Published in IET Computers & Digital Techniques
Received on 2nd July 2008
Revised on 2nd December 2008
doi: 10.1049/iet-cdt.2008.0072

In Special Issue on Networks on Chip

ISSN 1751-8601

Design-for-test approach of an asynchronous network-on-chip architecture and its associated test pattern generation and application

X.-T. Tran[1,3] *Y. Thonnart*[1] *J. Durupt*[1] *V. Beroulle*[2] *C. Robach*[2]

[1]*CEA-LETI, MINATEC – 17 rue des Martyrs, 38054 Grenoble, France*
[2]*Grenoble INP-LCIS – 50 rue Laffemas, 26902 Valence, France*
[3]*VNU-Coltech/SIS Laboratory – 144 Xuan Thuy Road, 10000 Hanoi, Vietnam*
E-mail: tutx@vnu.edu.vn

Abstract: Asynchronous design offers an attractive solution to address the problems faced by networks-on-chip (NoC) designers such as timing constraints. Nevertheless, post-fabrication testing is a big challenge to bring the asynchronous NoCs to the market because of a lack of testing methodology and support. This study first presents the design and implementation of a design-for-test (DfT) architecture, which improves the testability of an asynchronous NoC architecture. Then, a simple method for generating test patterns for network routers is described. Test patterns are automatically generated by a custom program, given the network topology and the network size. Finally, we introduce a testing strategy for the whole asynchronous NoC. With the generated test patterns, the testing methodology presents high fault coverage (99.86%) for single stuck-at fault models.

1 Introduction

To address the problems faced by system-on-chip (SoC) designers such as timing constraints, the globally asynchronous–locally synchronous (GALS) paradigm has been proposed [1]. In a GALS system, a number of synchronous islands communicate asynchronously with each other using an asynchronous communication environment. The networks-on-chip (NoC) architecture is perfectly adapted to the GALS platform where the network architectures (network routers and links) are fully implemented using asynchronous logic while the computational resources [i.e. intellectual properties (IPs)] are implemented with standard synchronous design methodologies [2].

Recent works have proposed some asynchronous implementations of NoC architectures [3–6]. The main advantages of asynchronous NoCs are robustness (insensitivity to delay variation because of physical constraints, temperature and voltage), ease of routing, low power management.

As the NoC paradigm is being brought to market, the test of NoC-based systems-on-chips (SoCs) for post-fabrication faults becomes a major challenge. Nowadays, in NoC-based systems, the embedded computational resources can be tested using traditional methods, while efficient testing methods for NoC architectures are really needed because of their regular structures.

Several design-for-test (DfT) propositions were made for testing NoC architectures. Most of them, however, focus on synchronous NoCs [2, 7–9]. To the best of our knowledge only two propositions were made for the test of asynchronous NoC architectures [10, 11].

In this paper, we present (i) the design and implementation of such a DfT architecture using a quasi-delay-insensitive

(QDI) asynchronous logic template and a 65 nm CMOS technology from STMicroelectronics, (ii) a simple method to generate test patterns for network routers and links, and (iii) a testing strategy for the whole NoC architecture.

The remaining part of this paper is organised as follows: Section 2 presents an overview of NoC-based systems testing. The asynchronous NoC architecture used in our work is shortly described in Section 3. The proposed testing approach for asynchronous NoCs is introduced in Section 4. Section 5 discusses the design and implementation of the proposed DfT architecture. The method to generate test patterns is presented in Section 6. Section 7 introduces a testing strategy for the whole network architecture. Finally, experimental results and conclusions are given in Sections 8 and 9, respectively.

2 NoC-based SoCs testing: state of the art

Like other SoCs, NoC-based SoCs have to be tested for manufacturing defects. Because of their regular structures, the testing strategy for NoC-based systems should address (i) the test of embedded computational resources (IPs) and their corresponding network interfaces (NIs), and (ii) the test of the interconnection infrastructure consisting of network routers and links between the routers.

2.1 Reuse of the network architecture for testing embedded computational resources

To test the embedded IPs in NoC-based SoCs, the IEEE 1500 standard's test wrapper [12] can be used for each embedded IP, while the network architecture can be used as a high bandwidth test access mechanism (TAM). Test data are encapsulated into packets that are transported on the network using the network protocol. The idea of using the network architecture as a TAM has been first proposed by Nahvi and Ivanov [13] at the University of British Columbia, Canada with the Novel Indirect and Module Architecture network. This network architecture has been intentionally developed for test purposes, not for communication purposes. Not long after that, several IP testing solutions reusing NoC as high bandwidth TAMs have also been proposed [14, 15]. The main difference between these propositions is the design of test wrappers and their adaptation to the network behaviour.

The principal advantages of this approach are the absence of extra TAM hardware cost and the availability of multi-path core test. In addition, test data process is decoupled from its transfer and scalability of test architecture is also improved. However, the NoC architecture should be searched for defects before being used.

2.2 Test of the interconnection architecture

The test of the interconnection architecture is a new challenge in NoC. In previous works on GALS systems, the interfaces between locally synchronous islands were reduced to point-to-point connections, which used synchronisation techniques at both ends. The associated testing solutions were test extensions to force a single clock at both ends of the connection, and use a conventional scan chain on this new clock domain. This method was used in [16]: the additional specific asynchronous logic was then tested using specific functional patterns. In NoC, though, the interconnection architecture is no longer a set of point-to-point connections, and the whole routing architecture needs to be tested.

A simple way to test the network is to route test data in the network with different paths using the network protocol. If the test responses are different from expected or if they are lost, one can say that the network is faulty. The advantages of this approach are its simplicity and its absence of area overhead. However, this approach has also several drawbacks. Although the network routers can be configured to transport test data to anywhere in the network, the testability of the network is still unresolved. We cannot access directly all inputs of the router-under-test for controlling purposes, and we cannot access directly all outputs of the router-under-test for observation purposes.

There are several DfT propositions for testing NoC architectures. For synchronous NoCs that may be seen as before as a collection of synchronous islands with point-to-point connections, a single test clock all over the chip could also be used. This is not optimal in terms of test application time, although, and more local solutions have been studied. Because network routers consist of FIFO buffers and routing logic parts; most of the discussions [2, 7] said that the test of network router should be done partially: using a built-in self-test (BIST) for FIFOs, and traditional methods for routing logic. Unfortunately, FIFOs are distributed all over the chip, which is a big challenge for this approach.

Since then several propositions [8, 9] have been done using serial scan, which is very expensive for the test of asynchronous NoCs in terms of area and test time. Besides, these works focus only on testing network routers, and do not address the test of network links across clock domains.

Asynchronous NoCs are designed and implemented in asynchronous logic without global clock. The test of asynchronous NoCs is more difficult than the test of synchronous NoCs because of the important number of feedback loops in asynchronous circuits and the lack of electronics design automation (EDA) supports for testing [17]. There are several DfT approaches for asynchronous circuits but area overhead is quite important. Efthymiou *et al.* [10] present a testing architecture for an asynchronous

doi: 10.1049/iet-cdt.2008.0072

NoC based on scan-latches insertion to break the feedback loops of Muller-C-elements. [The Muller element is a basic building block for self-timed digital design. It implements a join on signal transitions (events).] Unfortunately, the area overhead is then of about 43% of the total area, whatever the size of the design. Tran *et al.* [11] present a DfT architecture that uses a test wrapper around network routers. In such an approach, DfT area is not proportional to the total area but to the number of inputs/outputs, which may lead to a lower area overhead. However Tran *et al.* [11] does not present any implementation result.

3 Presentation of the base NOC architecture

This section intends to present the asynchronous NoC architecture that served as a basis for the present study. Beigné *et al.* [5] proposed an asynchronous NoC architecture providing low latency service named ANOC. The ANOC is composed of network routers, network links, asynchronous ⇔ synchronous interfaces (SAS interfaces) and NIs, as described in Fig. 1. The ANOC was used as an interconnect architecture for a GALS-based SoC, named flexible architecture unified system for telecoms (FAUST) [18].

The asynchronous network routers are the basic elements of network architecture and they have five bidirectional ports to connect four neighbouring routers (NORTH, EAST, SOUTH, WEST) and the nearest synchronous computational resource (RES) via a NI and a SAS interface. Network routers and links have been designed and implemented in QDI asynchronous logic style [19]. The communication between network routers is established by a handshake protocol (flit-level 'send/accept' protocol) with two virtual channels in order to improve system's QoS. The network links are 34-bit bidirectional channels, in which 32 bits are used for data and two bits are used for encoding packet information: begin-of-packet (BoP) and end-of-packet (EoP). Routing information is included in the header flit in a 'path-to-target' field shifted in each router after using the two lower bits for next direction (source routing).

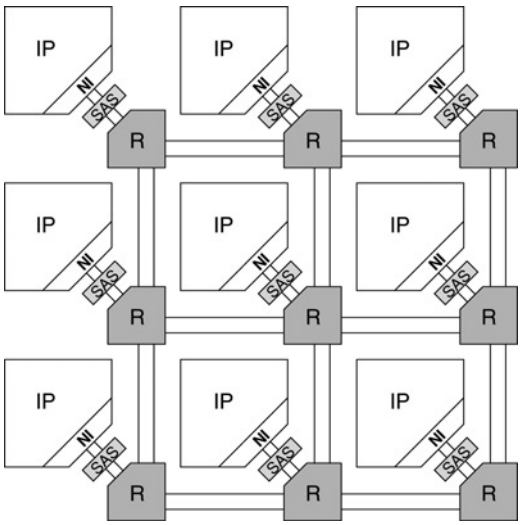

Figure 1 *ANOC architecture*

4 Proposed testing approach for ANOC

As mentioned in Section 2, structural testing approaches used for asynchronous circuits are expensive in terms of area overhead and test time. Moreover, the lack of testing supports from EDA companies makes the automatic test pattern generation (ATPG) for asynchronous circuits very difficult. Therefore our approach is based on the regularity and characteristics of the network architecture. The main challenge is to isolate each network element (routers and links), then to apply directly test patterns to the element-under-test and to obtain the test results.

To ease the test of the ANOC, we have proposed a DfT architecture as illustrated in Fig. 2. In this architecture, each asynchronous network router is surrounded by an asynchronous test wrapper in order to improve the controllability and the observability of the routers. The network links are reused to establish high bandwidth TAMs.

The test wrappers are used: (a) to insert test vectors to the network elements-under-test and to obtain the test results; (b) with network links, to establish high bandwidth asynchronous TAMs to transport test data.

To test a network link between routers, two test wrappers are used to insert test vectors and to obtain the test results.

All operations of the test wrapper are controlled by its own control module, named wrapper control module (WCM). To establish a test flow for the whole architecture, a dedicated 2-bit configuration chain (dash line) is built by connecting serially all wrappers' control modules. Test flows of the whole architecture are defined by an external controller. This controller is integrated in a unit, named generator–analyser–controller (GAC). The role of GAC unit is to generate test vectors, to generate test configurations and to

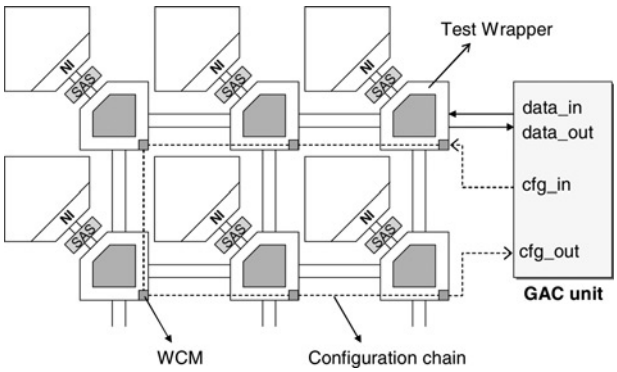

Figure 2 *Proposed DfT architecture*

analyse the test results. The GAC unit can be implemented on-chip as a BIST unit or off-chip as a computer program. In our experiment, the GAC unit has been implemented on an FPGA kit to be able to test the silicon chip.

The communication interface between the GAC unit and the ANOC architecture is composed of 34-bit input/output channels ('data_in' and 'data_out') and 2-bit input/output test configuration chains ('cfg_in' and 'cfg_out').

The proposed DfT architecture has been designed and implemented in QDI asynchronous logic style for many reasons: well adapted to the ANOC architecture, no dedicated test clock required, and no asynchronous $\Leftrightarrow$ synchronous interface overhead. In addition, QDI design is easier to be tested since every transition is necessary for the good operation of the circuit and QDI circuits are proven to stall in case of single stuck-at output faults [20]. These stalls are easily detected in the GAC unit as the impossibility to progress in the four-phase protocols of the asynchronous logic. Using a watchdog, it is indeed possible to detect that either the acknowledgement is never given back on an input or an expected output never arrives or never ends.

The reuse of the inter-router links allows not only to obtain a high bandwidth TAM but also to avoid wire congestion problems at layout processing phase. The proposed architecture has been designed for the ANOC, but can also be used for other QDI asynchronous NoCs or interconnects.

5 Design and implementation

In this section, we present how to design and implement the test wrapper, the basic element of the proposed DfT architecture. The DfT architecture is then simply realised by assembling these test wrappers.

5.1 Test wrapper architecture

The role of the test wrapper is to transport test vectors to the router-under-test and to obtain the test results. Given the number of input/output ports of the ANOC router, the test wrapper is composed of five input test cells (ITC), five output test cells (OTC) and a WCM, as described in Fig. 3. The ITCs and OTCs are alternatively interconnected to establish a boundary-scan path around the router. The reason to interconnect alternatively the ITCs and the OTCs is to optimise test wrapper's area cost and wire implementation. The role of the WCM is to control all test cells to do the test wrapper's function.

In addition, to reduce test time and to minimise test complexity, a bypass function is always preferred in DfT architectures. It allows reducing the length of test paths using short circuits between the inputs and the outputs of test wrappers. With this bypass function, only the router-under-test is set up in test mode. Therefore the router-under-test seems to be directly connected to GAC unit, whereas the other routers are inactive. Fig. 3 illustrates a bidirectional bypass (dashed bold lines) between EAST input/output ports and WEST input/output ports. The test data from the EAST input port of the test wrapper are directly transported to the WEST output port of the test wrapper and the data from the WEST input port of the test wrapper are directly transported to the EAST output port of the test wrapper, without passing the router or passing the other test cells.

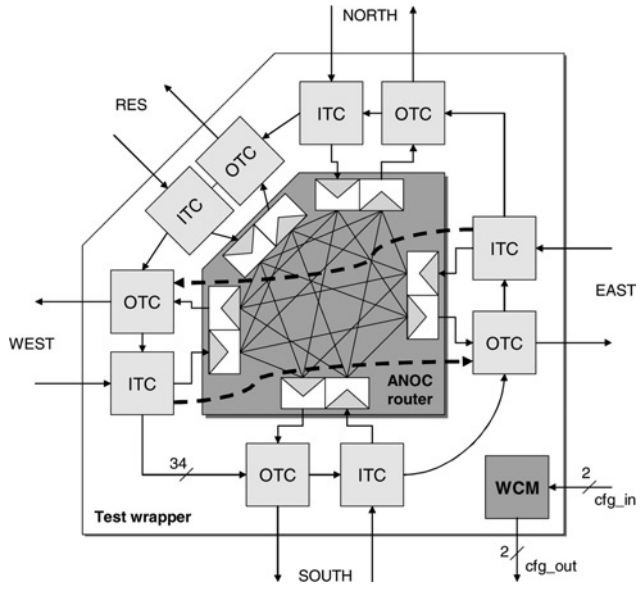

Figure 3 *Proposed test wrapper with router*

Because the external interface of test wrappers is similar to the external interface of network routers, there is no change to the network architecture, except that a 2-bit test configuration chain is added. As ANOC, the proposed test wrappers are fully designed and implemented in QDI asynchronous logic style. We use the four-phase return-to-zero (RTZ) signalling protocol for asynchronous channels, associated with a weak-condition half buffer reshuffling for pipelined stages. To obtain low-power consumption, the full data path (32 bits + BoP + EoP) is entirely designed with the 1-of-4 (MR4) encoded data [21], requiring 17 MR4 QDI connections (an MR4 connection includes four rails and an acknowledge signal). The 2-bit test configuration path is also designed with the 1-of-4 encoded data, requiring an MR4 connection. The other control channels are designed with the 1-of-2 encoded data (DR: dual-rail) or single-rail (SR)-encoded data connections (a DR connection includes two rails and an acknowledge signal, and a SR connection includes a rail and an acknowledge signal). Fig. 4 describes the structure of network links.

5.2 Test cells micro-architecture

The main components of the test wrapper are test cells (ITCs and OTCs). The function of test cells can be described as follows.

doi: 10.1049/iet-cdt.2008.0072

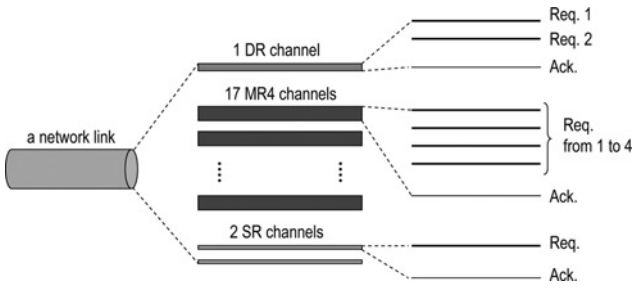

Figure 4 *Structure of a network link*

In normal mode, communication data are transported from network to router via ITCs and from router to network via OTCs, with no need of test control, aside from the global signal disabling test mode.

In test mode, the operations of an ITC can be explained as follows: test data from either the network or the previous test cell are stored on this test cell, and then the stored test data will be transported to either the router-under-test or the next test cell. It is similar for an OTC: test data from either the router-under-test or the previous test cell are stored on this test cell, and then the stored test data will be transported to either the network or the next test cell. All these operations are controlled by control channels provided by test wrapper's WCM.

To implement the above functions, we have developed the micro-architecture of the test cells (for both ITCs and OTCs) as illustrated in Fig. 5.

This micro-architecture is composed of two asynchronous multiplexers (MUX and MODE) and two asynchronous splitting blocks (S1 and S2). With this architecture, communication data are transported from 'noc-in' input to 'noc-out' output via MODE in normal mode. In test mode, test data from either 'noc-in' input or 'cell-in' input are stored on the asynchronous channel 'LOCAL' between MUX and MODE/next-cell, thanks to memorising characteristics of asynchronous channel design, then the stored test data will be transported to either 'noc-out' output or 'cell-out' output. These operations are controlled

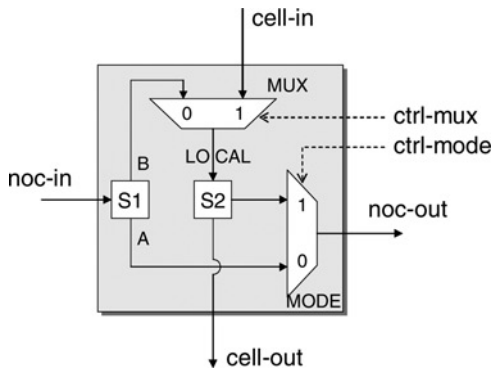

Figure 5 *Micro-architecture of ITC/OTC*

by two asynchronous channels, 'ctrl-mux' and 'ctrl-mode', provided by the WCM.

In normal mode, 'ctrl-mode' is only probed, and needs no acknowledge (using an asymmetric C-element) for minimal added latency and control.

To send data to either one of two sub-components, we use an un-controlled splitting block. The input channel 'noc-in' is hence split by S1 in two channels (A and B) that are connected to MODE and MUX. Similarly, the local memorising channel 'LOCAL' is split by S2 in two channels connected to MODE and 'cell-out'. These un-controlled splitting blocks are combinational and they can be easily implemented by combining the acknowledge signals (the acknowledge signals from MUX and MODE for S1, and the acknowledge signals from MODE and 'cell-out' for S2) in order to reduce both latency (a single gate) and area cost. The detailed implementation of these blocks is beyond the scope of this paper. In addition, in our architecture we use two accept signals ('accept0' and 'accept1') to control the data flows of the two virtual channels. Each multiplexer (MUX and MODE) generates a pair of accept signals. Thanks to their exclusive condition, the output accept signals at 'noc-in' are generated by merging the accept signals from these multiplexers. This can be done by a small combinational block. In order to simplify the representation of test cells, the implementation of send/accept signals is not included in Fig. 5.

The size of the test channels is the same as the size of the network links because the multiplexers and splitting blocks are implemented with 34-bit wide channels. Therefore each test channel has 17 MR4 QDI connections for data path, a DR connection for send signal, and two SR connections for two accept signals ('accept0' and 'accept1').

5.3 Test cells with the bypass function

As mentioned above, the bypass function is often preferred in DfT architectures. In our case, to implement a bypass function for the test wrapper, we have modified the test cells as illustrated in Fig. 6.

The most significant modification is made to the MODE multiplexer. For the ITCs, we add a bypass output channel ('bp-out') (Fig. 6*a*). For the OTCs, we add a bypass input channel ('bp-in') (Fig. 6*b*). The 'ctrl-mode' channel is now encoded by an MR3 encoded data to encode three mode values (normal, test, and bypass). In bypass mode, data from 'noc-in' will be transported to 'bp-out' output of the ITCs, and data from 'bp-in' input will be transported to 'noc-out' of the OTCs. To obtain minimal control, in bypass mode, 'ctrl-mode' is also probed as in normal mode. The following table resumes test cell's operations according to control channels' values (Table 1).

To receive a data from 'noc-in', store it on 'LOCAL' channel and then send it to 'noc-out', 'ctrl-mux' should be

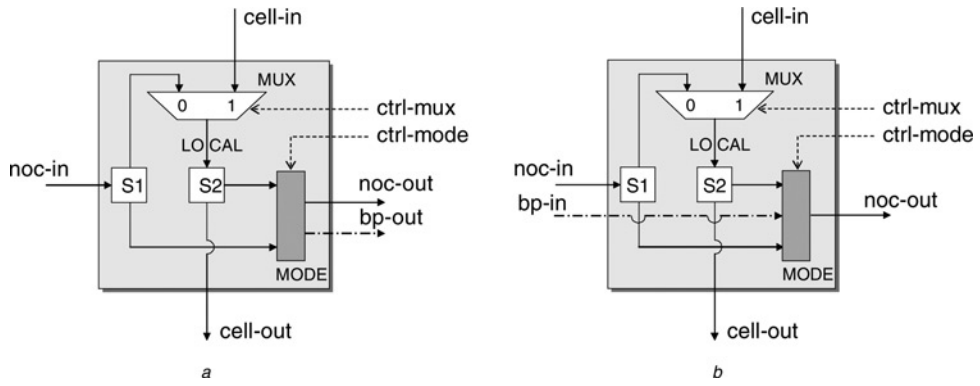

Figure 6 *Micro-architecture with bypass function*
a Input test cell
b OTC

Table 1 Description of test cells' operations

Ctrl-mode	Ctrl-mux	Description of test cell's operations
0	–	normal mode (enabled forever for minimal control)
–	0	receive data from 'noc-in' and store it
–	1	receive data from 'cell-in', then store it or transport it to 'cell-out' if there is a request from the next cell
1	–	transport stored data to 'noc-out'
2	–	bypass (enabled forever for minimal control)

Note: the sign (–) means that no value is written on the control channel

written with value '0' and 'ctrl-mode' should be written with value '1'.

5.4 Wrapper control module (WCM)

The operation of the test wrapper is decided by test configuration frames (TCFs) generated by GAC unit. In order to reduce the number of control channels provided by GAC unit, the whole TCF is intentionally split into 2-bit pieces, which are sent serially to WCM. In our architecture, each TCF includes 25 configuration pieces of 2 bits (encoded by the 1-of-4 data encoding scheme), as illustrated in Fig. 7.

The explanation of the TCF is presented in Table 2.

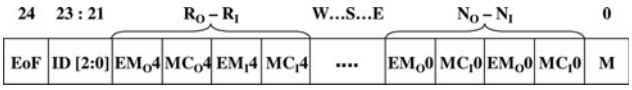

Figure 7 *Test configuration frame (TCF)*

Table 2 Explanation of the TCF

EoF	end of frame indicator
ID[2:0]	identification number of test wrapper
$EM_O[i]$	enable 'ctrl-mode' channel for OTC *i*
$MC_O[i]$	value of 'ctrl-mux' channel of OTC *i*
$EM_I[i]$	enable 'ctrl-mode' channel for ITC *i*
$MC_I[i]$	value of 'ctrl-mux' channel of ITC *i*
M	mode of test wrapper

Each configuration position is encoded by an MR4 data (2 bits), the value of each position therefore may be '0', '1', '2' and '3'[MR4] (i.e. {00}, {01}, {10} and {11} in binary, respectively). The value of '3'[MR4] is reserved for EoF indication, hence other configuration positions can use up to three values. With three configuration positions for test wrapper identification, the TCF can be used for NoC architectures up to 27 routers. For larger NoC architectures, we would need to add more identification positions.

The role of WCM is to gather the successive configuration values sent on the configuration chain until it forms the TCF for the test wrapper. When it has received a complete configuration frame by detecting an 'end-of-frame' (EoF) indication as the last piece of configuration, it has to determine whether the received configuration frame is reserved for the test wrapper or not. This is easily done by means of the 'identifier' (ID) field addressing the successive router test wrappers on the configuration chain. If the frame ID corresponds to one of test wrappers, WCM of the target test wrapper will generate control channels (according to the configuration frame's values) to control test cells.

To implement the above function, WCM is composed of a 'frame shifter', an 'EoF detector', an 'ID verifier' and a 'transferring block', as shown in Fig. 8.

doi: 10.1049/iet-cdt.2008.0072

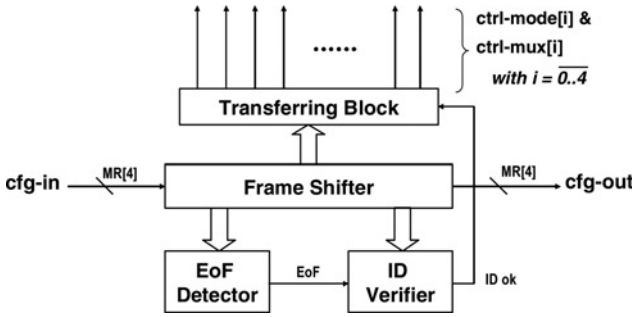

Figure 8 *Wrapper control module (WCM)*

The purpose of 'frame shifter' is to read a new configuration from the GAC unit via 'cfg-in' input, to shift the TCF one position (i.e. a 2-bit piece) to the right, and to write out the lowest significant configuration position to the configuration chain through 'cfg-out' output. The 'EoF detector' detects the completion of a TCF and the 'ID verifier' will verify the identification of the received TCF. The 'transferring block' is used as a driver to control all the test cells. It receives control values from 'frame shifter' when an EoF is detected with a good ID. The outputs of this block are 20 control channels ('ctrl-mode[i]' and 'ctrl-mux[i]', where i obtains values from 0 to 4 indicating NORTH, EAST, SOUTH, WEST, RES directions, respectively) used to control 10 test cells.

The value of mode (M) may be '0'[MR4] (normal mode), '1'[MR4] (test mode) or '2'[MR4] (bypass mode). The value written on the control channel 'ctrl-mode[i]' of ITCs or OTCs is the value of mode position. However, in test mode the 'ctrl-mode[i]' channel of the ITC i is written only when $EM_I[i]$ equals '1'[MR4], and the control channel 'ctrl-mode[i]' of the OTC i is written only when $EM_O[i]$ equals '1'[MR4]. The value written on each 'ctrl-mux[i]' channel of ITCs or OTCs depends on the value of $MC_I[i]$ or $MC_O[i]$, respectively: the 'ctrl-mux[i]' channel of the ITC i obtains '0' when $MC_I[i]$ equals '1'[MR4], it obtains '1' when $MC_I[i]$ equals '2'[MR4]; similarly, the 'ctrl-mux[i]' channel of the OTC i obtains '0' when $MC_O[i]$ equals '1'[MR4], it obtains '1' when $MC_O[i]$ equals '2'[MR4]. We can resume the relations between TCF and control channels explained above in Table 3.

Now, we take an example of testing the NORTH→ SOUTH routing path of a router, as illustrated in Fig. 9. In this example, we suppose that the eastern port (EAST) of test wrapper is connected to TAM.

We can easily generate the TCF for this test, see Table 4.

This TCF can be explained as follows: the most significant value (EoF = '3'[MR4]) indicates the end of TCF. The next three values (ID = '001'[MR4]) means that this TCF is used for the test wrapper number 1. The lowest significant value (Mode = '1'[MR4]) puts the test wrapper in test mode. E_I = '01' (i.e. $EM_I[1]$ = '0' and $MC_I[1]$ = '1') means that the ITC-1 (EAST) receives test data from the TAM and

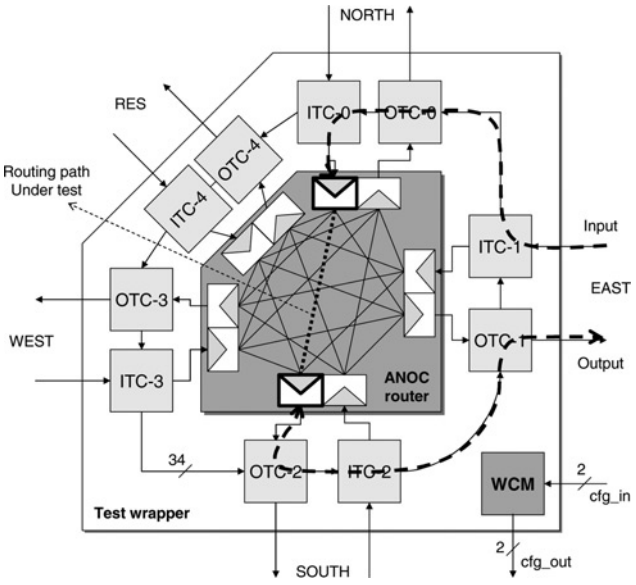

Figure 9 *Wrapper configuration for testing the NORTH → SOUTH routing path*

Table 3 Relation between TCF and control channels

	For direction i				OTC i		ITC i	
M	EM_O	MC_O	EM_I	MC_I	ctrl-mode	ctrl-mux	ctrl-mode	ctrl-mux
0	X	X	X	X	0	–	0	–
2	X	X	X	X	2	–	2	–
1	0	0	0	0	–	–	–	–
1	1	0	1	0	1	–	1	–
1	0	1	0	1	–	0	–	0
1	0	2	0	2	–	1	–	1

Note: the sign (–) means that no value is written on the control channel

Table 4 TCF for testing NORTH–SOUTH routing path

EoF	ID	R_O–R_I	W_O–W_I	S_O–S_I	E_O–E_I	N_O–N_I	M
3	001	00–00	00–00	01–02	12–01	02–12	1

shifts them to the OTC-0 (NORTH). N_O = '02'[MR4] means that the OTC-0 is in shifting mode, hence the test data are shifted to the ITC-0 (NORTH). The test data are then inserted to the router-under-test by the ITC-0 because N_I = '12'[MR4].

After router's operations, with S_O = '01'[MR4] the OTC-2 (SOUTH) receives the test result from the router-under-test and transports them to the ITC-2 (SOUTH). The ITC-2 is in shifting mode because S_I = '02'[MR4]. The test result is therefore shifted to the OTC-1 (EAST). With E_O = '12'[MR4] the OTC-1 receives the test result and send it to test analyser (integrated in GAC unit) via TAM.

We note that the TAM is established by other test wrappers and the network links. Other test configuration positions such as R_O, R_I, W_O, W_I equal '00'[MR4] mean that the following test cells: OTC-4 (RES), ITC-4 (RES), OTC-3 (WEST) and ITC-3 (WEST), are not concerned and are not controlled.

The objective of the above example is to show how to generate a TCF. Actually, these TCFs are automatically generated by a custom program according to the test strategy.

6 Test pattern generation

6.1 Fault model for QDI asynchronous circuits

Synchronous circuit operation is based on the logical values observed on nets at every clock cycle. Synchronous circuit testing is hence based on control and observation of the logical levels of the nets. Conventional synchronous ATPG methods use this property to force flip-flop outputs to different logical levels, and observe levels at flip-flop inputs in order to detect single stuck-at faults on the circuit-under-test. ATPG tools determine value combinations on all flip flops in order to detect a fault.

However, QDI asynchronous circuit operation is based on tokens flowing inside the logic. 1-of-N RTZ encoding is used to propagate tokens: every net alternates between logical '1' (data present) and logical '0' (no data) during operation. Fig. 10 describes this behaviour in case of 1-of-2 RTZ encoding QDI asynchronous circuits. We can see that a stuck-at net will impede this alternation, and block one of the handshaking loops, and all the following, as illustrated in Fig. 11. No data will be observed at the output. Furthermore, multiple stuck-at nets on different handshaking loops will block the design under test in the same way and can also be detected.

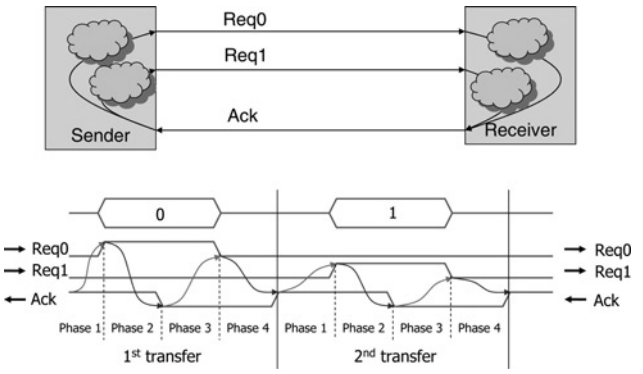

Figure 10 *Behaviour of 1-of-2 RTZ encoding QDI asynchronous circuits*

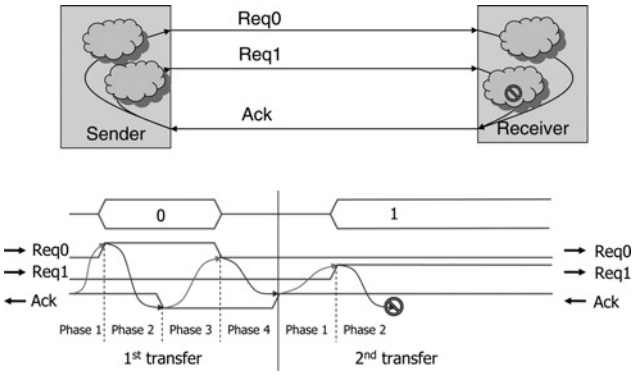

Figure 11 *Circuit is blocked by a stuck-at net 'Req1'*

In order to observe stuck-at faults on the nets of a QDI asynchronous circuit, it is thus sufficient to force a handshake on each net and check for liveness. There is no need for observation of the results. Test pattern generation is therefore simpler, but since there is no existing automated tool, patterns must be manually derived from a functional analysis of the circuit. Since NoCs are interconnection structures, not processing structures, there is very little control needed in order to excite all nets, and the test wrapper only needs to be configured to inject a vector at an input and collect it at an output.

6.2 Test patterns for network links

In ANOC, network links are implemented in asynchronous QDI logic. Each network link is composed of 17 MR4 channels, 1 DR channel and 2 SR channels, as illustrated in Fig. 4. MR4 channels are used to transport communication data, DR channel ('send') is used to indicate on which virtual

doi: 10.1049/iet-cdt.2008.0072

channel the data are being transported and SR channels ('accept0' and 'accept1') are used for indicating the availability of the corresponding virtual channels. An MR4 channel includes four request rails and an acknowledge rail. A DR channel includes two request rails and an acknowledge rail, and a SR channel includes a request rail and an acknowledge rail.

Considering the network's functionality, network links need no control at all: they transmit the values at a link input to the link output. The test of a network link is equivalent to 17 independent tests of 17 MR4 channels, plus the test of DR and SR channels. Exhaustive test vectors for 17 MR4 channels are given by the four possible values on each 1-of-4 digit of the data signals, that is '0.0.0. . . .0', '1.1.1. . . .1', '2.2.2. . . .2'; '3.3.3. . . .3'. These vectors are sent from the test wrapper of a router to the test wrapper of a neighbouring router. Besides, to test the virtual channel indicating signal ('send' signal), that is, DR channel, a 1-of-2 digit with two possible values ('0' and '1') needs to be used with these above test vectors. The 'accept' SR channels will be naturally tested in response to the acceptance of a flit on the 'data + send' signals by the neighbouring router's test wrapper. Table 5 presents the test vector set for each network link. Finally, we need only four test vectors to test a network link. Because all network links are identical, these test vectors are used to test all network links.

6.3 Test patterns for network routers

The functionality of an ANOC router can be explained as follows: the computational resources exchange data packets, which can be individually routed in the network. A packet is composed of successive flits (a flit is a network flow control unit that is composed of 32 data bits and 2 control bits). A packet is always composed of a header flit, which may be followed by zero or more body flit(s) and a tail flit, as mentioned in Section 3. The header flit provides routing information, indicated on 'path-to-target' field, which is a vector containing the successive directions to follow. Each router uses two lower bits of this vector to route the whole packet in the given direction, and shift the 'path-to-target' field so that it can be used by the following router. The body flit contains communication data and the tail flit informs the router that it is the end of the packet.

Each input unit is composed of an input block (IN) that receives data from the input. These data are then demultiplexed to two virtual channel blocks (VC0 and VC1). Similarly, each output unit is composed of two virtual channel blocks (VC0 and VC1) and an output block (OUT) that permit to multiplex the data from two virtual channel blocks and transport them to the output.

Table 5 Test vector set for each network link

Data (all 17 '1-of-4' digits)	Send (1-of-2 digit)
'0.0.0. . .0'	'0'
'0.0.0. . .0'	'1'
'0.0.0. . .0'	'0'
'0.0.0. . .0'	'1'

The test of a router can be realised in the following way: test vectors are inserted at one of the input ports and the test responses will be observed at the output ports. To drive the test vectors to the router-under-test, these test vectors have to be encapsulated as data packets and have to respect the format of data packet used on the network. Given the structure presented in Fig. 12, functional paths in the router are determined by:

- the BoP/EoP information which indicates the presence of a path and the need to shift it or not,
- the direction information which indicates where to route the data and
- the VC information which indicates in which sub-block the flits are to be stored.

The injecting test packet into different input units is realised by configuring test wrapper.

To obtain a complete coverage of the router-under-test, the format of a test packet should be as illustrated in Fig. 13.

The structure of an input/output port of the router is identical to the structure of a network link, as presented in the previous subsection, and is composed of 17 asynchronous MR4 channels for data, 1 DR channel and 2 SR channels for control signals 'send-accept'. So, we can use the same way to generate test vectors as done for network links. All the '1-of-4' digits of a test flit which do not participate to control should be forced independently to all possible values ('0', '1', '2' and '3'), that is 15 digits, as shown in Table 6.

We can see that these test vectors allow testing the 'data' part of different blocks (IN, VCvc.) of the input unit specified by

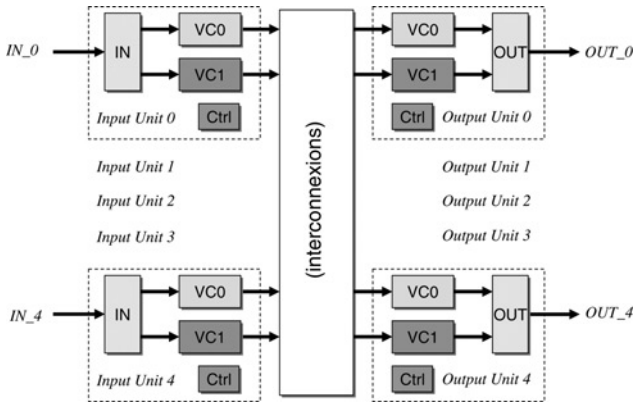

Figure 12 *Analysing the structure of an ANOC router*

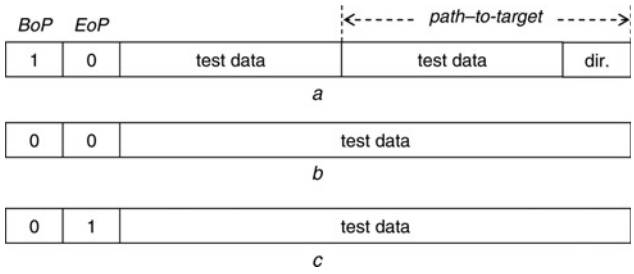

Figure 13 *Format of a test packet*

Table 6 Multi-flit test packet for a triplet 'input/output/virtual channel' of router

Data			Send, virtual channel
BoP/EoP	15 '1-of-4' digits	Direction	
2	'0.0.0. . .0'	'dir.'	'vc.'
0	'0.0.0. . .0'	'0'	'vc.'
0	'1.1.1. . .1'	'1'	'vc.'
0	'2.2.2. . .2'	'2'	'vc.'
1	'3.3.3. . .3'	'3'	'vc.'

test wrapper and the different blocks (VCvc., OUT) of the output unit specified by the routing direction ('dir.'). However, to cover the 'control' part of each input/output unit, we have to consider all possible combinations of control digits. Hence, the 1-of-4 digit that contains the BoP/EoP information, all 1-of-4 digits in 'path-to-target' field, and the 1-of-2 digit that indicates the virtual channel to use, must be set to all possible combinations in order to test every net of each input–output couple. To do that, we have defined three additional single-flit packets, as described in Table 7.

Finally, given an input, an output and a virtual channel, we need eight test vectors to cover the whole functionality range. Since a router has five inputs, four outputs connected to each input, and two virtual channels, this gives a total of 320 test vectors in order to test a whole router. These test vectors are inserted to the router-under-test by configuring the test

Table 7 Three single-flit test packets for a triplet 'input/output/virtual channel' of router

Data			Send, virtual channel
BoP/EoP	15 '1-of-4' digits	Direction	
3	'1.1.1. . .. . .1'	'dir.'	'vc.'
3	'2.2.2. . .. . .2'	'dir.'	'vc.'
3	'3.3.3. . .. . .3'	'dir.'	'vc.'

wrapper. The associated test configurations can be computed by a custom program and give a total of 640 TCFs since vector injection and collection are separated.

Section 8.4 will show the coverage on stuck-at faults given by those functional vectors. The following section will present how to test the whole ANOC network.

7 Testing strategy

In this section, we present a simple testing strategy that allows testing all network elements of the ANOC. With this strategy, only one network router is tested at a time. Then, we test all the network links that are connected to this router. Once the current router and its network links are tested, we will configure the test wrapper to put this router in bypass mode and go to test the next router and its network links. The test flow is defined by the test configuration chain and indicated by bold arrow, as illustrated in Fig. 14.

With this test flow, we can define the testing algorithm in Fig. 15.

The testing strategy presented above requires that the DfT architecture operates correctly. Hence, the DfT architecture needs to be tested in a preliminary phase before applying the test of the routers and links. This test can be done by setting all test wrappers in transfer mode and the corresponding test vectors can be generated using the same principle as the test vector generation of network links.

The implemented DfT architecture is targeted to test the network architecture (routers and links), but it can also be used to test the computational resources and their NIs. The detail of this work is out of the scope of this paper, and has been presented in [22]. Thanks to the reuse of the network links as a high throughput TAM to transport test data, no dedicated test bus is required.

8 Experimental results

8.1 Area overhead

The proposed asynchronous DfT architecture has been implemented using a 65 nm CMOS technology from

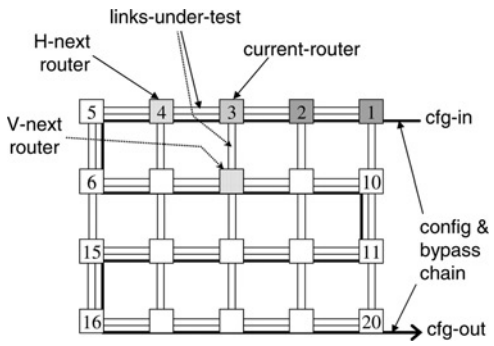

Figure 14 *Test flow for ANOC network*

doi: 10.1049/iet-cdt.2008.0072

TESTING ALGORITHM FOR ANOC NETWORK

```
Set current-router = 1;
While current-router ≤ N do
   /* N is number of network routers */
/***** Router test *****/
   Set current-router in test mode;
   /* see Figure 16 for different modes used in test process */
   Apply all router test vectors (320) to the router-under-test;
   /* A test vector is applied with its 2 TCFs */
/***** Network Link test *****/
   If current-router is not in the last row of the network then
      Set current-router in transfer mode;
      /* Transfer direction towards SOUTH (V-next router */
      Set V-next-router in loop-back mode;
      /* because of loop-back, a single vector tests both directions of bidirectional links */
      Apply all link test vectors (4) to the vertical links-under-test;
      /* links-under-test is the vertical links between current router and V-next router */
   End if;
   If the current-router is not the last router of the row-under-test then
      Set current-router in transfer mode;
      /* Transfer direction towards H-next router */
      Set H-next-router in loop-back mode;
      Apply all link test vectors (4) to the horizontal links-under-test;
      /* links-under-test is the horizontal links between current router and H-next router */
   End if;
/***** Prepare next test iteration *****/
   Set current-router in bypass mode;
   /* Set current-router in bypass mode to go to the next-router. Note that the bypass direction is
   de-fined by the test flow, see Figure 14 */
   current-router = current-router + 1;
End while;
```

Figure 15 *Testing algorithm*

doi: 10.1049/iet-cdt.2008.0072

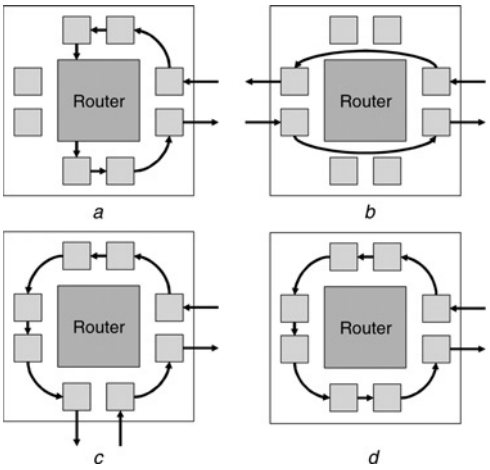

Figure 16 *Testing configurations*
a Test mode
b Bypass mode
c Transfer mode
d Loop-back mode

STMicroelectronics (with the TIMA 65 nm asynchronous library [23]). The area of a test cell is 8560 μm^2 and the area of WCM is 10 400 μm^2. In consequence, a test wrapper composed of five ITCs, five OTCs and a WCM block costs 96 000 μm^2, that is, 32.7% of the area of a testable asynchronous network router. In order to give a visible comparison about the area overhead, the FFT block in our system chip is implemented with an area cost of 1 600 000 μm^2 (250 Kgates).

However, the proposed DfT architecture is used to test not only the network architecture, but also as a TAM to test other parts of the ANOC-based system (computational resources and their NIs) that are not presented in this paper. The effective DfT area cost is hence to be shared between all computational resources of the whole chip. In consequence, the area overhead is about 3–5% of chip area depending on the size of the target chip (in the case of FAUST project [18], the area overhead of the proposed DfT architecture is 4.8%).

8.2 Additional latency

Test wrappers are used to improve the accessibility, testability of ANOC architecture, but they lead to additional latency for ANOC communication in normal mode. In our implementation, the additional latency is 0.34 ns for a pair of input and output ports compared to router latency of 2.00 ns (post-layout measurement), see Fig. 17.

This corresponds to the delay in two asymmetric C-elements and two AND gates. However, since the handshaking loops at the inputs/outputs of the router are not the most critical ones (in QDI asynchronous circuits, performance is determined by the length of the handshaking loops) there is no degradation of the NoC communication throughput at all, that is 500 Mflits/s in normal mode.

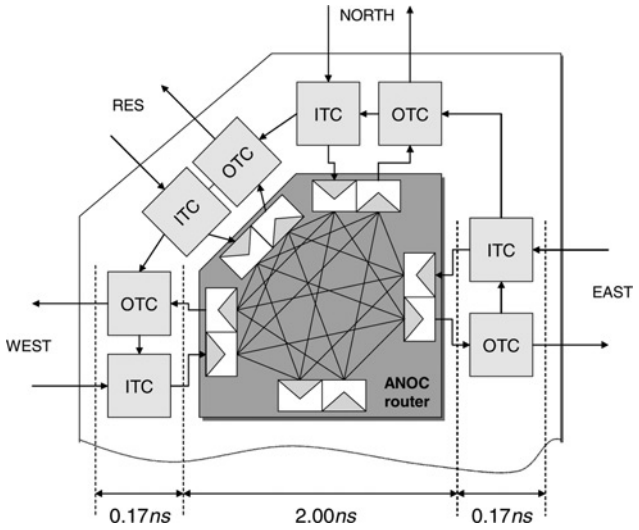

Figure 17 *Additional latency*

8.3 Test application time

By reusing network links as TAM, the proposed DfT architecture possesses high-bandwidth test paths. However, we need to send TCFs in order to control injection and collection of the test vectors. Throughput on the configuration chain is of about 2 ns per symbol, that is, 50 ns for a complete TCF.

In the test of network links, we need two TCFs per test vector for both inserting test vector and observing test result because two test wrappers are used. For router testing, in most cases, a single TCF would be sufficient to insert a test data flit and to observe the test result, and hence the maximum test speed would be about 20 Mtest-vectors/s; however, to simplify the TCFs generation, we use two TCFs per test vector in the test of network routers: one for inserting test vector and other for observing test result, which is the general case. This gives a test speed of about 10 Mtest-vectors/s. With generated test vectors, the test time for each network router is 32 μs and the test time for each bi-directional network link is 0.4 μs.

In consequence, for an ANOC of 20 network routers, the total test application time for the whole NoC is less than 0.7 ms, whereas the test application time for the FFT block in our system chip is about 7.114 ms.

8.4 Fault coverage

To evaluate the efficiency of the testing approach, we use the well known single stuck-at fault model on either inputs or outputs of the circuit cells. Because we did not dispose of a support tool to calculate fault coverage for asynchronous design, we have built a script-based program to insert automatically faults on all pins of every standard cell of the router (a single fault at once), simulate and calculate the detected faults. A fault is detected if the circuit stalls (no more transitions on outputs or acknowledge signals) before the expected end of the test. With the generated test

doi: 10.1049/iet-cdt.2008.0072

Table 8 Single stuck-at (SSA) fault coverage report for router testing

Circuit-under-test	SSA faults on outputs	SSA faults on inputs	SSA faults on both inputs and outputs
input controller	3178/3178 (100%)	6016/6016 (100%)	9194/9194 (100%)
output controller	5185/5198 (99.74%)	8925/8944 (99.78%)	14 110/14 142 (99.77%)
entire router	41 815/41 880 (99.84%)	74 705/74 800 (99.87%)	116 520/116 680 (99.86%)

vectors, the proposed testing method presents fault coverage of 99.86% for the network routers, 100% for the network links. Coverage of the DfT architecture itself is 99.75% using all possible test vectors. This high coverage is made possible because of the dataflow architecture of the network router, mainly based on multiplexers and demultiplexers on a pipeline-like structure, and the token-based design style used in ANOC: the only faults that were not detected in the router were localised before the inputs of the very few asymmetric C-elements used in the design to check for race conditions on control tokens used for several data tokens.

Table 8 presents a brief report on the number of injected faults, the number of detected fault, and the fault coverage of each router's parts as well as an entire network router.

The pin-based single stuck-at fault model used here is giving good results because the logical effect of a stuck-at fault is to prevent the occurrence of a transition, and therefore in most cases to absorb a token in the asynchronous system. Other types of faults may lead to the appearance of a new transition, which translates to the generation of a new token in another part of the system (shorts, etc.). These faults will not immediately lead to a stall, and could not be detected using the test vectors described before. With a repeated application of the test, though, the additional tokens created by this kind of faults would eventually fill the whole chain and stall the circuit because of congestion in the system. Supposing that the fault occurs at each application of the test, the system would stall at most after a number of applications of the test equal to the pipeline depth of the router, that is, nine in the case of ANOC. This might lead to a total test application time of about 7 ms for the NoC in order to try to catch this kind of faults.

8.5 Fault diagnosis and location of the defect

The proposed DfT architecture allows not only detecting the stuck-at faults, but also allows determining the location of defects which generate these stuck-at faults, that is, we can make a diagnostic with the proposed architecture. The advantage is that we can locate the detected faults in the circuit-under-test and then finding manufacturing problems that create these faults. In consequence, the diagnostic helps us to improve the production procedures in order to avoid these defects in the suite.

In fact, by analysing the router structure we can see that the control part of the router occupies about 10% logic gates of the router, and the number of logic gates of an input unit is approximately the number of logic gates of an output unit. In addition to the control part, each input unit and output unit can be dissociated in three sub-parts that have an equivalent number of logic gates, as illustrated in Fig. 12.

In case of a single stuck-at fault, the given set of 320 vectors allows to isolate immediately the sub-part containing the fault; that is one among 40 sub-parts. If the fault is on a data part, independence of the MR4 channels allows isolating the error on 1/600 of the router using 15 additional vectors computed during analysis. Furthermore, if all four values of this digit are rejected, one can conclude that the fault is on the acknowledgement, but if only one of the four values is rejected, the fault is on one of the 1-of-4 rails. Hence, a defect can be located on 1/3000 of the router. Since the router counts about 20 000 instances, this allows identifying the location of the defect with a precision of $\sim$7 gates.

8.6 Router reconfiguration in case of a single defect

Thanks to the bypass channels added for the test, functional connectivity within the network may be preserved by configuring the defective router to use the bypass channels rather than go through the defective logic. The routing paths need to be reconfigured accordingly, by splicing the path to target to remove the unused direction. This allows using defective chips in a degraded mode, either for application to reduce the costs of a new batch or for deeper inspection, during the test/debug phase.

9 Conclusions

Although asynchronous NoCs are likely to become good solutions for on-chip communication of GALS systems, they have serious testability issues that need to be addressed. Therefore a DfT-based testing approach for asynchronous NoCs has been proposed. In the case of ANOC architecture, a fully asynchronous DfT architecture has been designed and implemented. It is a configurable architecture that can be adapted to other asynchronous NoCs with different topologies.

In the paper, we have presented the design and the implementation of the proposed DfT architecture in a STMicroelectronics 65 nm CMOS technology, using a specialised asynchronous standard cell library from the TIMA laboratory [23]. The validation of the implementation was done using a custom SystemC/Verilog netlist co-simulation

platform. To prove our testing approach, the DfT architecture has been integrated into an actual asynchronous NoC-based SoC.

A simple method for generating test patterns is also proposed. The test vectors and corresponding test configurations are automatically generated by a custom program. With these test vectors, the testing approach presents high test coverage of 99.86% (using single stuck-at fault models).

10 Acknowledgment

This work is supported in part by Vietnam National University, Hanoi (VNU) under VENGME project.

11 References

[1] CHAPIRO D.M.: 'Globally asynchronous locally synchronous systems'. PhD thesis, Stanford University, 1984

[2] JANTSCH A., TENHUNEN H.: 'Networks on chip' (Kluwer Academic Publisher, 2003)

[3] BAINBRIDGE J., FUBER S.: 'CHAIN: a elay-insensitive chip area interconnect', *IEEE Micro*, 2002, **22**, (5), pp. 16–23

[4] LINES A.: 'Asynchronous interconnect for synchronous SoC design', *IEEE Micro*, 2004, **24**, (1), pp. 32–41

[5] BEIGNÉ E., CLEMIDY F., VIVET P., CLOUARD A., RENAUDIN M.: 'An asynchronous NoC architecture providing low latency service and its multi-level design framework'. Proc. 11th IEEE Int. Symp. Asynchronous Circuits and Systems (ASYNC), March 2005, pp. 44–53

[6] DOBKIN R., VISHNYAKOV V., FRIEDMAN E., GINOSAR R.: 'An asynchronous router for multiple service levels networks on chip'. Proc. 11th IEEE Int. Symp. Asynchronous Circuits and Systems (ASYNC), NY, USA, March 2005, pp. 44–53

[7] VERMEULENT B., DIELISSEN J., GOOSSENS K., CIORDAS C.: 'Bringing communication networks on a chip: test and verification implications', *IEEE Commun. Mag.*, 2003, **41**, (9), pp. 74–81

[8] AMORY A.M., BRIAO E., COTA E., LUBASZEWSKI M., MORAES F.G.: 'A scalable test strategy for network-on-chip routers'. Proc. Int. Test Conf. (ITC), Texas, USA, November 2005, pp. 591–599

[9] HOSSEINABADY M., BANAIYAN A., NAZM M., NAVABI Z.: 'A concurrent testing method for NoC switches'. Proc. Design Automation and Test in Europe (DATE), Messe Munich, Germany, March 2006, pp. 1171–1176

[10] EFTHYMIOU A., BAINBRIDGE J., EDWARDS D.: 'Test pattern generation and partial-scan methodology for an asynchronous SoC interconnect', *IEEE Trans. VLSI Syst.*, 2005, **13**, (12), pp. 1384–1392

[11] TRAN X.T., DURUPT J., BERTRAND F., BEROULLE V., ROBACH C.: 'A DFT architecture for asynchronous networks-on-chip'. Proc. IEEE European Test Symp. (ETS), Southampton, UK, May 2006, pp. 219–224

[12] IEEE 1500 standard for embedded core test, IEEE 1500 Working Group, http://grouper.ieee.org/groups/1500

[13] NAHVI M., IVANOV A.: 'Indirect test architecture for SoC testing', *IEEE Trans. CAD Integr. Circuits Syst.*, 2004, **23**, (7), pp. 1128–1142

[14] COTA E., CARRO L., LUBASZEWSKI M.: 'Reusing an on-chip network for the test of core-based systems', *ACM Trans. Des. Autom. Electron. Syst.*, 2004, **9**, (4), pp. 471–499

[15] AMORY A.M., GOOSSENS K., MARINISSEN E.J., LUBASZEWSKI M., 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

[16] GÜRKAYNAK F.K., VILLIGER T., OETIKER S., *ET AL.*: 'A functional test methodology for globally asynchronous locally synchronous systems'. Proc. Int. Symp. Advanced Research in Asynchronous Circuits and Systems, April 2002

[17] SPARSO J., FURBER S.: 'Principles of asynchronous circuit design – a system perspective' (Kluwer Academic Publishers, 2001)

[18] LATTARD D., BEIGNE E., BERNARD C., *ET AL.*: 'A telecom baseband circuit based on an asynchronous network-on-chip'. Proc. Int. Solid State Circuits Conf. (ISSCC), San Francisco, February 2007

[19] MARTIN A.J.: 'Programming in VLSI: from communicating processes to delay-insensitive circuits', in HOARE C.A.R. (ED.): (Ed.): 'Developments in concurrency and communication' (Addison-Wesley, 1990), pp. 1–64

[20] HAZEWINDUS P.J.: 'Testing delay-insensitive circuits'. PhD thesis, CS-TR-92-14, California Institute of Technology, 1992

[21] BAINBRIDGE J., TOMS W., EDWARDS D., FURBER S.: 'Delay-insensitive, point-to-point interconnect using m-of-n codes'. Proc. 9th IEEE Int. Symp. Asynchronous Circuits and Systems (ASYNC), Canada, May 2003, pp. 132–140

[22] TRAN X.T.: 'Méthode de Test et Conception en Vue du Test pour les Réseaux sur Puce Asynchrones: Application au Réseau ANOC'. PhD thesis, Grenoble INP, France, 2008

[23] MAURINE P., RIGAUD J.B., BOUESSE F., SICARD G., RENAUDIN M.: 'Static implementation of QDI asynchronous primitives'. Proc. 13th Int. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), September 2003, pp. 181–191

doi: 10.1049/iet-cdt.2008.0072