Selected best papers from ETS'06

Wrapper design for the reuse of a bus, network-on-chip, or other functional interconnect as test access mechanism

A.M. Amory, K. Goossens, E.J. Marinissen, M. Lubaszewski and F. Moraes

Abstract: A new core test wrapper design approach is proposed which transports streaming test data, for example scan test patterns, into and out of an embedded core exclusively via (some of) its functional data ports. The latter are typically based on standardised protocols such as AXI, DTL, and OCP. The new wrapper design allows a functional interconnect, such as an on-chip bus or network-on-chip (NOC) to transport test data to embedded cores, and hence eliminates the need for a conventional dedicated test access mechanism (TAM), such as a TestRail or test bus. The approach leaves both the tester, as well as the embedded core and its test unchanged, while the functional interconnect can handle the test data transport as a regular data application. The functional interconnect is required to offer guaranteed throughput and zero latency variation, a service that is available in many buses and networks. For 672 example cases based on the ITC'02 System-on-Chip (SOC) Test Benchmarks, the new approach in comparison with the conventional approach shows an average wrapper area increase of 14.5%, which is negligible at the SOC level, especially since the dedicated TAM can be eliminated. Futhermore, the new approach decreases the core test length by 3.8% on average.

1 Introduction

Modern semiconductor process technologies and design tools enable the design and manufacturing of increasingly complex SOCs. Such SOCs typically consist of a heterogenous mix of design blocks (cores) form a variety of in-house and external sources. The traditional form of functional interconnect between these cores is the on-chip bus (an array of wires with multiple writers under a mutual-exclusion control scheme). Typically, cores interface with the bus through standardised protocols, such as Advanced eXtensible Interface (AXI) [1] Device Transaction Level (DTL) [2], and Open Core Protocol (OCP) [3]. However, the common global on-chip bus is becoming a bottleneck in communication bandwidth and power dissipation. Multi-bus solutions have provided a temporary alleviation, but for the longer-term, a scalable solution is a network-on-chip (NOC) [4-7]. A NOC consists of a network of shared communication links and routers, which connect to the various cores through network interfaces (NIs). These NIs convert between the internal NOC protocol on one side and the core's protocol on the other side. For reasons of compatibility and reuse, the latter is typically one of the standardised bus protocols, such as AXI, DTL, and OCP. A number of NOCs [8-12] offer communication with guaranteed performance to implement real-time application and to make SOC designs more robust [13].

Only a modular approach, in which the various cores are tested as stand-alone units, effectively addresses the manufacturing-test challenges of SOCs [14]. Modular testing is enabled by on-chip design-for-test (DfT) hardware [15] in the form of a core test wrapper [16–18] that switches between functional mode and test mode access and isolation, and a test access mechanism (TAM) [19] that transports the test data from the SOC pins to the core terminals and vice versa. An evident idea is to reuse the on-chip functional interconnect to double as a TAM in the test mode(s) of the SOC; it saves the design effort and silicon costs associated with dedicated TAMs. This idea has been tried for traditional buses [20–25] as well as, more recently, NOCs [26–32].

This paper builds on the idea of reusing an on-chip bus, NOC, or other functional interconnect as a TAM. Unlike other papers, the focus here is on the core test wrapper design that enables this. It is our objective to modify neither the functional interconnect nor the core-under-test or its test, to maximise their reuse. For the functional interconnect, 'core test' is a regular application, in which the bus or NOC just transports data; it happens to be test data but that is irrelevant to the functional interconnect. The dedicated TAM ports of conventional wrappers are removed, as test data is now transported through functional ports. Most core tests are based on scan design, which is a real-time streaming application for the bus or NOC (once started, it needs to complete without interruption), but for

doi:10.1049/iet-cdt:20060152

Paper first received 11th September 2006 and in revised form 18th January 2007

A.M. Amory and M. Lubaszewski are with the Federal University of Rio Grande do Sul – UFRGS, Instituto de Informática, Av. Bento Gonçalves 9500, Porto Alegre, RS, Brazil.

K. Goossens is with NXP Semiconductors Research, High Tech Campus 37, 5656 AE Eindhoven, The Netherlands and is also with the University of Delft, Department of Computer Engineering, Mekelweg 5, 2628 CC Delft, The Netherlands

E.J. Marinissen is with NXP Semiconductors Research, High Tech Campus 37, 5656 AE Eindhoven, The Netherlands

F. Moraes is with the Catholic University – PUCRS, Faculdade de Informática, Av. Ipiranga 6681, Porto Alegre, RS, Brazil

E-mail: erik.jan.marinissen@nxp.com

An earlier version of this work has been published: Amory, A.M., Goossens, K. Marinissen, E.J., Lubaszewski, M., and Moraes, F.: 'Wrapper design for the reuse of networks-on-chip as test access mechanism.' *Proc. IEEE European Test Symp. (ETS)*, Southampton, UK, May 2006, pp. 213–218

which the required bandwidth can be chosen by adjusting the number of parallel scan chains. The streaming nature of scan test data requires a functional interconnect that provides uncorrupted, lossless, in-order data transport with guaranteed throughput ant zero latency variation, such as the Æthereal NOC [8]. We assume that the functional interconnect hardware is designed to functional specifications. This given bus or NOC hardware is programmed for the 'core test' application using its regular design flow [33]. Subsequently, the bandwidth used for test is tuned to fit the given functional interconnect, through adjustment of the number of parallel scan chains for the core by means of a new wrapper design optimisation algorithm.

This paper presents a test wrapper design method and corresponding optimisation algorithm that allows cores to be tested via on-chip functional interconnects that provide guaranteed performance. Also, we demonstrate their use with a NOC doubling as a TAM. They are equally applicable to other functional interconnects, such as traditional on-chip buses and crossbars. The remainder of this paper is organised as follows. Section 2 provides an overview of the related prior work. In Section 3, we list our assumptions regarding the communication behaviour and standardised protocol ports of the functional interconnect and formulate our problem definition. Our new wrapper design procedure is detailed in Section 4. Experimental results with respect to area and test length impact are given in Section 5, while Section 6 concludes this paper.

2 Related prior work

Reusing an existing functional interconnect as a means to obtain test access to embedded cores is an natural idea. Especially for functional tests, it is very natural to reuse the on-chip bus as a TAM. Harrod [20] has described ARM's original test approach, which was based on functional testing. The 32-bit advanced microcontroller bus architecture (AMBA) transports test stimuli from the integrated circuit (IC) pins via the external bus interface (EBI) to the core-under-test (CUT), and test responses back again. In test mode, a test interface controller acts as a bus master. In other work [21–23] the existing on-chip bus has been reused to apply functional tests from the on-chip microprocessor to embedded cores. An advantage of these approaches is the low additional area cost of the TAM. However, they require that the entire path from the tester (whether chip-external automatic test equipment (ATE) or on-chip embedded microprocessor) via the functional interconnect to the CUT is known and under full control. In this paper, we remove this constraint, by allowing any functional interconnect that offers guaranteed bandwidth and latency communication to operate as a TAM. Moreover, all approaches listed above depend on functional tests, for which the detection qualities are hard to assess, guarantee, and improve, and for which failure analysis is nearly impossible. Hence, most semiconductor companies prefer structural, scan-based tests instead. Feige *et al.* [24] demonstrated that ARM's functional-bus access approach is difficult to combine with scan-based testing. Hughes *et al.* [25] have described how ARM has moved from functional bus-based testing to structural, scan-based testing via a dedicated TAM.

An early paper to propose a NOC-like, packet switching network to serve as a TAM is that of Nahvi and Ivanov [34]. The paper presented the network as a dedicated test infrastructure, which obviously would imply high (but in their paper unquantified) silicon area costs. Cota *et al.* [26] are, to the best of our knowledge, the first to propose reusing a functional NOC as a TAM. The paper mainly focused on the scheduling of the test packets for the various cores in order to minimise test time. The approach requires knowledge of all kinds of NOC implementation details, such as the network topology, number of routers, packet size, time to unpack headers, and so on. Subsequent papers by the same author team [27, 28] take maximum power dissipation during the test into account as an additional scheduling constraint. Further extensions by Liu *et al.* include built-in self-test (BIST) and precedence constraints [29], variable-rate clocking and power constraints [30], and thermal constraints [31]. Also, Kim *et al.* [32] have suggested reuse of an existing functional NOC for test access purposes. Finally, Hosseinabady *et al.* [35] have recently focused on testing NOC switches through the network, but they did not detail the implementation of the switch under test, nor how the NOC remains operational while some of its switches are being tested.

None of the above papers describe how the streaming scan test data requirement is matched to the possibly bursty or packetised bus or NOC traffic. They do not specify how the communication protocol between the bus or NOC and the CUT is supported in order to keep the data flowing while the core is in its test mode. Also, the details of their test wrapper designs and the differences with conventional test wrappers such as TestShell [16, 17] and IEEE Std. 1500 [18] have not been disclosed.

3 Assumptions and problem statement

During SOC testing, the ATE needs to be connected to the CUT to load test stimuli and unload test responses. In a conventional SOC test set-up, the ATE and the CUT are connected by a dedicated on-chip TAM [19]. This set-up is schematically depicted in Fig. 1*a*. The test wrapper connects the CUT's test inputs via the TAM to the ATE stimulus channels; similarly, the wrapper connects the CUT's test outputs via the TAM to the ATE response channels. Fig. 1*b* depicts the new SOC test set-up under consideration in this paper. The test data transport from the ATE to the CUT and vice versa is now handled by the functional interconnect, (e.g. in the form of an on-chip bus, cross-bar, or NOC). This approach makes a dedicated TAM superfluous.

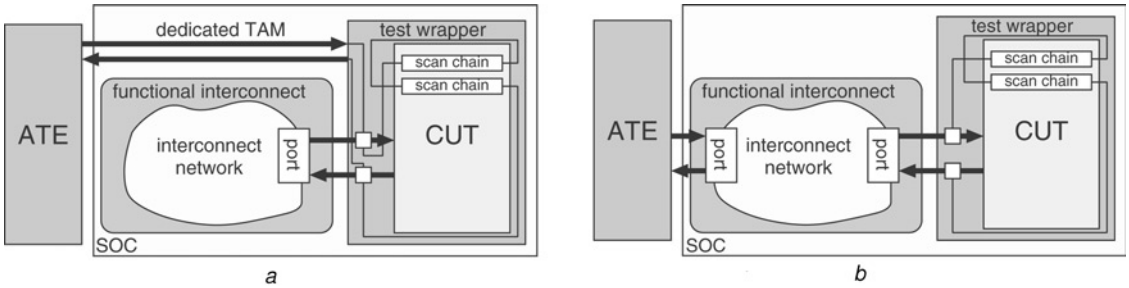

Fig. 1 *Conventional (a) and new (b) SOC test set-up*

The following two sections introduce the methods of achieving this, viz. abstracting from the functional interconnect implementation through guaranteed communication, and translating functional port protocols into a test protocol.

3.1 Guaranteed communication and test

Most tests have a streaming nature, that is once started, they need to complete without interruption. This is particularly true for scan-based, structural tests. If a scan operation is interrupted, the scan chains lose their content of stimuli and responses, unless additional DfT features are provided to freeze the scan content. Examples of such DfT features are stoppable clock signals, or scan flip-flops with a hold mode implemented by means of a feedback loop. These DfT features increase silicon area and negatively impact SOC performance and design flow, and are therefore undesirable. Also, most ATEs treat test stimuli and responses as streaming data, and are only capable of real-time comparison of the actual with the expected responses (possibly with an abort-on-fail in the case of a mismatch) and capturing responses. ATEs are typically not capable of more complex interactions with the SOC.

In order to keep the design flows of the functional interconnect, cores, and their tests unchanged, it is our objective to keep the streaming nature of the core test and the ATE intact, while at the same time using the functional interconnect in its normal mode of operation to transport test data as regular payload data. This requires the functional interconnect to provide uncorrupted, lossless, in-order data transport with guaranteed throughput and zero latency variation. If this is the case, the functional interconnect can be modelled as two connections (or 'virtual wires') with a given constant bandwidth and latency. One connection (from the tester to the CUT) is used for test stimuli, the other (from the CUT to the tester) carries test responses.

Fortunately, many functional interconnects already provide performance guarantees for several other reasons. Guaranteeing the communications of a core makes it independent from the rest of the system; it can then be designed in isolation, which benefits compositional design. Furthermore, many cores operate on real-time data, for which guarantees are required. NOCs that can give performance guarantees include Æthereal [8], Mango [9], Nostrum [10], and Sonics [12].

Using connections with guaranteed performance allows us to abstract from the implementation of the interconnect (bus, cross-bar, NOC, etc). This is valuable because inside the functional interconnect, the traffic might not be transported in a streaming manner at all. For example, a bitstream may be sent in bursts over a bus. Similarly, NOCs typically transport data in packets, which internally introduces data overhead and gaps in the transport [36]. Connections with guaranteed fixed bandwidth and latency allow us to ignore these details.

3.2 Functional port protocols and test

In the functional mode, the interconnect structure connects to each of the cores via one or more ports. As a precondition to core and interconnect reuse, it is common practice for these ports to use a standard protocol, such as AXI [1], DTL [2], or OCP[3]. In test mode, (a subset of) the same ports are reused to transport test data to and from the CUT. This approach requires that the ATE can be connected to the functional interconnect; in this paper, we assume that this is the case.

Functional port protocols define a number of signals and their semantics. For all major standard protocols, signals are divided in three groups: command, write, and read. Each group consists of zero or more signals, which are either input or output. An initiator port sends out the command and hence initiates the communication; a target port receives the command. A write port communicates data in the same direction as the command (i.e. from initiator to target), while a read port communicates data in the opposite direction; read-write ports can communicate data in both directions, although not all such bidirectional ports support simultaneous (full-duplex) read and write communication.

Fig. 2 illustrates the above for an example core with two DTL ports. Both ports in our example are DTL MMBD read-write ports. MMBD stands for memory-mapped block data, the most complex profile of the DTL protocol [2]. Our proposed method also applies to the other DTL profiles. The left-hand port is a target, while the right-hand port is an initiator. The signal names indicate the partitioning of the port signals into three groups; command, write, and read. All three signal groups have their own `valid` and `accept` signals that regulate the handshake process for data transfer. In addition, the command group has three more signals, that indicate address, read/write direction, and block size.

To transport test data over the functional interconnect to a core, that core needs to have at least one port that can serve as a test stimulus input and at least one port that can serve as a test response output. The test stimulus input role can be enacted by an initiator read or read-write port, or a target write or read-write port. Similary, a test response output role can be enacted by an initiator write or read-write port, or by a target read or read-write port. The example core in Fig. 2 has two ports that both can serve as a test input and a test output port, as both are read-write ports. An example of a valid scenario is to use the (left-hand) target port as a test input port and the (right-hand) initiator port as a test output port.

Every protocol port is used in one or more functional applications, with possibly different bandwidth and latency requirements. For example, an output port may generate standard-definition video in one application and high-definition video in another. The functional interconnect, and in particular the buffering capacity of ports, is dimensioned to support all required applications. Reusing the functional hardware, the connection between the ATE and a certain port has a bandwidth that we consider given.

We aim to let the functional interconnect transport test data as a regular application. Consequently, we only use the data signals for this task. In our example, test stimuli are input via `dtl_wr_data[32]` of the (left-hand) target port, while test responses are output via `dtl_wr_data[32]` of the (right-hand) initiator port. In addition, it is important that the functional protocol is executed correctly. For example, the command and the block

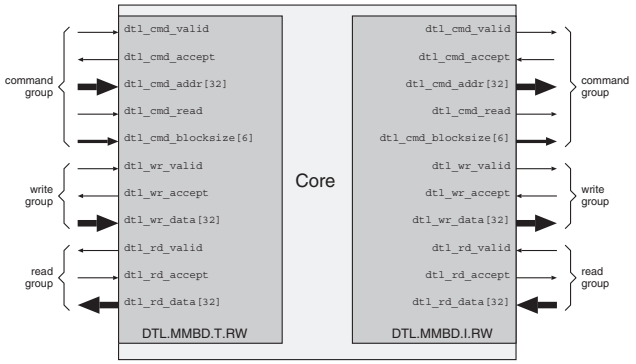

Fig. 2 *Example core with two DTL MMBD read-write ports*

size must be signalled, and the valid/accept handshakes must be completed. This is one of the requirements on our modified test wrapper design.

3.3 Problem statement

We assume a SOC containing a functional interconnect such as an on-chip bus or NOC and one or more cores. We focus on testing one core at a time. We assume also that an ATE is able to connect to the functional interconnect without specifying the details of that connection here. Via the functional interconnect, the ATE is able to send data to one or more of the CUT's protocol ports, while likewise the ATE is able to receive data from one or more of the CUT's protocol ports. Connections between the ATE and the CUT provide uncorrupted, lossless, in-order data transport, with guaranteed throughput and zero latency variation.

In our problem definition, the core will use exactly one protocol port as test input and exactly one other protocol port as test output. As not all bidirectional ports support full-duplex communication, read-write ports are used in only one direction, in order to allow scan-in and scan-out operations during test to overlap in time. Consequently, the core will need to have at least two protocol ports: one to serve as a test input and one to serve as a test output port. In the remainder of this paper, we will assume that at least two suitable ports are available.

For the CUT all information is given about its protocol ports through which it can communicate with the ATE. For each such protocol port i is specified the protocol, profile, direction of the port, data word width $w(i)$, maximum sustainable input bandwidth from the ATE $b^{in}(i)$, output bandwidth to the ATE $b^{\text{out}}(i)$, and corresponding transaction block size $s(i)$ [37]. Note that for a write-only or read-only port i, $b^{\text{in}}(i) = 0 \vee b^{\text{out}}(i) = 0$.

Furthermore the data is given as specified for the cores in the ITC'02 SOC Test Benchmarks [38]: all other inputs and outputs, that are not belonging to the protocol ports, the number of core-internal scan chains and their length, and the number of test patterns. Also specified is the test frequency f of the CUT, which might be different from the normal-operation frequencies of either the CUT or the functional interconnect.

We need to determine the design of a core test wrapper, which enables the CUT to be tested by the ATE via the functional interconnect, at test frequency f, without modifying the ATE, the interconnect, the CUT, or its test. The design of the wrapper is optimised such that the CUT's test length is minimised and we make best use of the offered bandwidth of the functional interconnect.

4 Wrapper design

4.1 Wrapper design overview

Fig. 3 shows overviews of both the conventional (Fig. 3*a*) and new (Fig. 3*b*) IEEE Std. 1500 [18, 39] compliant design, and illustrates their differences. In the INTEST mode of the conventional wrapper, the access of all functional inputs and outputs to the core is intercepted by the wrapper boundary register (WBR), indicated by dashed arrows, while ports to dedicated TAMs provide test access. In the new wrapper design, there is no dedicated TAM. Test access to the core is provided via a subset of the functional inputs and outputs, viz. via one selected input protocol port and one selected output protocol port. Functional inputs and outputs that do not belong to the selected protocol ports are intercepted by the WBR, and are again shown with dashed arrows.

The algorithm to design a core test wrapper and optimise its corresponding test length consists of the following five steps.

1. *Selection of test input and output ports:* In the case when multiple protocol ports are available that could serve as test input or test output ports, exactly one input port and exactly one test output port are selected. To minimise the resulting test length, we select two disjunct ports i and o (with $i \neq o$) of the CUT connected to the ATE such that the resulting test bandwidth b_{test} is maximised with

$$b_{\text{test}} = \min\left(b^{\text{in}}(i), b^{\text{out}}(o)\right) \tag{1}$$

As per (1), the test bandwidth is determined by the minimum of the bandwidths of the selected input and output port, as the bandwidths of the wrapper scan input and output need to be equal.

2. *Calculation of the number of wrapper chains:* Wrapper chains are the scan chains through the wrapped core, built-up from wrapper cells and core-internal scan chains [17]. At the core's test frequency f, the functional interconnect can deliver at most $\lfloor b_{\text{test}}/f \rfloor$ test stimulus bits per clock cycle via the selected test input port, and receive an equal amount of test response bits via the selected test output port. Hence, the wrapper should contain wc wrapper chains with

$$wc = \left\lfloor \frac{b_{\text{test}}}{f} \right\rfloor \tag{2}$$

to make maximal usage of the test bandwidth provided by the functional interconnect.

3. *Calculation of parallel-to-serial loading and serial-to-parallel unloading characteristics:* Stimulus bits

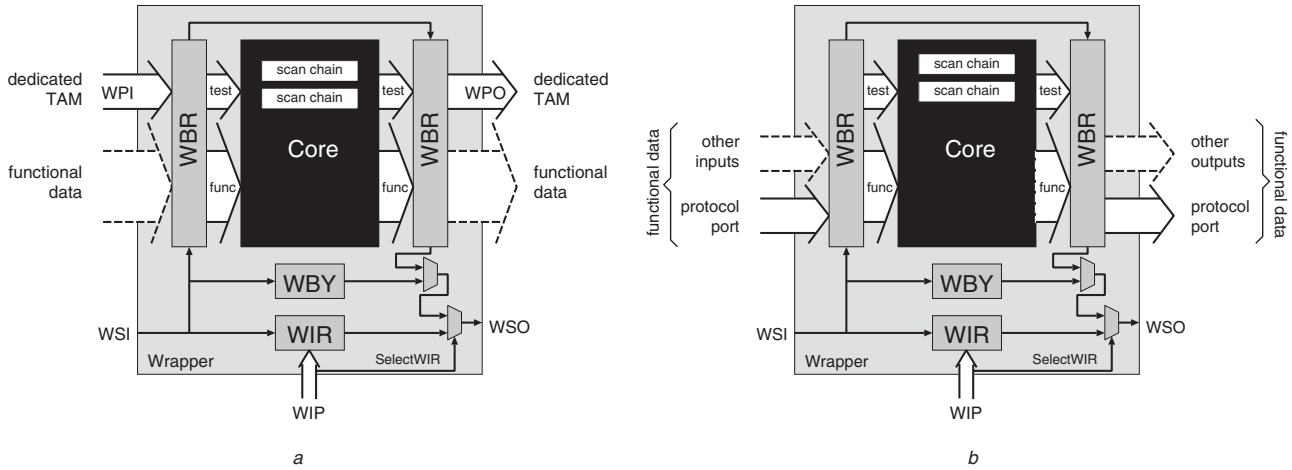

Fig. 3 *Overview of the (a) conventional and (b) new IEEE Std. 1500 compliant wrapper design*

Table 1: Example classification of port terminals

Class	Test input port: DTL.MMBD.T.RW	Test output port: DTL.MMBD.I.RW
SDI	`dtl_wr_data[29:0]`	—
RSDI	`dtl_wr_data[31:30]`	—
SDO	—	`dtl_wr_data[29:0]`
RSDO	—	`dtl_wr_data[31:30]`
DI	—	`dtl_rd_data[31:0]`
DO	`dtl_rd_data[31:0]`	—
CI	`dtl_cmd_valid, dtl_cmd_addr[31:0], dtl_cmd_read, dtl_cmd_blocksize[5:0], dtl_wr_valid, dtl_rd_accept`	`dtl_cmd_accept, dtl_wr_accept, dtl_rd_valid`
CO	`dtl_cmd_accept, dtl_wr_accept`	`dtl_cmd_valid, dtl_rd_valid, dtl_cmd_addr[31:0], dtl_cmd_read, dtl_cmd_blocksize[5:0], dtl_wr_valid, dtl_rd_accept`

arrive strictly periodically in words of $w(i)$ bits at the selected test input port i. There, they are equally divided over the wc wrapper chains and serially shifted into the wrapper and core. This process is repeated every $p(i)$ clock cycles, with

$$p(i) = \left\lfloor \frac{w(i)}{wc} \right\rfloor \tag{3}$$

Note that of the $w(i)$ data bits in each parallel word, only $\lfloor w(i)/wc \rfloor \times wc$ are actually used to carry stimulus bits; the remaining ($w(i)$ mod wc) bits carry unused data, and consequently will be assigned special wrapper cells in Step 4. The response side is handled likewise where every $p(o) = \lfloor w(o)/wc \rfloor$ clock cycles, $p(o)$ response bits are unloaded in parallel from the test output port o of each of the wc wrapper chains.

4. *Core terminal classification:* Based on their role in the test operation, core terminals are classified. The classification determines the actual wrapper design in Step 5 (i.e. the type of wrapper cell for the terminal, the position of the wrapper cell in a wrapper scan chain, and the control signals to the wrapper cell). The core terminal classification is described in more detail in Subsection 4.2.

5. *Actual wrapper design:* Based on the outcome of the core terminal classification, the wrapper is instantiated, according to the following sub steps.

- Assign wrapper cells to individual core terminals, as described in Subsection 4.3.
- Partition and order the wrapper cells and core-internal scan chains over the wc wrapper chains, as described in Subsection 4.4.
- Connect the control logic to the various wrapper cells.

4.2 Core terminal classification

Conventional wrapper design [17] distinguishes four classes of core terminals: functional inputs (FI), functional outputs (FO), scan inputs (SI), and scan outputs (SO). In the new approach, in which test data reaches the core via reused functional protocol ports, we add eight new classes, for the terminals of the functional protocol ports.

- SDI, SDO: Selected data inputs and outputs, that is the data terminals of the functional protocol port that have been selected to serve as test input (output) and carry actual stimulus (response) data.
- RSDI, RSDO: Remaining selected data inputs and outputs, that is those ($w(i)$ mod wc) data bits of the selected test input port i and ($w(o)$ mod wc) data bits of the selected test output port o that carry unused data.
- DI, DO: Data inputs and outputs, that is the data terminals of the remaining functional protocol ports not selected to carry test data.
- CI, CO: Control inputs and outputs, that is the non-data terminals of all functional protocol ports.
- FI, FO: Functional inputs and outputs, that is, all functional terminals that are not part of any functional protocol ports.
- SI, SO: Scan inputs and outputs.

The twelve classes provide a complete partitioning for all digital data terminals of the core, that is all classes are mutually exclusive and their union equals all terminals.

The classification of the functional protocol port terminals of the example in Fig. 2 is shown in Table 1. For both ports, the `dtl_wr_data[31:0]` terminals are selected as test input (output) ports and hence are classified as SDI and SDO respectively. Assuming $wc = 3$, both selected ports have (w mod wc) = (32 mod 3) = 2 unused terminals, say `dtl_wr_data[31:30]`, which are then respectively moved into the RSDI and RSDO classes. The `dtl_rd_data[31:0]` terminals are not selected as test input and output ports, and hence are left as DI and DO. The sets CI and CO include all control terminals, that is all `command` signal group terminals and the `valid` and `accept` signals of the write and read signal groups.

4.3 Wrapper cell design and assignment

In our wrapper, we use two types of IEEE Std. 1500 compliant wrapper cells. All terminals use a 'regular' wrapper cell, except for the CO-type terminals, which use a special variant of the 'regular' wrapper cell. Any IEEE Std. 1500 compliant wrapper cell can serve as a 'regular' wrapper cell. We prefer IEEE Std. 1500 wrapper cell

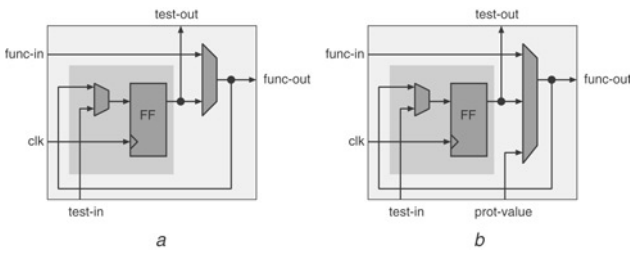

Fig. 4 *Implementation of wrapper cells (a) WC_SDI_COI and (b) WC_SDI_COT_G*

Table 2: Control generator parameters for test output port DTL.MMBD.I.RW

Terminal	Parameters(*v, p*)	Description
dtl_cmd_valid	(1, $s(o)\times p(o)$)	Every $s(o)$ words, a new command is issued
dtl_cmd_addr[31:0]	'0' = (0, 1)	Address is not used and hence kept at '0'
dtl_cmd_read	'0' = (0, 1)	Only the write side of this port is used and hence read = 0
dtl_cmd_blocksize[6]	($s(o)$, 1)	Block size is $s(o)$
dtl_wr_valid	(1, $p(o)$)	Every $p(o)$ clock cycles, a new word with test responses is available
dtl_rd_accept	'0' = (0, 1)	The read size of this port is not used and hence accept = 0

WC_SD1_COI [18], as depicted in Fig. 4*a*. This wrapper cell is small (one flip-flop only) and the combined activation of INTEST and EXTEST modes guarantees full test coverage of the wrapper cell itself, as the captured signal is tapped off after the functional multiplexer.

The CO-type core terminals require a special variant of the 'regular' wrapper cell because they need to assure that, as far as the on-chip bus or NOC is concerned, also in test mode, the functional port protocol is handled as normal, such that the transport of test data is not interrupted. For example, the valid/accept handshakes of the command, write, and read groups of a DTL port need to be completed as usual. The special wrapper cell for CO-type terminals consists of a generic part and a terminal-specific part. The generic part is a guarded variant of the 'regular' wrapper cell. For our preferred wrapper cell WC_SD1_COI, the guarded variant is IEEE Std. 1500 wrapper cell WC_SD1_COI_G as depicted in Fig.4*b*. Its functional output can be set to the value of signal prot-value. The circuitry that generates prot-value is specific for each CO terminal.

The generic version of the control generator has a periodic output behaviour, which is characterised by a binary-coded output value v and a period p. For $p-1$ consecutive clock cycles, the output value is $\bar{v}$, while for the next clock cycle, the output is v. The hardware implementation uses a simple counter. Some CO terminals require hard-coded '0'or '1' signals; this is captured by $(v,\ p) = (0,\ 1)$ and $(v,\ p) = (1,\ 1)$, respectively. The control generator implementation for these signals uses tie-off-cells.

Table 2 describes for the selected test output port of our example in Fig. 2 which output signals are required on its CO-type terminals in order to keep the functional port protocol running. The port is used as an output via its dtl_wr_data terminals, and hence dtl_cmd_read is kept at a hard-coded '0'. Every $p(o)$ clock cycles, a new word with test responses is available for parallel read-out from the scan chains. To write this word to the interconnect dtl_wr_valid is 'l' every $p(o)$ clock cycles. The specified bandwidth $b^{out}(o)$ is achieved for a given block size $s(o)$. Hence, the binary six-bit equivalent of $s(o)$ is output on dtl_cmt_blocksize[6]. Every $(s(o)\times p(o))$ cycles, a block of $s(o)$ words has been output, and this initiator port should issue a new command, in order to instruct the bus or NOC to accept another block. Terminals dtl_cmd_addr[31:0] and dtl_rd_accept are not used, and hence kept at hard-coded '0'.

4.4 Partitioning and ordering of wrapper chain items per wrapper chain

Wrapper chains are made up from wrapper cells and core-internal scan chains. We need to construct *wc* wrapper chains, but typically many more than *wc* wrapper chain items exist. Hence, we need to partition the wrapper cells and core-internal scan chains over the *wc* wrapper chains, and subsequently determine the order of the items per wrapper chain. Our approach to do this is a modification of the conventional wrapper design optimisation algorithm of [17].

In conventional wrapper design, the test length T_{conv} for a wrapped core is defined as

$$T_{\text{conv}} = (1 + \max(s_i,\ s_o)) \times pat + \min(s_i,\ s_o) \qquad (4)$$

where pat denotes the number of test patterns, and s_i and s_o denote the scan-in and scan-out length for the wrapped core, respectively [17]. To minimise T_{conv}, both s_i and s_o must be minimised. All wrapper input cells and all core-internal scan chains participate in a scan-in-operation and hence might contribute to s_i; likewise, all wrapper output cells and core-internal scan chains participate in a scan-out operation and therefore might contribute to s_o. The conventional partitioning algorithm in [17] first addresses the $\mathcal{NP}$-hard problem of partitioning the core-internal scan chains over the available number of to-be-constructed wrapper chains, before partitioning the wrapper input and output cells. This approach aims to minimise both s_i and s_o. The subsequent step of ordering of wrapper items per wrapper chain is done such that the wrapper input cells are followed by the core-internal scan chains, which in turn are followed by the wrapper output cells. A schematic view of the resulting conventional wrapper chain is depicted in Fig. 5*a*.

A schematic view of the wrapper chains resulting from our new wrapper design approach is shown in Fig. 5*b*. The main differences are formed by the category of SDI wrapper input cells, through which stimuli are loaded into the wrapper chain in a parallel fashion, and the category of SDO wrapper output cells, through which responses are unloaded from the wrapper chain in a parallel fashion.

- The test stimuli of a test pattern arrive at regular intervals of $p(i)$ clock cycles at the SDI wrapper input cells. All of them are shifted into the wrapper chains, apart from the last word, which can be directly consumed in parallel for testing.

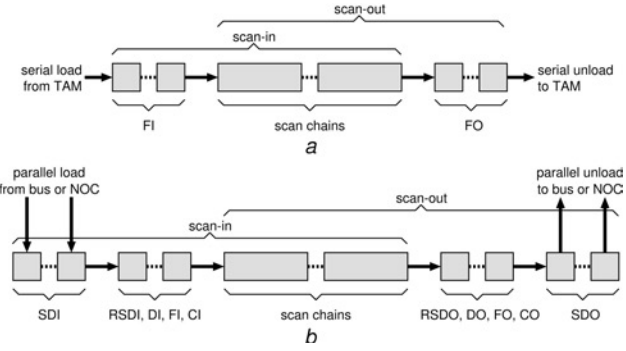

Fig. 5 *Schematic view of conventional (a) and new (b) ordering of wrapper items per wrapper*

Similarly, for each test pattern the first word of responses is directly transported away in parallel from the SDO wrapper output cells, after which the subsequent words are shifted out of the wrapper chains at regular intervals of $p(o)$ clock cycles. Hence, the test length T_{new} is redefined as

$$T_{\text{new}} = (1 + \max(t_i, t_o)) \times pat + \min(t_i, t_o) \quad (5)$$

with

$$t_i = \left(\left\lceil \frac{s_i}{p(i)} \right\rceil - 1\right) \times p(i) + 1 \quad (6)$$

$$t_o = \left(\left\lceil \frac{s_o}{p(o)} \right\rceil - 1\right) \times p(o) + 1 \quad (7)$$

- In order to minimise the test length T_{new}, the wrapper items should be partitioned such that all SDI cells and all SDO cells are evenly distributed over the *wc* to-be-constructed wrapper chains.
- For wrapper item ordering, it is important that all SDI cells are placed at the start of the wrapper chain, as they are the entry point of stimuli into the wrapper chain, otherwise, some wrapper chain items are unreachable for scan access. Similarly, all SDO cells should be placed at the end of the wrapper chain.

Algorithmically, the partitioning of wrapper items is done in a five-step approach, which is clearly based on the three-step approach in [17].

1. Assign the core-internal scan chains to the *wc* wrapper chains, such that the maximum sum of scan lengths assigned to a wrapper chain is minimised. Algorithms like LPT and COMBINE can be used [17]. The resulting partition is named $\mathcal{P}_s$
2. Assign the wrapper input cells in $RSDI \cup DI \cup CI \cup FI$ to the *wc* wrapper chains on top of $\mathcal{P}_s$, such that the maximum scan-in length of all wrapper chains is minimised. The resulting partition is named $\mathcal{P}_{\text{wc}}^{\text{in}}$
3. Assign the wrapper input cells in *SDI* to the *wc* wrapper chains on top of $\mathcal{P}_{\text{wc}}^{\text{in}}$. The resulting partition is named $\mathcal{P}_{\text{in}}$.
4. Assign the wrapper output cells in $RSDO \cup DO \cup CO \cup FO$ to the *wc* wrapper chains on top of $\mathcal{P}_s$, such that the maximum scan-out length of all wrapper chains is minimised. The resulting partition is named $\mathcal{P}_{\text{wc}}^{\text{out}}$
5. Assign the wrapper output cells in *SDO* to the *wc* wrapper chains on top of $\mathcal{P}_{\text{wc}}^{\text{out}}$. The resulting partition is named $\mathcal{P}^{\text{out}}$.

5 Experimental results

5.1 Simplified illustrative example

We have implemented our wrapper design for a simplified illustrative example core and NOC. The NOC is an automatically generated, simple Æthereal network [8], consisting of one router and four network interfaces, based on a 32-bits data DTL protocol [2]. As depicted in Fig. 6*a*, two NOC ports connect to the ATE source and sink, respectively, while the two other ports connect to the CUT.

The CUT has two DTL ports, Port 1 and Port 2, with 133 terminals each and a functional data word width $w = 32$. Besides the two ports, the CUT has no other functional terminals. The CUT has five internal scan chains of lengths 123, 123, 50, 50, and 23 flip-flops, and $pat = 10$. Port 1 is selected to receive stimuli and Port 2 is selected to send out responses. The test data flow for stimuli is from the source through the NOC into the CUT via Port 1, while test responses flow from the CUT's Port 2 via the NOC into the sink. $b^{\text{in}}(1) = 1600$ Mbits/s, $b^{\text{out}}(2) = 2400$ Mbits/s, and $f = 500$ MHz, and (as per Equation (2)), we can afford a wrapper with $wc = 3$ wrapper chains. As per Equation (3), test data arrives with period $p(1) = p(2) = 10$ clock cycles. For Port 1: $|\text{SDI}| = 30$, $|\text{RSDI}| = 2$, $|\text{DO}| = 32$, $|\text{CI}| = 62$, and $|\text{CO}| = 7$. For Port 2: $|\text{SDO}| = 30$, $|\text{RSDO}| = 2$, $|\text{DI}| = 32$, $|\text{CI}| = 7$, and $|\text{CO}| = 62$. The wrapper chain scan lengths are $s_i = s_o = 168$. The resulting wrapper design is shown in Fig. 6*b*. As per Equation (5), test length $T_{\text{new}} = 1781$ clock cycles. Note that this represents a 4.1% test length reduction compared to a conventional wrapper design with three wrapper chains, which would

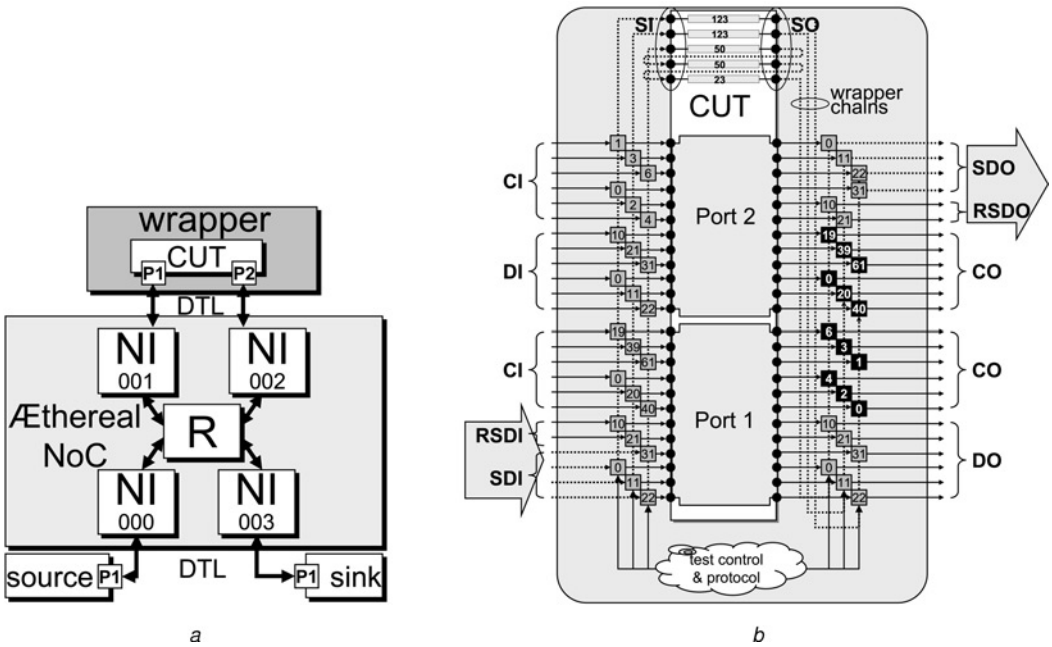

Fig. 6 *Illustrative example consisting of a NOC and CUT and b the detailed wrapper design for the CUT*

have a test length $T_{conv} = 1858$ clock cycles. This reduction is because of the fact that the last stimulus word of each pattern is loaded in parallel into the SDI wrapper cells and does not require any further shifting in, and likewise the first response word of each pattern is loaded in parallel into the SDO wrapper cells.

Our example was implemented in register-transfer-level VHDL and verified through simulation. The experiment showed a modest area increase for the new wrapper in comparison to a conventional wrapper. The number of gate-equivalents required to implement all wrapper cells went from 2910 to 3000 (an increase of +3.1%), while 489 gate-equivalents were required to implement the additional control logic to complete correct protocol operation. The total relative wrapper area increase was +19.9%. Note that wrappers are typically small compared to the overall SOC size, and hence, this area impact is negligible at the SOC level.

5.2 Wrapper area and core test length impact

In this section, we present wrapper area and core test length results from our new wrapper design approach in comparison with conventional wrapper design, obtained on a large subset of cores in the ITC'02 SOC test benchmarks [38]. A wrapper design and optimisation tool as described in this paper has been developed in C++; it uses the COMBINE wrapper design routine [17] as a basis. The tool calculates the required number of gate-equivalents to design the wrapper and the resulting test length in clock cycles. We do this for both the conventional wrapper design approach, which relies on a dedicated TAM, as well as for our new wrapper design approach, which reuses the functional protocol ports of the core.

The cores in the ITC'02 benchmarks do not have functional protocol ports (or, at least, they are not specified), while our method critically depends on their presence. Hence, we assume that every core has two DTL ports, of which one serves as a test input and the other as a test output. The assumed DTL test input port has 32 SDI terminals, 45 CI terminals, and 2 CO terminals. The assumed DTL test output port has 32 SDO terminals, 2 CI terminals, and 45 CO terminals. For the two DTL ports, each core needs 79 input terminals and 79 output terminals. Some of the 186 cores in the ITC'02 benchmarks do not have that many terminals and hence are not considered in our evaluation. Cores with bi-directional terminals are also not considered. This still leaves 42 cores for our experiments, which are listed in Table 3.

Table 3: List of considered ITC'02 SOC test benchmarks [17] cores

SOC	Cores considered
a586710	2, 3, 4, 7
d281	7
d695	2
f2126	1, 2
g1023	1, 2, 3, 4, 10, 11, 12, 14
h953	1
p22810	27
p34392	2, 10, 18
p93791	10, 32
q12710	1, 2, 3, 4
t512505	1, 2, 4, 8, 9, 14, 15, 16, 17, 23, 24, 25, 26, 29, 31

We assume that bandwidths $b^{in}(i)$ and $b^{out}(o)$ and test frequency f are such that the maximum affordable number of wrapper chains $wc = 16$. As $w(i) = w(o) = 32$, $wc = 16$ implies that $p(i) = p(o) = 2$. However, it is also possible to implement fewer, but longer wrapper chains, which means that we are not using the bandwidth to its maximum, and consequently accept a longer test length.

For all 42 cores considered and for a number of wrapper chains wc ranging from 1 to 16, we have calculated the number of gate-equivalents required to implement a wrapper and the corresponding test length in clock cycles, for both the conventional wrapper design algorithm [17] and our new approach. In total, this involves $42 \times 16 \times 2 = 1344$ wrapper calculations. Fig. 7 shows the average relative gate-equivalent count increase and the relative test length increase. The horizontal axes refer to cores with numbers in the order in which they are listed in Table 3.

Fig. 7*a* shows the increase in the number of gate-equivalents required to implement the new wrapper, relative

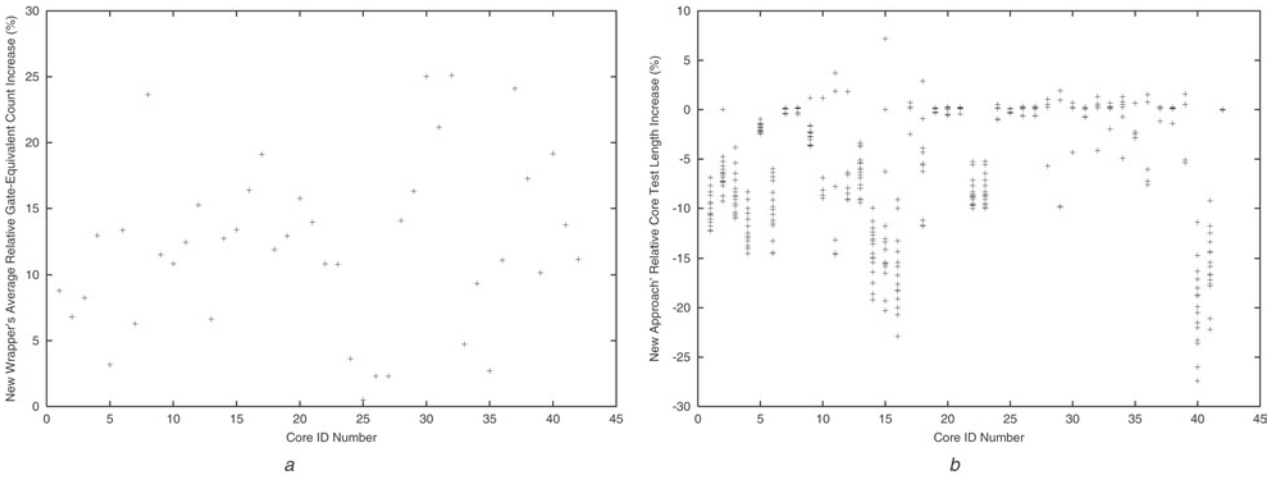

Fig. 7 *Proposed new wrapper design, compared to the conventional wrapper design, for 42 ITC'02 SOC test benchmark cores and* $wc \in [1 \ldots 16]$

a Average relative wrapper's gate-equivalent count increase
b Relative core's test length increase

to what was required for a conventional wrapper design. These numbers show only around 1% variation for varying values of *wc*, and hence we have decided to show the average increase for $1 \leq wc \leq 16$. The average wrapper area increase over all 672 cases is 14.5%, which is negligible at the SOC level. Fig. 7*b* shows that the test length impact of the new approach varies between −27.4 and +7.2% with, on average over all 672 cases, a test length decrease of 3.8%.

6 Conclusion

In this paper, we have proposed a core test wrapper design that allows test data to be transported via functional protocol ports, such as AXI, DTL, and OCP ports. This approach allows test data transport via an existing functional on-chip bus, NOC, or other functional interconnect. We have assumed that the functional interconnect connects to the test source and sink, be it either an external ATE, or on-chip BIST engine. We have required that the bus, NOC, or other functional interconnect provide guaranteed bandwidth and constant latency, which is necessary to operate streaming scan tests. We have exploited the available functional bandwidth as much as possible in order to minimise the core's test length.

Our new wrapper design no longer relies on a dedicated TAM, and hence has no dedicated TAM ports. Instead, the test data is transported via selected functional protocol ports. This requires careful wrapper design, such that the number of wrapper chains is tuned to the available test data bandwidth; the functional protocols are correctly executed while the test is running; and the protocol port wrapper cells are positioned at the head and tail of the various wrapper chains.

Although other papers have described the reuse of functional buses and NOCs as a TAM, no previous papers have worked out the details of the wrapper design for this scenario. The benefits of our approach are that no dedicated TAM is required, that test access to the embedded cores is guaranteed, and that the test expansion becomes simpler. By using guaranteed communication services, our approach hides any internal details of the functional interconnect. For the functional interconnect, the transport of test data is just another application, enabling unchanged re-use of the functional interconnect design flow.

The implementation of the new wrapper design for 672 example cases based on the ITC'02 SOC test benchmarks showed an average wrapper area increase of 14.5%, compared to a conventional wrapper. Note that at the SOC level, this area increase is negligible. The same 672 example cases show an average reduction of test length of 3.8%.

7 Acknowledgments

This work was partially supported by the Brazilian funding agencies CAPES and CNP_q-PNM, under scholarship grants 2371/04-9 and 141993/2002-2, respectively, and by the European MEDEA+ NanoTest project.

8 References

1 ARM: AMBA AXI Protocol Specification, 2003
2 Philips Semiconductors: Device transaction level (DTL) protocol specification Version 2.2, 2002
3 OCP International Partnership: Open core protocol specification 2.0 release candidate, 2003
4 Benini, L., and De Micheli, G.: 'Networks-on-chips: a new SOC Paradigm', *IEEE Computer*, 2002, **35**, (1), pp. 70–80
5 Jantsch, A., and Tenhunen, H. (Eds.): 'Networks on Chip' (Kluwer Academic Publishers, 2003)
6 Nurmi, J., Tenhunen, H., Isoaho, J., and Jantsch, A. (Eds.): 'Interconnect-centric design for advanced SoC and NoC' (Kluwer Academic Publishers, 2004)
7 De Micheli, G., and Benini, L. (Eds.): 'Networks on chips: technology and tools' (The Morgan Kaufmann Series in Systems on Silicon, Morgan Kaufman, 2006)
8 Goossens, K., Dielissen, J., and Rădulescu, A.: 'The Æthereal network on chip: concepts, architectures, and implementations', *IEEE Design & Test of Computers*, 2005, **22**, (5), pp. 21–31
9 Bjerregaard, T., and Sparsφ, J.: 'A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip'. Proc. Design, Automation, and Test in Europe, Munich, Germany, 2005, pp. 1226–1231
10 Millberg, M., Nilsson, E., Thid, R., and Jantsch, A.: 'Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip'. Proc. Design, Automation, and Test in Europe, Paris, France, 2004, pp. 890–895
11 Laffely, A., Liang, J., Tessier, R., and Burleson, W.: 'Adaptive system on a chip (aSOC): a backbone for power-aware signal processing cores'. Int. Conf. on Image Processing, Barcelona, Spain, 2003, pp. 105–108
12 Weber, W.D., Chou, J., Swarbrick, I., and Wingard, D.: 'A quality-of-service mechanism for interconnection networks in system-on-chips'. Proc. Design, Automation, and Test in Europe, Munich, Germany, 2005, pp. 1232–1237
13 Goossens, K., Dielissen, J., van Meerbergen, J., Paplavko, P., Rădulescu, A., Rijpkema, E., Waterlander, E., and Wielage, P.: 'Guaranteeing the quality of services in networks on chip' in Jantsch, A., and Tenhunen, H. (Eds.): 'Networks on chip' (Kluwer Academic Publishers, 2003), pp. 61–82
14 Marinissen, E.J., and Zorian, Y.: 'Challenges in testing core-based system ICs', *IEEE Commun. Mag.*, 1999, **37**, (6), pp. 104–109
15 Marinissen, E.J., and Waayers, T.: 'Infrastructure for modular SOC testing'. Proc. IEEE Custom Integrated Circuits Conf., Orlando, FL, USA, 2004, pp. 671–678
16 Marinissen, E.J., Arendsen, R., Bos, G., Dingemanse, H., Lousberg, M., and Wouters, C.: 'A structured and scalable mechanism for test access to embedded reusable cores'. Proc. IEEE Int. Test Conf., Washington, DC, USA, 1998, pp. 284–293
17 Marinissen, E.J., Goel, S.K., and Lousberg, M.: 'Wrapper design for embedded core Test'. Proc. IEEE Int. Test Conf., Atlantic City, NJ, USA, 2000, pp. 911–920
18 Da Silva, F. (Ed.): 'IEEE Std. 1500^{TM}-2005 standard testability method for embedded core-based integrated circuits' (IEEE, New York, NY, USA, 2005)
19 Goel, S.K., and Marinissen, E.J.: 'Effective and efficient test architecture design for SOCs'. Proc. IEEE Int. Test Conf., Baltimore, MD, USA, 2002, pp. 529–538
20 Harrod, P.: 'Testing reusable IP—a case study'. Proc. IEEE Int. Test Conf., Atlantic City, NJ, USA, 1999, pp. 493–498
21 Huang, J.R., Iyer, M.K., and Cheng, K.T.: 'A self-test methodology for IP cores in bus-based programmable SOCs'. Proc. IEEE VLSI Test Symp. Marina del Rey, CA, USA, 2001, pp. 198–203
22 Hwang, S., and Abraham, J.A.: 'Reuse of addressable system bus for SOC testing'. Proc. IEEE Int. ASIC/SOC Conf., Washington, DC, USA, 2001, pp. 215–219
23 Amory, A.M., Oliveira, L.A., and Moraes, F.G.: 'Software-based test for non-programmable cores in bus-based system-on-chip architectures'. Proc. IFIP Int. Conf. on Very Large Scale Integration, Darmstadt, Germany, 2003, pp. 174–179
24 Feige, C., ten Pierick, J., Wouters, C., Tangelder, R., and Kerkhoff, H.: 'Integration of the scan-test method into an architecture specific core-test approach'. Digest of papers of IEEE European Test Workshop, Barcelona, Spain, 1998, pp. 241–245
25 Hughes, P., Harrod, P., and Campbell, G.: 'Embedded CPU test strategies'. Digest of Papers of IEEE European Test Workshop, Corfu, Greece, 2002, pp. 281–285
26 Cota, E., Kreutz, M., Zeferino, C., Carro, L., Lubaszewski, M., and Susin, A.: 'The impact of NoC reuse on the testing of core-based systems'. Proc. IEEE VLSI Test Symp. Napa, CA, USA, 2003, pp. 128–133
27 Cota, É., Carro, L., Wagner, F., and Lubaszewski, M.: 'Power-aware NoC reuse on the testing of core-based systems'. Digest of Papers of IEEE European Test Workshop, Maastricht, The Netherlands, 2003, pp. 123–128
28 Cota, É., Carro, L., Wagner, F., and Lubaszewski, M.: 'Power-aware NoC reuse on the testing of core-based systems'. Proc. IEEE Int. Test Conf., Charlotte, NC, USA, 2003, pp. 612–621
29 Liu, C., Cota, É., Sharif, H., and Pradhan, D.: 'Test scheduling for network-on-chip with BIST and precedence constraints'. In Proc. IEEE Int. Test Conf., Charlotte, NC, USA, 2003, pp. 1369–1378

30 Liu, C., Iyengar, V., Shi, J., and Cota, É.: 'Power-aware test scheduling in network-on-chip using variable-rate on-chip clocking'. Proc. IEEE VLSI Test Symp., Palm Springs, CA, USA, 2005, pp. 349–354
31 Liu, C., and Iyengar, V.: 'Test scheduling with thermal optimization for network-on-chip using variable-rate on-chip clocking'. Proc. Design Automation, and Test in Europe, Munich, Germany, 2006, pp. 652–657
32 Kim, J.S., Hwang, M.S., Roh, S., Lee, J.Y., Lee, K., Lee, S.J., and Yoo, H.J.: 'On-chip network based embedded core testing'. Proc. IEEE Int. SOC Conf., Santa Clara, CA, USA, 2004, pp. 223–226
33 Goossens, K., Dielissen, J., Gangwal, O.P., Pestana, S.G., Rădulescu, A., and Rijpkema, E.: 'A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification'. Proc. Design Automation, and Test in Europe, Munich, Germany, 2005, pp. 1182–1187
34 Nahvi, M., and Ivanov, A.: 'A packet switching communication-based test access mechanism for system chips'. Digest of Papers of IEEE European Test Workshop, Saltsjöbaden, Sweden, 2001, pp. 195–200
35 Hosseinabady, M., Banaiyan, A., Bojnordi, M.N., and Navabi, Z.: 'A concurrent testing method for NoC switches'. Proc. Design Automation, and Test in Europe, Munich, Germany, 2006, pp. 1171–1176
36 Rădulescu, A., Dielissen, J., Pestana, S.G., Gangwal, O.P., Rijpkema, E., Wielage, P., and Goossens, K.: 'An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network programming', *IEEE Trans. CAD Integr. Circ. Syst.*, 2005, **24**, (1), pp. 4–17
37 Coenen, M., Murali, S., Rădulescu, A., Goossens, K., and de Micheli, G.: 'A Buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow-control'. Proc. Int. Conf. on Hardware-software Codesign and System Synthesis, Seoul, Korea, 2006, pp. 130–135
38 Marinissen, E.J., Iyengar, V., and Chakrabarty, K.: 'A set of benchmarks for modular testing of SOCs'. Proc. IEEE Int. Test Conf., Baltimore, MD, USA, 2002, pp. 519–528
39 da Silva, F., McLaurin, T., and Waayers, T.: 'The core test wrapper handbook – rationale and application of IEEE Std. 1500™, (Springer Science+Business Media, LLC, 2006)