Scan Test Strategy for Asynchronous-Synchronous Interfaces

Octavian Petre and Hans G. Kerkhoff

MESA+ Research Institute
Testable Design and Test Group
7500AE Enschede, The Netherlands
E-mail: O.Petre@utwente.nl

Abstract

In the next years, the well-known synchronous design style will not be able to keep pace with the increase of speed and capabilities of integration of advanced processes. Asynchronous design will become more and more common among digital designers, while synchronous-asynchronous interactions will emerge as a key issue in the future SoC designs.

This paper will present test strategies for 2-phase asynchronous-synchronous, and vice versa, interfaces. It will be shown how test vectors can be automatically generated using commercially available ATPG tools. The generated ATPG vectors will be able to test all the stuck-at-faults within the asynchronous-synchronous interfaces.

1 Introduction

The complexity and speed of SoCs are increasing and chip designers are desperately trying to keep up with them. Recently designed video processors contain more than 90 clock domains and more than 50 cores. New design paradigms, like core reuse of the already designed synchronous modules and asynchronous designs, are considered in order to cope with the ever increasing complexity. The future SoCs will contain multiple synchronous and asynchronous cores. As the software industry has demonstrated, the reuse of previously designed modules is an efficient way for building complex systems. However, the hardware engineer has not only to cope with the increase in complexity of systems but also with the deviation of the parameters of the devices induced by a particular technology.

Synchronous architectures have been the main design stream until now. Increased complexity and speed of the future ICs will hamper this design style. It will not be possible anymore to operate the entire design synchronously with a single clock.

Asynchronous design seems the best way to make use of high-speed technologies. However, commercially available CAD tools which are able to handle asynchronous design in an efficient way are almost non-existent, albeit there are numerous tools from academia like Petrify [3], Balsa [2] and some proprietary tools like Tangram [12].

Testing is also still in its infancy for these types of designs, although recently much effort has been invested to find suitable and easy-to-use testing strategies for asynchronous designs [10].

The future SoCs will probably be comprised of asynchronous and synchronous cores. At the interface between these blocks there will always be half-synchronous half-asynchronous modules, so-called synchronizers, which will facilitate the correct communication. Testing these synchronizers will require an integration of the test strategies for the asynchronous and synchronous blocks. Despite the fact that the fault coverage gain can be small, due to the small silicon area occupied by these devices, the synchronizers are crucial for the good operation of the whole SoC.

This paper will investigate, from the testing point of view, the interfaces between asynchronous and synchronous cores [8]. Section 2 will briefly explain the functionality of these interfaces while the next section (3) will show our test strategies for the presented modules. The results of fault simulations are presented in section 4 to prove that all the faults within the asynchronous-synchronous interfaces will be detected by the generated ATPG vectors. In section 5, interaction between multiple asynchronous and synchronous cores will be considered. In this section it will be shown that full-scan test strategies can still be applied to test such complex systems. Finally, the last section presents the conclusions.

2 Asynchronous - Synchronous Interfaces (ASI)

Asynchronous-synchronous interfaces using stretchable clocks or *Point to Point GALS Interconnect* as they are de-

Proceedings of the Eighth IEEE European Test Workshop (ETW'03)
1530-1877/03 $17.00 © 2003 **IEEE**

fined in [7, 8] represent a very efficient way to synchronize asynchronous and synchronous domains. These small-sized interfaces are filling the necessary gap, by bringing together the two design styles. The asynchronous design style will be used more and more due to its good ECM compatibility and low-power dissipation, while the synchronous modules will continue to exist mainly because of IP core reuse considerations.

There are slightly different types of synchronizers between asynchronous and synchronous domains based on the direction of the data. For both designs, the clock for the synchronous part is generated on-chip by a stoppable ring oscillator [9]. From a testing point of view this can be considered a drawback for low/medium-speed ICs, which share a common clock. However, the ASIs are targeted for high-speed architectures. Generating clocks on-chip will be the only viable design paradigm for future designs.

The following two subsections will explain the two types of asynchronous-synchronous interfaces in more detail.

2.1 Asynchronous - to - Synchronous Interface

Figure 1 presents an interface between an asynchronous core and a synchronous one. The asynchronous core is using the two-phase handshake protocol [4] for data exchange. In reference [8], this type of interface is called *"asynchronous producer and synchronous consumer"*.

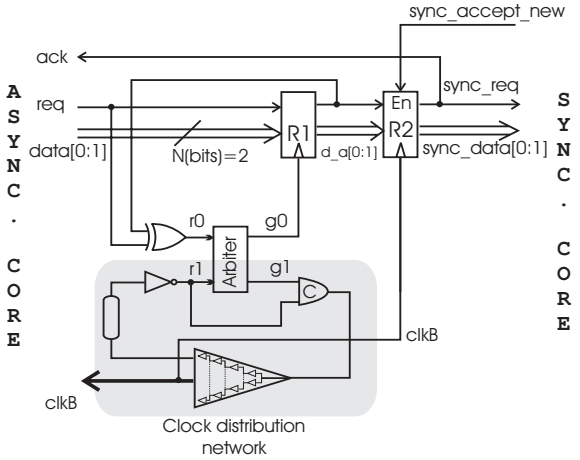

Figure 1. Asynchronous producer - synchronous consumer interface [8]

Different from the synchronizer in [8], the clock distribution network has been included in the delay path, the shadow area in figure 1, similar to what has been presented in [9].

The main idea of this ASI is to stop/stretch the synchronous clock (*clkB*) if its rising edge arrives nearly at the same time with the request signal (*req*) from the asynchronous core. This operation is performed by the arbiter shown in figure 1.

An exhaustive functional description of this circuit is given in [8].

2.2 Synchronous - to - Asynchronous Interface

Figure 2 illustrates the reverse data communication, between a synchronous producer and an asynchronous consumer. The asynchronous core is again using the two-phase handshake protocol [4] for data exchange. In reference [8], this type of interface is called *"synchronous producer and asynchronous consumer"*. The clock distribution network has been included in the delay-path as before.

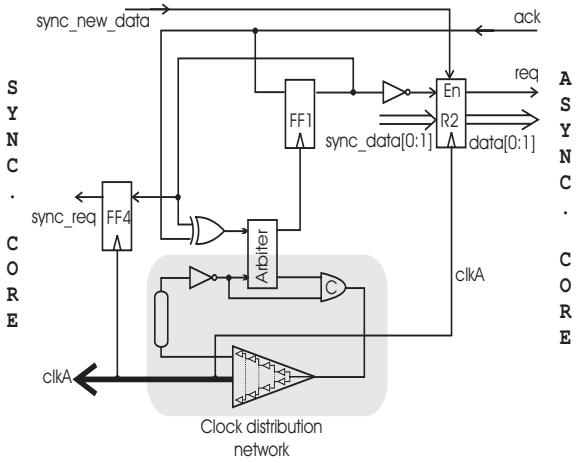

Figure 2. Synchronous producer - asynchronous consumer interface [8]

For this type of ASI, the synchronous clock might be stopped for a short time when an acknowledge signal (*ack*), received from the asynchronous core, is occurring at the same time with the rising edge of the clock.

An exhaustive functional description of this circuit is given in [8].

3 Test strategies

Asynchronous-synchronous interfaces are likely to be used extensively in the future. However, no new design style will be accepted for mass production unless, among other things, there is a good testing strategy associated with it. Many challenging designs are put on hold as a result of the inability to test them.

Nowadays, the most important test strategy for digital ICs is the scan technique. The technique is simple, easy

to implement, and gives very good results in practice even if the model used does not fully match the real faults. In the meantime some of the asynchronous designs have become scan-testable [10]. Synchronous design is, by default, scan-friendly. A natural conclusion would be to also use the scan-test technique to test the asynchronous-synchronous interfaces.

Testing the interface modules presented in figures 1 and 2 is not a straight-forward task if a full-scan technique is desired for compatibility with the digital design style. One can recognize hard-to-test asynchronous modules like the arbiter and the C-element. Also, the *req* or *ack* signals are propagating both in the control and data paths. An immediate solution would be to split these signals in test mode. However, this initial solution would result in a lower fault coverage. Our solution is to add 100% testable DfT hardware, followed by a remodeling of the interfaces. In this way they will become fully synchronous with respect to the ATPG tool.

The main assumption for our presented test strategies is that the asynchronous part can be made scannable. However, the asynchronous design style is quite diversified and some of them are not scan-test compatible. If the asynchronous part is not scannable, then the solution can still be applied but it would not be so efficient in terms of fault coverage.

Another example of potentially scannable asynchronous design is the Huffman circuit structure [5]. A scan structure can easily be implemented in this case by adding scannable flip-flops in front of the delay-lines.

3.1 Testing the Synchronous Producer - Asynchronous Consumer Interface

The main idea of the scan-test strategy for the asynchronous-synchronous interfaces was to observe that this schemes are basically buffers.

Figure 3 presents the test-scannable ASI for a synchronous producer - asynchronous consumer. Apart from the synchronous-asynchronous interface presented in figure 2, one may notice the relevant scannable asynchronous part represented by the combinational logic circuit (*CLC*) block and the FF3 flip-flop.

It can be seen that very little additional hardware has been added to the asynchronous-synchronous interface itself. This DfT hardware is represented by the wavy filling in figure 3, being basically a multiplexer and an inverter.

However, if this scheme would be fed into an ATPG tool, it could not be recognized as a scannable design. Therefore, it is necessary to remodel the scheme for the ATPG tool. The resulting scheme is shown in figure 4.

This scheme behaves, in test mode, exactly as the one in figure 3 if all the flip-flops are reset before the first test vector is scanned in. In practice this is always the case; before starting the scan procedure, all the flip-flops are reset.

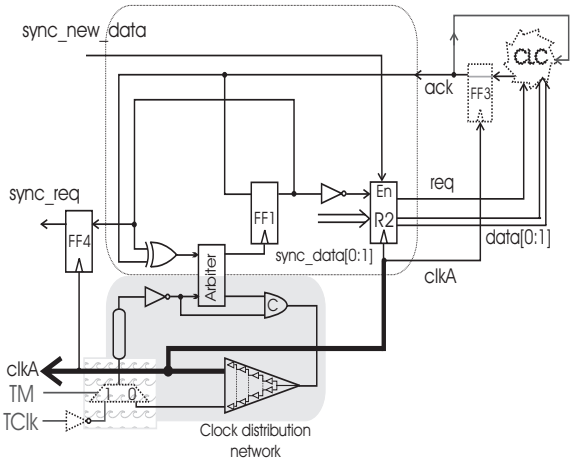

Figure 3. Test structures included in the synchronous producer - asynchronous consumer interface

During the scan mode, the *ack* signal is changing all the time, depending on the data that is scanned in. Therefore, the value stored in the FF1 flip-flop is the last value scanned in the FF3 flip-flop. Hence, from the ATPG point of view, all circuitry between the *ack* signal and the R2 register can be replaced by an inverter.

The test vectors can now be generated by a commercial ATPG tool for the circuit in figure 4. These vectors will be able to detect all stuck-at faults that may exist in figure 3. More detailed information proving that these test vectors are indeed detecting all the possible faults is given in section 4.

3.2 Testing the Asynchronous Producer - Synchronous Consumer Interface

Figure 5 presents the test-scannable asynchronous-synchronous interface. In this figure one can recognize:

- the initial synchronizer schematic presented in figure 1
- the relevant asynchronous part made scannable represented by the *CLC* block and the R3 register
- an inverter in front of an extra MUX controlled by the *TE* signal
- another inverter and MUX controlled by the *TM* signal

The *TE* signal represents the test-enable signal and is '1' during the scan operation. During the capture operation, the *TE* signal value is '0'.

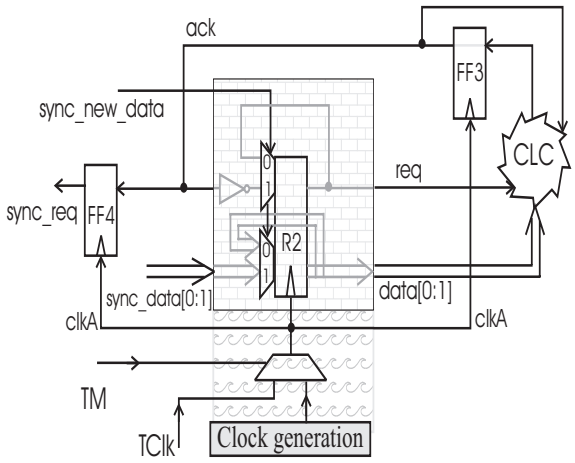

Figure 4. Test structure fed into the ATPG tool for the synchronous producer - asynchronous consumer interface

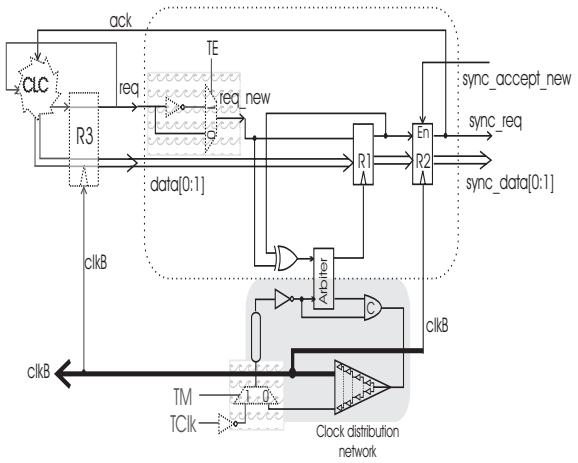

Figure 5. Test structures for the asynchronous producer - synchronous consumer interface

During the scan mode, the *req* signal is continuously changing depending on the data that is scanned in, while the *TE* signal is '1'. Therefore, the *req* value stored in the R1 register is the negation of last the *req* value scanned in the R3 register. After the last bit has been scanned-in, the *TE* signal will become '0' which will force a transition on the *req_new* signal. This transition will latch the *req* signal in the R1 register. Hence, from the ATPG point of view, all the circuitry between the *req* signal and the R2 register can be replaced by a buffer.

Compared with the design-for-test scheme of the synchronous - asynchronous interface presented in figure 3, one may notice an additional MUX block inserted in the *req* signal. This MUX has been introduced, as explained earlier, in order to force a transition on the *req_new* signal immediately after the scan has been completed. This is compulsory since the *data* bus signals must be latched in the R1 register. Without the inserted MUX, the *data* signals could have different values than the intended scan data.

The ATPG scheme to be used in order to generate the test vectors for this circuit is presented in figure 6.

The scheme in figure 6 behaves, in test mode, exactly as the one in figure 5 if a few conditions are met:

- The period of the test-clock signal, *TCLK*, should be sufficiently long in order to accommodate the propagation of the *req* signal through the complete logic path - inverter, MUX, XOR, ARBITER - until the R1 register.
- The *TE* signal should be delayed sufficiently long so that the arbitrary transition on the *req* signal, after the last bit was scanned-in, is able to propagate through the MUX, XOR, ARBITER and the R1 register. If this is not the case, then the latched values in the R1 register may have a random value.

The *TCLK* clock has usually a low frequency as compared to the functional one, while the delay of the *TE* signal can be programmed in the test equipment. Therefore, the above restrictions will neither affect the test time nor hinder the test strategy.

4 Fault Simulations - Verifying the ATPG Test Vectors

The presented test strategies require a complete verification. The main question is: *Are the test vectors, generated by commercial ATPG tools, sufficient for testing all the other stuck-at faults that where removed from the modeled schemes?* The answer is yes and the following paragraph will justify this answer.

An arbitrary scheme was considered, which consists of a synchronous core, an asynchronous core and an ASI. The asynchronous core was replaced with the scan-synchronous counterpart. Following this change, the asynchronous-synchronous interface has been replaced with the modeled ATPG version. The next step was to generate test vectors for the "modeled for ATPG" scheme. The ATPG tool used in our work is *TetraMAX* [11]. The generated test vectors have been used to perform fault simulations of all the faults not included in the "modeled for ATPG" scheme. Since the ASI contains an arbiter and a C element, it is not possible to fault-simulate the entire system using conventional fault simulators. As a result, *HSPICE* [6] has been chosen to

Table 1. Fault coverage for the considered scheme

Modeled for ATPG scheme		The original scheme	
Number of faults reported by the ATPG tool	280	Additional number of faults	+55
Number of undetected faults	2	Additional undetected faults	0
Number of ATPG vectors	13	Detected faults by ATPG vectors	55
Reported ATPG fault coverage	99.29%	Additional fault coverage	100%
Total number of faults: **335**			
Total number of detected faults: **333**			
Total fault coverage: **99,40%**			

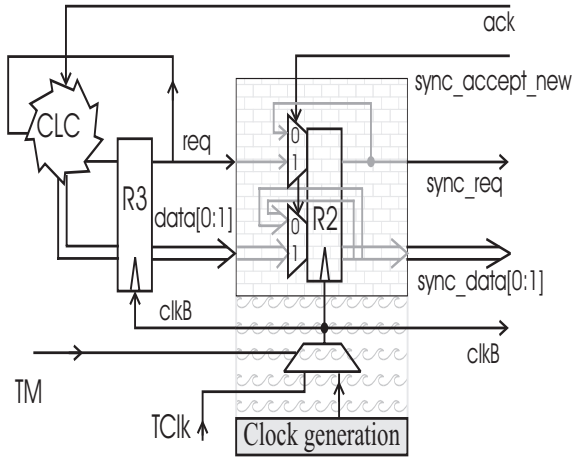

Figure 6. Test structure fed into the ATPG tool for the asynchronous producer - synchronous consumer interface

perform all the fault simulations. The SPICE model and the technology used were BSIM3v3 and UMC $0.18\mu m$ respectively. Other tools, like a VHDL/verilog simulator could also have been used.

By using the ATPG test vectors as input stimuli, we were able to prove that all the additional stuck-at faults were detected by the test vectors generated by *TetraMAX*. Table 1 shows the results for the considered circuit. The left columns present the data reported by the ATPG tool. The right columns show the data of the entire original (non-modified) structure. One may notice the additional faults that were not considered by the ATPG tool. However, these additional faults are 100% tested by the ATPG vectors. This is the case because the ASIs can be seen as buffers, and the 55 additional faults are most of the time functionally-equivalent [1] with the faults recognized by the ATPG tool. As a result, the total fault coverage is improving.

The 2 undetected stuck-at faults are related with the multiplexing operation of the clock signal. Table 1 shows also the number of test vectors generated by the ATPG tool, being 13. This is a small number of test vectors which are generated for the entire circuit. Therefore, for a real-life SoC, the burden of generating and applying additional test vectors, in order to test all ASIs, is negligible.

A flow chart of the steps presented above is shown in figure 7.

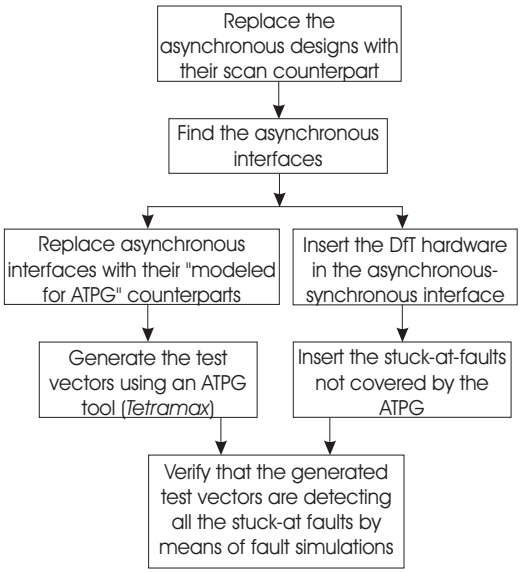

Figure 7. Flow chart for verifying the test vectors

5 Multiple Asynchronous Producers - One Synchronous Consumer

In a real life example, there is usually more than one asynchronous producer sending data to a synchronous consumer. In order to accommodate this situation, some modifications to the clock generation procedure should be performed. The resulting new scheme is presented in figure

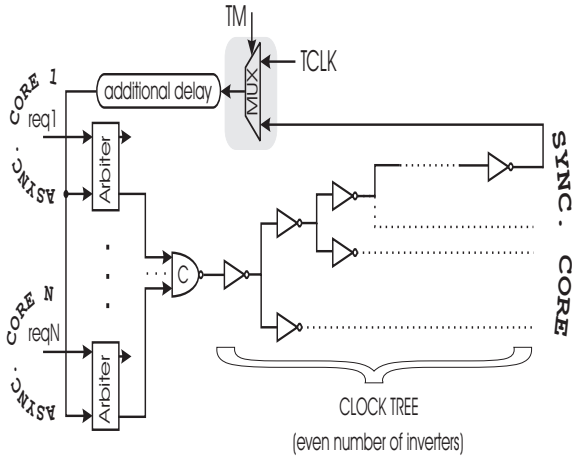

Figure 8. Multiple asynchronous requests for a single clock domain

In this figure, the DfT hardware has also been taken into account. This is represented by the MUX encompassed in the shaded area in figure 8. This MUX is necessary to control the clock in test mode. The test strategies for this type of configuration will remain the same.

The opposite case when there is only one synchronous producer and many asynchronous consumers does not pose any problems. There should be no modification for such types of interfaces. Also, the test strategies remain the same.

6 Conclusions

In the next years, the asynchronous-synchronous interfaces will become a common component in complex SoC designs. Therefore, it is essential to develop integrated test strategies with the surrounding cores. Albeit the area of the ASIs is small, their functionality is crucial for the whole SoC system. Testing is difficult since the ASIs contain asynchronous circuits inside.

An approach to test specific two-phase asynchronous-synchronous interfaces has been presented. These interfaces were previously tested in a functional way. The presented solution is able to test them in a structural way by using commercially available ATPG tools. Fault simulations have been carried out to prove that the structural test vectors generated by the ATPG tool are detecting all the stuck-at faults that may appear in the asynchronous-synchronous interfaces together with the additional DfT hardware. Using the previously scan test strategy, it has been proved that it is possible to test any combination of asynchronous synchronous cores as long as the asynchronous parts can be made scannable.

Acknowledgments

The authors would like to thank Rob Tijink for the CAD tools support. The Technology Foundation STW, applied science division of NWO, has financially supported this research.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design (revised printing)*. ISBN 0-7803-1062-4. IEEE Press, 1990.

[2] A. Bardsley and D. A. Edwards. The Balsa Asynchronous Circuit Synthesis System. *Forum on Design Languages (FDL2000)*, September 2000.

[3] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*, E80-D(3):315–325, March 1997.

[4] W. J. Dally and J. W. Poulton. *Digital Systems Engineering*. ISBN 0-521-59292-5 (hb). Cambridge University Press, 1998.

[5] S. Hauck. Asynchronous design methodologies: An overview. *Proceedings of the IEEE, Vol 83, No.1*, pages 69–93, January 1995.

[6] *HSPICE Golden Accuracy Circuit Simulator*. Synopsys, Inc. 700 East Middlefield Rd. Mountain View, Ca. 94043 (650) 584-5000 or (650) 943-4401.

[7] S. Moore, G. Taylor, P. Cunningham, R. Mullins, and P. Robinson. Using stoppable clocks to safely interface asynchronous and synchronous subsystems. *AINT'2000 Asynchronous Interfaces: Tools, Techniques, and Implementations*, pages 129–132, July 2000.

[8] S. Moore, G. Taylor, R. Mullins, and P. Robinson. Point to point GALS interconnect. *Eighth International Symposium on Asynchronous Circuits and Systems*, pages 69–75, April 2002.

[9] A. E. Sjogren and C. J. Myers. Interfacing synchronous and asynchronous modules within a high-speed pipeline. *Proceedings of the 17th Conference on Advanced Research in VLSI (ARVLSI '97)*, pages 47–61, September 1997.

[10] F. te Beest, A. Peeters, M. Verra, K. van Berkel, and H. Kerkhoff. Automatic scan insertion and test generation for asynchronous circuits. *International Test Conference*, pages 804–813, October 2002.

[11] *TetraMAX ATPG; High-Performance Automatic Test Pattern Generator Methodology Backgrounder*. Synopsys, Inc. 700 East Middlefield Rd. Mountain View, Ca. 94043 (650) 584-5000 or (650) 943-4401, May 1999.

[12] K. van Berkel, J. Kessels, M. Roncken, R. Saeijs, and F. Schalij. The VLSI programming language Tangram and its translation into handshake circuits. *EDAC'91: IEEE European Design Automation Conference, Amsterdam (The Netherlands)*, pages 384–389, February 1991.