



# GALAXY

GALS InterfAce for CompleX Digital  
SYstem Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

## Deliverable – D16

### *GALS Test flow*

<b>Grant Agreement No:</b>	214364
<b>Project acronym:</b>	GALAXY
<b>Project title:</b>	GALS InterfAce for CompleX Digital System Integration
<b>Funding Scheme:</b>	STREP
<b>Date of latest version of Annex I against which the assessment will be made:</b>	28.10.2008.
<b>Contractual Date of Delivery to the EC:</b>	31. Oct. 09
<b>Actual Date of Delivery to the EC:</b>	31. Oct. 09
<b>Author(s):</b>	Milos Krstic, Steffen Zeidler, Tatjana Nikolic (IHP), John Bainbridge (STX), M. Stanisavljevic (EPFL)
<b>Participant(s):</b>	IHP, INFINEON, EPFL
<b>Work Package:</b>	WP5
<b>Security:</b>	Public
<b>Nature:</b>	Report
<b>Version:</b>	3
<b>Total number of pages:</b>	46

#### **Abstract:**

This is a report that describes the possibilities for testing the GALS circuits. In this report we analyze the possibilities for test of the synchronous and the asynchronous part of a GALS system. In addition to that, we discuss the methods for joint testing of the all system components. In this report we summarize all major test methods for GALS systems and set the guidelines for GALS testing.

**Keyword list: asynchronous design, testing, scan, BIST**



# GALAXY

GALS InterfAce for CompleX Digital  
SYstem Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

<i>Function</i>	<i>Responsibility</i>	<i>Date</i>	<i>Signature</i>
<b>Written by:</b>	Milos Krstic, Steffen Zeidler, T. Nikolic, John Bainbridge, M. Stanisavljevic	1.10.2009	
<b>Checked by:</b>			
<b>Approved by:</b>			

Reserved to EC

<b>Approved by:</b>			
---------------------	--	--	--



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA



Never stop thinking



# GALAXY

GALS InterfAce for CompleX Digital  
SYstem Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

## CHANGE RECORDS

<i>ISSUE</i>	<i>DATE</i>	<i>§ : CHANGE RECORD</i>	<i>AUTHOR</i>
1	1-Nov-08	Asynchronous circuit testing	Steffen Zeidler
2	7-Sep-09	GALS Testing	T. Nikolic, M. Krstic
3	31-Oct-09	Revision	S. Zeidler, M. Krstic, John Bainbridge, M. Stanisavljevic



# GALAXY

GALS InterfAce for CompleX Digital  
SYstem Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

## BIBLIOGRAPHIC RECORD

Project Number:	214364
Project Title:	GALAXY
Deliverable Type:	Report
Deliverable Number:	D16
Contractual Date of Delivery:	31. Oct. 2009
Actual Date of Delivery:	31. Oct. 2009
Title of Deliverable:	GALS Test Flow
Work package contributing to the Deliverable:	WP5
Authors:	Milos Krstic, Steffen Zeidler, T. Nikolic, John Bainbridge, M. Stanisavljevic
Abstract	This is a report that describes the possibilities for testing the GALS circuits. In this report we analyze the possibilities for test of the synchronous and the asynchronous part of a GALS system. In addition to that, we discuss the methods for joint testing of the all system components. In this report we summarize all major test methods for GALS systems and set the guidelines for GALS testing.
Keywords	asynchronous design, testing, scan, BIST
Confidentiality Level	Public
Name of Client:	EC
Distribution List:	GALAXY, EC, internet
Authorised by:	Milos Krstic
Issue:	3
Document ID:	D16
Total Number of Pages:	46
Contact Details:	<a href="mailto:krstic@ihp-microelectronics.com">krstic@ihp-microelectronics.com</a>



## TABLE OF CONTENTS

<b>1</b>	<b>INTRODUCTION .....</b>	<b>7</b>
<b>2</b>	<b>REFERENCES .....</b>	<b>9</b>
2.1	<b>ACRONYMS .....</b>	<b>9</b>
2.2	<b>REFERENCE DOCUMENTS .....</b>	<b>10</b>
<b>3</b>	<b>TESTING DIGITAL SYSTEMS .....</b>	<b>14</b>
3.1	<b>DESIGN FOR TESTABILITY (DFT) .....</b>	<b>14</b>
3.2	<b>FAULT MODELS .....</b>	<b>15</b>
3.3	<b>DFT TECHNIQUES .....</b>	<b>16</b>
3.4	<b>STUCK-AT FAULT TESTING .....</b>	<b>17</b>
3.5	<b>SUMMARY .....</b>	<b>17</b>
<b>4</b>	<b>TESTING ASYNCHRONOUS CIRCUITS .....</b>	<b>18</b>
4.1	<b>INTRODUCTION .....</b>	<b>18</b>
4.2	<b>ASYNCHRONOUS HANDSHAKE CIRCUITS .....</b>	<b>18</b>
4.3	<b>TESTING ASYNCHRONOUS CIRCUITS .....</b>	<b>21</b>
4.4	<b>CONCLUSION .....</b>	<b>28</b>
<b>5</b>	<b>GALS TESTING .....</b>	<b>29</b>
5.1	<b>GALS TESTING PROBLEMS .....</b>	<b>29</b>
5.2	<b>SCAN ISSUES WITH MULTI-CLOCK-DOMAIN DESIGNS .....</b>	<b>30</b>
5.3	<b>AT-SPEED SCAN ARCHITECTURES FOR GALS SYSTEMS .....</b>	<b>31</b>
5.4	<b>STUCK-AT FAULT TESTING OF GALS-SYSTEMS .....</b>	<b>32</b>
5.5	<b>FUNCTIONAL TEST FOR GALS SYSTEMS .....</b>	<b>33</b>
5.6	<b>FUNCTIONAL TEST PATTERN GENERATION AND FAULT COVERAGE FOR GALS INTERFACES .....</b>	<b>35</b>
5.7	<b>FUNCTIONAL TESTING OF NOC GALS INTERFACE .....</b>	<b>37</b>
5.8	<b>BIST FOR GALS SYSTEMS .....</b>	<b>37</b>
5.9	<b>RECOMMENDED TEST STRATEGY FOR GALS .....</b>	<b>41</b>
5.10	<b>DESIGN-FOR-TEST IN SILISTIX CHAIN NOC .....</b>	<b>44</b>

## LIST OF FIGURES

Figure 1:	Evolution of DFT advances in digital circuit testing .....	14
Figure 2:	Asynchronous handshake circuits .....	19



# GALAXY

GALS InterfAce for CompleX Digital  
SYstem Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

---

Figure 3:	Asynchronous handshake protocols.....	19
Figure 4:	GALS System .....	20
Figure 5:	Two ACL units with ack <sub>2</sub> -s-a-0 .....	23
Figure 6:	Asynchronous design with (a) and without scan chain (b) .....	24
Figure 7:	Asynchronous BIST .....	26
Figure 8:	General on-line testing scheme.....	27
Figure 9:	Basic at-speed test schemes: (a) skewed-load and (b) double capture.....	31
Figure 10:	Functional testing of a handshake channel using test extension element.....	34
Figure 11:	Simplified block level schematic with connections between TEE and LSI .....	34
Figure 12:	The shaded areas represent portions of the circuit that are not covered by these tests. Functional test vectors are used to detect the stuck-at faults in these areas.....	36
Figure 13:	The red shaded area represent portion of the circuit that is not covered by these tests. Functional test vectors are used to detect the stuck-at faults in these areas.....	37
Figure 14:	BIST configuration .....	38
Figure 15:	BIST components .....	39
Figure 16:	Global BIST configuration .....	39
Figure 17:	Central BIST Controller (CBC) configuration .....	41
Figure 18:	Simplified Test Flow.....	44
Figure 19:	Scan-latch locations for sequential scan .....	45

## LIST OF TABLES

Table 1:	Test method comparison .....	27
----------	------------------------------	----



---

## 1 INTRODUCTION

---

Very Large Scale Integrated (VLSI) circuits designed using modern Computer Aided Design (CAD) tools are becoming faster and larger, incorporating millions of smaller transistors on a chip. VLSI designs can be divided into two major classes: Synchronous and Asynchronous circuits. Synchronous circuits use global clock signals that are distributed throughout their sub-circuits to ensure correct timing and to synchronize their data processing mechanisms [SU02, SA05]. Asynchronous circuits contain no global clocks. Their operation is controlled by locally generated signals [OH02]. Asynchronous circuits [EBE04] have many potential advantages over their synchronous equivalents including lower latency, low power consumption, and lower electromagnetic interference [EBE04][SAK07]

Synchronous design methodologies have been plagued by the necessity to distribute the centralized clock all throughout the chip with an acceptably low skew. This task has become even more difficult with the ever decreasing feature size of modern technologies. Decreasing feature size results in devices that are smaller, have less parasitics and therefore operate faster. However these devices have smaller drive strengths, whereas the interconnection parasitics remain more or less the same. As a result interconnection delays start to dominate, making it especially difficult for centrally distributed signals (that need to reach to each corner of the chip) to propagate in time [GUR02].

The recent trend to integrate more and more functions on a single chip, called System-on-Chip (SoC), provides additional challenges. SoCs require a great deal of modularity, where previously designed hardware modules can easily be embedded in a larger design. This typically results in a system that consists of several hardware blocks that have been designed to operate with different clock frequencies. To derive synchronous local clocks from a centralized clock in such a system is an involved task without a trivial solution [GUR02].

Asynchronous design techniques have always attracted attention as an alternative, especially for SoC integration, as they do not rely on a centralized clock. This alleviates problems related to clock distribution and enables the integration of hardware blocks with different clock domains. However, asynchronous design techniques also have a number of well known shortcomings, most notably complicated design methodologies and lack of reliable tools and test methodologies [GUR02].

The Globally-Asynchronous Locally-Synchronous (GALS) is a relatively new VLSI system design methodology that promises to combine the advantages of both synchronous and asynchronous operations [CHA84]. In GALS, rather coarse grained synchronous functional blocks are surrounded by self-timed wrappers that include a local clock generator and asynchronous communication ports. The functional blocks operate synchronous to the local clock but communicate asynchronously with similar blocks [GUR02].

Globally-asynchronous locally-synchronous (GALS) systems may become a solution for nowadays challenges in the field of VLSI design. The ITRS road-map [SIA06] predicts that, as a solution to the clock distribution problem, GALS will become mainstream design technique in the near future. In a GALS system, a number of synchronous islands of logic communicate asynchronously using a suitable interconnect. Unfortunately, the testability of asynchronous systems is considered one of their major drawbacks [SAK07]. Fully synchronous chips are becoming not feasible anymore due to clock distribution and power consumption problems. The value of GALS lies in combination of well known synchronous design methods and relative simple asynchronous communication channels. The key components are the communication control ports around the synchronous modules and the stretchable clock also called a wrapper. This clock has an unbound delay and is controlled by events the asynchronous channel [BL02]. GALS is a solution to combine the advantage of asynchronous and synchronous circuits design.

However, in order to successfully apply GALS system it is needed to enable efficient and reliable testing technique for such systems. In general, the testing of SoCs becomes an increasing challenge as these devices become more complex. A SoC design is typically built block by block. Efficient testing is



# GALAXY

GALS InterfAce for CompleX Digital  
SYstem Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

---

also best done by block by block. Recently, pre-designed cores are also used in the SoCs. Testing individual circuits, individual blocks and individual cores have established technologies [SAK07].

GALS systems confine the asynchronous circuitry solely to a self-timed wrapper around the locally-synchronous islands. While this approach offers many advantages, and especially eliminates the costly asynchronous design of large functional blocks, the asynchronous elements in the self-timed wrapper can not be verified by conventional methods. For asynchronous parts one has to use additional features, such as functional testing as described in [GUR02]. Alternatively, it is possible to apply the embedded functional testing on the system level based on BIST [KG05]. Finally, it is also possible to apply synchronous test methods such as scan techniques also to the asynchronous wrappers [TB02]. In this case, asynchronous wrapper becomes just a part of the locally synchronous module that can be tested using the classical scan approach for system testing.

In the following text we will summarize the methods for the testing of synchronous and asynchronous modules and discuss how those methods could be successfully combined for the testing of GALS systems



---

## 2 REFERENCES

---

### 2.1 ACRONYMS

<b>AFSM</b>	Asynchronous Finite State Machine
<b>ATE</b>	Automatic Test Equipment
<b>ATPG</b>	Automatic Test Pattern Generation
<b>BIST</b>	Built-In Self-Test
<b>CBC</b>	Central BIST Controller
<b>CUT</b>	Circuit under Test
<b>DI</b>	Delay insensitive
<b>DCVSL</b>	Differential Cascode Voltage Switch Logic
<b>DfT</b>	Design for Testability
<b>DUT</b>	Design under test
<b>GALS</b>	Globally Asynchronous Locally Synchronous
<b>IPG</b>	Invariant Property Generator
<b>LFSR</b>	Linear Feedback Shift Register
<b>LS</b>	Locally Synchronous
<b>MCE</b>	Muller C-element
<b>MISR</b>	Multiple-Input Shift Register
<b>MUTEX</b>	Mutual Exclusion
<b>QDI</b>	Quasi-delay insensitive
<b>SoC</b>	System on Chip
<b>ST</b>	Self-timed
<b>STA</b>	Static Timing Analysis
<b>TAP</b>	Test-Access Port
<b>TF</b>	Transition Fault
<b>TPG</b>	Test pattern generator
<b>TRA</b>	Test response analyser



## 2.2 REFERENCE DOCUMENTS

\* Where we put the reference outside of the sentence at the end of a paragraph that means that the whole paragraph is cited from the referenced document

Ref.	Document Title
[AB94]	M. Abramovici, M. A. Breuer, and A. Friedman, Digital Systems Testing and Testable Design. IEEE Press, 1994.
[AL98]	V. C. Alves, F. M. G. Franca, and E. P. Granja, "A BIST scheme for asynchronous logic," in Proceedings of the 7th Asian Test Symposium (ATS'98), pp. 27–32, 1998.
[BA96]	S. Banerjee, S. T. Chakradhar, and R. K. Roy, "Synchronous test generation model for asynchronous circuits," in Proceedings of the 9th International Conference on VLSI Design: VLSI in Mobile Communication, pp. 178–185, 1996.
[BAR87]	P. H. Bardell, W. H. McAnney, and J. Savir, Built-In Test for VLSI: Pseudorandom Techniques, John Wiley & Sons, Somerset, NJ, 1987.
[BEN03]	B. Benware, R. Madge, C. Lu, and R. Daasch, Effectiveness comparisons of outlier screening methods for frequency dependent defects on complex ASICs, in Proc. IEEE VLSI Test Symp., pp. 39–46, April/May 2003.
[BS97]	E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," Lecture Notes in Computer Science, vol. 1294, pp. 513–525, 1997.
[BL02]	Blaauwendraad, B., Student thesis, Title: TIR, design and testing of a Simple GALS, Linköping University, Department of Electrical Engineering, 2002
[BU00]	M. L. Bushnell and V. D. Agrawal, Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits, Springer, Boston, 2000.
[CAR87]	J. Carter, V. Iyengar, and B. Rosen, Efficient test coverage determination for delay faults, in Proc. IEEE Int. Test Conf., pp. 418–427, September 1987.
[CHA84]	D. M. Chapiro, Globally Asynchronous Locally Synchronous. Systems. PhD thesis, Stanford University, 1984.
[CRO99]	A. Crouch, Design for Test for Digital IC's and Embedded Core Systems, Prentice-Hall, Upper Saddle River, NJ, 1999.
[WST08]	Laung-Terng Wang, Charles E. Stroud, Nur A. Touba, "System-On-Chip Test Architectures Nanometer Design For Testability", Morgan Kaufmann Publishers is an imprint of Elsevier, 2008
[EF04]	A. Efthymiou, C. Sotiriou, and D. Edwards, "Automatic scan insertion and pattern generation for asynchronous circuits," in Proceedings of the Conference on Design, vol. 1, pp. 672–673, February 2004.
[EBE04]	Aristides Efthymiou, John Bainbridge, Douglas A. Edwards, Adding Testability to an Asynchronous Interconnect for GALS SoC, Asian Test Symposium 2004: 20-23
[GIZ06]	D. Gizopoulos, editor, Advances in Electronic Testing: Challenges and Methodologies Series: Frontiers in Electronic Testing, Springer, Boston, 2006.
[GU02]	F. Gürkaynak et al., A Functional Test Methodology for Globally-Asynchronous Locally-Synchronous Systems, Proc. of ASYNC, pp. 181-189, 2002.
[GU06]	F. Gürkaynak, GALS System Design: Side Channel Attack Secure Cryptographic Accelerators, PhD thesis, Hartung-Gorre Verlag, 2006



# GALAXY

GALS InterfACE for CompleX Digital  
System Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

[GUO06]	F. K. Gurkaynak, S. Oetiker, H. Kaeslin, N. Felber and W. Fichtner: "GALS at ETH Zurich: Success or Failure ?", Proceedings of the Twelfth IEEE International Symposium on Asynchronous Circuits and Systems, Grenoble France, pp. 159-168, March 13-15, 2006.
[HER96]	K. Heragu, J. H. Patel, and V. D. Agrawal, Segment delay faults: a new fault model, in Proc. IEEE VLSI Test Symp., April 1996, pp. 32-39.
[HH03]	M. Heath and I. Harris, "A deterministic globally asynchronous locally-synchronous methodology for validation, debug and test," in Proceedings of the 12th IEEE North Atlantic Test Workshop (NATW'03), 2003.
[HU94]	H. Hulgaard, S. M. Burns, and G. Borriello, "Testing asynchronous circuits: A survey," Integration, the VLSI Journal, vol. 19, pp. 111-131, March 1994.
[HR04]	P. D. Hype and G. Russel, "A comparative study of the design of synchronous and asynchronous self-checking risc processors," in Proceedings of the 10th IEEE International On-Line Testing Symposium (IOLTS'04), pp. 89-94, 2004.
[IE1149]	IEEE Std 1149.1-2001 IEEE Standard Test Access Port and Boundary-Scan Architecture
[IYE88]	V. Iyengar, B. Rosen, and I. Spillinger, Delay test generation 1: Concepts and coverage metrics, in Proc. IEEE Int. Test Conf., pp. 857-866, September 1988.
[JA03]	Vinay B. Jayaram, Experimental Study of Scan Based Transition Fault Testing Techniques, MSc Thesis, Virginia Tech, 2003
[JHA03]	N. Jha and S. Gupta, Testing of Digital Systems, Cambridge University Press, London, 2003.
[KB951]	A. Khoche and E. Brunvand, "A partial scan methodology for testing self-timed circuits," in Proceedings of the 13th IEEE VLSI Test Symposium (VTS'95), pp. 283-289, 1995.
[KB952]	A. Khoche and E. Brunvand, "Testing self-timed circuits using partial scan," in Proceedings of the 2nd Working Conference on Asynchronous Design Methodologies (ASYNC'95), pp. 160-169, 1995.
[KB97]	A. Khoche and E. Brunvand, "Critical hazard free test generation for asynchronous circuits," in Proceedings of the 15th IEEE VLSI Test Symposium (VTS'97), pp. 203-208, 27 April-1 May 1997
[KK97]	M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Saldanha, and A. Taubin, "Partial scan delay fault testing of asynchronous circuits," in Proceedings of the 1997 IEEE/ACM International Conference on Computer- Aided Design (ICCAD'97), pp. 728-735, 1997.
[KG05]	M. Krstic and E. Grass, "BIST technique for GALS systems" in Proceedings of the 8th Euromicro Conference on Digital System Design, pp. 10-16, 2005.
[Krstic]	M. Krstic, "Request Driven GALS Architecture", PhD Thesis, BTU Cottbus, Germany, 2006.
[LIN05]	X. Lin and J. Rajski, Propagation delay fault: A new fault model to test delay faults, in Proc. IEEE Asian and South Pacific Design Automation Conf., pp. 178-183, January 2005.
[LI00]	M. J. Liebelt and C.-C. Lim, "A method for determining whether asynchronous circuits are self-checking," in Proceedings of the 9th Asian Test Symposium (ATS'00), pp. 472-477, 2000.
[MA91]	A. J. Martin and P. J. Hazewindus, "Testing delayinsensitive circuits," in Proceedings of the 1991 University of California/Santa Cruz conference on Advanced research in VLSI, pp. 118-132, 1991.
[MAR99]	Eric Jan Marinissen, Yervant Zorian, Rohit Kapur, Tony Taylor, and Lee Whetsel, Towards a Standard for Embedded Core Test: An Example, Proceedings of the International Test Conference, September 1999, pp. 616 - 627.
[MC86]	E. J. McCluskey, Logic Design Principles: With Emphasis on Testable Semicustom Circuits, Prentice-Hall, Englewood Cliffs, NJ, 1986.
[ME82]	G. C. Messenger, "Collection of charge on junction nodes from ion tracks," IEEE Transactions on Nuclear Science, vol. 29, no. 6, pp. 2024-2031, 1982.



# GALAXY

GALS InterfAce for CompleX Digital  
System Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

[MRL04]	Y. Monnet, M. Renaudin, and R. Leveugle, "Asynchronous circuits sensitivity to fault injection," in Proceedings of the 10th IEEE International On-Line Testing Symposium (IOLTS'04), p. 121, 2004.
[MOU00]	S. Mourad and Y. Zorian, Principles of Testing Electronic Systems, John Wiley & Sons, Somerset, NJ, 2000.
[ND00]	B. Nadeau-Dostie, Design for At-Speed Test, Diagnosis and Measurement, Springer, Boston, 2000.
[OH02]	Eunjung Oh, Soo-Hyun Kim, Dong-Ik Lee, Ho-Yong Choi, High-Level Test Generation for Asynchronous Circuits from Signal Transition Graph, IEICE Trans on Fundamentals of Electronics, Communications and Computer Sciences Vol.E85-A No.12, 2002, pp. 2674-2683
[OVG02]	Stephan Oetiker, Thomas Villiger, Frank K. Gürkaynak, Hubert Kaeslin, Norbert Felber, and Wolfgang Fichtner, High Resolution Clock Generators for Globally-Asynchronous Locally-Synchronous Designs, Handouts of the Second ACiD-WG Workshop of the European Commission's Fifth Framework Programme, Munich, Germany, January 2002.
[PAR01]	K. Parker, The Boundary Scan Handbook, Springer Science, New York, 2001.
[PI02]	J. Pintaske, "Chip-design – warum nicht asynchron?," Elektronik, vol. 25, pp. 34–41, 2002.
[RAJ98]	J. Rajski and J. Tyszer, Arithmetic Built-In Self-Test for Embedded Systems, Prentice-Hall, Englewood Cliffs, NJ, 1998.
[RK94]	D. A. Rennels and H. Kim, "Concurrent error detection in self-timed vlsi," in Proceedings of the 24th International Symposium on Fault-Tolerant Computing (FTCS'24), pp. 96–105, 15–17 June 1994.
[RON99]	Marly Roncken, Defect-Oriented Testability for Asynchronous IC's, Proceedings of the IEEE 87 (1999), no. 2, 363–375.
[SA04]	Mathew A. Sacker, Andrew D. Brown, Andrew J. Rushton, Peter R. Wilson, A behavioral synthesis system for asynchronous circuits, IEEE Trans. VLSI Syst. 12(9): 978-994 (2004)
[SAK07]	P. Sakthivel and P. Narayanasamy, An Approach to the Design of Optimal Test Scheduling for System-On-Chip Based on Genetic Algorithm, In book: Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, Publisher Springer Netherlands, pp. 25-28
[SAX02]	J. Saxena, K.M. Butler, J.Gatt, R.Raghuraman, S.P Kumar, S. Basu, D.J. Campbell and J. Berech, "Scan-Based Transition Fault Testing – Implementation and Low Cost Test Challenges", Proceedings of IEEE International Test Conference, 2002, pp. 1120-1129.
[SIA03]	SIA, The International Technology Roadmap for Semiconductors: 2003 Edition—Design, pp. 30–36, Semiconductor Industry Association, San Jose, CA ( <a href="http://public.itrs.net">http://public.itrs.net</a> ), 2003.
[SIA06]	SIA, The International Technology Roadmap for Semiconductors: 2006 Update, Semiconductor Industry Association, San Jose, CA ( <a href="http://public.itrs.net">http://public.itrs.net</a> ), 2006.
[STR02]	C. E. Stroud, A Designer's Guide to Built-In Self-Test, Springer, Boston, 2002.
[SU02]	Yin-He Su, Ching-Hwa Cheng, Shih-Chieh Chang, Novel Techniques for Improving Testability Analysis, IEICE Trans on Fundamentals of Electronics, Communications and Computer Sciences Vol.E85-A No.12, 2002, pp. 2901-2912
[SY05]	D. Shang, A. Bustrov, A. Yakovlev, and D. Koppad, "On-line testing of globally asynchronous circuits," in Proceedings of the 11th IEEE International On-Line Testing Symposium (IOLTS'05), pp. 135–140, July 2005
[SY06]	D. Shang, A. Yakovlev, F. Burns, F. Xia, and A. Bustrov, "Low-cost online testing of asynchronous handshakes," in Proceedings of the Eleventh IEEE European Test Symposium (ETS'06), pp. 225–232, May 2006.



# GALAXY

GALS InterfAce for CompleX Digital  
System Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

[SH96]	V. Schöber and T. Kiel, "An asynchronous scan path concept for micropipelines using the bundled data convention," in Proceedings of the International Test Conference (ITC'96), pp. 225–231, October 1996.
[SH01]	V. Schöber, Der testfreundliche Entwurf asynchroner Schaltungen. PhD thesis, Fachbereich Elektrotechnik und Informationstechnik, Universität Hannover, Germany, June 2001.
[SF01]	J. Sparsø, S. Furber, R. van Leuken, R. Nouta, and A. de Graaf, Principles of Asynchronous Circuit Design: A Systems Perspective. Kluwer Academic Publishers, Boston, 2001.
[SMI85]	G. Smith, Model for delay faults based upon paths, in Proc. IEEE Int. Test Conf., pp. 342–349, October 1985.
[TB02]	F. T. Beest, A. Peeters, M. Verra, K. van Berkel, and H. Kerkhoff, "Automatic scan insertion and test generation for asynchronous circuits," in Proceedings of the International Test Conference (ITC'02), pp. 804–813, 2002.
[TB03]	F. T. Beest, A. Peeters, K. V. Berkel, and H. Kerkhoff, "Synchronous full-scan for asynchronous handshake circuits," J. Electron. Test., vol. 19, no. 4, pp. 397–406, 2003.
[TOU06]	N. A. Toubia, Survey of test vector compression techniques, IEEE Design & Test of Computers, 23(4), pp. 294–303, July/August 2006.
[TRA88]	C. A. Traver. Design for Testability Techniques for Globally-Asynchronous Locally-Synchronous Systems, PhD thesis, University of Virginia, May 1988
[WAI87]	J. Waicukauski, E. Lindbloom, B. Rosen, and V. Iyengar, Transition fault simulation, IEEE Design & Test of Computers, 4(5), pp. 32–38, April 1987.
[WIE95]	Rik van de Wiel, High-Level Test Evaluation of Asynchronous Circuits, Asynchronous Design Methodologies, IEEE Computer Society Press, May 1995, pp. 63–71.
[WWW06]	L.-T. Wang, C.-W. Wu, and X. Wen, VLSI Test Principles and Architectures: Design for Testability, Morgan Kaufmann, San Francisco, 2006.
[YS92]	F. Yang and R. Saleh, "Simulation and analysis of transient faults in digital circuits," IEEE Journal of Solid State Circuits, vol. 27, pp. 258–264, March 1992.

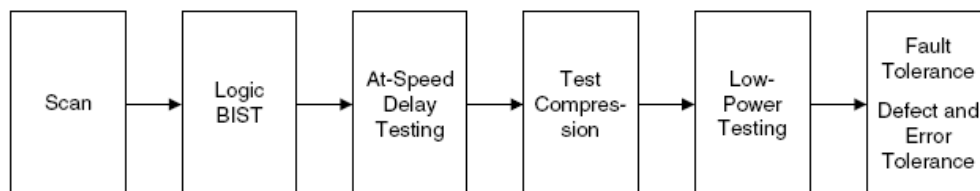


## 3 TESTING DIGITAL SYSTEMS

### 3.1 DESIGN FOR TESTABILITY (DFT)

SoC designs contain a variety of components, including digital, memory, as well as analog and mixed-signal circuits, all of which need to be tested. DFT is essential for reducing test costs and improving test quality. Scan is widely used for digital logic testing. However, one of the challenges in the nanometre design era is dealing with the rapidly growing amount of scan data required for testing complex SoC designs. The bandwidth between an external ATE and the device under test is limited, creating a bottleneck on how fast the chip can be tested. This has led to the development of test compression and logic BIST architectures, which reduce the amount of data that needs to be transferred between the ATE and device under test. Another key issue for scan testing is applying at-speed tests, which are crucial for detecting delay faults. [WST08]

The semiconductor industry heavily relies on *scan* and *logic built-in self-test (BIST)* technique [MC86, AB94]. Scan converts a digital sequential circuit into a scan design and then uses *automatic test pattern generation (ATPG)* software [BU00, Jha03, WWW06] to detect faults that are caused by manufacturing defects (physical failures) and manifest themselves as errors, whereas logic BIST requires using a portion of the VLSI circuit to test itself on-chip, on-board, or in-system. To keep up with the design and test challenges [SIA03, SIA06], more advanced DFT techniques have been developed to further address the test cost, delay fault, and test power issues [GIZ06, WWW06]. The evolution of important DFT techniques for testing digital circuits is shown in Figure 1. [WST08]



**Figure 1: Evolution of DFT advances in digital circuit testing**

The success of scan-based DFT techniques from the mid-1970s through the mid-1980s led to their adaptation for testing interconnect and solder joints on surface mount PCBs. This technique, known as boundary scan, eventually became the IEEE 1149.1 standard [IE1149] and paved the way for floating vias, microvias, and mounting components on both sides of PCBs to reduce the physical size of electronic systems. Boundary scan provides a generic test interface not only for interconnect testing between ICs but also for access to DFT features and capabilities within the core of an IC [IE1149] [PAR01]. A *test access port (TAP)* controller is included to access the boundary-scan chain and any other internal features designed into the device, such as access to internal scan chains, BIST circuits, or, in the case of FPGAs, access to the configuration memory. [WST08]

Design for testability has always been a weak point of asynchronous systems and one of the most important reasons why asynchronous techniques have not gained industrial popularity yet. Recently, GALS techniques are proposed as an effective way of complex digital system integration. GALS asynchronous wrappers should allow the integration of large synchronous blocks into complex digital systems. In order to achieve wide commercial use of the GALS technique, it is needed to provide an adequate way of testing it. [KG05]



## 3.2 FAULT MODELS

In order to properly test digital circuits we first have to define what kind of defects (faults) can appear in a digital system. Here is a short overview of those faults.

### Stuck-at fault model

Stuck-at is widely used fault model and it is covering many different physical faults [AB94]. The concept of this fault is very simple, and for each line in the system two possible stuck-at faults may appear: stuck-at 0, and stuck-at 1. For this type of fault it is considered that the faulty line is in permanent faulty state giving a constant 0 or constant 1. Additionally, one can consider stuck-at faults lines at both inputs and output of logical gates or only at outputs. Considering only output stuck-at faults simplifies the test but doesn't cover all that can be detected with a test for both input and output stuck-at faults.

### Delay fault model

A delay defect is a defect that causes an extra delay in the circuit. An example is a spot defect that causes a resistive short or open. To make the delay testing problem solvable, delay defect behaviour must be abstracted to a delay fault. [WST08]

### Transition fault model

The transition fault model is similar to the stuck-at fault model in many respects. The effect of a transition fault at any point P in a circuit is that any transition at P will not reach a scan flip-flop or a primary output within the stipulated clock period of the circuit. According to the transition fault model [6], there are two types of faults possible on all lines in the circuit: a slow-to-rise fault (STR) and a slow-to-fall fault (STF). A slow-to-rise fault at a node means that any transition from 0 to 1 on the node does not produce the correct result when the device is operating at its maximum operating frequency. Similarly, a slow-to-fall fault means that a transition from 1 to 0 on a node does not produce the correct result at full operating frequency. [JA03]

The primary advantage of the TF model is that test generation does not need to consider circuit timing. A stuck-at fault test generator can be modified to meet the additional requirements for generating TF tests [WA187]. Because a stuck-at fault can be considered a very slow TF, a TF test set will detect all the corresponding stuck-at faults. The TF model has more constraints than the stuck-at fault model, so the TF coverage is normally lower than stuck-at fault coverage. Top-off vectors can be generated to test the stuck-at faults not detected by the TF test set. The primary disadvantage of the TF model is that its resolution is limited by the difference in delay between the longest and shortest path through the fault site [WST08].

### Inline-Delay Fault Model

Production experience shows that many delay defects are due to resistive interconnect vias, which cause both the rising and falling transitions through that line to be slow [BEN03]. This can be modelled by the inline-delay fault. This fault is analogous to the TF, except that only one of the STR or STF faults must be detected at each fault site. A test set for inline-delay faults is smaller than a test set for TFs. [WST08]

### Gate-Delay Fault Model

A spot defect that causes a small delay fault for a particular transition on a gate input or output can be modelled as a gate-delay fault, also termed a local delay fault [CAR87]. Here "small" is a delay that is larger than the minimum slack but smaller than the maximum slack for that line. Testing such



faults requires testing a long or longest path with the appropriate transition through the fault site. The quality of a test set is defined by how close the minimum detected delay fault sizes are to the minimum detectable fault sizes [IYE88]. [WST08]

### Path delay fault model

The path-delay fault model [SMI85] models the distributed delay on a path. A path is a sequence of gates from PIs to POs, with a transition on each gate output along the path. The gate input on the path is referred to as the on-input, or on-path input, whereas the other inputs are side inputs or off-inputs. If the circuit contains a path that is slow for a rising or falling transition, then it contains a path-delay fault. Unless explicitly mentioned, we refer to the transition direction at the input to the path. The path-delay fault model assumes that any path can have any delay, so fault coverage is defined as the fraction of all paths (or all synthesizable paths) tested. [WST08]

The path-delay fault model is more general than the fault models discussed above, because a path-delay fault test will detect the corresponding transition, inline, or gate-delay faults, as well as distributed delay resulting from process variation (*global delay faults*) or supply noise. The primary drawback of the path-delay fault model is that the number of paths can be exponential in the size of the circuit, so computing and achieving high coverage is difficult. Path delay testing is primarily used to test a set of longest (or *critical*) paths provided by static timing analysis. [WST08]

A fault model intermediate between gate and path delay is the segment delay fault model. It assumes that a segment along a path is slow [HER96]. Because the segment length is bounded, the number of segment faults is linear in the circuit size. The propagation delay fault model combines the transition fault and path delay fault models [LIN05]. It assumes that the sum of the local delay and the distributed delay of the fault propagation path causes a failure, with at least one robust propagation path. [WST08]

## 3.3 DFT TECHNIQUES

In this section we will discuss the most popular DFT techniques, applied for almost all synchronous circuits. In particular, we will discuss scan and BIST techniques, and their application with Automatic Test Equipment (ATE).

As described in [WST08], scan design is implemented by first replacing all selected storage elements of the digital circuit with scan cells and then connecting them into one or more shift registers, called scan chains, to provide them with external access. With external access, one can now control and observe the internal states of the digital circuit by simply shifting test stimuli into and test responses out of the shift registers during scan testing. The DFT technique has proved to be quite effective in improving the product quality, testability, and diagnosability of scan designs [CRO99, BU00, JHA03, GIZ06, WWW06]. Although scan has offered many benefits during manufacturing test, it is becoming inefficient to test deep submicron or nanometre VLSI designs. The reasons mostly relate to the facts that

(1) Traditional test schemes using ATPG software to target single faults have become quite expensive, and

(2) Sufficiently high fault coverage for these deep submicron or nanometer VLSI designs is hard to sustain from the chip level to the board and system levels. [WST08]

To alleviate these test problems, the scan approach is typically combined with logic BIST that incorporates BIST features into the scan design at the design stage [BU00, MOU00, STR02, JHA03]. With logic BIST, circuits that generate test patterns and analyze the output responses of the functional circuitry are embedded in the chip or elsewhere on the same board where the chip resides to test the digital logic circuit itself. Typically, pseudo-random patterns are applied to the *circuit under test* (CUT) while their test responses are compacted in a *multiple-input signature register* (MISR) [BAR87, RAJ98,



ND00, STR02, JHA03, WWW06]. Logic BIST is crucial in many applications, in particular, for safety-critical and mission-critical applications. [WST08]

Since the early 2000s, test compression, a supplemental DFT technique to scan, is gaining industry acceptance to further reduce test data volume and test application time [TOU06, WWW06]. Test compression involves compressing the amount of test data (both test stimulus and test response) that must be stored on ATE for testing with a deterministic (ATPG-generated) test set. This is done by using code-based schemes or adding additional on-chip hardware before the scan chains to decompress the test stimulus coming from the ATE and after the scan chains to compress the test response going to the ATE. This differs from logic BIST in that the test stimuli that are applied to the CUT form a deterministic (ATPG-generated) test set rather than pseudo-random patterns. [WST08]

There are also many others test techniques, that are mainly used a supplement to the mentioned one. For example one could use a functional testing, and test the circuit while making its normal operation. Additionally, it is possible to use current sensing test techniques, such as IDDQ testing, for identifying short circuits in the system.

### 3.4 STUCK-AT FAULT TESTING

The basic test, that is already for years an industry standard, is the test of the circuits for stuck-at faults. To test for stuck-at faults, two steps are involved. The first step is to generate a test vector that excites the fault and the next step is to propagate the faulty effect to a primary output or a scan flip-flop. Automatic test pattern generation (ATPG) tools are typically used to generate the test vectors. The stuck-at fault model is being used virtually everywhere in the industry to screen defects caused by stuck-at faults. It is relatively easy to generate patterns for stuck-at faults and pattern volume is also comparatively low. [JA03]

In the past few years, structured scan-based methods, most often full-scan, are being increasingly used to generate test patterns that are capable of achieving high coverage. The advantage lies in the short cycle time for developing them, combined with the relative ease in debugging them. Test application times may not be short if many flip-flops are included in long serial scan chains. Designers often make use of multiple parallel scan chains to reduce the test application time. However, as design sizes increase, most types of automatic test equipment (ATE) are incapable of supporting the increased number and depth of scan chains. Scan based methods still remain the most viable alternative to functional tests and many improvements are being devised for such methods. [JA03]

### 3.5 SUMMARY

In the previous text we have introduced basic DfT issues and techniques. For classical synchronous design there are many mature test solutions available.

All those techniques, or some combination of them, are used for testing digital circuits. Most of these techniques are invented for the synchronous circuits. However, many of them could be also successfully applied for the asynchronous circuit testing. There are also special techniques that are designed only to test asynchronous circuits. This we will analyze in the following chapter.



---

## 4 TESTING ASYNCHRONOUS CIRCUITS

---

### 4.1 INTRODUCTION

When considering the evolution of shrinking feature sizes of integrated circuits (ICs), it is obvious that more complex designs will be integrated within one single chip that will become a SoC. Most ICs today are designed to be synchronous, i.e., they are controlled by a global clock signal, which triggers all memory elements, simultaneously. But with the growing complexity of modern ICs the overhead in design and chip area for the clock tree rises extremely. This is because the complex clock tree contains amplifiers, buffers and delay elements which are required to ensure that the clock signal arrives at all memory elements at the same time to guarantee the correct behaviour of the circuit. But clock distribution is only one of several problems regarding synchronous systems [PI02]. Simultaneously switching registers causes high electrical peaks. The fall of voltage can result in a malfunction of the whole circuit. Another problem is the electromagnetic interference (EMI) generated by high clock frequencies. This physical effect has several drawbacks. One is that EMI can negatively influence other system components and can cause a crosstalk resulting in faulty circuit behaviour.

Additionally, synchronous circuit designs require an adjustment of the clock period to the worst-case delay behaviour. Before manufacturing the circuit its longest delay path must be identified. The required time for this path determines the clock frequency. Also the temperature and the operating voltage must be observed to ensure the correct behaviour when those environmental parameters change.

Asynchronous circuit design can be a solution of the above mentioned problems. The advantage of asynchronous in contrast to synchronous circuits lies in lower power consumption, because the modules are active only in the presence of data, better delay behaviour, modularity of such systems, and additionally in better security aspects. This was studied in [BS97, MRL04].

But asynchronous systems also have some drawbacks compared to synchronous designs. The major aspect that shall be addressed here lies the lack of test techniques for asynchronous circuits. Recently, many test methods have been specifically developed for synchronous systems. Indeed, in the last decades testability issues of asynchronous circuits were further investigated. Therefore, several test approaches for synchronous circuits have been adapted to asynchronous designs and new approaches were particularly developed to cope with the problems regarding asynchronous circuit testing. However, testing asynchronous circuits is still a difficult task.

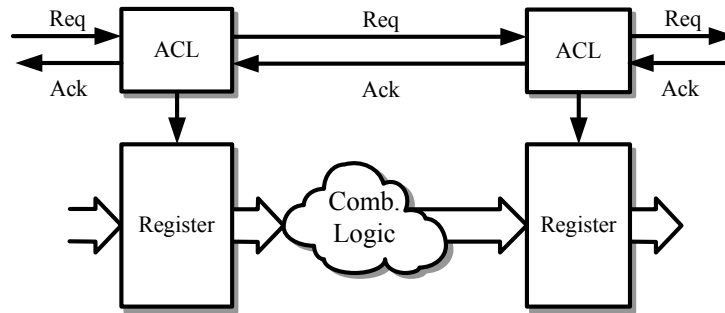
This chapter gives an overview about existing problems in testing, solutions to those problems and Design for Testability (DfT) for asynchronous circuits. Such a comprehensive survey was already presented by H. Hulgaard et al. in [HU94] and A.J. Martin et al. gave an overview about testing a special category of asynchronous circuits in [MA91]. The intention of this chapter is to introduce and summarize new techniques and approaches. Thereto, section 3.2 of this chapter introduces asynchronous circuits, discusses their properties and gives an insight into types of asynchronous circuits. Section 3.3 deals with test approaches for asynchronous circuits and presents the most common fault models. The last section concludes this chapter.

### 4.2 ASYNCHRONOUS HANDSHAKE CIRCUITS

The basic characteristic of asynchronous circuits is that they are not controlled by a global clock. Hence, a complex clock distribution tree is not necessary to control the memory elements. In general, these circuits are composed of asynchronous modules which store the actual state of the circuit and some computational elements between those asynchronous modules. The asynchronous modules themselves consist of memory cells and asynchronous control logic (ACL) which generates a local clock signal triggering the according memory cells to store the current state. The computational elements are realized by combinational logic mostly using standard gates. But also non-standard gate

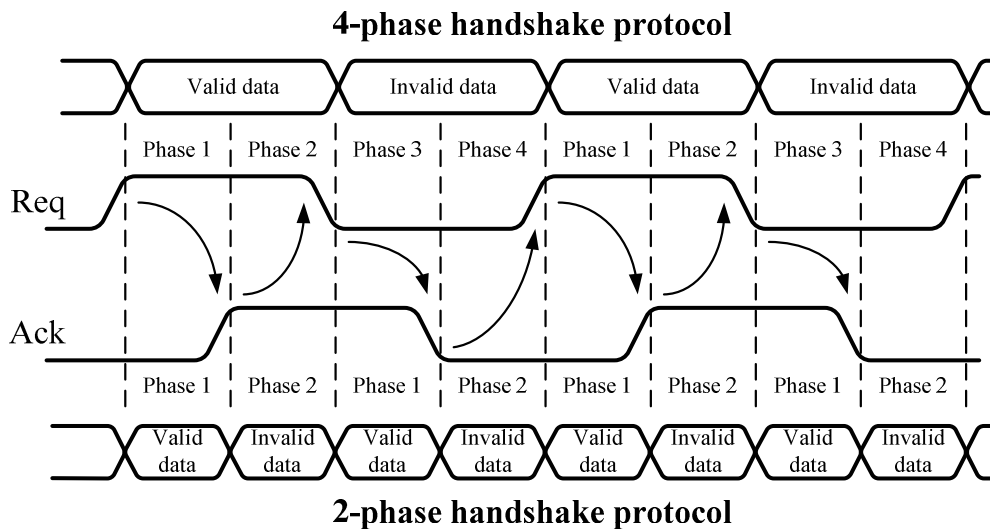


implementations, e.g. DCVSL gates, are possible. The communication between asynchronous modules is organized by a bidirectional handshake protocol using a request (*req*) and an acknowledgment (*ack*) signal. A request is assigned when data should be transmitted. If the data was received then an acknowledgment is generated. Figure 2 shows the general schematic of asynchronous handshake circuits.



**Figure 2:** Asynchronous handshake circuits

A handshake protocol can be categorized in different ways. One major distinction regards the number of phases. Here, there are two major kinds of handshaking protocols to be distinguished, 2-phase and 4-phase as shown in Figure 3. Both have different properties regarding power and time consumption and the meaning of signal levels and transitions. The 4-phase handshake protocol is level-based, i.e. the four different phases of the protocol depend on the level of the handshake signals. This is simple to implement, but it requires more power and time to execute a full handshake compared to the 2-phase handshake protocol, which is transition-based. This means that the phase of the handshake depend on signal events, i.e. transitions, rather than on the level of the handshake signals. Because the 2-phase handshake protocol requires only half of signal switches it is more efficient in time and energy consumption.



**Figure 3:** Asynchronous handshake protocols

Another distinction considers how data bits are transmitted. A handshake-protocol with explicit request and acknowledgement signals and one wire per each data bit is called single-rail protocol. The counterpart to this is the dual-rail handshake protocol. It uses two wires (*dt dt*) per each data bit where a valid '1' is encoded with (01) and a valid '0' with (10). When both signals are low then the data is not valid and this is the so called NULL-value. The last case, if both signals are high, corresponds to the forbidden state. Dual-rail handshake protocols have no explicit request signal. The request is encoded

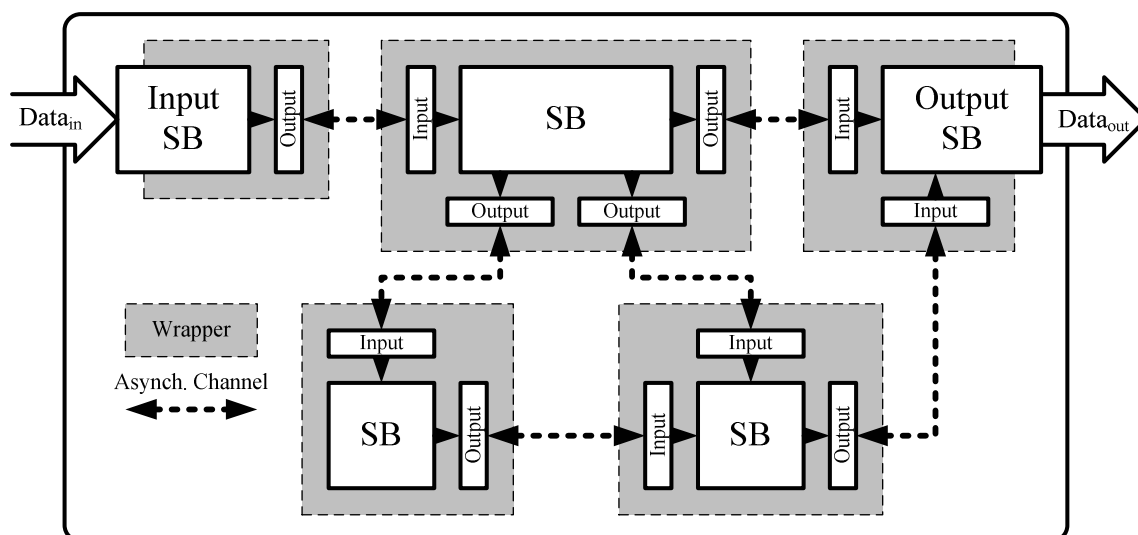


within the data lines when all data bits have a valid value. There also exist other protocols, e.g. m-of-n-encoding, but due to space limitations these types should not be discussed here.

The last distinction considers the initiator of the handshake. In a push-protocol the sender is the active part, i.e. that the request goes from the sender to the receiver. The opposite is also possible and is called a pull-protocol. Here, the receiver initiates the transmission by sending a request that is acknowledged by the sender. A concrete handshake protocol realization is a combination of the mentioned categories. J. Sparsø and S. Furber have given a comprehensive overview of the most important protocols in [SF01].

Beside the categorization of the used protocol purely asynchronous circuits can also be classified regarding gate and wire delays as being self-timed (ST), speed-independent (SI), delay-insensitive (DI) or quaside delay-insensitive (QDI). An asynchronous circuit which works correctly with positive bounded gate and wire delays is said to be delay-insensitive. SI circuits consider only gate delays as being positive bounded. Delays in wires are assumed to be zero. The mixed forms are QDI circuits which have positive delays in gates and in wires, as well. The difference to DI circuits is that forks are assumed to be isochronic, i.e. the delays of the fork's branches are equal. The last class of circuits (ST) exhibit more elaborate timing assumptions under which they work correctly.

Another type of circuits are globally-asynchronous locally-synchronous (GALS) systems, which combine the advantages of both design styles. Such systems consist of synchronous modules/blocks (SB) operating with individual clock domains. Figure 4 illustrates such a GALS system. The communication between SBs is organized by asynchronous channels, e.g. realized by asynchronous FIFOs. Other ways for interconnecting SB use synchronizers or pausable clocks. The SBs use transmission enable signals to indicate that they are ready for transmission. The conversion between the synchronous communication enable and the asynchronous handshake signal is done by a wrapper module.



**Figure 4:** GALS System

Based on their structure and the corresponding behaviour asynchronous circuits offer several advantages compared to their synchronous counterparts. For example, due to self-timed properties asynchronous circuits operate at their maximum speed and consume power only when data must be processed. Furthermore, these circuits are robust against environmental parameters (supply voltage, temperature etc.) and the asynchronous switching of memory cells results in lower electromagnetic radiation. The last major advantage is the possibility of modular design. This enables asynchronous modules to be separately developed rather than designing an entire design at once.



Although these advantages are well known, the industry often avoids asynchronous designs, because of less tool support, the lack of knowledge in dealing with these systems and other problems which are not present in synchronous circuits. Critical hazards and races are only one example. In synchronous designs the propagation of those effects is inhibited by adjusting the clock period such that signal values have enough time to stabilize. In asynchronous circuit where each signal transition has a meaning, hazards can be disastrous. This shows that the usage of a synchronous design style simplifies the development of systems and it must be mentioned that the generation of complex systems was only possible with the restriction to synchronous designs. As a consequence methods for testing the manufactured circuits were especially developed for synchronous systems. Hence, the test equipment is organized to operate synchronously which is the reason that purely asynchronous circuits are difficult to test. However, many approaches for asynchronous circuits were investigated to make these systems testable. The next section gives an overview about existing methods and discusses their properties.

## 4.3 TESTING ASYNCHRONOUS CIRCUITS

Testing is an important part in chip production, since a design will not be passed to mass production without its verification. In order to enhance the testability, designs must often be extended with additional hardware to access and control the circuit during test. The majority of DfT-techniques were developed for synchronous rather than for asynchronous systems. As a consequence, industrial companies often avoid asynchronous circuits, because of many problems they have to overcome when testing such designs. One important issue is the communication between the design under test (DUT) and the test equipment. Most test automatons operate synchronous, apply and compare patterns cycle by cycle. The problem is that testers are not capable of reacting to responses of the DUT. Hence, a real asynchronous communication is impossible. To realize the communication the design must be extended to have a synchronous interface which denotes additional hardware that should preferably be avoided.

When testing GALS systems a similar problem appears. Indeed, GALS systems communicate synchronously with the environment, hence, the communication can be organized. However, these systems tend to be non-deterministic in their timing behaviour, i.e. a response of the DUT can occur in a particular range of clock cycles. Here, the problem is that a tester expects a response at a specific clock cycle rather than in a range of cycles. If the expected response does not occur in the specific cycle then the whole test is treated as failed. A solution for this problem is called synchro-tokens and was presented by M.W. Heath and I.G. Harris in [HH03]. This approach attempts to make GALS systems deterministic during test using a token ring with a node on each of two communicating SBs. Each node counts local clock cycles to determine when the token is expected to arrive and when it should depart. Hence, a node can stop the clock when the token has not arrived and can ignore all transitions on the asynchronous channel when the token has appeared too early. Using this scheme a GALS system can be made deterministic.

Another problem of asynchronous circuit testing is redundant logic that is used to avoid critical hazards and races. Because a fault within redundant logic will not change the logic function of the circuit, it cannot be detected by any test. The critical point with this is that although a fault within redundant logic does not affect the circuit's function it can mask other faults. Thus, the masked fault cannot be detected by a test (cf. [AB94]).

### ***Fault Models in Asynchronous Circuits***

Before a test method is chosen or developed for a given design one has to consider which types of faults probably occur within the design and which faults the test method is able to detect. Due to their sensitivity to every signal transition, asynchronous circuits are more susceptible to more types of faults than synchronous implementations. As previously mentioned, hazards can cause an asynchronous circuit to malfunction. Hence, test methods developed for synchronous systems could not be adapted to asynchronous designs without taking such additional effects into account. However, many fault models



for synchronous systems can also be applied to asynchronous circuits. This section briefly lists a set of feasible fault models for asynchronous circuits.

Fault models can be defined on different abstraction levels of the circuit. They should preferably be simple to limit the fault universe, i.e. the number of considered faults. Because of the explosion of possibilities, it is often assumed that only a single fault occurs in a circuit. But also multiple faults are considered. Mostly, it is sufficient to determine, whether a circuit is faulty or not. Hence, the most commonly used model for permanent faults is the single stuck-at fault (s-a-f) model, which assumes one wire or a gate output to be stuck at a constant value. The s-a-f model is dominated by many others, i.e. a complete test for single stuck-at also detects other faults which are not integrated within this model. For example, M. Kishinevsky et al. present a method for detecting delay faults using the stuck-at fault model in [KK97]. These faults do not affect the logical behaviour of a combinational circuit. Only the timing is influenced. However, the test for delay faults is essential in ST circuits, because of timing assumptions under which these circuits work correctly. For this reason delay faults are often the subject of investigations in asynchronous circuit testing. Similarly, critical hazards and races are also often considered in asynchronous circuit testing since they can invalidate a test.

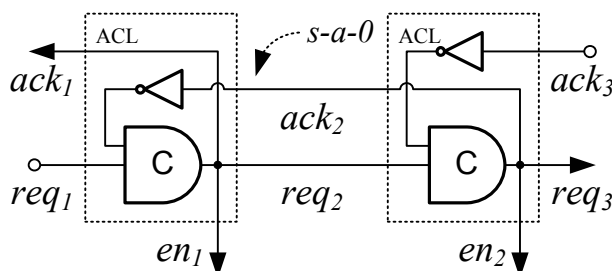
Transition faults are another type of faults referring to faulty timing behaviour of asynchronous circuits. Such faults can be premature and inhibit transitions on control logic signals, which often lead to violations regarding the order of handshake protocol transitions. These violations are relatively easy to observe from the functional sight of a system and so they can be used as a first indication whether a circuit is faulty. As a consequence, the circuit can be further exploited regarding structural faults, e.g. stuck-short/opens or bridging and coupling faults. These fault models on transistor and logic level, respectively, are in most cases too complex to be applied to the whole circuit. Therefore, the fault should be isolated using the mentioned functional analysis.

The next category of faults is transient faults, which only temporarily affect the circuit's function. Such faults could appear, if an alpha particle or an atmospheric neutron forces an erroneous signal transition [YS92, ME82] or can be induced by crosstalk. Furthermore, transient faults can be intentional to influence the circuit behaviour. For instance, hackers attack a system to access protected memory areas or secret information using radiation from lightning, laser emission etc (cf. [MRL04]).

To detect the mentioned faults different approaches can be used. Each approach has several advantages and some drawbacks and is feasible for a variety of fault models. The following subsections discuss the properties of the approaches and present some examples and specific realizations.

## ***Self-checking Properties of Asynchronous Control Logic***

The first approach to be discussed utilizes the property of asynchronous control circuits to halt in the presence of a s-a-f. This property was discussed in [HU94]. Figure 5 illustrates two ACL units which communicate using a single-rail handshake protocol. Initially, the outputs of all Muller-C elements (MCE) are low. Assume that  $ack_2$  is s-a-0 and a full handshake should be performed on the left side of the circuitry, i.e.  $req_1$  is set to 1 (denoted by  $req_{+1}$ ) followed by  $ack_{+1}$ ;  $req_{-1}$ ;  $ack_{-1}$ . After its propagation delay the first MCE will fire because of  $req_{+1}$  and the signals  $ack_1$  and  $req_2$  will be assigned to high. Then  $req_{+2}$  will cause the second MCE to set its output high after its propagation delay. But because of  $ack_2$ -s-a-0 that inhibits  $ack_{+2}$  the first MCE is not able to reset its output when  $req_{-1}$  happens. Hence, the circuit will deadlock and it can be shown that every s-a-v ( $v \in \{0;1\}$ ) will also force the circuit to halt. This deadlocked state can be detected by waiting a certain amount of time using a checking scheme that can be realized



**Figure 5:** Two ACL units with  $ack_2$ -s-a-0

### Test Pattern Generation

The generation of test patterns is a complex process during test flow and many aspects, e.g. the cost and quality of the test patterns, has to be considered. Test patterns can be generated to perform functional tests, i.e. to determine whether the DUT performs its specified function, or to test for structural faults. The former is mostly performed during the design process and verification of a pilot line. But because the set of functional faults is unbounded it cannot be realized automatically. In comparison to functional faults, the set of structural faults is limited. Hence, a test for detecting such faults can be automated [AB94], referred to as ATPG.

However, the generation of tests for asynchronous circuits has some obstacles, that makes this process more difficult than for synchronous systems. As already mentioned, asynchronous circuits contain redundant logic to avoid hazards and races. Hence, a test pattern set with 100% fault coverage cannot be achieved. Another problem in ATPG for asynchronous circuits regarding hazards and unstable states is that these effects can invalidate a test because they are not considered during test generation. This means that a test fails due to some unstable signals although the circuit is fault free. Approaches to avoid test invalidation were presented by S. Banerjee et al. in [BA96] and by A. Khoche and E. Brunvand in [KB97]. The former approach uses a synchronous model that is generated for ATPG. This model operates in fundamental mode, i.e. input patterns are applied and output responses are observed only when the circuit is stable. In [KB97] a 6-valued algebra that includes hazards is applied to ST circuits. The considered ST circuits consist of modules that are composed of XOR gates, MCEs and gated latches. Therefore, the propagation of the six values has to be considered only for these elementary components. The algebra was integrated into a modified D-algorithm that performs the pattern generation.

A further problem of ATPG for asynchronous circuits is that test patterns generated for detecting structural faults in some cases are forbidden or impossible to occur in normal operation mode. These patterns can lead into invalid states or oscillation which invalidates a test. Also the huge amount of sequential elements, e.g. MCEs, makes ATPG for asynchronous circuits problematical. To ease the pattern application in sequential elements, scan chains can be integrated into the DUT. This is presented within the next section.

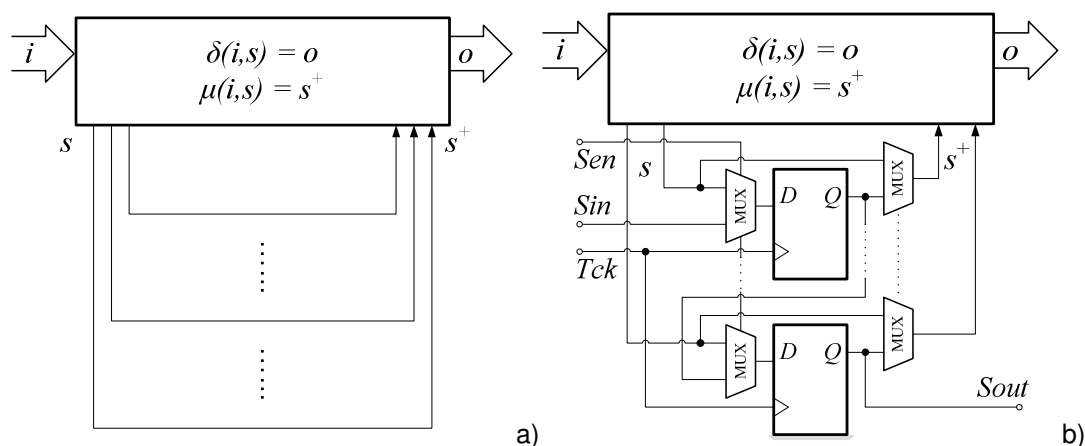
### Scan Test Approaches

Scan test is a common DfT technology that is widely accepted by the industry. It is a strong method for detecting structural faults as well as functional faults if a boundary scan is used for some module of the DUT. To test a circuit using scan, all memory elements obtain an additional input and are chained up to one or multiple sequential shift registers. The memorized state of the circuit can then easily be examined by scanning out the values stored within the internal scan registers. Because conventional scan registers require a global clock signal for scan operation this technique was mainly adopted in synchronous circuits. But due to its feasibility to detect structural faults the scan design has been also adjusted to asynchronous circuits.

In general, asynchronous circuits have feedback loops without memory elements. This is shown in Figure 6(a). In order to make this design scannable, the feedback loops are broken by inserting scan



latches which are transparent in normal mode. Figure 6(b) illustrates this. As well as for synchronous circuits two scan approaches can be distinguished: full-scan and partial-scan. A full-scan offers high fault coverage but also has a huge hardware overhead due to the necessity to make all memory and sequential elements scannable. Thus, a full scan introduces scan capabilities into both local and global feedbacks. A partial scan sacrifices fault coverage in order to save hardware costs. Mostly, this is achieved by inserting scan capabilities only into global feedbacks. Such a partial-scan technique was presented by A. Efthymiou et al. in [EF04]. The inserted scan latch is used to apply test patterns to 1-of-5 pipeline latches of the CHAIN interconnect. These pipeline latches consist of regular MCEs without any scan capabilities. Hence, the approach takes advantage of the regular arrangement of the MCEs to apply the same patterns to all those MCEs, simultaneously. Using this scheme, they were able to reduce the total area by 60% as compared to full-scan and achieve a fault coverage of 99.5%.



**Figure 6:** Asynchronous design with (a) and without scan chain (b)

Another possibility to identify locations for scan latch integration uses analysis of the DUT. A. Khoche and E. Brunvand have presented a partial-scan technique in [KB951, KB952] to get a better coverage for stuck-at-faults than approaches using the self-checking capabilities of ST circuits. They used the testability and structural analysis as well as pattern generation aspects to select the elements that has to be made scannable and which elements should simply be transparent in test mode. Furthermore, they proposed specially adjusted latches to perform tests.

An approach to detect delay faults using partial scan was proposed by M. Kishinevsky et al. in [KK97]. As already mentioned, this was done by reducing the test for delay faults to a test for stuck-at faults. They also used a structural analysis to select the paths that shall be tested for delay faults. Once a path is selected, a pattern sequence for s-a-f is generated to initialize and to activate the considered delay fault.

A full scan technique is presented by F. te Beest et al. in [TB02, TB03]. The proposed method introduces scannable MCEs and adds a synchronous test mode to the circuit, in which the entire circuit is controlled by external clocks. Additionally, the MCEs and other asynchronous standard cells are remodelled such that conventional test generation tools can be used for ATPG. The method results in a test generation flow, capable of automatically testing any handshake circuit with test-quality equal to synchronous circuits. They used their method to evaluate several circuits and reported a stuck-at fault coverage over 99%.

The problem of conventional full or partial scan integration within an asynchronous design is the requirement of a synchronous clock tree. This means that the tree is added to the circuit only for scan operation; hence, a huge area overhead is required only for scan purposes. Additionally, a synchronous clock tree is one of those things that should be avoided with the asynchronous design style due to its required timing and power considerations. A solution to this problem can be an asynchronous scan path, e.g. given by V. Schöber et al. in [SH96, SH01]. They developed special asynchronous scan registers and modules to control the scan registers during test mode. The advantage of this approach is



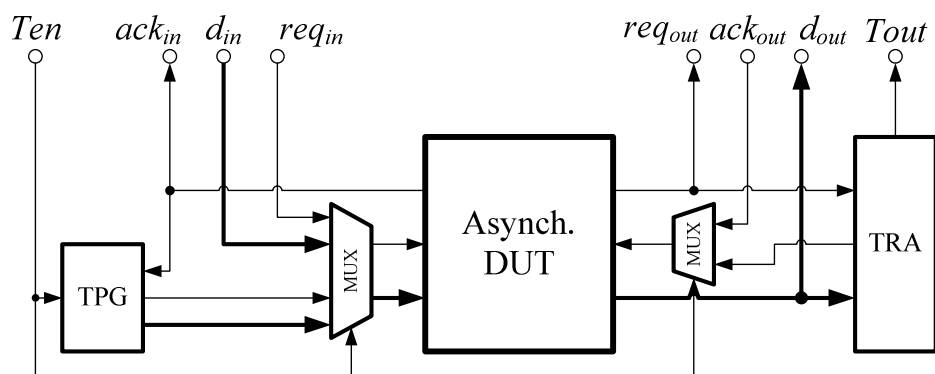
that it maintains the asynchronous behaviour even in scan mode. Consequently, no synchronous clock tree is necessary.

Scan test is capable to test for structural faults and is feasible for performing stepwise tests. However, it is not applicable to test the design at its operations speed, referred to as at-speed testing. Therefore, other schemes must be used, e.g. built-in self-test and on-line fault detection approaches that are presented within the next sections.

## ***Built-In Self-Test***

Built-In Self-Test (BIST) is one way to integrate test capabilities directly into the chip. This enables the advantage that a test can be performed on the operation speed of the DUT, referred to as at-speed testing. Furthermore, it allows the execution of tests when the DUT is already integrated within an operating system. In general, besides the DUT a BIST consists of a test pattern generator (TPG) and a test response analyzer (TRA). Figure 7 shows the general structure of a BIST for asynchronous circuits. In comparison to synchronous BIST, asynchronous communication channels are required for the interconnection of the BIST components. Hence, the TPG must be able to generate a request and must stop its operations until the acknowledge signal arrives. Similarly, the TRA must wait for a request to occur and must generate an acknowledgement. In [AL98] V.C. Alves et al. have presented a simple scheme for organizing a BIST in an asynchronous environment. The approach assumes that the DUT exhibits an end-of-operation signal similar to the request signal of asynchronous handshake circuits. The only difference is the absence of a acknowledge signal. The approach uses a graph-theoretical scheme based on tokens to determine the active component. Their first attempt toggles between all three components TPG, DUT and TRA, where only one component is active. This is enhanced by activating both TPG and TRA in parallel, since the pattern generation and the response analysis can be done simultaneously.

However, a conventional BIST also has some drawbacks, e.g. the lack of diagnosis capabilities. This is because the analysis of DUT's responses is done by the TRA. Depending on its specification the TRA either generates a pass/fail information or a signature out of the DUT's responses. The former option enables a simple analysis, but there is a problem when the pass/fail signal is erroneous. This can be avoided using the second possibility. In comparison to the pass/fail signal, a signature can be further analyzed and a fault within the signature can be detected. But this must be done by an external checker that knows the expected signature. In both cases, one is not able to extract when and where a fault has occurred. Hence, a BIST scheme has less diagnosis capabilities than e.g. scan test. In order to enhance this, a hierarchical BIST can be utilized as it was done by M. Krstic and E. Grass in [KG05]. This technique for GALS circuits will be described in detail in the next section. This technique was integrated into a GALS baseband processor to verify separate components. The baseband processor was divided into a transmitter and a receiver which themselves were subdivided into several SBs communicating via asynchronous channels. The BIST was used to test these asynchronous communication channels as well as the SB. Therefore, the BIST is composed of several TPGs and TRAs to perform individual tests. Thus, different tests can be performed, e.g. local tests including one asynchronous channel and one synchronous block. Also global tests are possible when the TPG and TRA of the GALS bounds are utilized to perform tests. Furthermore, the transmitter and the receiver are connected together in a BIST internal loop, such that stimuli at the input of the transmitter can propagate through the entire GALS architecture to the outputs of the receiver. Hence, a global test that checks the entire design can be performed. The drawback of this BIST design is that the TRAs deliver a pass/fail information that is received by the central controller. An internal fault within the central controller can distort this information.



**Figure 7:** Asynchronous BIST

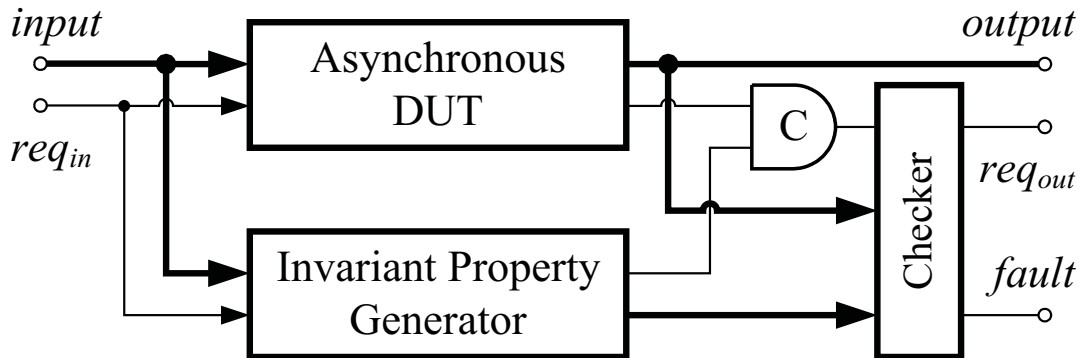
### ***On-Line Testing and Concurrent Error Detection***

A similar approach to observe faults in an IC is on-line testing and concurrent error detection (CED). In comparison to a BIST that explicitly generates test patterns, an on-line testing scheme tests the circuit during its normal operation. Here, additional redundant hardware components are used to estimate, whether the functional unit to be tested operates correctly. One module, namely the Invariant Property Generator (IPG), generates an expected invariant property using the DUT's inputs. The output of the DUT should have this property. Thus, to check for a fault, both the expected property and the property of the actual output, must be compared. Therefore, a checker generates the property from the output and compares this with the expected property. The invariant property may be the parity, i.e. the number of ones modulo 2 within the output vector. But also codes, e.g. Hamming or Berger codes, are used to implement a CED scheme. For example, P.D. Hyde and G. Russel present a comparative study of a synchronous and an asynchronous RISC processor using Dong's code in [HR04].

In order to implement on-line testing methods within asynchronous circuits the IPG and the DUT must be synchronized before the checker can perform the comparison. Otherwise, the checker may indicate an error until both computations are complete. A possible synchronization is shown in Figure 8. Here, both DUT and the IPG receive the request signal from the environment and generate an individual request when their computations are completed. These two request signals are synchronized by means of the MCE that generates the request signal propagating to the checker. After the comparison within the checker, the request is sent to the environment which is then able to receive the result and the fault indication signal. The drawback of this on-line testing methodology is the additional synchronization that may decrease the system's operations speed.

Other techniques do not necessarily require synchronization. In [SY05] D. Shang et al. have firstly presented an on-line checker for handshake protocols. This checker was enhanced in [SY06] to achieve lower area costs. The goal of the checker was to check the protocol for stuck-at faults, premature firings and order violations of occurred transitions. Therefore, the checker scheme is divided into four check modules corresponding to the four stages of the two handshake signals and four control units used to control the active stage check unit. Each stage checking module is organized to check one stage of the protocol. Except their inputs, all these modules have the same structure. They consist of a delay element, one D-Flip-Flop and some logical gates. The delay element determines the minimum delay  $d_{min}$  between two signal transitions to detect premature firings. Hereby, the major drawback is the use of multiple delay elements within the separate stage checking modules. These delay elements require most of the area when they are realized using inverter chains.

In [RK94] a checker scheme for dual-rail logic based on Differential Cascode Voltage Switch Logic (DCVSL) is given by D. Rennels and H. Kim. They implemented a checker in CMOS to detect the forbidden value (11) of the dual-rail code. Furthermore, they have shown how to organize a DCVSL circuit using this checker. To verify their design approach they inserted a few thousands faults into two experimental circuits. No error was undetected, but they mentioned that inserting a couple of faults is not representative for the fault coverage.



**Figure 8:** General on-line testing scheme

### Comparison

The mentioned test approaches have different properties regarding fault coverage, controllability and observability, at-speed testing etc. A comparison of the approaches is shown in Table 1. It is clear that using the self-checking aspects of asynchronous circuits does not influence the performance and the area. Also at-speed testing is possible. However, the observability, diagnose capability and fault coverage are low, since a deadlock indicate at least one fault within the control logic of the circuit only, but it is not possible to determine where the fault has occurred. Scan test seems to be the opposite regarding the considered properties. It offers the best fault coverage, observability and controllability and also diagnoses capabilities due to its possibility of scanning in test stimuli and scanning out responses of separate circuit components. But this also implies more test time and the necessity of replacing all standard flip-flops with scannable ones negatively affects the area and performance of the design. Additionally, the introduction of a clock tree for the scan operation into asynchronous circuits is very extensive. The last two testing techniques, BIST and on-line testing are very similar regarding the requirements. Both techniques allow at-speed testing and are relatively easy to control and to observe when a feasible interface for the tester is added to the asynchronous DUT. BIST techniques often utilize linear feedback shift registers for implementing a TPG and a TRA. Those components are very cheap in area overhead. But these components must be integrated into the DUT using multiplexers, thus, it influences the performance even in normal operation mode. On-line testing techniques do not have this performance lost when the result of the check is not required. Only in test mode an additional delay is required for the check. Because BIST-techniques often use pseudo random patterns rather than real operation patterns, the functionality of the DUT is harder to test. Furthermore, the result is often compressed in the form that many test responses are combined within the TRA to one output signature. Hence, faults can mask other faults and this reduces the fault coverage and the diagnose capabilities. However, a BIST is easy to implement and the required test time is low. The design and integration of an on-line testing scheme can be very complex and often it is not easy to find a feasible invariant property. In addition the test requires much time just like functional tests since the test patterns must be applied separately. Nevertheless, the responses can separately be analyzed resulting in better diagnosis and fault coverage.

**Table 1:** Test method comparison

Test method \ Requirement	Self-Checking	Scan Test	BIST	On-line Test
Observability & controllability	☆	☆☆☆☆☆	☆☆	☆☆
Fault coverage	☆	☆☆☆☆☆	☆☆	☆☆☆
Influences in area	☆☆☆☆☆	☆	☆☆☆	☆☆
Influences in performance	☆☆☆☆☆	☆	☆☆☆☆☆	☆☆☆☆☆
At-speed testing	☆☆☆☆☆	☆☆	☆☆☆☆☆	☆☆☆☆☆
Diagnose capabilities	☆	☆☆☆☆☆	☆☆	☆☆☆
Test time	☆☆☆	☆	☆☆☆☆☆	☆☆☆



# GALAXY

GALS InterfAce for CompleX Digital  
SYstem Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

---

## 4.4 CONCLUSION

Within the last sections asynchronous circuits have been introduced as a perspective for complex SoCs. The replacement of the global clock signal by locally clocked modules is shown as a solution for design problems regarding clock distribution, power consumption and EMI. But due to the resulting lack of controllability and observability as well as difficulties with the communication between DUT and ATE, asynchronous circuits are assumed to be difficult to test. However the last sections have shown that in the last decades many enhancements on testing asynchronous circuits have been investigated and developed. However, the testability of asynchronous circuits has to be further improved to make the asynchronous design methodology applicable in industrial system.



---

## 5 GALS TESTING

---

### 5.1 GALS TESTING PROBLEMS

With the increased complexity of the digital circuits it became beneficial to partition a complex design into the set of smaller localized designs that still function internally synchronously but mutually communicate asynchronously. However, in order to apply the GALS methodology, we need a clear path how to test such structured systems. As expressed in [GU06], as soon as a design using self-timed circuit techniques is mentioned, the first question that pops up is: "How are you going to test this circuit?"

To propose an optimal test strategy for GALS systems, it is necessary to define test requirements. For a GALS application, the test should be clearly separated from the functional part in order that testing cannot affect the correct functioning of the system. Additionally, the test should be easily initialized and controlled. The test circuitry should provide easy observability and controllability of all interesting points in the system. On the other hand, it is desirable to allow hierarchical testing, and testing at the speed of operation. Finally, all tests should show robustness towards the asynchronous behaviour of the wrapper and the highest possible fault coverage.

To summarize, GALS testability research has the following main objectives:

- Provide sufficient fault coverage for all the wrapper elements (port controllers and local clock generators).
- Verify that the LS blocks can communicate with the wrapper.
- Ensure that all GALS modules communicate with each other according to specifications.
- Provide a method for standard test equipment to interface with the individual LS modules, so that they can be tested using conventional scan based methods.
- Enable testing wrapper elements without modification of the LS modules. [GUR02]

Self-timed circuits have always been regarded as being difficult to test. There are two main properties of self-timed circuits that make traditional testing approaches infeasible: [GU06]

1. It is not possible to hold the state of a self-timed circuit using a global signal.

In synchronous systems, once the clock is halted, the state of the system is frozen. It can be observed and manipulated with ease. There are well established methods (i.e. scan based testing) and proven tools to support this approach. It is possible to support similar functionality for self-timed circuits as well [GU06]. For example it is possible to break all combinational loop and insert scannable elements at breaking position, as defined in the pervious chapter. Additionally, it is possible to insert scan function in the special asynchronous sequential elements, such as C-element. But the required test functionality must be part of the circuit definition from the beginning.

2. Self-timed circuits are (in principle) sensitive to all transitions of their inputs.

This increases the amount of failure sources. In synchronous systems, as long as all nodes have the correct value at the time of a clock transition, the system will function correctly. Parasitic transitions of intermediate nodes have no negative effect on functionality in a synchronous system. In self-timed circuits, such glitches can have terminal consequences. Not only that, but signal transitions that are faster or slower than normal may result in the circuit malfunctioning. [GU06]. Therefore, the testing of asynchronous circuits must pay more attention not only to the static (as in the synchronous case), but also to dynamic faults.



There have been numerous approaches to address the needs for asynchronous testing, as we discussed in the previous chapter. When compared to classical asynchronous circuits, it is obvious that the amount of asynchronous testing is much less in a GALS system, since only a limited amount of asynchronous elements (which in turn only comprise 10-20 gates per port) need to be tested. While having a limited number of nodes alleviates the testing problem by some degree, the basic problems of asynchronous testing remain the same: The port controllers and the local clock generators are prone to stuck-at faults, bridging faults, delay faults, and hazards. [GUR02]

Generally, the design for testability (DFT) of synchronous digital systems is based on the scan chain approach. There are similar techniques also in asynchronous domain [TB02]. On the other hand, there are claims that a functional test is sufficiently effective for asynchronous circuits. For example, a GALS test technique based on a functional test is proposed in [GU02]. Built-in Self-Test (BIST) could be a possible compromise between functional and scan-based testing and it is already successfully used in the asynchronous world [AL98]. In the following sections we will discuss about the methods and the strategy for testing of GALS circuits, having in mind the methodology for asynchronous and synchronous circuits described in previous chapters.

## 5.2 SCAN ISSUES WITH MULTI-CLOCK-DOMAIN DESIGNS

As we know, the most of the designs in the industry today have more than one clock domain. It is quite common strategy now to combine different clock domains in the common DfT scan structure. There are several techniques how the test of the multi-clock domain circuit can be performed. This will be described in the following text.

Very frequently happens that although in application the system is sourced from several clock signals, for the test the single clock source is used. However, we have to be sure that the tester period is made equal to the slowest of all clocks and all clocks are pulsed in this period sequentially. However, this requires a lot of change, as sequential ATPG has to be supported.

On the other hand, it is lot easier to generate the tests for one clock domain at a time. One clock pin is used for launch and capture while the scan-in and scan-out operations are done using all clocks. The procedure is then to scan-in data using all the clocks, use one clock pin to fire the launch and capture pulses and finally scan out using all the clocks. When ATPG for one clock domain is complete, the entire process is repeated for a different device clock. The disadvantage is that any “cross-interacting logic” remains untested in this case. [JA03]

If one common clock is being used to test designs, all flip-flops in domains not being tested have to be masked. This places a requirement that the clock tree for this test clock must be able to run at the fastest frequency being tested. Further, all the flip-flops in other domains need to be identified beforehand, so that they can be masked accordingly. [JA03]

In multiple clock domain designs, it may so happen that flip-flops triggered by different clock domains are on the same scan-chain. In such a case, it becomes imperative to place LOCKUP latches at the boundaries of these clock domains, to eliminate clock skews between domains while shifting through the scan chain. Such lockup latches delay the data for half a clock cycle, from the rising to the falling edge, thus providing a high tolerance to skew between domains. This requirement is especially necessary for transition fault tests, though desirable for stuck-at tests [SAX02]. [JA03]

In some designs, designers do not use LOCKUP latches, but intentionally skew the device clocks to make up for the use of LOCKUP latches. This technique works while testing for stuck-at faults using low frequencies. However, this method will most often not work for at-speed testing, as it is almost impossible to simultaneously shift the scan chain, perform launch and at-speed capture in the same test pattern. As a result, the only alternative is to use LOCKUP latches in the scan chain at domain crossings. Additionally, it is imperative to place the trailing edge-triggered flip-flops prior to the leading edge-triggered flip-flops in the scan chain. [JA03]

All those measures could be also applied for the GALS systems. However, it looks unnecessary to build a common scan chain for different LS blocks triggered with the different clock source. Building a



common scan chain for different LS block leads to losing of many advantages of the GALS design and we have to deal with the timing closure of the scan chain at the classical way. Therefore, the easiest choice for the GALS test designer should be to separate clock domain issue and to focus on separate testing of LS modules. On the top of this approach, the designer has to take care about the system and inter-domain testing, without involving the global scan approach in to the DfT flow.

### 5.3 AT-SPEED SCAN ARCHITECTURES FOR GALS SYSTEMS

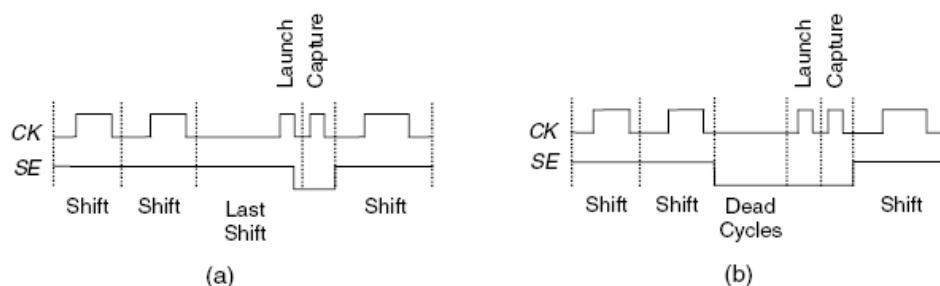
Although scan design is commonly used in the industry for slow-speed stuck-at fault testing, its real value is in providing at-speed testing for high-speed and high-performance circuits. This can be also applied for circuits that contain multiple clock domains, each running at an operating frequency that is either synchronous or asynchronous to the other clock domains. [WST08]

There are two basic capture-clocking schemes for testing multiple clock domains at-speed [WST08]:

- (1) *skewed-load* (also called *launch-on-shift*) and
- (2) *double-capture* (also called *launch-on-capture* or *broad-side*).

Both schemes can test path-delay faults and transition faults within each clock domain (called intra-clock-domain faults) or across clock domains (called inter-clock-domain faults). Skewed-load uses the last shift clock pulse followed immediately by a capture clock pulse to launch the transition and capture the output test response, respectively. Double-capture uses two consecutive capture clock pulses to launch the transition and capture the output test response, respectively. In both schemes, the second capture clock pulse must be running at the domain's operating speed or at-speed. The difference is that skewed-load requires the domain's scan enable signal *SE* to switch its value between the launch and capture clock pulses making *SE* act as a clock signal. Figure 9 shows sample waveforms using the basic skewed-load and double-capture at-speed test schemes. [WST08]

Because scan designs typically include many clock domains, which do not interact with each other, clock grouping can be used to reduce test application time and test data volume during ATPG. Clock grouping is a process used to analyze all data paths in the scan design in order to determine all independent or noninteracting clocks that can be grouped and applied simultaneously. [WST08]



**Figure 9:** Basic at-speed test schemes: (a) skewed-load and (b) double capture

This strategy is a viable option for GALS systems. However we have to take into the account asynchronous paths that contain also sequential logic (C-elements) and loops. In order to enable this technique, one should, in the test mode, break all loops and include the C-elements and the scan-chain.



## 5.4 STUCK-AT FAULT TESTING OF GALS-SYSTEMS

Stuck-at fault approach is the most common approach in the synchronous world. There are many commercial CAD tools (for example Synopsys Tetramax etc.) supporting completely ATPG for stuck-at model for standard synchronous systems.

Using a stuck-at fault testing approach for GALS wrapper elements is also a known approach [TRA05]. Gürkaynak et al in [GU02] have investigated the suitability of a stuck-at fault testing approach for their GALS implementation. Since all port controller net lists are obtained using the simple university CAD tool 3D, they consist of several simple And/Or structures. A test method was developed that was able to take advantage of this specific structure. At first sight it seems very straightforward to solve the testability problem using conventional ATPG methods. However there are several problems: [GU06]

- In order to produce hazard-free outputs, most of the ports contain redundant logic (that guarantees hazard-free operation). As a result some internal nodes can not be tested without introducing additional test inputs and/or outputs.
- Some gates that are required to realize the additional testability need to be placed within the asynchronous signal paths. This would affect the timing of these blocks. As a result the ports would have to be re-synthesized according to the delays introduced by the test blocks.
- Since the GALS port controller is essentially an asynchronous state machine, even if sufficient inputs and/or outputs are added to achieve reasonable error coverage, the problem of finding a sequence of test vectors to excite the circuit is not a trivial problem as it involves sequential rather than combinational test pattern generation.
- The correct functionality of the GALS port can not be verified by stuck-at faults alone since the GALS ports are extremely sensitive to path delays, and may fail to function properly if the path delays are not within certain limits.
- While the high-level port descriptions remain the same across different technologies, the actual mapping to standard cells may differ due to the available standard cell library. As a result the test methodology can not be made technology independent.

As a combined result of the above mentioned problems, stuck-at fault based testing approaches were less than satisfactory for GALS implementation. The solutions that were investigated required either extensive additional test inputs or outputs which degraded the operation speed considerably or yielded very poor test coverage. Furthermore the problem of interfacing to external synchronous testing hardware remained unresolved with this approach.

Testing is an essential part of the IC manufacturing process. Since there are very few self-timed circuits that have been manufactured in the industry [Ron99], the quality of test solutions for self-timed circuits, when compared to their synchronous counterparts, is clearly lacking. [GU06]

The self-timed test problem is tackled in two main directions:

1. Using a functional approach - Certain faults in self-timed circuits lead to clearly observable behaviour, usually such circuits stop functioning altogether. Several approaches have been proposed that rely mainly on such 'self-checking' behaviour [MA91, Wie95, GUR02].
2. Modifying the self-timed circuit to support scan-based testing - Since scan based testing is well-known and effective, most self-timed test methodologies try to add scan capability to state holding elements [PF95, KB95, BPTB02]. Unfortunately, full-scan based self-timed test methods incur a very large area penalty, at times doubling the circuit area. In order to keep the overhead at acceptable values, partial-scan methods have been suggested as well. [GU06]

The GALS methodology developed by Muttersbach used a synchronous fall back method, in which, for test purposes, all AFSMs were bypassed and synchronous state machines were used instead. The resulting system was a fully synchronous system that can be tested normally. This method was more of an emergency solution and several alternatives were considered that actually tested the AFSMs as well. The AFSMs used in the GALS system are very limited in complexity. Therefore, instead



of devising a general method that is capable of testing any given AFSM, practical methods that ensure adequate test coverage for the AFSMs used in the GALS methodology were investigated. [GU06]

At first, a method that added scanable elements to the asynchronous connections was considered. This method, similar to the one presented in [BPtB02] introduces a very large area overhead. Such an arrangement also interrupts the asynchronous handshake signals between GALS modules, slowing the communication and reducing the throughput. Since all AFSMs are tested in isolation, this method fails to detect delay faults that occur because signal transitions between two AFSMs are either too slow or too fast. On top of that, it was shown that the stuck-at test coverage of this approach is not above 90%. [GU06]

In a GALS system, only a very small portion of all stuck-at faults are in the AFSMs. As an example, in *Acacia*, the total number of stuck-at faults is 154,604. Only 182 (0.118%) of these faults are within the AFSMs. This was the main motivation to develop a functional approach to test the AFSMs within the GALS system. This approach [GUR02] adds a Test Extension Element (TEE) to the each GALS module. The TEE is clocked by a synchronous test clock. During test mode, the TEE is able to decouple the *Pen* signal from the LS island and initiate data transfers on all self-timed connections. In this way, all data connections within the GALS system can be tested individually. This idea is very similar in principle to the IEEE P1500 standard proposed for testing embedded cores [MAR99]: Inter-module communication is tested by initiating data transfers between modules. This technique will be described in more details in sections 5.5 and 5.6. [GU06]

## 5.5 FUNCTIONAL TEST FOR GALS SYSTEMS

Before the scan-based structural test techniques were implemented, functional pattern testing was the only test method being employed in manufacturing test. Many problems are associated with using functional at-speed patterns. The major problems include the cost associated with developing them, difficulty in debugging them and the high cost of test hardware. Fault simulation is a major problem as very few tools exist that can fault grade functional patterns for delay fault coverage. Since the quality cannot be easily assessed, the level of fault coverage will be unknown. Even if such tests are generated, a large pattern volume is required to achieve the necessary coverage. As clock rates and design sizes increased, testing using functional patterns became more impractical. [JA03]

However, there are still many areas where functional testing is still used. Such examples are high-performance circuits where timing and hardware overhead introduced by scan is not acceptable.

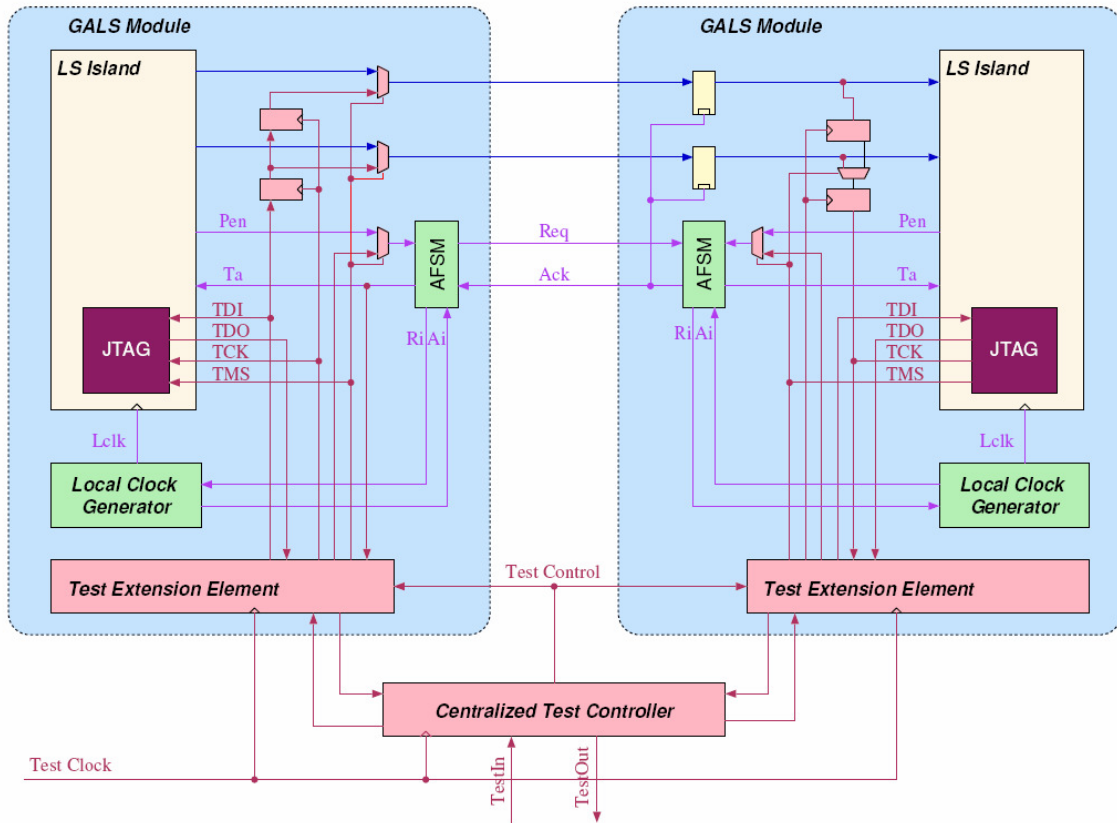
Additionally, for testing of GALS interconnects the functional testing could also make lots of sense. For example, instead of testing the internal structure of the GALS ports, in [GUR06] is proposed a method to control and monitor the communication between pairs of ports. In this method means to control each port are added to the wrapper. For testing purposes the GALS system operates in a test mode where the port controllers are isolated from the LS block and are controlled by the test circuitry. It is therefore possible to initiate data transfers between individual GALS ports. Testing is achieved by observing these data transfers. As the test system controls only the interface between the LS block and the GALS ports, the asynchronous data path is not interrupted by test hardware and the asynchronous communication rate is maintained during testing. A functional test is feasible, to test asynchronous elements in the GALS system, since the asynchronous circuits are of limited complexity [GU06].

In this work the test extensions for GALS System were developed to provide a common interface for test purposes. Unlike the stuck-at fault based methodology that concentrates on testing individual GALS port controller elements, the functional test methodology uses the transfer channel as the main primitive in terms of testing. The transfer channel, shown in Fig. 10, consists of a point-to-point connection including an input port and an output port of two separate wrappers.

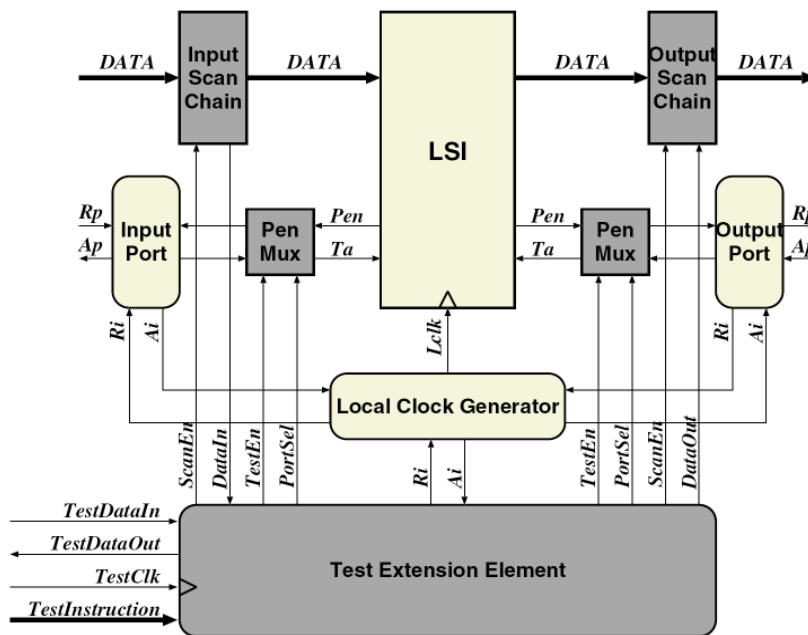
The test methodology basically consists of controlling the individual transfer channels independently from the LS block. The necessary circuit to decouple the port controllers from the LS blocks is part of the test extension element (TEE) [GU06]. The TEE provides all GALS modules within a GALS system with a unique interface to the test controller. Each TEE is customized to a specific LS island and contains hardware to access data inputs, data outputs, the local clock generator, and port



control signals within the wrapper. Figure 11 shows a simplified block schematic of a TEE connected to a LS island [GU02].



**Figure 10:** Functional testing of a handshake channel using test extension element



**Figure 11:** Simplified block level schematic with connections between TEE and LSI



## 5.6 FUNCTIONAL TEST PATTERN GENERATION AND FAULT COVERAGE FOR GALS INTERFACES

As concluded in the previous section functional testing should be always performed since stuck-at and dynamic faults detection and coverage is of the highest importance. Here we show an example of functional testing for stuck-at faults.

In a typical GALS system, only a very small portion of all stuck-at faults are in the AFSMs. As it is mentioned in section 5.4, in Acacia [GU06], the total number of stuck-at faults is 154,604. Only 182 (0.118%) of these faults are within the AFSMs. This was the main motivation to develop a functional approach to test the AFSMs within the GALS system.

The majority of the faults are located within the locally synchronous islands and can be detected using standard stuck-at fault testing methods. However, it is necessary to provide access to standard test interfaces. The local clock generator can be configured to run in a test mode, where an external synchronous clock is used instead of the locally generated clock. While system functionality can not be maintained in this mode, scan-chain based test methods can be applied to test all locally synchronous islands. Such a test is sufficient to provide test coverage for most of the chip [GUO06].

Faults within:

- Local clock generators,
- Self-timed port controllers,
- Glue logic within the self-timed wrappers,
- Input and output registers of the locally synchronous islands,

can not be detected using this method. The local clock generators are designed with a special configuration port that allows them to be configured and monitored during operation. This interface can be used to provide a test solution, and all faults within the local clock generator can be detected reliably. The remaining faults are detected using a functional approach. The procedure is the following: first, a list of all remaining faults is obtained. Then, for each remaining fault, the netlist of the design is modified and the system is fed with functional vectors that stimulate the chip to execute data transfers between all GALS modules. Faults that manifest themselves at the circuit outputs are marked as detected [GUO06].

For the control of LS islands in test mode a simplified TEEs can be used. Individual TEEs are controlled by a centralized test controller (Figure 10). The centralized test controller can be implemented in hardware, giving the system a built-in self-test capability, or implemented as a test program on automated test equipment. The TEE also enables the test controller to access the LS island through a JTAG interface. In this methodology, all LS islands are assumed to have their own test solution.

A special local clock generator developed by Stephan Oetiker [OVG02] can enable switching between the locally generated clock and an external synchronous clock for configuration. In a way, the functionality of the TEE can be integrated into the LS island and the local clock generator. During test mode, all GALS modules are run with the same synchronous test clock. This allows standard tools to be applied to generate the ATPG patterns. All LS islands can be fully tested using this method. However, the stuck-at faults within the AFSMs, the local clock generators, and the data interface including the latches used for retaining data, can not be detected with these patterns. Figure 12 shows the recommended configuration. All scan chains of individual LS islands are connected and standard ATPG test vectors can be used to test the circuit. The shaded areas represent the portions of the system that contain stuck-at faults that can not directly be detected using the scan chains [GU06].

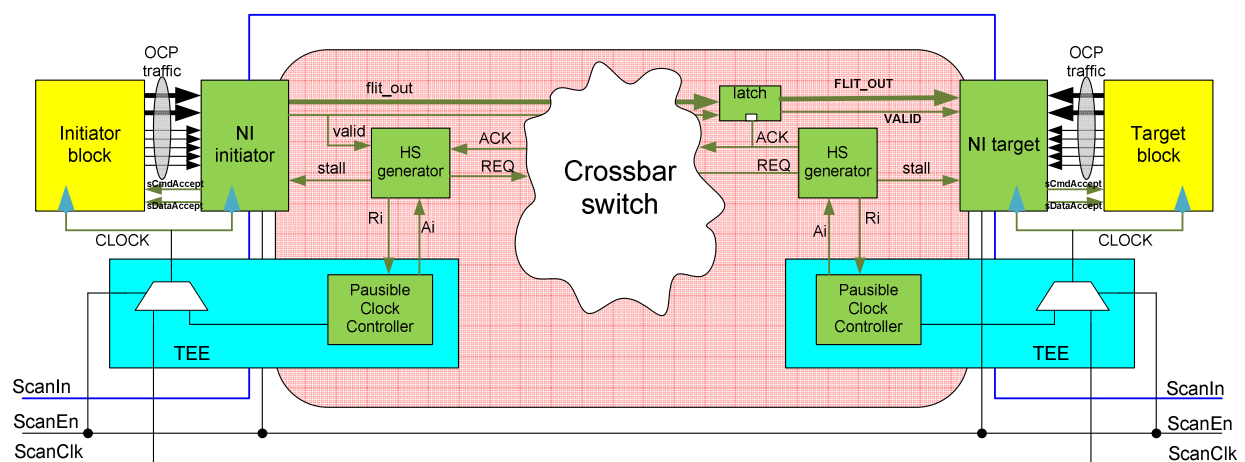




### 5.7 FUNCTIONAL TESTING OF NoC GALS INTERFACE

Following the example from the previous section a functional testing approach is proposed for the GALSified NoC Network Interface (NI) (Figure 13). GALSified NI (green blocks in Figure 13) is designed in a way to have a local plausible clock and again a special local clock enables switching between the locally generated clock and an external synchronous clock for configuration. In a way, the functionality of the TEE can be integrated into the LS island and the local clock generator.

On order to implement functional testing all the latches and flip-flops within the Network Interface (NI) should be changed to scanable flip-flops. Again TEE can be very simple with only ScanEn and ScanClk signals and special local clock generator implementation.



**Figure 13:** The red shaded area represent portion of the circuit that is not covered by these tests. Functional test vectors are used to detect the stuck-at faults in these areas.

First netlist standard test vectors for synchronous modules (blocks+NI) are generated and the majority of the faults are located within the locally synchronous islands and can be detected using standard stuck-at fault testing. Since AFMSM is very simple and small (10 equivalent gates), the percentage of undetectable faults that need to be further considered at gate level simulation is relatively small.

Again, for each remaining fault, the netlist of the design is modified and the system is fed with functional vectors that stimulate the chip to execute data transfers between all GALSified NIs. Each node that was observed to have changed its value more than 4 times during this simulation is considered to be detectable for stuck-at-1 and stuck-at-0.

### 5.8 BIST FOR GALS SYSTEMS

Built-in self-test methods form an alternative for applying a test vector from the outside of a chip. In the previous chapters we have seen the basic BIST methodology for synchronous and asynchronous systems. An important advantage of BIST is the test speed especially for large chip. Testing can be done at the internal speed of the chip. This is faster than using an external test setting. To prevent the use of a big and costly chip area, (pseudo)random sequences generated are used instead of one created by an ATGP program. One important feature of the BIST that it can be successfully applied also during the chip operation (usually offline). A linear feedback shift register can be used to generate pseudo-random sequences. [BL02] In the case of logic BIST, the output response comparison functions are also built into the chip. BIST could be a possible compromise between functional and scan-based testing for GALS systems, and it is already successfully used in the asynchronous world [Krstic].



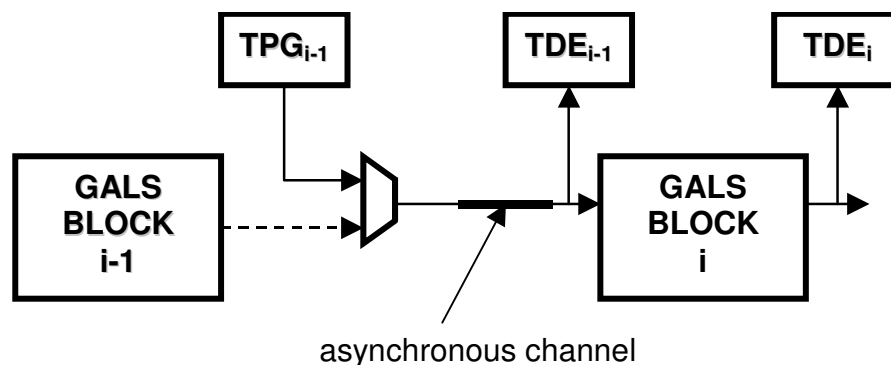
BIST has obvious positive sides (at-speed testing, no need for external ATPG, etc.) but there are also some negative. The common problems include:

- There is an incremental area penalty for the BIST controller, which may not be easily justifiable.
- The requirement of an at-speed scan-enable calls for added changes to the design and test hardware.
- There is an added routing complexity.
- Complications with timing closure are always a concern. [JA03]

One additional issue of BIST approaches is the power increase, and there are several techniques how to limit the power consumption of the BIST testing.

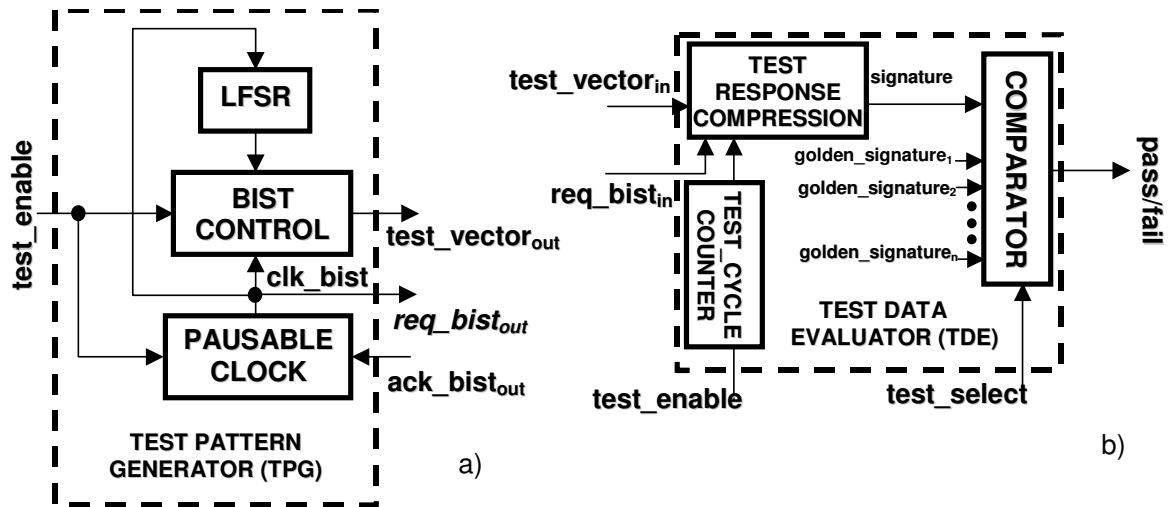
One detailed BIST architecture for GALS systems was introduced in [KG05].

The architecture of one asynchronous channel is given in Figure 14. In order to test the operation of the channel, several test components are placed. A test pattern generator (TPG) is positioned at the output of the asynchronous GALS wrapper and the test data evaluator (TDE) is placed at the input of the asynchronous wrapper. In principle, any placement of TPG and TDE around the wrapper is conceivable. It is proposed to use the depicted configuration because the priority is testing of the asynchronous channel. However, the testing of the handshake signals is possible only implicitly by compacting the valid data transferred via the asynchronous channel. One of the control signals (*REQ* or *ACK*) are used to trigger the respective TDE. With an additional TDE, as shown in Figure 14 it is possible to evaluate the operation of the locally synchronous part, too. The TDEs and TPGs must be designed in such way that they operate completely asynchronously and follow the handshake protocol.



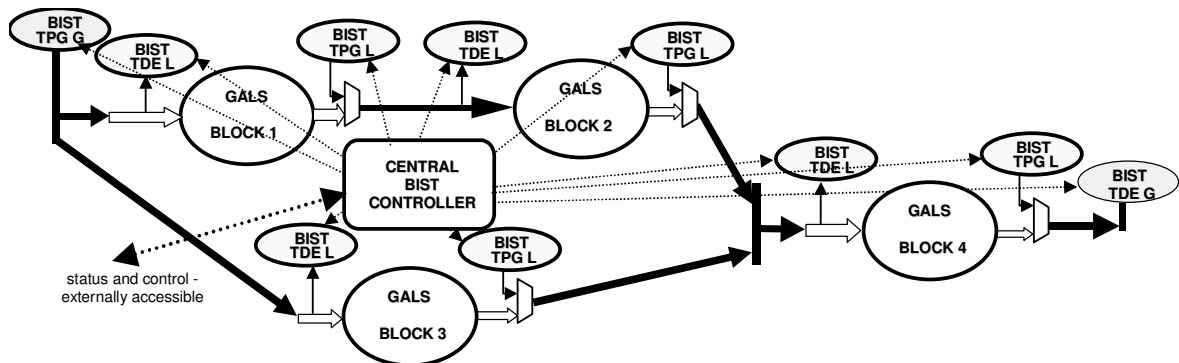
**Figure 14:** BIST configuration

The general architecture of the BIST building blocks is shown in Figure 15. Additionally, interfacing of several BIST blocks is illustrated in Figure 16.



**Figure 15:** BIST components

A test pattern generator (TPG), given in Figure 15a, has to generate patterns that are suited to perform the testing of internal operations of an asynchronous wrapper and the operation of a locally synchronous module. The TPG consists of a Linear Feedback Shift Register (LFSR), a BIST controller and a pausable clock generator. The LFSR has to generate pseudorandom patterns. The BIST controller has to control the LFSR and to generate test vectors based on the LFSR output and, optionally, on some predefined test patterns. This predefined data structure is usually connected with the functional requirements of the specific GALS block. The purpose of this additional logic is to enable non-random test vectors that open the LS datapaths for particular operations.



**Figure 16:** Global BIST configuration

For example, sometimes it is necessary to combine the pseudorandom data pattern with certain defined sequences in order to activate deeper datapath structures of a complex GALS system. Finally, a clock generator is needed in the TPGs, because our GALS blocks are request-driven. We need to trigger the synchronous part of the TPG in parallel with the token generation for the GALS block. However, in general it is not always easy to use an ordinary external clock source for this purpose because this clock cannot be stretched if the acknowledge on the GALS line does not arrive before the next rising edge of the clock. Therefore, a pausable clock is needed, based on a tuneable ring oscillator to drive the request and BIST clock. For better component utilization, it is proposed to use just one ring oscillator for all local TPGs, because only one of them is needed for a particular test procedure.

The test data evaluator (TDE) should check the output test data and indicate the result of the testing. The TDE consists of a test response compression circuit and a comparator, as shown in Figure 15b.



The test response compression block is based on signature analysis and, accordingly, incorporates one LFSR in its structure.

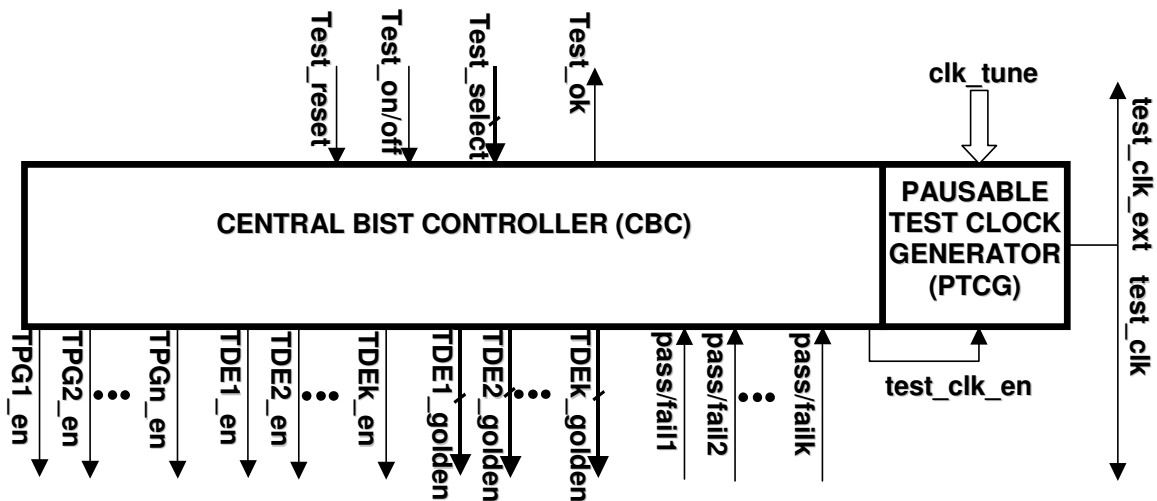
The TDE is triggered with the respective handshake signal. It records the changes on the data lines only when a valid control token is present. Because of that, with this BIST approach, the timing nondeterminism could be tolerated. For comparison, with the actual signature it uses one of  $n$  precomputed golden signatures. Which golden signature will be used depends on the performed test. The golden signature could be chosen via the *test\_select* control signal, defined by the global BIST controller. The number of test cycles is defined in the test cycle counter. This circuit is counting the number of accepted valid test vectors and when this number is equal to the depth of the test vector set, the test response compression block will be disabled and a pass/fail signal is generated.

In this solution two different types of BIST have been implemented: global and local. The global BIST (Fig. 16) is initiated at the circuit boundaries and, therefore, is performed fully synchronously. Consequently it is based on the general well-known BIST strategy. Additionally, during the global test, it is possible to enable the local test compaction, placed between the GALS blocks, in order to increase the number of check points and to achieve a better fault coverage. When the global test is finished, our test controller can activate the local tests in order to further evaluate various components of the GALS system. During every local test one local test pattern generator is activated and the results are stored in one or more test data evaluators. With such a strategy, testing may give broad and profound results about the correctness of the circuit operation and the operation of specific GALS blocks. With an increased number of test points and with local testing it is possible to isolate faulty components and possibly perform diagnostics of the fault.

For global testing of the system, a global test pattern generator (TPG G) is used. This generator may function completely synchronously. The generated data should propagate through the system and a global test data evaluator (TDE G) should process output data and compare it with the expected value. Additionally, there are several local TPGs and TDEs (TGG L and TDE L) that are driven completely asynchronously.

In order to control the BIST registers and to collect and process the results of test, one central BIST controller (CBC) is proposed. This CBC includes a pausable test clock generator in its structure. CBC can be programmed to prepare a particular test during system reset when the GALS circuits are stable. Then the BIST can start immediately after the deactivation of the system reset. After some period, which should be sufficient for performing the test, the test status data is delivered.

In general, the purpose of the CBC is to allow a simple off-chip communication mechanism that can be used for test parameter setting and test result reading. From the signals in Figure 17, *Test\_reset* is used for the reset of the testing circuitry, *Test\_on/off* is used for the activation of the testing operation and *Test\_select* should select which specific vector set is activated. Signal *Test\_ok* indicates the success of the test, when all test vectors are processed. On the other hand, the CBC controls the execution of the particular test, based on the external setting. For that reason, *TPGi\_en* and *TDEi\_en* enable the respective TPGs and TDEs, *TDEi\_golden* selects the golden signature value for a respective TDE, and *pass/fail<sub>i</sub>* should indicate the result of the testing. In addition to that, the CBC enables the operation of the pausable test clock generator (PTCG) when the setting of all parameters of the current vector set is finished and all TPGs and TDEs are ready to perform a particular test. Moreover, it is possible to adapt the CBC such that it supports some of the existing test standards as EJTAG etc. This activity of all BIST I/O pins is completely timing deterministic and synchronised with the external clock. Hence, a hardware tester can be used for automatic testing. The described BIST structures are applied in the GALS baseband processor.



**Figure 17:** Central BIST Controller (CBC) configuration

In general, BIST clearly separates the test circuitry from the functional part. On the other hand, in the case of a scan-based approach, the test structure is closely coupled with the functional components. BIST is also very useful regarding simplicity of the initialization and control procedure. For the scan-based approaches test generation and management is not easy but there are many tools which support the designer. For functional tests the test generation is a critical point. There is a big problem how to access deeper pipeline stages in the system with the functional test. Furthermore, the number of vectors is usually huge making the cost of testing unacceptable. It is much easier to organize hierarchical testing based on the BIST a on the scan methods. [Krstic

Regarding observability and controllability it is clear that scan testing offers the best results. Functional testing has the problem that many errors can be masked inside the design and are not observable on the I/O pins. In general, the functional test and BIST suffer from low test coverage in comparison with scan based approaches.

However, regarding time for testing, it is much cheaper to reach a certain level of test coverage with BIST then in case of the functional test. Finally, at-speed testing is much more difficult with scan methods. The only possibility for this type of tests is to use either BIST or functional tests.

Very important advantage of the BIST tests for GALS systems is a possibility to run very complex functional tests internally (without providing external test vectors). The hardware testers are strictly cycle based and cannot react to asynchronous output signals of the circuit. The GALS arbitration processes may preclude cycle level determinism. PVT (process, voltage, temperature) variations further contribute to timing non-determinism. BIST significantly reduces the effort for generating a test program and enables us to use a synchronous tester.

Therefore, BIST represents one reasonable and preferable option for testing of GALS systems (rather than global scan insertion). Several reasons support that: the number of scan elements could huge and the time for testing will be unacceptable, functional testing could verify the correct system operation with a high probability, the scan-insertion will diminish the performances of the system. Additionally the scan approach doesn't cover dynamical faults, race states, it increases the global test clock tree for scan-cells that we wanted to avoid with GALS. However, applying functional test for GALS will require expensive testers and increase the cost of testing.

## 5.9 RECOMMENDED TEST STRATEGY FOR GALS

Previous chapter was dedicated to the theory of testing with a focus on asynchronous circuits. This section summarizes specific on recommendations on test development for GALS.



- Use scan approach wherever this seems to be feasible and possible

Create one or more scan-path to reach all latches or flip-flops together of the synchronous parts. The scan-path may operate only when the circuit is in the test mode and can be generated by tools for synchronous designing. Automate test pattern generation tools can therefore be applied in order to minimize the test time and maximize the fault coverage. An external clock is needed in order to shift the states of the scan-path. Reasonable approach would be that each LS block has its autonomous scan chain(s). However, it is also possible to combine different LS domains into the single scan chain as in [GU06]. In any case we can apply any of the scan methods that are anyway used for multi-clocking systems, as described in Section 5.2 and 5.3. Asynchronous wrappers could be included into scan chain, using the previously described approach (breaking of internal loops and making all sequential elements scannable). In this case, particular asynchronous wrapper should normally belong to scan chain of respective locally synchronous.
- Use test monitors for handshaking circuits

The control circuit of the asynchronous part can be tested in normal operation/functional mode. If the handshaking between modules get stuck in compare with a golden chip, the chip is faulty and therefore useless. Therefore a test pattern has to be designed which has to activate all the asynchronous channels in the chip [BL02]. Since it is very difficult to generate test patterns able to catch all possible dynamic faults that may appear in an asynchronous circuit under nanometre process environment, it is very useful to implement test monitors that can online follow the functioning of handshake circuits.
- Combine testing of synchronous and asynchronous components with BIST

For large chips the method of using scan-paths may be use to much time. In those cases building the self-test capabilities provide a suitable solution just as the do for synchronous designs. Each module can have it's own BIST structures. They can even work concurrently [BL02]. Additionally, it is possible to have also the global BIST test that will test the complete functionality of the system. It is very useful that BIST test achieve as good as possible test coverage and this can be checked with appropriate tools.
- Always perform functional test.

Although for the classical synchronous designs the main focus is on structural testing, for GALS and asynchronous circuits it is very important to verify operation for dynamic faults. Therefore, some sort of the functional system testing is needed, either over BIST or by testing asynchronous channels as in [GU02]. Those functional tests may also, for the limited circuit complexity as in the case of the asynchronous wrappers, achieve very high test coverage for the stuck-at fault model, but it can also cover many dynamic faults.
- Use multi-level approach for GALS testing

It is not necessary to base the test of the GALS system on a single test strategy applied for all system components in the same time and with integrated DfT structure. It is highly advisable to utilize GALS modularity and organize also testing at the different hierarchical levels. In particular, local test methods should be implemented on the locally synchronous level (possibly including respective asynchronous wrapper). Applied local test methods should be based on the proven and commercially available tools and methods. For example, the usual strategy would include scan test on the local synchronous level. With this approach we have to grant the access for ATE to each locally synchronous block.

However, it is also advisable to generate the test method that will operate on the global level and test the complete system functionality. This global test could be based on functional or BIST approach. Implementation of standard stuck-at scan methods on the global level is as we have shown possible but in the context of the global asynchrony looks unnecessary. However, it is on the system designer to decide whether investment into global test scan methods pays-off or not.



- Use commercial DFT tools as much as possible

There is a large spec of already available DTF CAD tools for DFT insertion and ATPG test generation based on the different fault-models (for example Tetramax etc.). Those tools are today very effective and mature and for state-of-the-art GALS system those tools should be as much as possible utilized. DTF scripts should be optimized for GALS and asynchronous logic, to enable successful testing. However, in principle if asynchronous logic is prepared for scan insertion and the loops are broken, the commercial ATPG tools could be directly used. Utilization of academic DFT tools is also possible but usually difficult due to their immaturity, restriction in system complexity, and lack of support and documentation.

- Enable use of the industrial hardware testers and ATE in general

Today industry standard is application of complex hardware testers for manufacturing test. The market leaders, such as Verigy, Teradyne, Advantest, and LTX Credence have developed large set of different testers optimized for SoC testing, memory testing, or mixed-design testing. However, most of those testers are based around the synchronous paradigm and cycle based. Therefore, the direct introduction of event-based asynchronous logic to the cycle based testers is usually not possible without careful evaluation of test strategy. Therefore, the input and output signals of the CUT have to be synchronized and prepared for cycled sampling and strobing. An additional potential problem is timing nondeterminism that is present in the asynchronous and potentially GALS circuits due to the arbitration process. Hardware testers usually have difficulties to deal with non-deterministic signals. Therefore, non-deterministic behaviour of IO ports has to be avoided. One elegant solution for this problem could be application of BIST test on the system level, however with synchronous and deterministic communication with tester.

One important aspect of ATE support is ability to access the local units of the GALS system. If we focus on multi-level test approach for GALS systems we must grant the access for the tester to each locally synchronous block. One way to do it is implementation of JTAG ports able to access each block separately.

If the asynchronous components are included in scan test structure we could also test the system for dynamic faults by loading the test vectors with a low frequency clock and executing the test achieving real throughput of the system.

- The test strategy for highly complex GALS systems and NoCs is in principle the same as described

The proposed strategy is shown simplified at Fig. 18.

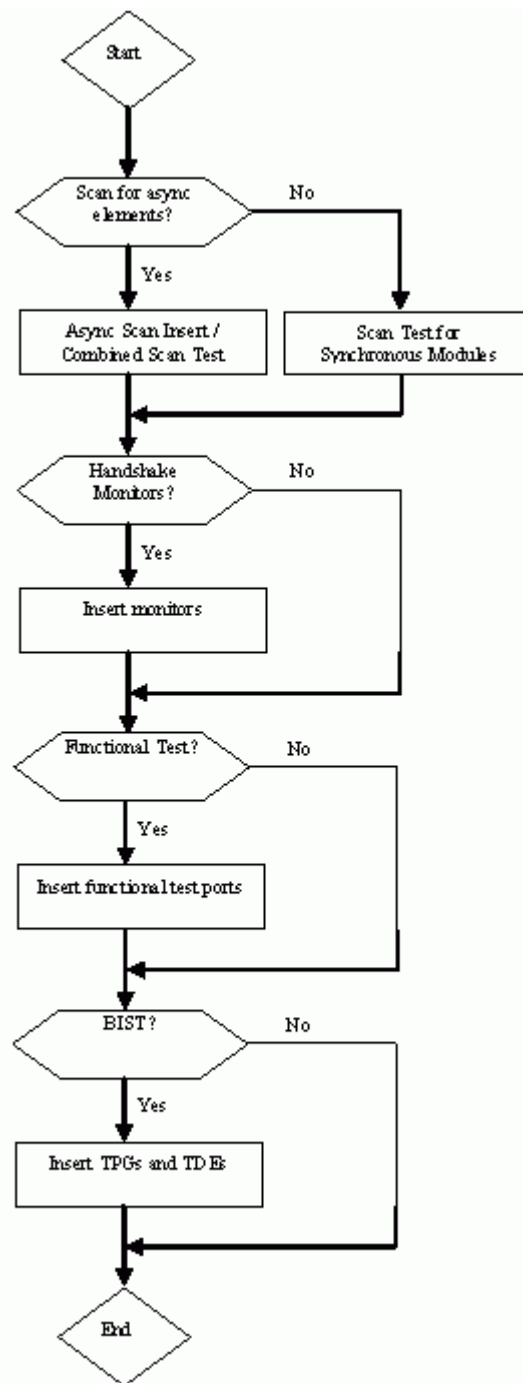


Figure 18: Simplified Test Flow

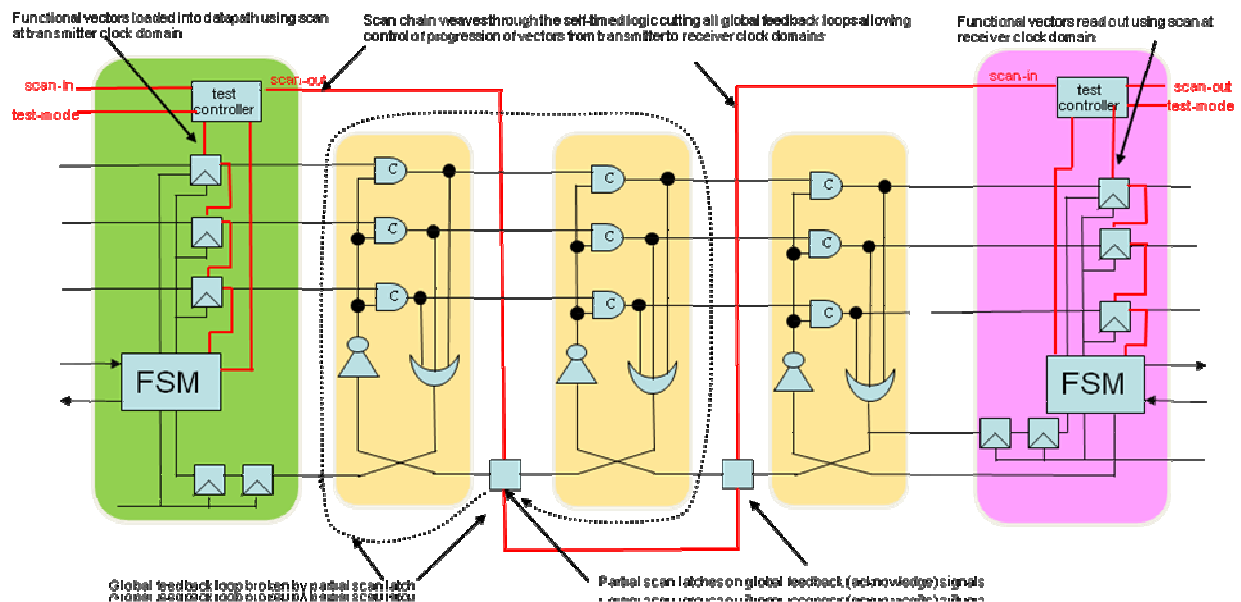
## 5.10 DESIGN-FOR-TEST IN SILISTIX CHAIN NoC

NoC brings interesting opportunities for improving the design-for-test (DFT) methodology of SoCs by providing a high-speed access mechanism to each IP-block's primary interfaces. Macrocells test access methods can be easily facilitated over the NoC and is a good fit with the need to separately and concurrently test many blocks within the NoC architecture. It also provides the modularity needed to separate testing of the self-timed NoC implementation from the usual test procedure of the endpoint



logic. Such separation is required because the self-timed implementation of the CHAIN NoC means that it is incompatible with the conventional scan-insertion (and in some cases also the ATPG tools) from mainstream vendors. However, in this sense these components are no different than other blocks that are instantiated as hard macros such as memories and analog blocks, and like those blocks the vendor has to provide a DFT solution.

Multiple test strategies can be used for the CHAIN NoC components. The first, and least intrusive to a normal EDA flow is to use full-scan, where the clockless CHAINlibrary components are constructed such that every C-element features an integral scan-latch. This approach is compatible with all existing scan-chain manipulation tools and conventional ATPG approaches. The more advanced, and lower cost approach relies on a combination of functional patterns and sequential, partial scan. In both cases similar 99.xx % stuck-at fault coverage as is achieved as in regular clocked logic test. Consideration of a pipelined path from a transmitter to a receiver as illustrated in Figure 19 can illustrate how the sequential scan approach works.



**Figure 19: Scan-latch locations for sequential scan**

Scan-latches are placed on targeted nodes, typically feedback loops, state-machine outputs and select-lines that intersect the datapath. These are shown explicitly in the simplified pipeline of Figure 19 but in reality they are encapsulated in (and P&Red as part of) the hardmacro components. Testing the network is then a three-stage process:

- The first pass of the test process uses just the conventional scan flops in the transmit and receive hardmacros to check the interface with the conventionally clocked logic.
- Then in transport-layer test mode the same scan flops are used to shift functional vectors into the datapath at the transmitter. The circuit is switched back into operational mode (where the global-feedback partial-scan latches connect straight through without interrupting the loops) and the vectors are then transmitted through the network at-speed and then read out at the receiver using it's scan-chain. This achieves good coverage of all of the datapath nodes and many of the control-path nodes through the network. It is not essential, just more efficient to use this step since all faults can also be detected using the final pass below.
- The final pass uses the partial-scan latches on the global feedback loops in non-feedthrough mode to break the global loops allowing access to monitor and control the acknowledge signals. This enables the propagation of the test vectors to be single-stepped through the pipeline giving further increased coverage and improved ability to isolate the location of any faults that are detected.



# GALAXY

GALS InterfAce for CompleX Digital  
SYstem Integration

Confid. Level: Public  
Date : 01/11/2009  
Issue: 1

---

The full set of required patterns are generated by the CHAINworks tools in STIL format for use with conventional testers and test pattern processing tools. Achievable coverage is verified using conventional third party concurrent fault simulators.

Highly efficient delay-fault testing is performed using a variant on the functional-test approach. Patterns injected at a transmit unit are steered through the network to a receiver and the total flight time from the transmitters clocked/asynchronous converter to the receivers asynchronous/clocked converter is measured. Any significant increase above the expected value [as reported by Silistix high-level NoC-architectural synthesis tool] is indicative of a delay fault somewhere on the path between the two ends. This approach is very efficient for detecting the absence or presence of delay faults, but does not help with localization of exactly where a delay-fault is. However, the scan access facilitates such localization of any faults detected, albeit through a circuitous route.