive.

Test is a process which directly impacts these economic issues. It is something that many designers would prefer to have nothing to do with: “Let the test people figure out how to test it”. Such an approach, with a “wall” between design and test becomes less and less viable as chips become more complex. Chip quality is a critical factor of product cost. Quality requirements map directly to test, as this is the only method of improving quality beyond the intrinsic yield of the fabrication process. Since test has to be good enough to meet quality requirements, it directly affects time to market and cost.

The importance of test, and its requirements, have migrated into design. By applying design for testability (DFT) techniques, it is possible to prevent costs associated with test from making a part uneconomic to produce. The effort spent in the design phase then becomes a mechanism to avoid the “over the wall” approach, by allowing a “tunnelling” through the wall, whereby designers and test engineers work together.

This paper addresses the why and how of DFT for digital ICs. Some information is first presented on failure mechanisms and fault models in order to establish a foundation. Benefits of DFT are then discussed, which is followed by details of scan design and general testability guidelines. This is followed by typical design rules which must be followed in order to be able to generate tests satisfactorily. Finally, some issues regarding cost/benefit tradeoffs and the necessity of using several different types of test are discussed.

2. Three Phases of Testing

Figure 1 illustrates a typical IC production test flow. The quantity $A$ represents die from the fab facility, and $B$, $C$ and $D$ represent the number of parts which pass each test block. The first block, also called “continuity” testing, tests for gross failures before applying power to the part. The second block is called “functional” test, but includes any digital test given to the part. Finally, the parametric tests verify such items as tristate leakage and input/output voltage levels. The fourth box represents system tests, which indirectly test the chip. Failures which occur here ($Z$ in the diagram) represent “test escapes”, also known as Type II errors [1].

The diagram also illustrates some important differences in various definitions of the term yield. For IC processes, the usual definition is the quantity $D/A$, but for test purposes individual yields are more important. In particular, if one attempts to relate fault coverage to expected quality levels using any of a number of theoretical expressions, e.g. [2-5], then the value for yield needs to be $C/B$, since it is the functional test block to which these equations apply.

Figure 1. The three phases of testing.
3. Fault Models

To enable tests to be automatically generated, and a metric available to grade the effectiveness of any given test set, models are needed for chip failure mechanisms. Although the fundamental reason that chips are defective is physical defects that occur during fabrication, it is impractical to directly model these defects for an entire chip since it would involve process simulation. A more abstract fault model is always used, which represents the effect of a physical defect at the transistor or gate level. A given fault model is then used as a target for automatic test generation, and as the basis of test grading by fault simulation.

Because of the multitude of physical defects which can occur, no single fault model can describe all, or even most, failure mechanisms. A variety of fault models is required to ensure a test suite has sufficient defect detection. The most common fault models are outlined below.

A. Stuck-at

In this model, the effect of a physical defect is to make one or more nodes behave as if they were permanently stuck at 1 or 0. It found widespread use for digital bipolar circuits, for which it is a fairly accurate model of actual faulty behavior [6]. For CMOS circuits it is a much poorer indicator of behavior and many defect behaviors cannot be modelled as stuck-at faults [7-9]. Despite this fact, it is still by far the most commonly used model, both for test generation and fault simulation.

B. Bridging

The most common failure in CMOS integrated circuits is a short between physically adjacent conductors, giving rise to a bridging fault. Whereas some of these can be represented by stuck-at behavior (e.g. a short to $V_{dd}$ or $GND$), the majority cannot. Since two nodes are shorted together, in general, weak logic values result, and if feedback is involved, latching or oscillation can result. There have been a number of approaches to simulate and generate tests for these faults [10-16], but none have so far found their way into a commercially available tool. In this regard it is fortunate that tests generated using the stuck-at model typically have a very high coverage of bridging faults [17,18].

The difficulty of dealing with bridging faults at the logic (voltage) level has led to alternative detection methods, in particular using power supply current monitoring [19]. We will discuss DFT to allow IDDQ testing later.

C. Delay

The delay fault model represents an attempt to quantify failures that loosely fall into the category of “timing failures”. There are two main models used, namely the gate delay fault and path delay fault [20]. In the former, a defect is assumed to cause excessive delay at a particular gate, while in the latter it causes excessive delay along a particular path. In contrast to stuck-at and bridging faults, two patterns are required to test delay faults, one to set up the initial state and the second to launch the transition.

D. Fault Grading

The above fault models are used to assess the effectiveness of a test, by determining the fault coverage. It is very important to realize that this coverage represents the proportion of modelled faults detected, and not the proportion of total faults detected. Thus, a single stuck-at fault coverage says nothing about bridging coverage or delay coverage. Coverage of these “non-modelled” faults is accidental, and represents “peripheral” coverage. In order to guarantee their detection, rather than hope for it, they must be directly targeted.

4. The Complementary Nature of ATPG and DFT

It is beyond the scope of this paper to describe in detail the various automatic test pattern generation (ATPG) algorithms that have been developed (see [21] for more details). The two principles of importance are controllability and observability. To generate a test for a fault on a given node, that node must first be stimulated to the appropriate value (controllability) and then the result of that stimulation must be propagated to some node which is observable (observability).

For combinational circuits the problem is manageable and there are a number of effective commercial tools available. For sequential circuits, on the other hand, significant difficulties are encountered. In Fig. 2, the combinational outputs $D_1$..$D_k$ are not directly observable, and the inputs $Q_1$..$Q_k$ are not directly controllable. State sequences must be used to set the flip-flops to desired values and to observe the
next state. Although algorithms exist for test generation (see [22] for some examples), they typically cannot produce test sets with acceptable coverages in acceptable amounts of time for circuits of the type shown in Fig.2 without DFT modifications. Furthermore, manually written functional vectors, although probably having higher coverage than that typically achieved by sequential ATPG vectors, require enormous amounts of effort to obtain sufficient coverage to produce a sufficiently high quality chip.

It is this limitation in available ATPG systems that is one of the main drivers for DFT. ATPG is not very effective without DFT; conversely, without a good ATPG tool many of the benefits of DFT cannot be realised. DFT is more than simply satisfying ATPG requirements, however. Some of the wider set of issues driving DFT will now be discussed.

5. Why Design for Testability?

In the most general sense, the aim of DFT is to improve either or both controllability and observability of a circuit. In doing so the circuit becomes more testable and higher fault coverages can be achieved. The benefits in doing this can be discussed under a number of headings.

A. Fault Coverage and Quality

Even if the exact relationship between fault coverage and quality is not known, it is clear that increasing coverage implies increasing quality. The metric commonly used for quality is defect level (DL), which is expressed by the ratio

\[
\text{defect level} = \frac{\# \text{bad parts which test as good}}{\text{total} \ # \text{parts passing the test}}
\]

Units used are parts per million (ppm) and it is not uncommon to have requirements in the order of a few hundreds of ppm or less. To achieve this, fault coverages in the high 90s must be achieved. Furthermore, because of the multitude of defect types, 100% coverage of any one fault model is no guarantee of high quality [23].

DFT allows higher achievable coverages, which translates directly into higher quality parts.

B. Board and System Quality

Following on from IC quality, board and system quality depend on chip testability. Poorly tested chips result in low quality, which makes board yields lower than they should be. Also, testability features of chips can be used to help board test, an obvious example being boundary scan.

C. Test Application Time

Easily testable ICs have reduced test application time since long, complex sequences are avoided. This translates directly into a per part cost saving.

D. Debug

Although not a test issue, debug is a critical problem which can be made substantially more tractable by DFT. This is because DFT provides improved observability and controllability of internal chip nodes, exactly what is needed for debug purposes. The impact on debuggability by “DFT” can represent a major justification for its use [24].

E. Time to Market

Easily testable ICs reduce time to market. This is because much less time is involved in obtaining tests with sufficiently high fault coverage by automatic means. Time-consuming generation of manual tests is correspondingly reduced. Further, the process of getting the tests running on a tester takes considerably less time for automatic vectors than it does for functional vectors, because the latter are generally extracted from simulation, with events taking place at arbitrary times. This is incompatible with the synchronized event changes that a tester can support, and the translation of one to the other is time-consuming and error-prone.

A further reduction in time to market is the debug aspect. It takes much less time to debug a part with good internal access than one with only the pins to work with.

F. Failure Analysis

Similarly to debug, if failure analysis is required to expose the root cause of a defective chip, then the procedure is made much easier by the presence of DFT structures. In fact, in most cases DFT structures and ATPG vectors make failure analysis possible, when it would not be if functional vectors were the only tests available.

Such analysis can also be of benefit in yield improvement, since process problems can be quickly identified, rather than waiting for traditional SRAM process monitors to be put through the process. Sudden shifts in process, which can cause “yield crashes” can also be identified and corrected.

6. Scan Design

A. Principle of Operation

This technique is a “structured” approach, in that it imposes a global design style, which is
independent of any particular design. The intent is to alleviate the problems discussed above with general sequential circuits as depicted in Fig. 2. In Fig. 3 the flip-flops have been replaced with 2-port devices. When scan enable is asserted, all the flip-flops are connected in one long shift register, the “scan chain”, with the $Q$ of one flop connected to the $D$ of the next.

The controllability problems with the $Q$ state values are eliminated because they can be set directly by scanning in the appropriate values. Similarly, there are no observability problems with the $D$ values because once they are clocked into the flip-flops they can be scanned out and observed one bit at a time at the scan out port. In this way, the difficult sequential test generation problem has been transformed into the much more manageable combinational problem.

Note that providing direct access to all internal storage nodes also makes a huge impact on debug. If a chip fails it is relatively straightforward to determine exactly which flip-flops have incorrect values. A scan chain is akin to having a “dump” facility built in.

Since many designs have a large number of flip-flops, and more than one clock domain, it is common to have multiple scan chains, rather than a single long one. This allows shorter shift times, less tester memory (on a per pin basis) and separate scan chains for different clock domains.

The implementation of the scan cells in Fig. 3 is known as “muxed-scan”, because of the mux in front of each flip-flop. It also characterized by using the system clock as the scan clock. There are many other scan implementations (see [25] p. 434), of which the most well-known is IBM’s LSSD [26], which uses separate clocks for scanning, rather than the system clock.

The benefits of combinational ATPG are achieved in full scan designs, where all flip-flops are scanned. In practice, there is always a small number of elements, frequently latches, that for performance reasons are not scanned. Combinational tools can typically handle this situation, however. This is to be contrasted to partial scan designs, where a significant number of flip-flops are not scanned. The reason for doing this is area saving, since a scan flip-flop is larger than its non-scan version, but sequential test generation is now required.

Adopting scan design means certain design rules need to be followed. These will be discussed next.

B. Design Rules for Scan Design

Simply providing a scan chain is not sufficient to guarantee high coverage tests are able to be generated. Design rules must be followed, which restrict design styles to those which result in circuits which ATPG tools can deal with. It is not possible to detail all scan design rules since they vary with the scan methodology used, however, some general rules can be stated which capture the essentials.

1. No unconnected inputs. All cell inputs must be driven by another cell or by a primary input.
2. Clock control in test mode. All flip-flops on a given scan chain are unconditionally clocked by the same signal edge from the chip pins. This is particularly applicable when gated clocks are used. Since ATPG systems require synchronous operation, gated clocks cause problems and need to be over-ridden. Also, with multiple clock domains, either ensure the clocks can be tied together, or allow independent control of each separate clock signal.
3. Data clock-independent. Flip-flop data must not be gated with the clock.
4. Asynchronism disabling. All flip-flop preset and clear inputs must be able to be set to the inactive state from the chip pins. This is to prevent data from being corrupted during scanning.
5. No drive fights. For tristate buses, no more than one driver may be enabled on a bus at any time, or at least all enabled drivers must drive the same value onto the bus. If a state where no drivers are enabled can exist, buskeepers must be used. ATPG tests place circuits into arbitrary states, including states which never happen in normal mode. These illegal states can cause conflicts, for example on internal tristate buses. If the vectors which cause these conflicts have to be removed from the test, fault coverage can suffer.
6. I/O enable state change independent. For I/O pads, the tristate enable signal must not change during a test, since this can cause a drive fight with the tester.

![Figure 3. Scanned sequential logic.](image-url)
7. No asynchronous feedback loops. Fig. 4 shows a common situation involving bus transceivers, together with a possible solution. Without the additional logic, a loop is formed when both transceivers are turned on, a situation which is unable to be handled by ATPG tools.

![Figure 4. Asynchronous loops in bus transceivers.](image)

8. Bypass embedded RAM or other cores, or wrap them in scan chains. While generating tests for other logic, ATPG programs do not know how the embedded cores operate, with the result that logic outputs that go into the core become unobservable and core outputs that feed into the logic are uncontrollable. Poor coverage results unless something is done.

9. Disable RAM writes and latch gates during shifting. While scanning, flip-flop outputs are feeding into the logic. If a RAM has been set up with known values on its outputs then the scanning operation must be prevented from causing new data to be written into the RAM, corrupting the known values.

10. Make latches transparent. Latches behave in some ways like RAM, and have to be prevented from introducing unknown values into downstream logic. A simple way is to force them into transparent mode during testing.

7. Other DFT Guidelines

There are other guidelines which apply whether a design uses scan or not. These consist of a list of “sensible” things to do. Exactly which are used will depend on a particular design, since not all techniques are appropriate for all designs. A list of guidelines, with a brief explanation follows.

1. Make RESET easy for ATPG and tester to do. A chip has to be in a known state before testing can commence, and long, complex initialization sequences are difficult, time-consuming and costly.

2. Make CLOCKS controllable by the tester. There needs to be a bypass provided so that system clocks can be driven by the tester, rather than by free-running internal clock generators. The latter pose formidable problems in synchronization of the data between the chip and tester.

3. Add TEST POINTS to enhance controllability/observability. Fig. 5 shows the most general form, where, for example, logic B can be tested by driving its inputs directly without having to propagate them through logic A. The mux controlled by TEST_SEL_1 can be replaced by a single gate if only one value is desired to be injected by this means (e.g., an AND gate to obtain a 0).

![Figure 5. Test point insertion.](image)

4. Beware of REDUNDANT LOGIC. Whereas synthesis tools are effective in eliminating accidental redundancies, there are situations where redundancy is required, for example in ECC circuits and in hazard protection. Special allowance must be made to test these circuits or the protection provided may be lost.

5. Split LARGE COUNTERS into 4-bit pieces, or SCAN them. Without any DFT, a 32-bit counter needs $2^{32}$ clock cycles to fully test it. This is uneconomically long on a tester. By segmenting the counter, a complete test can be done in a fraction of the time.

8. IDDQ Testing

A. What is IDDQ Testing?

This is a method of testing which relies on monitoring the steady-state value of the chip power supply current. The main premise is that in CMOS circuits, the “normal” state of a gate provides no current path from $V_{DD}$ to $GND$ (either the $p$ block or the $n$ block is conducting but not both). The only current observed is therefore leakage current, which is low. Any abnormally high current therefore indicates the presence of a defect. Although a single measurement can give reasonable coverage (typically 30%) current
needs to be measured for a large number of different circuit states to maximize coverage.

The method provides unsurpassed observability, since the power supply is present at every gate. Bridges are particularly amenable to detection since a short between gates driving opposite values results in a low conduction path from $V_{DD}$ to GND. The resultant high current is directly observable, without having to be concerned about logic values which result from the bridge. Note that sensitization of the bridge is the same as for logic testing: IDDQ does not improve controllability.

B. Rules for IDDQ Testing

The use of IDDQ testing requires additional rules to the scan rules described earlier. Clearly the drive fight rule is applicable to IDDQ test as well, but others are listed below.

1. No active pullups or pulldowns. These result in high current when the node is at the opposite state.
2. No degraded voltages. Logic inputs must not be driven by a single $n$ pass transistor since the degraded logic 1 will cause the $p$ block in the driven gate to partially conduct, causing elevated current. If switch logic of this type is used, a restoring device must be used as shown in Fig. 6.
3. Power dissipating circuitry such as RAMs or analog blocks must have a standby mode, or be placed on separate power supplies.

9. The Costs of Designing for Testability

A. Silicon Area

The largest area impact occurs in the adoption of scan design. A scan flip-flop has more logic associated with it than a non-scan flip-flop and therefore is larger. Additional test signals also have to be routed. The actual overhead will vary according to the ratio of flip-flop gates to logic gates. It will also depend on the area taken up by RAMs and other cores.

Typically overhead is minimized by connecting the scan chain in an order which is dependent on layout, this information being obtained from placement tools.

In any case, it is frequently not the actual area overhead which is the critical factor. Standard cell ASICs are commonly produced in fixed die sizes, so if the area increase can still be accommodated in the same die size then the cost impact is small. Also, if a design is pad-limited, then core area increases make no difference to overall die size.

B. Circuit Speed

The muxed-scan approach shown in Fig. 3 clearly has a performance impact, since the addition of the mux before the normal D input introduces additional delay. Other implementations such as LSSD can have smaller additional delays because the multiplexing of system and scan data is done using different clocks, rather than a mux in the signal path. Additional second order delays also occur due to additional clock loading and flip-flop output fanout to accommodate the scan path.

Use of the techniques in section 7 also introduces additional delay, particularly ifmuxes are added for additional controllability and/or observability.

C. Design Style Restrictions

The design rules discussed above prohibit certain design styles. Whereas the end result is a much more testable design, the restrictions could be viewed as a "price" to pay.

D. Extra Design Work

In order to satisfy the rules, extra design work is required during the design cycle. Designs may have to be reworked when design rule checking indicates rule violations.

Some of the above items are difficult to quantify, but the overhead can generally be managed to be less than 10%.

10. The Costs of Not Designing for Testability

A. Design Time

Without DFT, designing good test patterns is very difficult. Since the patterns are almost certainly functional, the only people who can do this are the designers, which therefore adds to design time.

B. Resources

Coupled with the above, circuit designers are tied up with writing tests, when they should be designing the next chip. Thus future designs are also impacted.

C. Time to Market

Parts cannot be shipped without a good test. Because large amounts of time are required to get tests
to sufficient fault coverage, time to market is negatively impacted.

D. Manufacturability

Even with the best manual tests, it is highly unlikely that coverage is going to be in upper 90s. The poorer tests that result in many more bad parts escaping. This then becomes a manufacturing issue, particularly when the customer complains about shipped quality.

E. Profit

In general, functional tests are very long when compared to scan tests. Without DFT, long and complex sequences are needed, which produce tests which take a long time to run. These inefficient tests, and the test escapes, are very expensive and reduce profit.

F. Complexity

Part debug is very difficult with no DFT, and board and system turnon are also very tough. The lack of controllability and observability has impact at all levels.

However the above is counted, there is usually substantially more than 10% penalty by ignoring DFT issues.

11. The ROI for DFT

From analyzing the costs and benefits of DFT a few issues become evident. Firstly there is a net savings in designer effort. Even though more time has to be invested during the design cycle, there is a large payback when the test generation step is reached. For a small incremental cost, disasters are avoided. These disasters occur when it is discovered that a test of appropriate coverage cannot be obtained, and the design has to be turned. This is always an expensive proposition.

Known results in known time is another big advantage. With no DFT, the process of obtaining and debugging functional vectors takes an unpredictable amount of time, and it is inevitably significantly longer than the much more predictable results when DFT is used.

Finally, the provision of DFT makes chip, board and system debug easier. The return on investment for DFT is extremely high.

12. A Composite Test Strategy

Once DFT has been implemented in a design, several different types of tests are able to be generated. Examining the list of common fault models described in section 3 we can identify tests which target these.

A. Stuck-at

This being the “standard” model, there are several ATPG systems available which will generate tests to detect all detectable faults. Coverages are extremely high, but usually not 100% of all faults due to test mode signals (usually required to ensure that the design satisfies the design rules) blocking some faults in test mode.

B. Bridging

IDDQ tests are very effective in detecting bridges. The tests can be generated using ATPG and applied using scan, or selected from a set of functional vectors.

C. Delay

A common method of obtaining timing coverage is to use functional vectors run at-speed. However, more direct targeting of these faults is to use “AC scan”, whereby transitions are generated by using the scan chain [27,28].

D. Putting it all Together

High quality is assured only by high coverage of as many defect types as possible. Consequently, all the above tests are necessary, and recent experimental results [29] show how all tests detect failures that are missed by any other test. If any test is not used, quality will worsen, and it is then an economic decision whether this is acceptable or not.

13. Issues Not Covered

Space does not permit giving details of many other forms of DFT. Built-in self test (BIST) and boundary scan are two important areas which deserve mentioning. A good reference is [21] and there are many articles on these subjects in the International Test Conference proceedings, and other conferences.

14. References


