Model Generation of Test Logic for Macrocell Based Designs*

E. de la Torre, J. Calvo, J. Uceda
Universidad Politécnica de Madrid
ETSI Industriales de Madrid
c/ José Gutiérrez Abascal 2, 28006 Madrid
Phone : 34-1-4117517, Fax: 34-1-5645966, e-mail: eduardo@upmdie.upm.es

Abstract

This paper presents a set of tools for generation, simulation, evaluation and synthesis of VHDL models of test logic for macrocell based embedded microcontroller based systems. The generated models are described at behavioural level so they fit with system descriptions suited for fast simulation. An IEEE 1149.1 Boundary Scan implementation is used, providing manufacturing test, on line test and monitoring capabilities.

1. Introduction

Higher time-to-market needs for custom embedded control applications are demanding tools for fast and efficient system design and prototyping. The use of high level description languages, specially VHDL[1], and the possibility of simulating the system at the very early stages of the design phases have been of the major benefit for the system designer. In this way, virtual system prototyping using fast simulation models allows the designer to efficiently tune system performance, complexity and other system parameters, given an initial, and very often changing, set of system requirements.

At system level, several design specification stages can be identified (see figure 1). The top level layer is an abstract representation of the functionality, where hardware and software may not be identified. From there, a sequence of HW/SW partitioning, HW module partitioning, scheduling and allocation operations will decide a system architecture. This architectural definition level represents the behaviour of the selected architecture, on which the functionality of the system can be defined, for digital systems, with the precision of a clock cycle, and where the interfaces among different subsystems may be completely defined. After this, RTL and logic synthesis will lead to the detailed design phase that opens the physical design stage.

The architectural level is important for emulated prototyping or virtual prototyping, as many issues like architecture performance estimations, module connectivity checks, software programming and debugging, etc. can be done with the use of such behavioural architectural model [6]. The advantage of this model is that it provides much better simulation performance than lower level structural simulators. The purpose of the test environment here presented is to raise the test design phase up to this specification level.

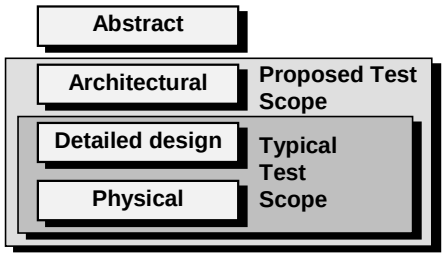

Figure 1. Specification stages and test scope

One of the ways to speed up the design process at the architectural definition level is not to design the system starting from scratch, but to use large pre-built blocks with a given complete functionality called macrocells [2]. Throughout this introduction section, a discussion of the characteristics of macrocell based designs will be done, as well as the testing problems for these type of circuits from the system designer point of view. Section 2 will show the basic needs to solve these problems. These needs will be taken as requirements for the test design environment presented in Section 3. Section 4 shows the characteristics of the generated VHDL test models. After this, application results and conclusions will be shown.

A. Macrocell Based Designs

A macrocell, as defined in the OMI (the Open Microprocessor systems Initiative European program), is a medium to very high complexity block with a given functionality, known interconnect interfaces (if possible, following a standard or a common definition) and different definition levels or properties definitions called *views* (for example, architectural, RTL, layout, test, etc.).

With a macrocell approach, the task of the system

* This work has been partially funded by ECU (ESPRIT #8235)

designer consists on connecting several macrocells so that system requirements are fulfilled. However, this basic task is not as simple as it seems, as several problems appear:

- The large granularity of the basic construction blocks make difficult to achieve the precise requirements matching with minimum costs. This problem, however, may be solved with highly configurable and parametrisable macrocell libraries: this is a point where more emphasis should be put by macrocell designers.
- The partition between hardware and software has the same granularity problem. However, given a microprocessor core and a selection of the I/O devices given by the system requirements, the target system can be programmed on a virtual prototype and performance may be checked very early in the design process.
- System simulations may have performance problems if no simulation oriented models are available. For example, regular block generators, like classic RAM, ROM or PLA compilers produce simulation models based on structural information that cause large computing overheads at system level simulations.
- The use of vendor-provided macrocells may cause problems in specific design stages if the set of views is not complete. For example, the lack of a test view for a complex macrocell could collapse the test pattern generation phase, specially if no structural information is available for the cell.

For the specific case of embedded control systems, the macrocell concept requires an additional clarification: with the exception of the microprocessor core blocks (integer unit, cache memories, etc.), where block sizes are large, the rest of functions, basically I/O devices, are obtained by hooking small to medium size macrocells. Therefore, as we are dealing with low end macrocells sizes, the problem is that the number of gates required for a given DFT implementation on or around a given macrocell may be too large compared to the macrocell size itself. Therefore, this fact will restrict the use of scan chains around all macrocells because of large DFT area overheads.

Other problems, taken apart of the previous discussion because they are the motivation of the work here presented, are the testing difficulties:

B. The test problem

System designers usually consider the test phase as something that has to be done much later, and by someone else. However, complex designs require test planning at the system level, specially if we consider the test aspects of a system from a wide perspective, and take into account not only the manufacturing test issue, but on-line testing and system monitoring as well.

An additional inconvenience in large systems, is the near to exponential increment of test costs with respect to system size. High complexity integrated circuits and systems require well studied and finely-tuned test strategies. On the other side, on-line detection of hardware and software errors is a task that may also require a large amount of system resources, measured in terms of extra silicon area, dedicated operating system support, etc.

The test views of a macrocell should include manufacturing test vectors, whether the cell uses structured DFT techniques, functional test vectors, and/or BIST. Some methods have been proposed, based on the Boundary Scan standard IEEE 1149.1[3,4], as standards for manufacturing test in macrocell based designs [5]. The aim of this method is to apply the test vectors of every macrocell test view serially through IEEE 1149.1 compliant scan registers that surround the macrocells. This solution is, however, very expensive in terms of additional area if the macrocells have small sizes, as it happens in the case of embedded systems.

On the other side, existing microprocessor testing solutions [7 to 17] show that the way to test a microprocessor or microcontroller chip is not the application of a single method, but the selection of several methods, each of them valid for a different submodule. Among these ones, scan path techniques, connection of macroblocks I/O signals to the primary chip I/O pins in test mode, functional testing, BIST and boundary scan (BS) implementations are the methods most frequently used. It is also important to note that since the appearance of the boundary scan standard, many microprocessors incorporate more or less sophisticated versions of the standard, from which other test features rely upon.

Testing has always been, in general, in the middle of a cost trade-off in between the test design costs, including DFT, and test application costs. In addition, test functionality is seldom considered a basic system requirement, and it is difficult to justify efforts in test design and DFT. So, an important issue is to determine the benefits of a set of selected testing strategies against the implantation costs, and this includes the determination of these costs at the moment of deciding the strategies to follow.

2. Test Design Environment Requirements

From the test problems previously identified, it comes out the need of an environment or a set of tools that offer the system designer the following basic capabilities:

- Generation of behavioural simulation-oriented models of test logic. The aim is to produce a model that does not penalise simulation time when the test subsystem is installed in the complete system model.
- The model generator should also produce high level test simulation functions, so that the designer can easily work with the test logic without need of knowing the low level details, like how to assert the boundary scan required signals, how internal registers and subregisters

are arranged, how to manage the TAP controller, etc.

- Tools for test vector serialisation and assembly are required. The simulation environment should provide the designer with a way to interact dynamically with the test subsystem in simulation time.
- Synthesis oriented VHDL models are required to easily move into later design stages.
- Estimation of test costs for a given solution, by inferring the DFT area (without need of synthesis) and test application time.

In this sense, high level system prototyping is an advantage if test cost estimations can be done from high level specifications. The possibility of automatically generating the model of the test subsystem integrated with the rest of the system, together with the capability of estimating test application cost (in terms of required test vectors) and also the estimation of increment of area caused by DFT are of great benefit for the overall test problem within the general system design flow.

The the test model and the proposed test environment simplify the steps of the estimation cycle. Figure 2 shows how the test cost optimisation cycle is improved with the test environment proposed in this paper.

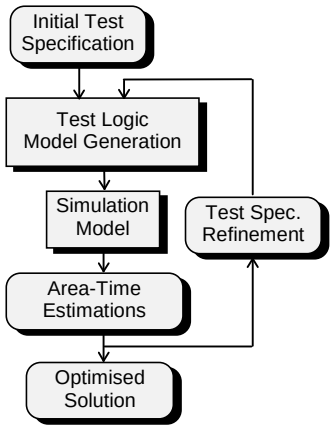

Figure 2. Test strategy refinement loop

The missing point in the previous figure is how to select a new test specification to produce a minimisation of test costs. It can be done with specific tools, like a knowledge based tool, or through a dedicated cost-oriented study. This decision making tool is out of the scope of this paper.

3. Proposed Test Environment

The general aim of virtual prototyping is to provide the system designer, including hardware and software designers, with a combination of tools and system models that are useful for the embedded system design, simulation and verification stages. In the same way, the test subsystem model and the tools proposed within the test environment aim to simplify the design and verification tasks for the issues related with the test subsystem starting at the architectural level.

Figure 3 shows a block diagram of the proposed test environment, where the different tools, models and data information dependencies can be seen.

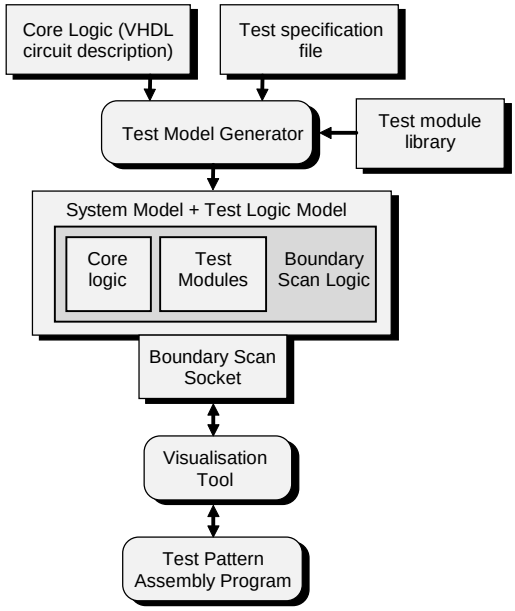

Figure 3. Test environment module diagram

The *core logic* contains the VHDL models of the original system. This description can either be at the integrated circuit level or the printed circuit board (system) level. In the first case, the core logic description is represented by a structural VHDL file showing how the internal macrocells are interconnected. In the second case, the PCB description is also defined by a structural description showing the instantiations of the different integrated circuits and their interconnection lines.

The characteristics of the test strategies to use in the system are indicated in the *test specification (TS) file*. This file contains, for each circuit, the definition of the test functions to implement in the test subsystem, the organisation of BS registers and subregisters, the standard and user-specific BS instructions to use, the instantiation of additional test modules that are to be included in the integrated circuit together with the core logic, etc.

Both core logic and TS file are the input data to the Test Model Generation tool (TMG). The output of the TMG is another VHDL that integrates the original system and all the test logic as specified in the TS file.

The test modules to integrate in the circuit are taken by the TMG from a library of configurable test blocks (signature analysers, watchdog controllers, bus masters/monitors, bus range checkers, etc.). The insertion of a test block must be specified in the TS file, and the TMG will instantiate this module into the TSM, add the required BS instructions for the correct addressing of

module registers and perform the adequate signal connections with both core logic and BS logic.

DFT area estimations are produced by the TMG tool, also. These estimations are produced from a set of technology dependent parameters that are obtained from the synthesis of some simple predefined modules, optimised for the selected target technology. Technology parameters are automatically obtained from a script that synthesises these modules, and it has to be done once for every target technology.

The application of test vectors through the standard IEEE 1149.1 test access port is performed through the Boundary Scan Socket module. This module is in charge of applying and receiving data through the BS signals. The models are prepared for two different simulation environments: the first one, suited for chip manufacturing test validations, is a full VHDL testbench template that incorporates functions and procedures to allow operation at higher level than basic boundary scan signal handling. The second simulation environment, more suited for board monitoring and on-line testing activity, uses the C language interface of the VHDL simulator to connect with the visualisation tool and the test pattern assembly programs.

The generation of the value sequences that are fed to the system through the BS may be created from high level test functions using the Test Pattern Assembly Program (TPAP). The TPAP may be used to perform functional simulation for the integrated circuit or for the entire board models. It incorporates algorithms for generation of high-level test functions, suchs as PCB interconnection testing, cluster testing and BS consistency check.

Simulation results might be inspected using the waveform viewer of the VHDL simulator. However, the interpretation of such results in the case of long test vector sequences, as in the case of the BS logic, is a very hard issue. The visualisation tool (VT) shows the results obtained from the simulation not only as the resulting sequence of test responses but as the result of the application of the high level test functions. The visualisation tool also lets the user to select the test programs to apply with a script file. User driven commands given through the visualisation tool may be given independently on the execution of the system testbench, allowing on-line monitoring emulation.

4. The Test Model

The model can be generated and configured from a test specification file. This means that the addition of BS logic and specific test modules, interconnection modifications and high level block parametrisations are automatically performed with the information provided in the TS file.

One of the characteristics of boundary scan testing is the huge length of serial test data required to load and download the boundary registers or any other serial register connected between TDI and TDO. This means that a lot of test clock cycles have to be simulated and therefore a significant amount of time is used. To solve this, the test model provides a way to perform data load/unload independent of the system clock(s) and of the test clock, so that serial data is available at the appropriate register elements with low simulation effort. However, the number of required test cycles is evaluated so that an estimation of the total and partial test lengths can be performed for a given test plan.

For other purposes, the model also includes the capability of test cycle by test cycle simulation. This allows, for example, signature calculations for blocks with unpredicted self-test signature values, precise verification of correct test vector application, all type of on-line operation, etc.

Subcycle timing is not checked, although logic delay effects have been considered to minimise timing penalties.

In some circumstances, the operation of BS with the integrated circuit plugged together with other circuits on the PCB may produce incorrect operation or damage to the circuitry because signal driving directions may change with respect to normal operation mode. The robustness of the IEEE 1149.1 standard is in the direction of providing the necessary HW architectural mechanisms to avoid that possibility. Incorrect operation may, however, produce undesired results. In this sense, the virtual model provides information about possible incorrect or hazardous operation during the test mode. It detects effects like the existence of multiple drivers in a signal, leaving inputs in high impedance, etc.

The TSM is also a gateway to the final logic implementation on the chip and PCB: the model generation tool produces a fully synthesisable VHDL model of the test logic.

There are three different test models, suited for different uses:

a) *High level specification model at chip level* (see Figure 4). This model includes the instantiation of the required BS register logic and other user defined internal registers, the generation of the control logic required for operation of the specified BS instructions, the instantiation of additional library test modules and their interconnection with the core logic and the BS control ports.

The testbench instantiates the resulting block with all the added circuitry, the BS socket and either the visualisation tool interface or the VHDL testbench profile, as required by the user.

The application scope of this module includes the validation of the internal module interconnection, the verification of the BS instruction level routines, the validation of the manufacturing test routines that require the aid of the incorporated BS logic (like direct off-chip module connection, etc), execution of BIST structures (if

any) controlled from the BS port, and the validation of the correct operation of the modules that are instantiated from the test module library.

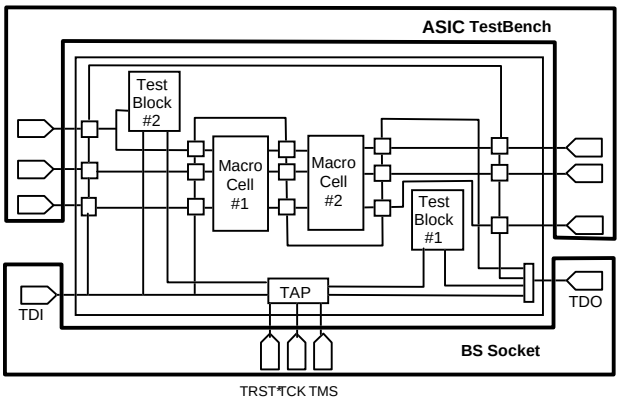

Figure 4. Asic test Model

b) *High or low level integrated circuit models connected at PCB* (see Figure 5). The model at this stage includes the instantiation of the integrated circuit models as in the first point, and the interconnection at PCB level of the BS logic between several circuits. The testbench also includes the BS socket to connect to the visualisation tool and the test assembly program.

The application scope of this model includes the validation of the global BS activity at either PCB level or system level, that is: simulation of the complete system, with the capability of observing the PCB activity together with the BS activity. This allows, among others, the validation of the algorithms developed for PCB manufacturing test and simulation and validation of the system monitoring strategies and on-line self test schemes.

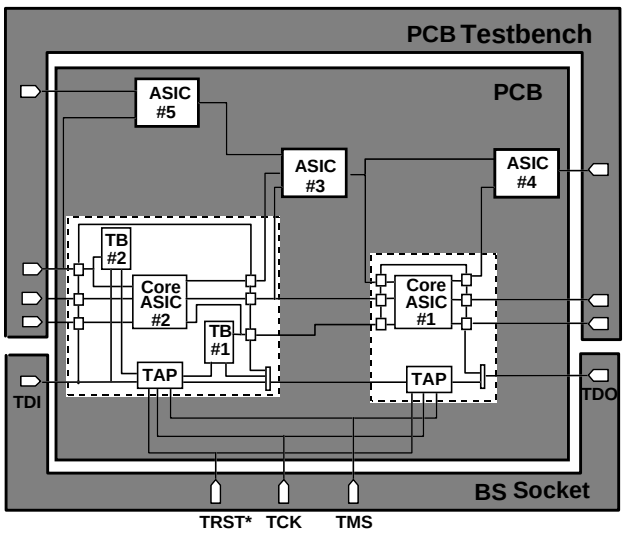

Figure 5. PCB model

c) *Low level description of the complete integrated circuit.* The low level description is a fully synthesisable description of the whole chip logic, and thus, this stage is oriented to the estimation of DFT area and to the synthesis of the incorporated test blocks.

There is no benchmark generated for this block, as this function is included in the first model type.

5. Application Results

The test environment has been developed within the ECU project (OMI Initiative), on which a target embedded control application is implemented based on a custom Sparc™ board. The main microcontroller chip integrates a Sparc™ compliant integer unit, instruction and data caches, a PI-bus controller and several I/O devices. The behavioural system model is integrated with development system tools so that the board model can be attached to a general use software debugger (gcc) and to a high level hardware observer.

The test environment has been used with this target board at chip and PCB levels. At chip level, different features have been implemented and tested functionally:

- Boundary scan compliant logic in the integrated circuit periphery.
- Subregisters for control and observation of I/O ports.
- Internal scan register for the IU macrocell.
- Event counter/range checker for the internal instruction address bus.

There are presently under development more library test modules, such as an internal bus master/observer and a configurable watchdog controller.

In order to compare model synthesis results and accuracy of DFT area estimations, comparisons with Synopsys™ JTAG Compiler have been done. Several circuits with a variable number of boundary scan cells have been tested. The test models implemented for this benchmark are simple implementations of the required BS logic to minimally satisfy the IEEE 1149.1 standard (mandatory instructions, boundary, bypass and instruction registers). Table 1 shows DFT area estimations, synthesis area results for the models generated with our tool, and area results with Synopsys™ JTAG Compiler. The first percentage column shows deviations of the area estimations with respect to the synthesised area result of our model. The second one shows the differences between our model and the Synopsys™ model. Every synthesis area result shows two numbers: the first one takes only into account area for logic cells; the second one includes logic cell area and interconnection area. The test models generated with the TMG have been synthesised with Synopsys™ Design Compiler. Area units are given in mils. The largest circuit, coreECU, is the microprocessor core model developed in the ECU project.

It can be observed in the table that area estimation deviations are less than 3 percent for large circuits, and

that the model generated with the TMG is slightly larger than with Synopsys™ JTAG Compiler. In fact, our model is always around 26 mils larger than the Synopsys™ model, no matter the number of cells. Therefore, area differences are very small for large circuits.

Circuit	# BS Cells	Estim. area	TMG area	Diff (%)	Syn. JTAG Compiler	Diff (%)
FrqDiv	5	380.5	376.6	1,04	350.6	7.42
		928.7	894.7	1,80	785.4	13.91
BCD7seg	11	527.9	529.4	-0.28	503.4	3.95
		1242.6	1204.8	3.14	1095.5	7.95
Alarm	17	675.2	683.6	-1.22	657.6	3.95
		1556.5	1518.6	2.50	1409.3	7.75
RTClock	26	896.3	904.1	-0.87	879.6	2.79
		2027.3	1972.2	2.77	1864.9	5.78
Comp	27	920.8	928.7	-0.85	904.2	2.71
		2079.6	2022.7	2.81	1914.9	5.63
muxAlHr	40	1240.1	1252.3	-0.98	1226.3	2.12
		2759.8	2681.8	2.91	2572.5	4.25
coreECU	325	8239.7	8360.9	-1.45	8334.8	0.34
		17413.4	17166.6	1.44	16930.4	1.39

Table 1. Estimation and synthesis comparison results

Regarding simulation performance, the operation of the Sparc™ board model with the test model included (only for the main chip) is only 5 to 6 percent slower than the original system model simulations. This benchmark was done having the system emulating a C program execution and without activating the test logic. Test logic was fixed in the run-test/idle state, while having the test clock (TCK) at the same frequency of the system clock. This show that the penalty time for adding the test model is not very important. A test script running continuous sampling operation on the test logic produces an increment in simulation time around 16 percent.

The time needed for generating the test model from the original VHDL code and the test specification file is negligible.

6. Conclusions

A behavioural test logic model and a test environment for simulation, estimation and synthesis have been proposed. The main contribution of the work is to rise the test design phase at the system designer's level, providing an efficient method of implementing and simulating standard test logic in macrocell based systems.

Test design time is reduced and overlapped, allowing the test optimisation cycle (implement-estimate-simulate) at the high level system design phase.

Among the future trends, it is important to highlight that the proposed environment can be easily adapted to implement a cost-oriented automatic test plan selection tool. This would allow the implementation of highly optimised test designs.

The use of two standards, VHDL for the model, and Boundary Scan for the test access logic produces wide interoperability and usability.

Macrocell based designs can be a promising solution if they can be highly configurable and parametrisable, and all design views are available from the macrocell providers. The embedded control world is a good application field for macrocell designs.

References

[1] IEEE Standard VHDL Language Reference Manual (ANSI/IEEE Std 1076-1993). IEEE Computer Society. June 1994

[2] ELI macrocell standard. Open Microprocessor systems Initiative. XXX

[3] IEEE Standard Test Access Port and Boundary-Scan Architecture. (IEEE Std 1149.1-1990 and IEEE Std 1149.1a-1993). IEEE Computer Society. October, 1993

[4] The Boundary Scan Handbook. K.P. Parker. Prentice Hall. 1992

[5] OMI standard 307: Macrocell manufacturing test. Matra MHS. June, 1995

[6] J. Goicolea, R. Guzmán, M.A. Salas, S. Olcoz, D. Navarro, A. Roy, 'System Designer Approach to the Development of Embedded Systems Using VHDL', APCHDL, October 1994

[7] K. Holdbrook, S. Joshi, S. Mitra, J. Petolino, R. Raman, M. Wong, 'microSPARC™: A Case Study in Scan-based Debug', International Test Conference, 1994

[8] D. Bhavsar, J. Edmonson, 'Testability Strategy of an ALPHA AXP Microprocessor'. International Test Conference, 1994

[9] D. J. Mirizzi, W. Jerrels, D. Ohmart. 'Implementation of Parallelsite Test on an 8-Bit Configurable Microcontroller', International Test Conference. 1993

[10] D.D. Josepshson, D. J. Dixon, B. J. Arnold. 'Test Features of the HP PA7100LC Processor', International Test Conference . 1993

[11] R. Patel, K. Yarlagadda. 'Testability Features of the SuperSPARC Microprocessor'. International Test Conference. 1993

[12] H. T. Nagle, R. R. Fritzemeier, J. E. Van Well, M. G. McNamer. 'Microprocessor Testability'. IEEE Transactions on Industrial Electronics, May, 1989

[13] T. J. Powell, F. Hwang, B. Johnson. 'Testability features in the TMS370 family of microcomputers'. International Test Conference, 1988

[14] A. Crouch, M. Pressly, P. Laskso, J. Circello, 'Testability Features of the MC68060 Microprocessor', International Test Conference, 1994

[15] Hunter, K. Torku, J. LeBlanc, 'The PowerPC(tm) Microprocessor Test Methodology'. International Test Conference, 1994

[16] W. Harwood, M. McDermott, 'Testability features of the MC68332 modular microcontroller', International Test Conference, 1989

[17] J.A. Lyon, M. Gladden, E. Hartung, E. Hoang, K. Raghunathan, 'Testability features of the 68HC16Z1', International Test Conference 1991