

15

Testing a System on a Chip

15.1 Introduction

The shift toward very deep submicron technology has encouraged IC designers to increase the complexity of their designs to the extent that an entire system is now implemented on a chip. This new design paradigm of system on a chip (SOC) has changed the approach to design and testing. To increase the design productivity and decrease time-to-market, reuse of previously designed modules is becoming common practice in SOC design. However, the reuse approach is not limited to in-house designs, but is extended to modules that have been designed by others as well. Such modules are referred to as *embedded cores*. This approach to design has encouraged the founding of several companies that specialize in providing embedded cores to service multiple customers. It is predicted that in the near future, cores, of which 40 to 60% will be from external sources [Smith 1997], will populate 90% of a chip. Except for a very few, individual companies do not have the wide range of expertise that can match the spectrum of design types in demand today.

Core-based design that is justified by the need to decrease time-to-market has created a host of challenging problems for the design and testing community. First, there are legal issues regarding the intellectual property (IP) for the core provider and the user. Second, there are problems in integrating and verifying a mix of proprietary and external cores that is more involved than simply integrating ICs on a printed circuit board (PCB). The third issue, which is the basic concern of this book, is the need for an efficient testing strategy.

Before embarking on discussing the possible testing strategies and their corresponding design approaches, we first examine the various types of cores and core-based design approaches. The testing strategies presented here are taken from academic and industrial sources.

15.2 Classification of Cores

A typical SOC configuration is shown in Fig. 15.1. It consists of several *cores* that are also referred to as *modules*, *blocks*, or *macros*. Often, these terms are used interchangeably. These cores may be DSP, RAM modules, or controllers. This same image of an SOC may be perceived as a PCB with the cores being the ICs mounted on it.

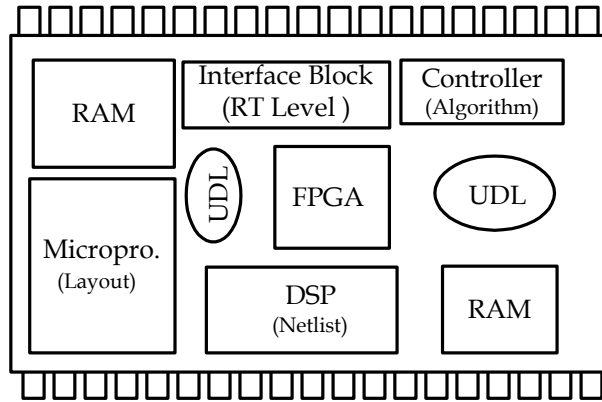


Figure 15.1: A system on a Chip.

It also resembles standard cells laid on the floor of an IC. In the latter cases, the blocks are of identical types. That is, they are all ICs in the PCB case or all standard cells in the IC case. For an SOC, they may be a mix of types, as we describe next.

Table 1. Categorizing Reusable Blocks [Hunt 1996]

Type	Flexibility	Design Flow	Representation	Libraries	Process Technology	Portability
Soft	Very flexible Unpredictable	System Design	Behavioral	Not applicable	Independent	Unlimited
		RTL Design	RTL			
Firm	Flexible	Floor planning	RTL, blocks Netlist	Reference	Generic	Library Mapping
		Placement				
Hard	Inflexible Predictable	Routing Verification	Polygon data	Process specific library and design rules	Fixed	Process mapping

Reusable cores are classified in three categories: hard, firm, and soft [Gupta 1997]. *Hard cores* are optimized for area and performance and they are mapped into a specific technology and possibly a specific foundry. They are provided as layout files that cannot be modified by the users. *Soft cores*, on the other extreme, may be available as HDL technology-independent files. From a design point of view, the layout of a soft core is flexible, although some guidelines may be necessary for good performance. This flexibility allows optimization to the desired levels of performance or area. *Firm cores* are usually provided as technology-dependent netlists using library cells whose size, aspect ratio, and pin location can be changed to meet the customer needs. Table 15.1 summarizes the attributes of reusable cores. From this table there clearly is a trade-off between design flexibility on one hand and predictability and hence time to market performance complexity on the other. Soft cores are easily embedded in a

design. The ASIC designers have complete control over the implementation of this core, but it is the designer's job to optimize it for area, test, or power performance.

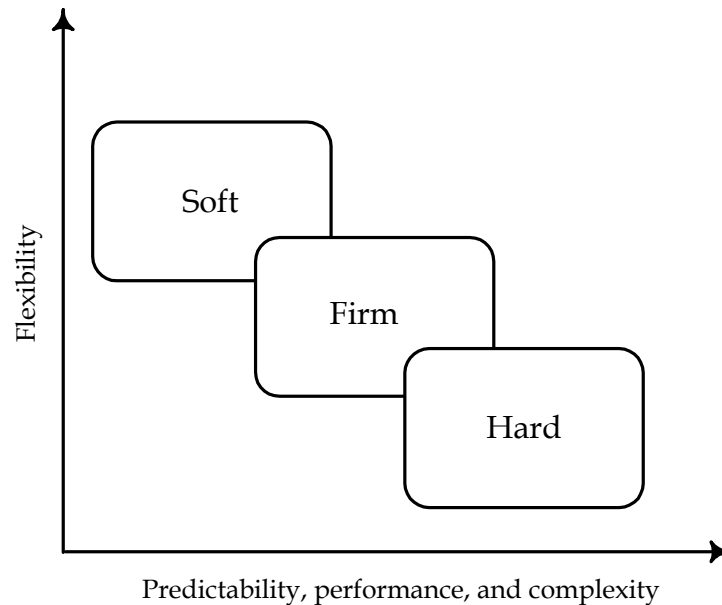


Fig. 15.2 Tradeoffs among Types of Cores [Hunt 1996]

Hard cores are very appropriate for time critical applications, whereas soft cores are candidates for frequent customization. The relationship between flexibility and predictability is illustrated in Fig. 15.2. The cores can also be classified from a testing perspective. For example, there is typically no way to test a hard core unless the supplier provides a test set for this core, whereas test set for the soft core need to be created if not provided by the core provider. This makes hard cores more demanding when developing a test strategy for the chip. For example, it would be difficult to transport through hard cores a test for an adjacent block that may be another core or a UDL component. In some special cases, the problem may be alleviated if the core includes well-described testability functions. We discuss this feature in Section 15.6.3.

15.3 Design and Test Flow

As we have advocated in the beginning of the book, an integrated design and test process is highly recommended. This approach cannot be more appropriate than it is for core-based systems. Conceptually, the system on a chip paradigm is analogous to integration of several ICs on a printed circuit board. However, there is a fundamental difference. Whereas in a PCB the different ICs have been designed, verified, fabricated, and tested independently

from the board, fabrication and testing of an SOC are done only after integration of the different cores. This fact implies that even if the cores are accompanied by a test set, incorporation of the test sets is not that simple and must be considered while integrating the system. In other words, reuse of design does not translate to easy reuse of the test set. What makes this task even more difficult is that the system may include different cores that have different test strategies. Also, the cores may cover a wide range of functions as well as a diverse range of technologies, and they may be described using different HDL languages, such as Verilog, VHDL, and Hardware C to GDSII.

The basic design flow, shown in Fig. 3.3, still applies to SOC design in the sense that the entire system needs to be entered, debugged, modified for testability, validated, and mapped to a technology; but all of this has to be done in an *integrated framework*. Before starting the design process, an overall strategy needs to be chartered to facilitate the integration. In this respect, the specification phase is enlarged and a test strategy is included. This move toward more design on the system level and less time on the logic level has been emphasized in the book from the onset and illustrated in Fig. 1.1.

The design must first be partitioned. Then decisions must be made on such questions as:

- Which partition can be instantiated by an existing core?
- Should a core be supplied by a vendor or done in-house?
- What type of core should be used?
- What is the integration process to facilitate verification and testing?

Because of the wide spectrum of core choices and the diversity of design approaches, SOC design requires a *metamethodology*. That is, a methodology that can streamline the demands of all other methodologies used to design and test the reusable blocks as well as their integration with user defined logic. To optimize on the core-based design, an industry group deemed it necessary to establish a common set of specifications. This group, known as the Virtual Socket Interface Alliance (VSIA), was announced formally in September 1996. Its intent is to establish standards that facilitate communication between core creators and users, the SOC designers [IEEE 1999a].

An example of using multiple cores is the IBM-designed PowerPC product line, based on the PowerPC 40X chip series [Rincon 1997]. The PowerPC microcontroller consisted of a hard core and several soft cores. For timing critical components such as the CPU, a hard core was selected, while soft cores were used for peripheral functions such as the DMA controller, external bus interface unit (EBIU), timers, and serial port unit (SPU). The EBIU may be substituted by, say, a hard core from Rambus.

A change in simulation and synthesis processes is required for embedded cores due primarily to the need to protect the intellectual property of the core provider. Firm cores may be encrypted in such a manner as to respond to the simulator without being readable by humans. For synthesis the core is instantiated in the design. In the case of a soft core, sometimes the parameters are scaled to meet the design constraints. To preserve the core performance, the vendor may include an environment option to prevent the synthesis program from changing some parts of the design. This will protect the core during optimization. However, the designer may remove such an option and make some changes in the design. A hard or a firm core is treated as a black box from the library and goes through the synthesis process untouched.

15.4 Core Test Requirements

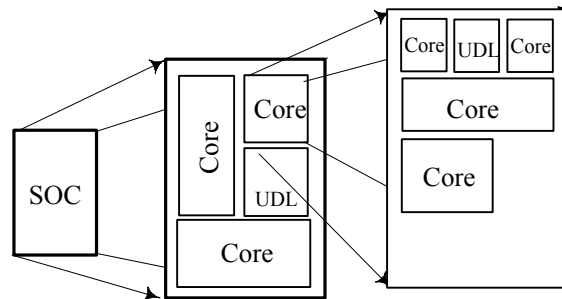


Figure 15.3 Hierarchical Organization of Cores

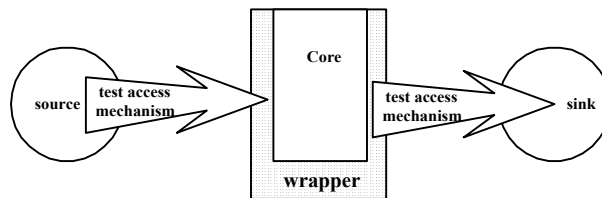


Figure 15.4 Test Architecture

There are many issues that arise in attempting to test a core-based SOC: The system is too large; it has a heterogeneous complexion since it integrates a wide range of core types; and the cores themselves may be core-based, as illustrated in Fig. 15.3. In traditional ASIC, the design consists of several components usually selected from a cell library. Based on fault models, test patterns are generated for the complete design. This may be done manually or automatically, but it is usually applied to the primary inputs of the design and its responses are observed at the primary outputs. For core-based circuits, it is possible to merge the soft cores with the UDL components and a

test can then be generated by the core user for the merged design. Such an approach is not feasible with non-mergeable cores, such as hard cores. Cores might also come with different embedded test schemes that the core provider includes in the design such as a BIST controller or scan path for internal test. Assume for a moment that internal test sets are available for each core. The next issue is to devise mechanisms and a strategy for test execution in the individual cores; that is, it is necessary to have test access in order to justify the test patterns at the inputs terminal of the core. In addition, it is important to propagate the test response from the output terminals as depicted in Fig. 15.4. In this paradigm, the patterns are supplied from the source of test data and the response verification is considered as the sink of the test data. Another important requirement is testing the interconnections and the “glue” logic among the cores on the chip. In summary, the main test requirements for SOC testing are:

- Internal test for embedded cores
- Access to the embedded core
- Isolation of one or more cores that will be tested simultaneously
- Facility to test the interconnections among the cores
- Facility to test all UDLs added logic to “glue” the cores

At the SOC level, to orchestrate application of the test sets to any core and the observation of the response, it is important to plan a strategy at the onset of the design. It is important to decide about the need for a *test controller* or a *test bus*. It would be too late to realize such needs after the SOC design is completed.

Recognizing the above requirements and the need for interoperability, a number of initiatives have begun to address the integration of testing with core-based chip design. Several organizations, including an IEEE Test Technology Technical Council (TTTC) [www.computer.org/tab/ttc] meet regularly to develop a standard test access for cores from various providers [Zorian 1997].

15.5 Conceptual Test Architecture

One of the major challenges in the system chip realization process is the integration and coordination of the on-chip test and diagnosis capabilities. Compared to the conventional chips, the system chip test requirements are far more complex than the PCB assembly test, which, for instance, consists of interconnect and pin toggling tests. The system chip test is a single composite test. This test is comprised of the individual internal tests for each core, the UDL test,

and the test of their interconnects. As discussed earlier, each individual core or UDL test may involve surrounding components. Certain peripheral constraints (e.g., safe mode, low power mode, bypass mode) are often required. This necessitates access and isolation modes. In addition to the test integration and interdependence issues, the system chip composite test requires adequate test scheduling. This is needed to meet a number of chip-level requirements, such as total test time, power dissipation, area overhead, and so on [Zorian 1993]. Also, test scheduling is necessary to run intracore and intercore tests in a certain order so as not to impact the initialization and final contents of individual cores. With the foregoing scheduling constraints, the schedule of the composite system chip test is created.

In addition to the foregoing differences between testing traditional chips and system chips, we have to note that system chips also have the typical testing challenges of the very deep submicron chips, such as defect/fault coverage, overall test cost, and time-to-market. The generic test architecture for cores in SOC consists of three structural elements, as depicted in Fig. 15.4.

1. *Test pattern source and sink.* The test pattern source generates the test stimuli for the embedded core, and the test pattern sink compares the resource(s) to the expected response(s).
2. *Test access mechanism.* The test access mechanism transports test patterns. It can be used for on-chip transport of test stimuli from a test pattern source to the core under test, and for transport of test responses from the core under test to a test pattern sink.
3. *Core test wrapper.* The core test wrapper forms the interface between the embedded core and the environment. It connects the terminals of the embedded core to the rest of the IC and to the test access mechanism.

All three elements can be implemented in various ways, such that an entire palette emerges of possible approaches for testing embedded cores. In subsequent sections we review the various alternatives and classify current approaches.

15.5.1 Source and Sink of Test Data

The test pattern source generates the test stimuli for the embedded core. The test pattern sink compares the response(s) to the response(s) expected. Test pattern source as well as sink can be implemented either off-chip by external automatic test equipment (ATE), on-chip by built-in self-test (or embedded ATE), or as a combination of

both, as we mentioned first in Chapter 1. Source and sink do not need to be of the same type; for example, the source of an embedded core can be implemented off-chip, while the sink of the same core is implemented on-chip. The choice for a certain type of source or sink is determined by (1) the type of circuitry in the core, (2) the type of predefined tests that come with the core, and (3) quality and cost considerations [Zorian 1998].

We distinguish three main types of circuitry used in system chips today: (1) logic, (2) memory, and (3) analog and mixed-signal. Simple cores consist of one circuitry type only; complex cores consist of multiple simple cores, possibly of different circuitry type. These three types of circuitry exhibit different defect behavior, and hence their tests are quite different in nature [Agrawal 1994]. This also requires different types of sources to generate the stimuli and sinks to compare the responses. ATE systems as well as BIST schemes for logic, memory, and analog are traditionally quite different. This diversity of circuit types may require a mix of off- and on-chip test applications.

The variety in types of core tests is much larger than the three circuitry types listed above. In addition, tests cannot only be classified by the type of circuit they test, but also by the type of measurement they require (voltage vs. current), by the way they are generated (functional vs. structural), by the amount of core-internal adaptation they require (scan vs. test points), and so on. All these classification bases were discussed in one or more of the earlier chapters. For on-chip testing, the test patterns for digital logic core are typically generated by an LFSR dedicated to a core or shared with other cores.

15.5.2 Test Access Mechanism

The test access mechanism (TAM) takes care of on-chip test pattern transport. It can be used (1) to transport test stimuli from the test pattern source to the core under test, and (2) to transport test responses from the core under test to the test pattern sink. The test access mechanism is by definition implemented on-chip. Although for one core the same type of test access mechanism is often used for both stimuli as well as response transportation, this is not required, and various combinations may coexist. The selection of the mechanism requires a trade-off between the bandwidth of test data transport, possible extra hardware, and test application time [Zorian 1998].

When implementing a core test access mechanism, we have the following options:

- A test access mechanism can either reuse existing functionality to transport test patterns or be formed by dedicated test access hardware.

- A test access mechanism can either go through other modules on the IC, including other cores, or pass around those other modules.
- One can either have an independent access mechanism per core or share an access mechanism with multiple cores.
- A test access mechanism can either be a plain signal transport medium or may contain intelligent test control functions.

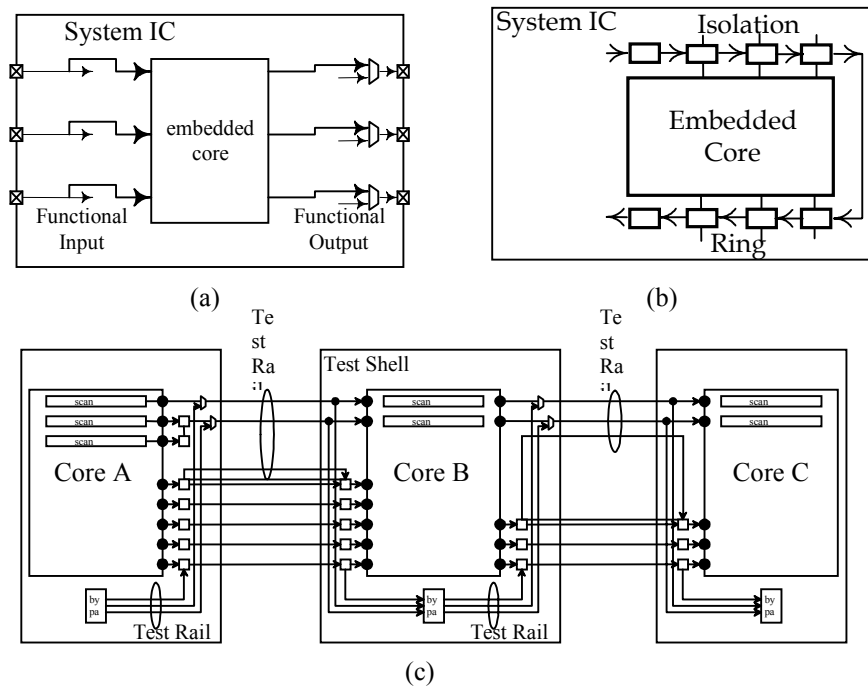


Figure 15.5 Test Access Mechanisms and Wrappers (a) Direct Access, (b) Isolation Ring, (c) TestShell and TestRail
The most straightforward TAM is the traditional direct access test bus shown in Fig. 15.5a. To access

individual cores, which are competing on the IC pins, a multiplexing system needs to be added to connect the TAM to the IC pins [Immaneni 1990, Monzel 1997]. Another well-known mechanism is Boundary-Scan [Whetsel 1997]. A variation of this serial technique is shown in Fig. 15.5b [Touba 1997]. The TestRail access mechanism combines the strengths of the test parallel TAM and serial Boundary-Scan test. It is shown in Fig. 15.5c in the context of a core wrapper, TestShell [Marinissen 1998]. Use of these mechanisms is illustrated in Section 15.6 after wrappers have been described.

15.5.3 Core Test Wrapper

The core wrapper is the interface between the embedded core and the rest of the chip. This wrapper should allow the following mandatory operational modes:

1. *Normal mode*: in which the wrapper is transparent to various interactions of the core with the other circuitry on the IC
2. *Internal core test mode*: in which the TAM is connected to the core for the transport of testing data
3. *External test mode*: in which the TAM supplies the test data to test UDLs and interconnections between cores on the SOC

Other optional modes may be used, such as a bypass mode. It is possible to combine several modes. The core wrapper connects the core I/Os to the test access mechanism. Whereas the I/Os of the core are determined by its functionality, the width of the access mechanism is a function of the bandwidth of the test sink and source and the amount of chip area available. There are probably more I/Os than the access mechanism can accommodate and some space compaction will be required in test application mode. The latter case would require a serial-to-parallel conversion of the core input signals and parallel-to-serial conversion of the output signal. This conversion function may be implemented in the core test wrapper or the TAM.

Examples of core test wrappers are the *test collar* [Varma 1997] and the *TestShell* [Marinissen 1998]. The first example is illustrated in Fig. 15.5b, where each I/O of the core is attached to a flip-flop in a fashion similar to Boundary-Scan Cells (BSCs). In this example the TestShell is shown in Fig. 15.5c where TestRail connects the I/Os of the various cores. The TestRail is the TAM and its width is three. The length of the TestShell varies according to the number of elements in each scan chain, but the TestRail is of constant width. For example, core A has three scan chains; one chain is connected directly to the TestRail, but the other two had to be multiplexed to the third wire in the TestRail.

15.6 Testing Strategies

In the remainder of the chapter we describe several approaches that address one or more aspects of SOC's testability. The examples are to demonstrate use of the SOC test concepts and DFT techniques presented throughout the book. In this respect they serve the same purpose that the examples of microprocessors did in Chapter 13. The examples mostly use DFT techniques that have been proven successful in traditional IC design. They utilize a mix of the test architecture components described in Section 15.5. We categorize them by one of the access mechanisms discussed earlier and presented in Fig. 15.5.

15.6.1 Direct Access Test Scheme

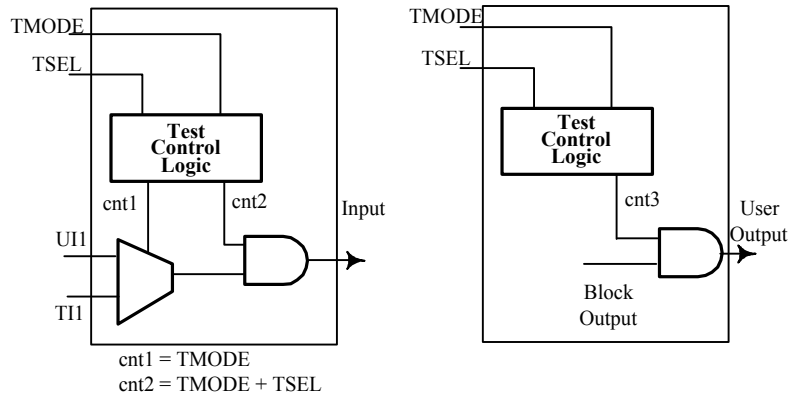


Figure 15.6 (a) Input and (b) Output Modifications [Immaneni 1990].

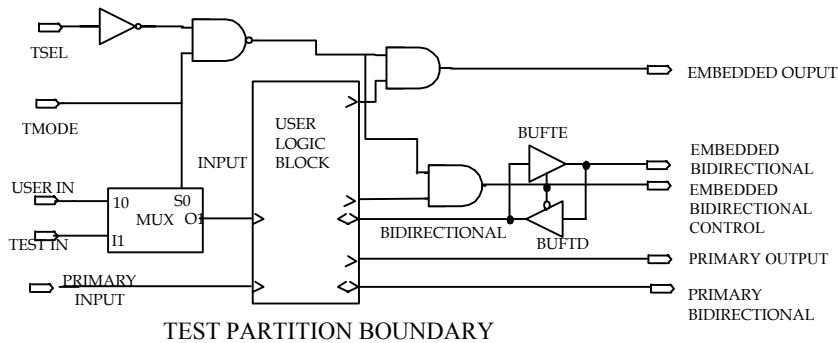


Figure 15.7 DATS UDL [Immaneni 1990].

The direct access test scheme (DATS) is symbolized by Fig. 15.5a. This is the approach taken to test ASIC design in which the Intel 80C186 and 80C51 are embedded as cores [Immaneni 1990]. Its intent is to make the core's inputs, outputs, and the bi-directional ports accessible outside the chip by mapping them onto the chip's pins. In this fashion, any embedded core can be isolated, simulated, and tested independently of the rest of the chip. Test vectors can then be generated to check the interconnections between the various components of the chip. The scheme requires the modification of the I/O ports that are not primary I/Os of the chip to be modified as illustrated in Fig. 15.6. TMODE and TSEL are two added control pins to the chip.

The circuit functions in normal mode whenever $TMODE = 0$. The module under test, be it a core or a UDL, is in test mode when its $TMODE = 1$ and $TSEL = 1$. Meanwhile, all the other modules are not active in testing and their $TSEL = 0$. A change in a user-defined logic is as illustrated in Fig. 15.7, where only the embedded I/Os are modified while those that are connected directly to the pins of the chip are "untouched." The core also needs to follow the

standard and, in addition, to be restructured to minimize the I/O modifications without jeopardizing it. We skip the modification made for these cores for the sake of showing how the scheme can be applied to the ASIC. In Fig. 15.8, two cores, block 1 and block 3, and a UDL, block 2, are shown. TMODE at PIN3 is distributed to all components, while the multiplexing on the I/O pins is made only whenever necessary. The advantage of this approach is in its simplicity and in testing the core as if it is the only circuit on the IC. The drawbacks are the competition with other cores on the limited pins of the IC. A large multiplexing circuitry is thus needed to accommodate all cores.

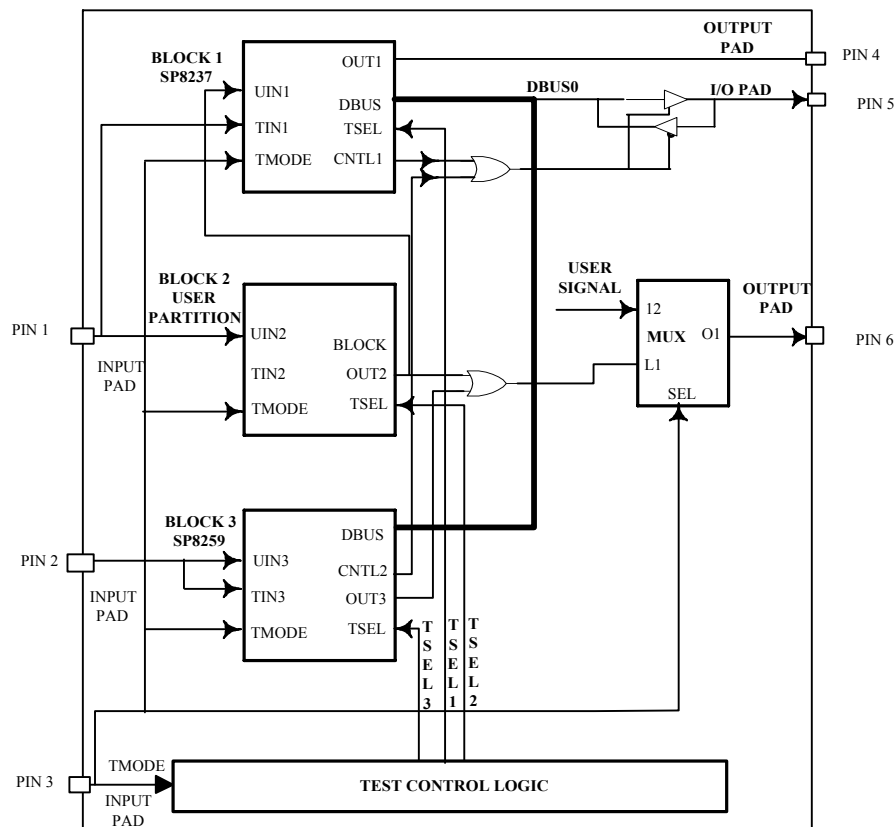


Figure 15.8 DATS Implementation Example [Immaneni 1990].

15.6.2 Use of Boundary-Scan

In Section 15.3, we likened the SOC to a PCB. As such, it is natural to apply the PCB testing approach, Boundary-Scan to test the SOC. In Chapter 10, we presented the major features of Boundary-Scan, IEEE Standard 1149.1. It has the advantage of facilitating the testing of any chip from the edge of the board. It has the flexibility to bypass a chip or integrate a DFT structure such as internal scan and BIST. Also, the availability of private instructions permits the customization of individual cores. A test access port (TAP) and its controller are used to

orchestrate the testing scenario. By extending this testing technique to the SOC, one would extrapolate that each core on the chip will have a TAP and its controller. Cores originally designed with Boundary Scan include a TAP. When embedding them in a system, it is not advisable to remove the TAP since it might be used to regulate other DFT features; An extra TAP can be used to control cores that do not adopt IEEE Standard 1149.1.

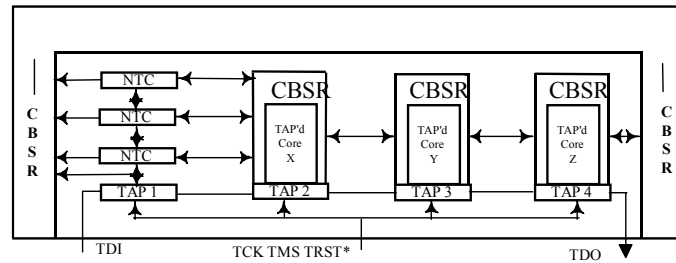


Figure 15.9 An Example of IC Containing Four TAPs [Whetsel 1997].

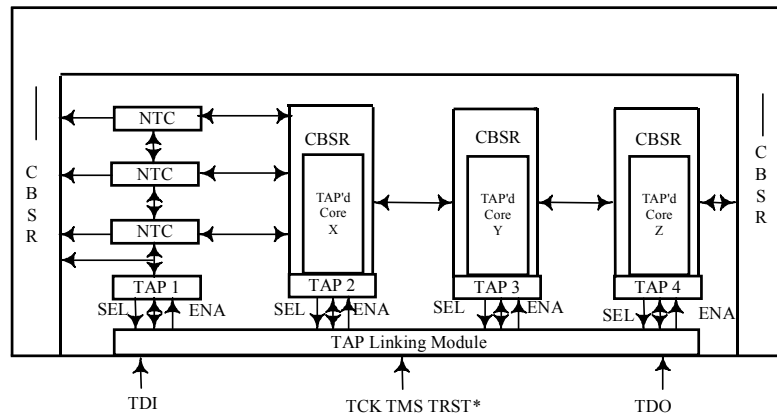


Figure 15.10 Design in Fig. 15.7 incorporating test access ports [Whetsel 1997].

An approach based on boundary scan for testing SOC, is illustrated in Fig. 15.9 and includes a mix of cores with a TAP controller (TAPed) and non-TAPed cores (NTC). TAP1 controls all NTC cores. The test data input (TDI) and test data output (TDO) are connected through all TAPs. This proposed arrangement does not comply with Standard 1149.1, which assumes one TAP per chip. It is important to make believe that all these TAPs are equivalent to one TAP. This is achieved by adding extra circuitry: the *TAP multiplexing link* (TML) and signals, as illustrated in Fig. 15.10. The control signals, SEL_i and ENA_i are to facilitate communication between the TML and any TAP_i. All TDO pins are connected to chip TDO through a multiplexing system. The TML enables and connects one or more TAPs to the accessed core via the IC's test pins. This arrangement allows testing of the core whose TAP is enabled. Similarly, the ENA signals can enable or disable the TAPs. These signals could also be included in the design of a TAP controller to force the TAP to go to or remain in the Run-Test-Idle state when disabled.

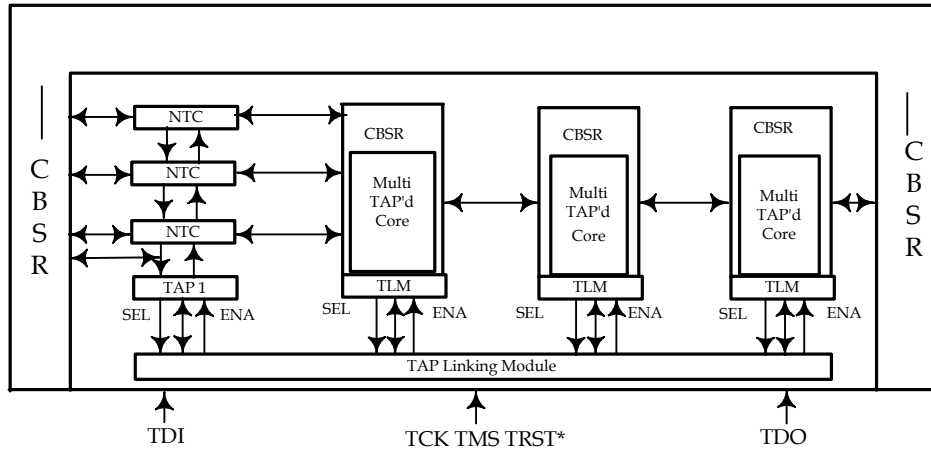


Figure 15.11 Reusable TLM Architecture [Whetsel 1997, Fig. 11].

As depicted in Fig. 15.2 and described earlier in the chapter, any core could itself be composed of other cores. Cores including TML controllers are embedded in a chip at their turns, as illustrated in Fig. 15.11. This is the equivalent of Fig. 5.10 with each TAP being replaced by a TML, except for the TAP used for NTC cores. Thus the proposed testing approach can be done on a hierarchy of cores.

The main advantage of this approach is that it is based on a mature, already proven technology. The scheme has merit when short test sets are used, as in the case of testing the interconnects. However, due to its serial nature, it has the disadvantage of a long test application time for long test sets.

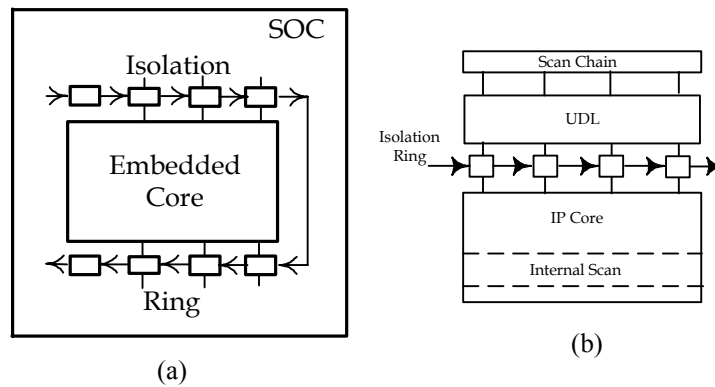


Figure 15.12 A wrapper (a) Isolation Ring (b) Test Access

A variation of the Boundary-Scan testing approach was recommended by [Touba 1997]. A flip-flop is placed at each I/O pin. All such flip-flops form an *isolation ring* around the core as a collar, as shown in Fig. 15.12a. This ring serves as a wrapper and provides full controllability and observability of the core as well as observability

for the UDL driving it. For example, if a UDL has full scan, the test is applied through the scan chain and observed on isolation rings around the core as illustrated in Fig. 15.12*b*. Also, the test to the core is supplied and the response is observed through the isolation ring. Instead of a full ring, it is possible to use only a partial ring and to modify the output space of the UDLs in such a way as to allow justification of the test vectors on the neighboring core [Pouya 1997]. Some of the inputs include flip-flops (IR) and some do not (NIR). The IR set is included in the ring and the others are not. To observe the outputs of the UDL, each output driving an NIR pin is XORed with an output driving an IR pin, and the resulting signal is fed to the partial ring. This amounts to the space compaction discussed in Chapter 12. As the test set is applied to the core through a UDL module, there is no guarantee that the patterns arriving at the core will detect all faults. Thus selection of input, IR, to be included in the ring has to be done with care to guarantee high fault coverage. The selection scheme of the IRs depends on the interfacing modules.

15.6.3 Use of Scan Path

The intent of the scan-path approach is to make cores other than the one under test transparent and to use them to transfer test data in and out of the chip. This approach is based on using the scan chains of the cores as a (TAM). It assumes that all cores include scan path and will make a minimal change to the I/O of the tested modules, cores, or UDLs. Such an assumption is realistic since scan path design is now widely practiced. Typically in the cases of soft and firm cores system designers include the scan path, whereas in hard cores a scan path is already included. The scan-path chains of all the cores are then used to justify (1) test vectors to an internal component and, (2) test responses to the chip's outputs.

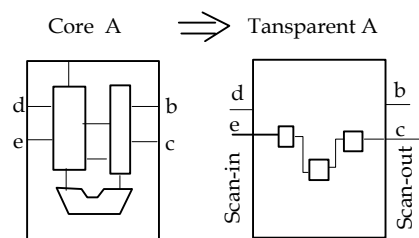


Figure 15.13 Transparent model.

The scan chains of the various modules of the system are used to regulate the test data traffic in and out of the chip. For this each module can be made transparent by representing it with its scan chain only as shown in Fig. 15.13. We call this representation the *transparent model* of the module. A model is transparent "in the sense that test data can be propagated through them without information loss" [Murray 1996]. Also, the system as a whole may

include feedback, and this makes testing more difficult. Thus it is important to look at the circuit as a whole and assure that no feedback is present in testing mode. After determining the scan path for all cores, the S-graph for all UDL logic is formed and is merged with those of the cores. The combined S-graph can then be searched for feedback loops and the smallest number of them is open to make the entire system acyclic. This will result in different scan chains through the chip that need to be connected in a *judicial* way to allow the application of test patterns and observation of the responses from any component on the chip.

The scan chains of the individual cores can be connected in one of the two principal schemes. In the first scheme, all scan-in (and scan-out) of the individual cores is connected to an IC pin through a multiplexer network. In the other scheme, all scan chains are connected in a daisychain [Aerts 1998]. Both schemes are illustrated in Fig. 15.14. The first scheme is actually equivalent to a direct access mechanism, and the cores will be tested one at a time. The test application time in the second scheme is as long as the total number of scan chains. To reduce this time, a bypass flip-flop is placed across each chain. The output of the scan chain and of the bypass flip-flop are multiplexed to control whether the core is working in bypass or scan mode. This arrangement gives various alternatives to testing the cores. For example, it is possible to test only one core and bypass the others, or viceversa, to bypass only one and test the others. Another alternative is to test all cores concurrently and as soon as one core runs out of patterns, put it in bypass mode [Aerts 1998].

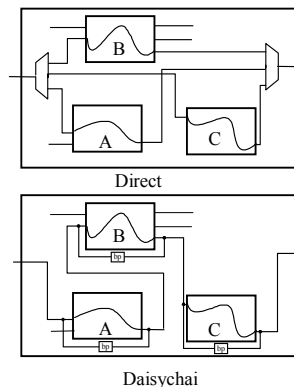


Figure 15.14 Scan-Path as Test Access Mechanism

Instead of using traditional scan testing as presented above, we can reduce the dependency on the external ATE bandwidth by using an on-chip test pattern generation as test data source and/or a space compaction of the cores' output signals in the data sink. The approach is illustrated with the example shown in Fig. 15.15. We will examine the testability of components *B* and *U*. We replace all the other modules by their transparent images, the

scan chains. First, we describe the overall approach to testing the UDL and the core it drives, then we elaborate on details of performing the testing and the required extra hardware [Mourad 1999].

- *Testing U.* A test applied on U can be observable through the scan chain of B and then through C to the primary outputs of the chip
- *Testing B.* The test for B is applied through U , then its response is shifted through C .

There are two main issues to consider next: observability of the test response at the outputs of U that are not connected to scan-in input of core B , and the justification of the test to core B at all of its inputs. Solving these problems is, however, dependent on the type of testing approach of these modules. We consider two alternatives with different merits (advantages and disadvantages that we need to balance). The first approach uses space compaction of the outputs; the second is based on scan-BIST.

15.6.3.1 Space Compaction. Space compaction schemes have been discussed in Chapter Twelve in conjunction with BIST testing. The situation in BIST involved too many outputs than can be accommodated in the signature analyzer. For this, the parity of some, if not all, the outputs will replace the actual output. Such a compaction for UDL U is shown in Fig. 15.15a. Any input combination at the decoder control lines will result in a test pattern for B . Therefore, the signals to the control lines of the weighted decoder are, in effect, the patterns to the core under test (CUT). They are supplied through the scan chain. Typically, the number of control lines is, $K = \lceil \log_2(M) \rceil \ll M$. Consequently, this scheme decreases the external bandwidth by the ratio M/K . In addition, the overall area cost of this technique is much lower than that of a full collar wrapper [Mourad 1999].

15.6.3.2 Scan-BIST. Let us assume this time that BIST is used to test the CUT and a multiple input signature register is used to compress the response to a PR test set. The output of the MISR can then be propagated to a primary output of the chip through the transparent images of the cores. Again, a MUX is used to select between normal operation and test as illustrated in Fig. 15.15c. The same MISR can be reused to test other cores on the chip. A comprehensive BIST schedule and control method has been developed to deal with tradeoffs for BIST power dissipation, test time optimization, and BIST resource sharing [Zorian 1993]. This universal BIST scheduler, is either customized to execute a predetermined BIST schedule for a manufacturing test or is programmed on the fly through the JTAG port to execute the test of individual cores during silicon debug and diagnosis.

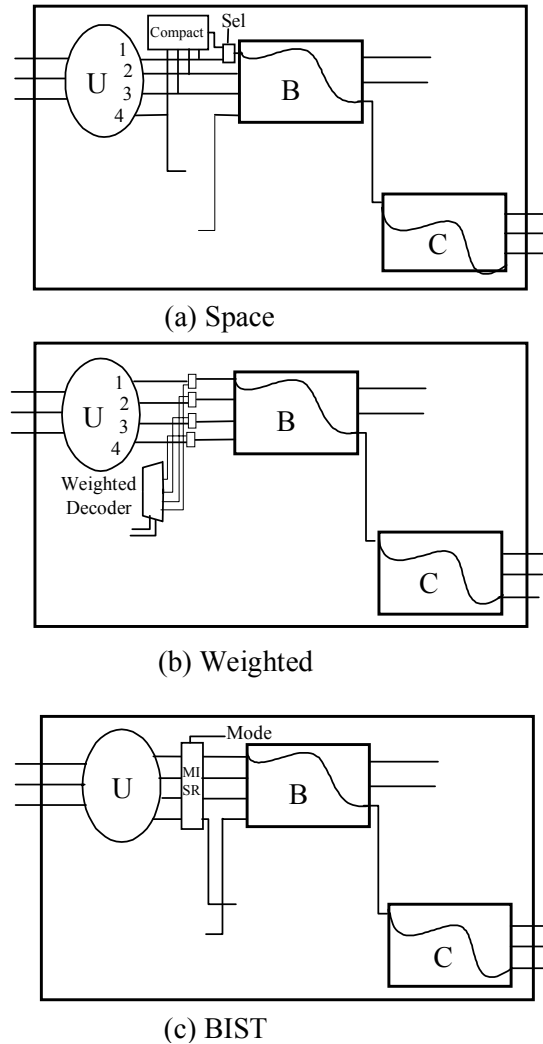


Figure 15.15 Reducing the External Bandwidth

15.7 To Explore Further

In this chapter we have described a state-of-the-art SOC design and test. The entire area needs to be “explored further.” However, we recommend that the reader follow up closely with the progress of the standards mentioned earlier: VSIA and P1500.

15.7.1 Virtual Socket Interface Alliance

The virtual socket interface (VSI) alliance consists mainly of representatives from the semiconductor industry companies [VSI 1998]. The organization’s task is to identify and define interface standards for design reuse of

virtual components. Seven development working groups (DWGs) have been formed to study and recommend standards for various aspects of the design and test integration, such as IP protection, implementation verification, on-chip bus, and manufacturing-related test. The scope of the latter DWG include [Zorian 1997]:

- Test access
- Test isolation
- Interconnect test
- Shadow logic test
- Test integration
- Test logic integration
- Failure identification

Among the expected deliverables by the DWGs are:

- Test data interchange specification: Test vector formats (e.g. STILL), test structures, and interface specifications
- Guidelines for system chip integrators: Specification of system chip test architecture and test guidelines, testing of UDLs and interconnects, system-level tests, test integration and debug
- Guidelines for VC providers: Guidelines on test methods, DFT, test isolation, test access, and debug
- Guidelines for mergeable IP providers: Guidelines on test methods and DFT rules.

Whenever possible, the VSI Alliance tries to endorse existing or emerging standards. In the scope of the manufacturing-related test DWG, these are IEEE P1500 [IEEE 1999b] and IEEE P1450 which is better known as STIL (standard test interface language) [Matson 1997, Wohl 1997, IEEE 1999a].

15.7.2 IEEE P1500 Standard

IEEE P1500 is a standard focusing on the critical aspects of ease of reuse and interoperability with respect to testing of cores originating from distinct core providers. This is done by standardizing core test knowledge transfer and test access for embedded cores.

In September 1995, the Test Technology Technical Council (TTTC) of IEEE Computer society initiated a Technical Activity Committee on embedded-core test, in order to identify common needs in this domain. After several meetings in conjunction with major conferences, it has concluded that there were indeed common needs and hence a design for standardization [Zorian 1997]. In June 1997, the IEEE Standard Board granted permission to start a standard activity and this was then the official start of IEEE P1500.

In order to improve efficiency of both core providers and core users, the IEEE P1500 standard facilitates the

interpolarity with respect to testing when cores of various providers come together in one SOC, but it does not cover the core's internal test methods or DFT, nor SOC test integration and optimization. IEEE P1500 focuses on standardizing those areas that are at the interface between core provider and core user. Through a scalable standard, IEEE P1500 contributes to ease plug-n-play for testing, while maintaining the required flexibility to cope with different cores and system chips.

IEEE P1500 has active participation from leading experts in relevant industry segments, such as systems companies, Electronic Design Automation (EDA) vendors, core providers, IC manufacturers, and ATE suppliers. Several Task Forces carry out the detailed technical work of P1500. The standard draft document is posted on the P1500 Web site [IEEE 1999b], where one can also find the minutes and presentation of Working Group meetings.

The two main elements of the IEEE P1500 standard are a language, called CORE Test Language (CTL) and a scalable core test architecture. The first is meant to standardize the core test knowledge transfer. CTL is based on another IEEE standard language, viz., IEEE Std, 1450.0, the Standard Test Interface Language (STIL) [IEEE 1999a], which is extended to accommodate specific core test constructs. IEEE P1500 only standardizes the test wrapper around the core and its interface to one or more TAMs. The test wrapper is comprised of the following basic components: a Wrapper, Boundary Register, a Wrapper Instruction Register and a ByPass Register

IEEE P1500 is currently in the development phase of the standard. The first version of the standard focuses on non-merged digital logic and memory cores. It is planned to cover analog and mixed-signal cores, as well as the DFT guidelines for mergeable cores, in future expansions. Because the standardization work is not finalized at this point of time, we do not include its details architectural details and the language specifics, but we recommend you to further explore the standard either by contacting TTTC web site, (<http://computer.org/tab/ttc>) or by visiting the P1500 web site (<http://grouper.ieee.org/groups/1500>).

References

- Aerts, J. and E. J. Marinissen (1998), Scan chain design for test time reduction in core-based ICs, *Proc. IEEE International Test Conference*, pp. 448 – 457.
- Agrawal, V. et al. (1994), Built-in self-test for digital integrated circuits, *AT&T Tech. J.*, Vol. 73, No. 2, p. 30.
- Beenkar F., B. Bennetts, L. Thijssen (1995), Testability concepts for digital ICs - macro test approach, Kluwer Academic, Norwell, MA.
- Gupta, R. K. and Y. Zorian (1997), Introduction to core - based system design, *IEEE Des. Test Comput.*, Vol. 14, No. 4, pp. 15 – 25.

Hunt, M. and J. A. Rowson (1996), Blocking in a system on a chip, *IEEE Spectrum*, Vol. 36, No. 11, pp. 35 – 41.

IEEE (1999a), P1450 Web site <http://grouper.ieee.org/groups/1450/>.

IEEE (1999b), P1500 Web site <http://grouper.ieee.org/groups/1500/>.

Immaneni, V. and S. Raman (1990), Direct access test scheme: design of block and core cells for embedded ASICs, *Proc. IEEE International Test Conference*, pp. 488 – 492.

Marinissen, E. J. et al. (1998), A structured and scalable mechanism for test access to embedded reusable cores, *Proc. IEEE International Test Conference*, pp. 448 – 457.

Matson G.,(1997), Structuring STIL FOR scan test generation based on STIL, *Proc. IEEE International Test Conference*.

Monzel J. (1997), “Low cost testing of system-on-chip, *Proc. Digest of 1st IEEE International TECS*, Nov.

Mourad, S., and B. Greene (1999), Scan-Path based testing of system on a chip, *Proc. IEEE International Conference of Electronics, Circuits and Systems*, pp. 1081 – 1084, Cyprus.

Murray B. T., and J. P. Hayes (1996), Testing ICs: Getting to the core of the problem, *IEEE Computer*, Vol. 29, No. 11, pp. 32 – 38.

Pouya, B., and N. A. Touba (1997), Modifying user-defined logic test access to embedded cores, *Proc. IEEE International Test Conference*, pp. 60 – 67.

Ricon, A. M. et al. (1997), Core design and system on a chip integration, *IEEE Des. Test Comput.*, Vol. 14, No. 4., pp. 26 – 35.

Smith G. (1997), Test and system level integration, *IEEE Des. Test Comput.*, Vol. 14, No. 4.

Touba, N. A., B. Pouya (1997), Testing embedded cores using partial isolation rings, *Proc. 15th IEEE VLSI Test Symposium*, April, pp. 10 – 15. Also, *IEEE Des. Test Comput.*, Vol. 4, No. 4, pp.52 – 58.

Varma, P. and S. Bhatia (1997), A structured test reuse methodology for core-based system chip, *Proc. IEEE International Test Conference*, pp. 294 – 302.

VSI (1998), VSI Alliance Web site <http://www.vsi.org/>.

Whetsel, L. (1997), An IEEE 1149.1 based test access architecture for ICs with embedded IP cores, *Proc. IEEE International Test Conference*, pp. 488 – 491.

Wohl, P., V. T. Williston and J. Waicukauski (1997), A unified interface for scan test generation based on STIL, *Proc. IEEE International Test Conference*, pp. 1011 – 1019.

Zorian, Y. (1993), A distributed BIST control scheme for complex VLSI devices, *Proc. 11th IEEE VLSI Test Symposium*, pp. 6 – 11.

Zorian, Y. (1997), Test requirements for embedded core-based systems and IEEE P-1500, *Proc. IEEE International Test Conference*, pp. 191 – 199.

Zorian, Y. et al. (1998), Testing embedded-core based system chips, *Proc. IEEE International Test Conference*, pp. 135 – 149.

Problems

- 15.1.** List the three main types of IP cores and compare the steps used in their design and testing.
- 15.2.** You are to design an SOC that includes three hard cores, one soft core, and some logic that you defined. Use the format of Fig. 14.2 to represent the synthesis and optimization steps needed to complete the design.
- 15.3.** The supply of the test patterns to the various cores on an SOC and the retrieval of the response to these patterns are two of the bottlenecks in testing these ICs. What would you recommend to alleviate this problem?
- 15.4.** What test strategy would you suggest for a SOC that uses only hard cores and some user-defined logic?
- 15.5.** SOC is likened to PCB. Is boundary-scan appropriate to use for SOCs? List advantages and disadvantages.
- 15.6.** What is meant by reuse in the context of SOC? How can test reuse facilitate testing of SOCs?