

# The Evolution and Future of DFX in Modern ICs

**Al Crouch**  
Inovys Corp.

## Abstract

In the past Design-for-Test, known as DFT, was the most popular concurrent engineering term in use. However, recent changes in the nature of process tracking, yield analysis and the difficulty of manufacturing has led to a spate of new “Design-for” terms; the most widely used and confused term being DFM, but DFY, DFR, DFD and DFPA have also been used recently.

## The Wealth of Embedded Capabilities

In modern electronics in the nanometer era, many problems and issues have been managed in ASICs, SoCs, Microprocessors, DSPs, NoCs and other chips with large digital content by adopting and embedding more test and measurement circuitry on the die. The purpose of this logic is to enhance the measured fault coverage; to reduce the volume of test vectors used; to help automate the creation of those vectors; to reduce the time and work required to diagnose defects, errors, and bugs to an actionable root cause; and to monitor the analog and parametric effects associated with digital logic (e.g., voltage sensitivity, thermal performance, power

consumption). The majority of this extra circuitry has the purpose of enabling or enhancing structural-test (deterministic tests based on fault models) and so it is called Design-for-Test or DFT logic. However, there has recently been an increase in other embedded logic to enhance debug-and-diagnosis, functional verification (assertions), and yield analysis – so the term DFX has been used (where the “x” represents a variable that stands for Test, Debug, Manufacturing, Reliability, and Yield).

## Historically Speaking – DFT Evolves

When I first started applying my trade as a Design-for-Test engineer (more years ago than I care to admit), there really didn’t exist any such thing named Design-for-Test, but there were two active concepts that would grow into what we call DFT today: Testability and Concurrent-Engineering. These were in the days where a big chip was defined as 35K-Gates and a 20MHz clock rate was blisteringly fast and the problem was defined as accessing those 35 thousand gates through just 49 or even 144 package pins (the gate to pin ratio). The term and science of “Testability” had migrated from

the board test and military products segment and was originally an after-the-design concept where a design was checked against rules or a checklist for being testable (and so the testability could be measured as to how many rules or questions were answered positively out of the total number of rules asked). Some of these checks and rules involved being able to reset the logic or control its clocking, but a great many of the rules resolved around the idea of checking for observability and controllability criteria as it directly related to the application of vectors for “coverage.” Coverage was then simply defined as either the ability to set a node or net in the design to a logic 1 or 0 or to toggle it – it wasn’t the elegant fault-model and analysis that it is today. Concurrent engineering was a similarly

through the normal interface as if the chip were being used in its application) were the dominant solution to manufacturing testing of chips, however, functional vectors are made to implement and verify operation – not manufacturing criteria. Manufacturing test is ideally the verification that the structure of the chip does not include manufacturing errors or defects; it is not cost-effective to use manufacturing test as the verification that the designers created the correct logic functions – this could be done once on a testbench with a representative chip, but should not be done repeatedly as part of the production test process. This means that out of all of the possible functional vectors that could be applied (which in my experience, even in the early days, was millions), some selection process

***The concept of collecting or observing data while the natural functional software was being applied is commonly called “Trace.”***

borrowed term and was the concept of conducting an evaluation or analysis during the design process as opposed to after the design was completed.

Concurrent-Engineering and Testability eventually evolved into DFT and in several discrete stages – and each stage of the evolution was driven by some limitation or trade-off. For example, the first major driver that brought Concurrent-Engineering and Testability together was the size of the tester memory within the ATE of the time.

Originally, functional vectors (chip operation vectors that entered the chip

must be implemented to choose the ones most suited for manufacturing test (usually fault-simulation) – and the number of vectors to be selected were economically limited to the size of the Automatic Test Equipment (ATE) memory (because reloading the ATE for a second pass back then, even with a small test program memory, consumed the better part of an hour). It was this selection process that drove one of the first evolutionary stages of what could be called DFT.

The limits placed on the amount of test vectors that could be economically

applied by an ATE led to the concept of “measurable fault coverage” and the use of automated or software sequential vector generation (as opposed to manual or ad hoc sequential vector generation). The adoption of “fault-based” vectors coupled with the limited ATE vector memory and the requirement of a target coverage then led to the next step in the evolution – the analysis of the design for testpoints to enhance the coverage. The testpoints were placed where there was a controllability or an observability problem that resulted in a large amount of vectors to achieve a little bit of coverage (for example, having to actually count to 4 billion with a 32-bit counter to achieve all the states necessary to test the structure of the logic that makes up the counter). This led to a methodology that included a lengthy analysis of the schematic (gate-level) to identify where to put the few testpoints (for example,

using SCOAP – Sandia Controllability and Observability Analysis Program); and if the number of package pins were limited, then a scan shift-register method was used to access these few testpoints (early on, the type of scan access was the 1149.1 JTAG type of scan, not the scan popularized today). At a historical time period when chips had multiyear design cycles, spending significant time evaluating and modifying the gate-level portion of the design with the intent to reduce the number of vectors required to achieve a high measurable target coverage and using a partial-scan methodology, was entirely acceptable.

As time moved forward, Moore’s Law marched on and design cycles shrank while design sizes increased dramatically; and more and more chip designs were being represented in an HDL/RTL model form that allowed simulation and in many cases automated synthesis. At this point, in some

market segments, the luxury of time to conduct the analysis on the gate-level description was no longer an option and “partial-scan” gave way to “mostly-scan.” Mostly-scan represented automatically converting most of a design to the scan-shift methodology, but not the perceived critical timing paths. So, instead of adding a few JTAG-like scan registers to be accessed by the JTAG TDI-TDO serial path – this scan methodology was made by converting the existing functional flip-flops to scan flip-flops by adding a multiplexor in front of the D-input to create an additional Scan-Data-In (SDI) port and also routing a controlling multiplexor signal known as Scan-Enable (SE), and then adding routing connections to the Q or Q-bar output (or sometimes a dedicated flip-flop cell SDO signal) in such a way to connect the flip-flops together into serial shift-registers which were then called scan chains.

In the early days of mostly-scan, the shifting clock frequencies were slow (generally between 1MHz and 20MHz) and since the economics of scan testing were not fully understood – there were usually too few scan chains to make full economic use of scan. For example, in many cases there were just 1 to 8 scan chains and the pins used to feed those chains were dedicated pins or borrowed pins whose timing requirements could easily stand the extra loading or support an extra multiplexor.

The automation of vector generation for mostly-scan was one of the key economic savings – much faster and much more compact than automated sequential vector generation for most of the chip (combinational vector generation reduced the problem to solving tracing equations on isolated chunks of combinational logic and only across one clock cycle). However, not fully understanding the economic trade-

offs involved with scan led to a period of “questioning the value” of scan since badly inserted scan could consume large amounts of chip area (as much as 28 percent had been reported) since scan is a route intensive methodology; and if there was a single or just a few scan chains and with dedicated pins – this added the cost of package pins and created vector files that only fit on ATE with dedicated scan test options or that required multiple test insertions to fit. This drove the next evolution which was more of an economic learning function by the ATE industry, the EDA ATPG industry, and the chip design industry – that scan is more affordable when optimized and optimizations were possible and automatable. The scan insertion process was largely included within the logic synthesis programs and some tools started applying routing optimization to the physical layout of scan chains using algorithms such as nearest-neighbor (achieving a more palatable 5 percent area overhead); and the chip designers started borrowing functional pins to make “parallel scan” chains (usually between 32 and 100 depending on the number of available signal pins). Scan clocking (the shift data rate) was also increased in many cases to between 50MHz and 100MHz (depending on the tester available) to reduce the applied test time.

Several things happened at once which drove the flavor of the scan methodology to be included in chips – but for the most part, no matter what the specific test architecture, the digital logic was now tested by scan and the term DFT was fully entrenched. In fact, many organizations now had DFT engineers on staff (a person or team whose job it was to operate the scan insertion and ATPG tools). One of the

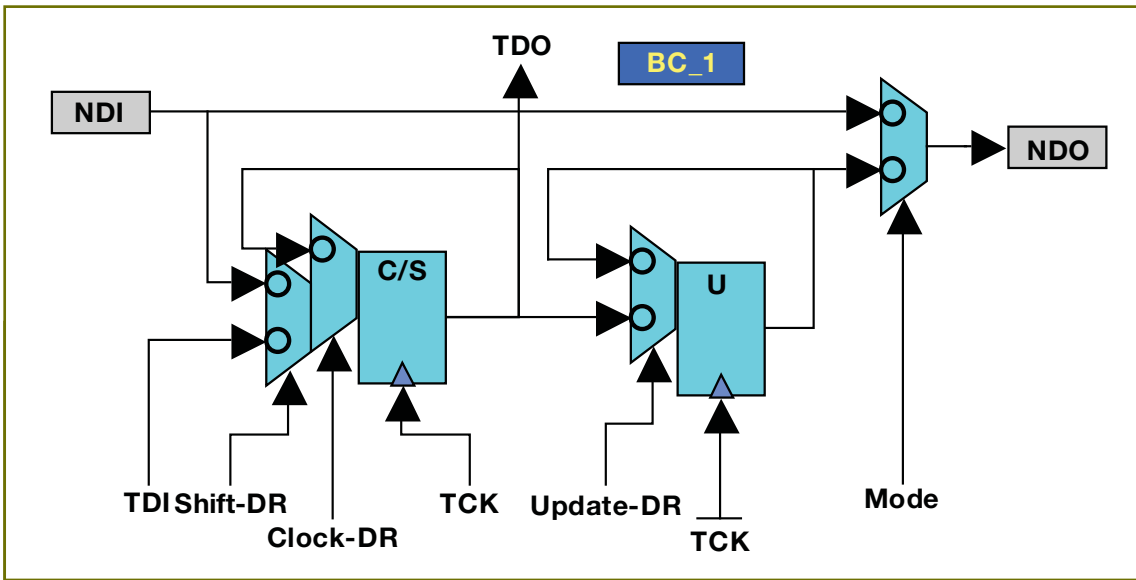


Figure 1. A JTAG Scan & Update Cell as a Testpoint

major next steps forward was the use of Core-Based design – the reuse of an existing microprocessor or microcontroller core coupled with memory cores. This placed the requirement on the core provider to ensure the achievable coverage of the core – and for hard cores (layout macros) that the test architecture, test vectors, and achievable coverage of the test vectors be “packaged and portable.” A great number of solutions were tried, but largely “full-scan” and “scan test wrappers” won out. The portable and test-in-isolation requirement was such a problem that several Standards bodies looked at the issue and eventually, the IEEE 1500 Core Test Standard was ratified.

The other interesting test methodologies that matured with the adoption of core-based design was the portable scan-based “logic built-in self-test” (BIST) and the portable memory BIST units – both of these methodologies move the vector generation and vector application into the chip itself. These BIST methodologies were not new technologies, but were adaptations that had been borrowed from design environments such as space electronics (where self-contained test was required) or very high-performance chips such as cutting-edge microprocessors (where the coverage metric included both stuck-at coverage and at-speed frequency or timing coverage).

This brings us to the current state of DFT where the latest economic driver is now test time with a lesser driver being Intellectual-Property (IP) protection (scan testing provides a remarkable amount of register information and could allow register-mapping of an end design if no steps are taken to obfuscate the scan data output). The scan methodology has now evolved into “embedded vector compression” as a response to these issues. The

embedded vector compression methodology uses concepts from logic BIST – a re-seeding pattern generator of some sort – to enable a small scan vector in the tester to drive a pattern generator which creates the actual larger scan vector applied to the scan chains (for example, a 32-bit serial bit stream that looks like a scan vector in the tester may expand to drive 320 scan chains that are 32 clock cycles deep). The time saving or compression now comes from supporting a very large number of scan chains, such as 1,000, being driven by a decompressor instead of just a few scan chains driven by package pins. In most cases of scan testing, the number of scan chains is limited to the number of available chip pins – for example, 100,000 flip-flops can make 1 scan chain that takes 100,000 clocks to load-unload it once (and there are usually thousands of scan vectors in a pattern set) if 3 pins are used for SDI, SDO and SE; or 100 scan chains that take 1000 clocks to load-unload once if 201 pins are used. The embedded compression methodology allows just a few scan pins to feed a decompressor logic unit and the decompressor feeds hundreds to a few thousand scan chains though some noncorrelation and distribution logic resulting in very short load-unloads (for example, the 100,000 flip-flops now organized into 1,000 scan chains have a load-unload of 100 clocks). The scan chain outputs are similarly routed through a compactor unit that reduces them to just a few pins – the effect of this is to fully encrypt the test data supplied by and observed by the tester. This makes for an interesting self-contained vector application method that is portable (a core can be delivered with a decompressor and compactor and can be integrated as

complete and testable unit) and provides a measure of IP protection.

The most commonly supported DFTs included within the majority of die manufactured today are Manufacturing Scan, Logic BIST, and Embedded Deterministic Scan Compression for digital logic; Scan Test Wrappers as Isolation-and-Test-Access Mechanisms (TAMs) for embedded cores; Memory BIST for most forms of memory; and JTAG Boundary-Scan for digital I/O. There has also recently been more support for specialty items such as internal loop-back and embedded-pattern generation for high-speed I/O. There have also been the inclusion of other logics meant to assess the electrical environment and to provide real-time data capture for debug purposes, but these logics fall under the heading of a different “Design-for” category.

### The Case of DFD

The proliferation of cores based on microprocessors and digital signal processors (DSPs), such as ARM, MIPS, ColdFire, PowerPC and so on brought on another “design-for” – Design-for-Debug. Microprocessors have always supported internal access to logical registers such as the program counter, stack pointer, and user registers to enhance the software development stage (part of this capability is actually a requirement imposed by the software compiler companies). The concept of collecting or observing data while the natural functional software was being applied is commonly called “Trace.” Other similar functions such as replacing or hiding bad data (overwrite or mask) are also commonly supported. This access, since it was incorporated in the hardware, was naturally used by the design team during first silicon bring up to enhance yield analysis and performance evaluation.

Recently, the other logic that is not part of the processor core has been treated similarly, but in a more general fashion. A concept known as “hardware assertions” has begun to be applied and can be viewed as embedded verification. The main problem with DFD is that it is not standardized – it is true that there are only so many functions that can be applied for debug purposes (compare functions, trace functions, etc.) but the protocol of the access is different with each core. A common example is that a single chip can include multiple cores and each core may have its own debug-access-port, so to conduct debug on a core embedded within a chip, may require a different debug interface at the chip’s edge for each core. This problem is currently being addressed by a standardization effort through the OCP-IP (Open Core Protocol – International Partnership: the complete socket standard ensuring rapid creation and integration of interoperable virtual components).

### What About DFM?

With the adoption of sub-wavelength nanometer designs – designs that include feature sizes smaller than the wavelength of light used to expose them lithographically – a new set of problems have arisen. Basically, lithographic issues create design weaknesses that result in timing, thermal, and voltage sensitivities. The attempt to understand and prevent these problems has been called DFM, Design-for-Manufacturing. There was some confusion in the industry on using the DFM term because DFT and DFD imply that logic is added to the design to make them more testable and debug-diagnosable, but DFM was a term largely associated with evaluating the mask for lithographic issues. The term that meant

to add logic to the chip to assist with DFM issues was more correctly named DFY (Design-for-Yield). DFM as a term has evolved to mean the evaluation of the layout during the design phase to enhance the lithographic process. This includes OPC (Optical Proximity Correction), RET (Reticle Enhancement Technology), and CMP-Fill (Chemical Mechanical Polishing metal fill of blank areas to help the physical polishing process).

When there are DFM issues, they can create Systematic yield impacts. However, these yield issues are not easily diagnosable as are particulate or process or machinery based problems – systematic yield issues from DFM relate back to the design and require design information to

trace to a root-cause. For example, a difficult to manufacture particular standard cell such as a high-drive AND-gate may create nominal operation of some gates on some chips in some logics under some conditions and may create chip failure signatures that look largely random. It takes sophisticated yield analysis to trace chip failures to gates and to then make the correlation that a group of fails all involve exactly the same gate-type even though that gate may be in different areas of the chip. To adequately detect DFM faults requires very expensive test techniques such as multiple temperature testing; multiple applied frequency screening; and voltage modulation or voltage stress testing. Including logic that can assist in

the evaluation of fails, can decrease the cost function involved and that can help localize the fails to layout objects such as gates, nets, vias; or to environmental conditions such as voltage and temperature; or to operating conditions such as the applied clock frequency – including DFY logic and this logic is usually held in different regard than DFT logic. This logic is becoming necessary in many current designs as a means to track operation issues that are evident in-system back to the yield-issues they represent for the chip or core provider.

DFM issues have recently been implicated in a class of problems that have been called NTF – no trouble found. This is the case when delivered chips that have passed testing are placed in their board or system environment and then fail – when removed and evaluated on a tester, they pass. When these fails are traced down, it turns out that a large portion of them have their source in DFM issues that have escaped manufacturing test at the chip level. So this is a growth space that may drive the next generation of DFT, DFD and DFY.

### The Future

The future holds incredible levels of integration – tens to hundreds of cores on one die; complex on-chip data transfer schemes such as internet busses on-chip; complex form factors such as 3D die stacks with vertical inter-chip and intra-chip busses made of through-silicon-vias (TSVs); and the requirement to isolate to a sub-hierarchy such as a logic core, memory core, or die in a die-stack to allow yield to be tracked and enhanced. The chips, die, cores and busses of the future will also be subject to new failure modes and process sensitivities that may include quantum effects and thermal regulation

issues. This means that the DFT, DFD, DFY, and DFM that make up DFx today may be joined by other new DFs that may include DFPA (Power-Aware) or DFH (Heat/Thermal) analyses and logics. History has shown, however, that the most-likely future is one where the current DFx technologies become commonplace, more automated, and more part of the design landscape so that there is little or no thought to most of it – cores will come with embedded-compression scan and memory-BIST testing and interfaces that are just plug-and-play for most applications; and the organization of the logic that makes up DFx will be more formalized and selection, scheduling, configuration, and use will be more automated under some chip-level standardized DFx processor. ■

### About the Author

#### Al Crouch

Al Crouch is chief scientist and director of DFx R&D at Inovys, a provider of DFT-based ATE and DFx software. He has over 20 years in the semiconductor industry and has previously worked for companies such as TI, DEC and Motorola. He is an inventor on more than 13 issued patents and the author of a best-selling DFT engineering textbook.

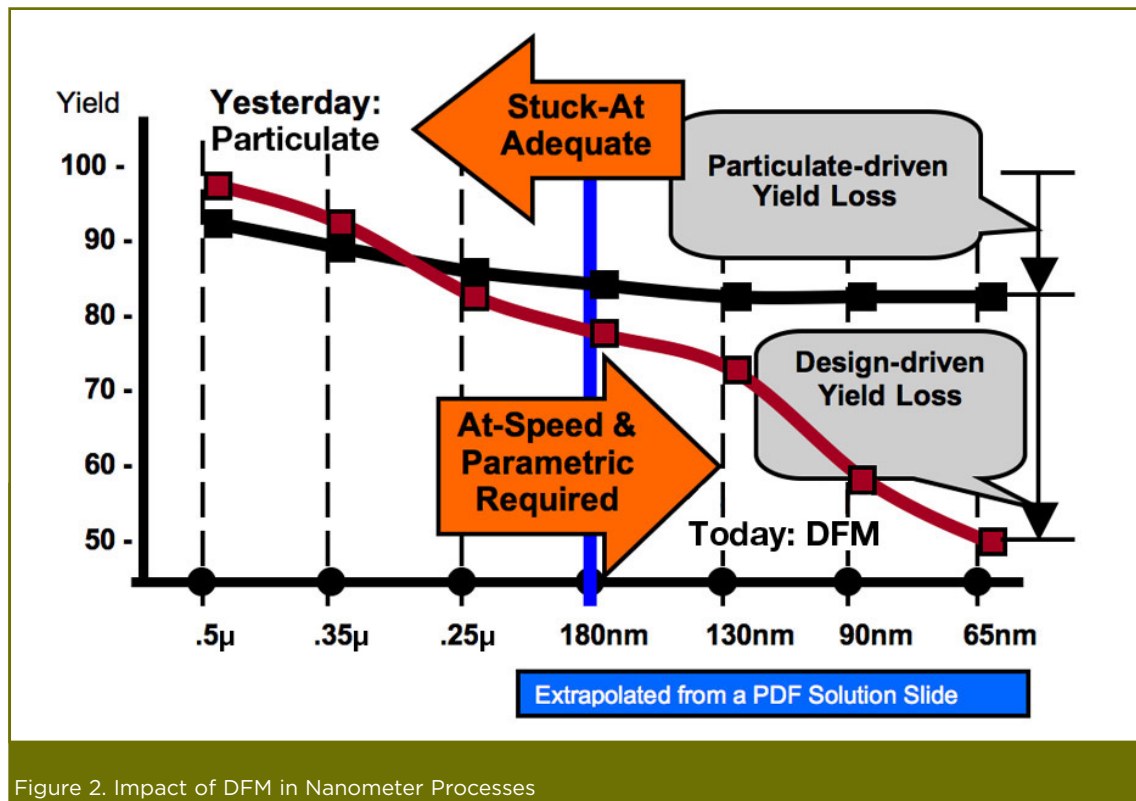


Figure 2. Impact of DFM in Nanometer Processes