Abstract – The purpose of this paper is to present a novel built-in Clock Domain Crossing (CDC) test and diagnosis methodology for Globally Asynchronous, Locally Synchronous (GALS) systems. The methodology allows design and prototype validation, low maintenance and repair costs, and production / lifetime at-speed test. Moreover, high resolution diagnosis is obtained, to identify which device(s) and/or communication channel(s) is (are) faulty. This is not trivial in GALS systems, for which the CDC issue is challenging. The underlying principle of the proposed methodology is to embed a CDC test and diagnosis (CDC T&D) structure in each locally synchronous domain. Complete device-to-device communication channels are tested, including transceivers, buses, and board connectors. Identical test patterns (generated to detect static (stuck-at, shorts and open faults) and dynamic (crosstalk) faults) are used in each FPGA. The proposed CDC T&D methodology is validated in a case study, the acquisition electronics of a complex multi-board, multi-FPGA (nine Xilinx™ xc2v4000-4bf957) system. Test and validation results are presented.

Keywords – GALS, Synchronism, CDC (Clock Domain Crossing) Test and Diagnosis

I. INTRODUCTION

Complex, multi-board, multi-device, multi-clock electronic systems pose severe problems in the test and diagnosis arena. Many of these systems are GALS (Globally Asynchronous, Locally Synchronous) systems [1][2].

In GALS systems, each synchronous subsystem, usually referred as clock domain, operates with its own clock signal. Hence, synchronization is a relevant problem [3]. In fact, communication among clock domains demands harmonization of the different clock phases, in order to guarantee that data is safely transferred among clock domains. Clock Domain Crossing (CDC) strategies and hardware techniques have been proposed to deal with this problem [3][4]. Survey of CDC hardware techniques is presented in [4].

Test is necessary for design and prototype validation, and to constrain maintenance and repair costs. Hence, production and lifetime test is needed, which quests for BIST (Built-In Self Test) solutions. Many physical defects and signal integrity problems manifest themselves as dynamic faults, thus requiring at-speed test. Physical defects can become active due to aging. Moreover, high resolution diagnosis is required, in order to identify which device(s) and/or communication channel(s) are faulty.

Moreover, functional test of asynchronous circuits is not an easy task, since most external ATE (Automatic Test Equipment) is cycle-based and cannot provide event-based handshake signals. Nevertheless, GALS functional test requires that co-existing domains can be tested, as well as all the entire communication infrastructure [7]-[11].

In order to avoid the ATE problem, built-in CDC test and diagnosis (CDC T&D) methodology is proposed in this work. High resolution diagnosis is also a key factor of success to lower maintenance and repair costs. In this paper, the focus is on T&D of the communication channels among locally synchronous clock domains in a GALS system, particularly, the issues associated with clock domain crossing. The CDC T&D structure (CDC Tester) is embedded within the synchronous domains, and the CDC Interface is used for CDC communication.

The paper is organized as follows. In section II, the CDC T&D methodology is presented, emphasising the architectural aspects, and the test and diagnosis procedures. In section III, the case study implementing the methodology is described. Section IV presents the experimental results, and section V summarizes the main conclusions.

II. TEST AND DIAGNOSIS METHODOLOGY

Target systems, in the context of this paper, are GALS systems, constituted by a set of locally synchronous clock domains, interconnected either by single or multiple buses, or by HUB/SWITCH nodes.

The underlying idea of the proposed methodology is to house all the functionality, even the T&D functionality, inside the synchronous domains and to guarantee the correct communication and communication synchronism, by using a CDC Interface.

From a T&D point of view, two basic strategies are considered. First, the GALS system is self-testable at device, board and system level, using a Built-In Functional Test (BIFT) methodology [13]. With BIFT, the system and its modules are tested for the target application, for different operation scenarios, from a functional (not structural) point of view. Although a significant overhead is associated with...
BIFT, a dramatic test cost saving is achieved. BIFT also enables lifetime test and diagnosis. Second, a communication channels self-T&D methodology is proposed, for the multi-clock domain environment, referred as CDC testing and diagnosis.

The focus of this paper is on the second strategy. A CDC T&D methodology is proposed that may allow us: (1) to verify at-speed that the physical communication infrastructure is defect-free and does not induce signal integrity problems; (2) to identify and locate defects, if errors are detected. Verification is made off-line. Each channel is tested at the speed at which it is used in the system’s functional mode.

A T&D Architecture

In Figure 1, a bus-based GALS system architecture is depicted, highlighting the locally synchronous clock domain modules. As shown, each clock domain is composed by the Core, the Bus Controller, the CDC Tester and the CDC Interface. Each core can perform the mission functionality or a self-test routine.

At system level, the communication infrastructure may encompass physical buses, transceivers and board connectors. Point-to-point, or device-to-device communication paths must be tested, at-speed, in both directions, as bidirectional signal flow may occur in normal operation. Hence, each clock domain may act in the system as transmitter or as receiver.

In CDC test mode, data flowing in the buses are test vectors generated (or stored) in each CDC Tester. Defect-free communication channels must deliver the same data they receive. As shown in Figure 3, both the transmitter and the receiver domains must store the same data. We refer it as the golden data. Hence, each CDC Tester has a Test Pattern Generator (TPG) module, which stores identical golden data in its embedded memory. It also houses a TAE (Test Answer Evaluator) module. TAE (Figure 3) generates data to uncover errors in data transmission in the communication channels and in the CDC Interface, and to drive the diagnosis process. In CDC test mode, each transmitter device act as TPG, and the receiver(s) device(s) act as TAE. Test vectors are transmitted by one module to the CDC nodes and received by the remaining modules. In each receiver, a comparison with its own golden data is carried out. The result is stored and/or analysed in the TAE.

The TAE is constituted by a comparator, an Accumulator and a RAM. This architecture allows a high resolution diagnosis. A first hint on the source of error can be derived from the accumulator contents. However, to achieve a more detailed diagnosis, it is necessary to observe the data stored in the RAM contents, which is constituted by the SourceID (identification of the transmitter domain), the TPG Addr (address of the test vector used for comparison) and the error. In the diagnosis procedure, we describe how this data is used for diagnosis.

In order to generate the golden data, the TPG process uses fault models for open, shorts, crosstalk and delay faults. At present, test vectors are generated according to a Modified Counting (MC) algorithm (Counting sequence (CS) has been proposed by Kautz [9]) to uncover static faults (open, shorts and 2-nets coupling). MC algorithm solves the problem of aliasing [8] presented by the CS algorithm. The MA (Maximum Aggressor) fault model (proposed in [7]) has been used to uncover static and dynamic faults (glitches and delayed responses, due to cross coupling or other effects).

The total test length (TL) of the MC algorithm is $TL_{MC} = \log_2(N+2)$, where $N$ is the number of nets, or of bit lines in the bus. Obviously, the test pattern length, TL, is the closest integer equal or greater than $\log_2(N+2)$. For MA fault model, $TL_{MA} = 6N$ vectors are required for the detection of the four

---

**Figure 1 – System architecture highlighting CDC tester insertion**

**Figure 3 – CDC Tester architecture**

**Figure 3 – TAE architecture**
types of faults [7]. According to [7], with this set of vectors, 100% crosstalk fault coverage can be achieved. Hence, the total test length of the golden data is \( TL = 6N + \log_2(N+2) \).

Using the test vectors generated by MC+MA algorithms allows the discrimination between static and dynamic faults, enhancing diagnosis.

The number of transactions involved in a test run is: \( \# \text{transactions} = \# \text{interfaces} \times (\# \text{vectors} + \text{communication overheads}) \). For a system constituted by 10 interfaces, a 64bit Bus with a communication overhead of 10 transactions, the total \( \# \text{transactions} \) is around 4000. For a Bus rate of 10Mtransactions/second and considering 1 test run/second, it corresponds to 0.04% of Bus occupation.

In CDC test mode, the Bus Controller must be able to identify and correct an error in the test header, and is responsible for: a) Starting the test, or; b) Identifying that a CDC test session is active; c) Converting data to/from bus protocol; and d) Sending and/or receiving data test vectors.

B Test Procedure

When the CDC test mode is active, the BUS arbiter selects a clock domain as transmitter. This domain sends the test header, and activates its own CDC tester that prepares the golden data to be send to the other domains. When the bus controller, in each (other) domain, receives the test header, it initiates the test procedure, by activating its own CDC tester and waiting for incoming test vectors.

In transmitter mode, the corresponding CDC tester acts as TPG and the CDC Interface writes in the bus the golden data, which is broadcasted to all devices interfacing the bus. In each clock domain, in receiver mode, the CDC Interface reads test vectors from the bus, and sends them to CDC tester, which acts as TAE by comparing the incoming test vectors with its golden data, stored in its TPG module.

All \( n \) clock domains are selected as transmitters in sequence. Hence, the CDC test session is composed of \( n \) runs. The result of comparisons, for each run and in each receiver, can be used for different purposes. If the received and stored vectors match, no transmission errors are detected. If vectors don't match, error vectors are generated. For each failing vector, \( T_i \), the failing bits are identified. After the \( n \) runs, a test report is generated.

C Diagnosis Procedure

Diagnosis can be carried out in a hierarchical way. By simply observing the accumulator contents, a first level diagnosis can be performed, to identify some possible error sources. As an example, assume a single-bus GALS. In each domain, there is an accumulator and a RAM. When one domain behaves as transmitter, the others behave as receivers.

Observing the accumulators in the different receiver domains the following possibilities exist: 1) \( \text{ACC}_i = 0 \not\rightarrow \text{no error} \); 2) \( \text{ACC}_i \neq 0 \rightarrow \text{error in domain } i \); 3) \( \text{ACC}_i \neq 0 \text{ and } \text{ACC}_j \neq 0 \rightarrow \text{errors} \); 4) \( \text{ACC}_i \neq 0 \text{ and } \text{ACC}_j = 0 \rightarrow \text{it is possible to assume that the error source is in a path shared by both domains} \); and 5) \( \text{ACC}_i = 0 \text{ and } \text{ACC}_j \neq 0 \rightarrow \text{it is possible to assume that the error source is in a path used only by domain } j \).

In order to increase the diagnosis resolution, a second level diagnosis is needed, requiring additional data. This will be demonstrated in the simulation results section.

Using only the accumulator contents, more detailed and reliable diagnosis can be obtained by observing the test results when another domain is chosen as transmitter. Comparing two tests results can provide additional information. As an example, assume a single-bus, 3-domain system (see Figure 4). Table 1 represents a given diagnosis data of this faulty system. Assume that only stuck-at (static) faults may occur.

Error in bit 4 is present in all communications involving domain 1. Hence, an error in bit 4 exists in a IN/OUT path of domain 1. This error is not in the BUS; otherwise, it would be present in all the table entries. Error in bit 10 is present only when domain 3 is transmitting. Thus, the error source must be located in the OUT path of domain 3. Finally, it is possible to infer that domain 2 is fault free.

Relatively easy inference of error sources is harder (low resolution), when coupling and/or dynamic faults are present. There are also indirect errors (transceiver commands, clocks, etc.) that further increase the difficulty of the diagnosis. In this case, the accumulators' contents are not sufficient and it is necessary to use the RAMs contents (level 2) to achieve a reliable diagnosis. RAM data contains the transmitter ID, the test vector that uncovered the error and the error itself. Identification of the true error bit vector can be made by inference from the sequence of error / test vector according to existing methods [8].

Diagnosis data processing can be performed on-system, or by the host computer. Data processing with the accumulators and/or RAM depends on the intended diagnosis resolution. The test report can be delivered to the host computer, for further processing. Access to the test results can be carried out by a number of ways, i.e., a Test Bus, constituted by a reduced number of data lines, or by a Test Connector. Additionally, JTAG can be used to access the results.

III. CASE STUDY

The case study used to demonstrate the proposed

<table>
<thead>
<tr>
<th>Transmitter</th>
<th>Domain 1</th>
<th>Domain 2</th>
<th>Domain 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain 1</td>
<td>Error in bit 4</td>
<td>Error in bit 4</td>
<td>Error in bit 4, 10</td>
</tr>
<tr>
<td>Domain 2</td>
<td>Error in bit 4</td>
<td>Error in bit 4</td>
<td>Error in bit 10</td>
</tr>
<tr>
<td>Domain 3</td>
<td>Error in bit 4</td>
<td>No error</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4 – Example of a faulty GALS system

Table 1- Error table for Figure 4
methodology is the DAE (Data Acquisition Electronics) of a multi-board, multi-FPGA (Xilinx™ xc2v4000-4bf957 FPGA) and multi-bus system. A comprehensive description of the DAE system can be found in [12]-[14]. DAE is constituted by 4 DAQ (Data Acquisition) boards and 1 TGR/DCC (Trigger / Data Concentrator) Board, communicating among them through two buses (64-bit GBus, 2x19-bit DBus). DAE Data inputs come from a set of Front-End Electronics (FE) modules (collecting data from sensor arrays). DAQ boards carry out the first data filtering to identify relevant data. Each DAQ board houses 2 DAQ FPGA. The TGR/DCC board is responsible for final identification of relevant data and for its packaging prior to sending it to the host computer for image reconstruction.

Each DAQ FPGA communicates with different clock domains, namely, with the FE (point-to-point), and through the buses with the remaining 7 DAQ FPGA and the TGR/DCC FPGA. DBus and GBus have similar protocols. The synchronization between DAQ FPGA and FE is carried out by a different structure presented in [6], [15] and [16].

In the context of this paper, the interesting part of the DAE system is concerned with the DBus and GBus, the infrastructure of the communication channels among locally synchronous clock domains.

In Figure 5, the DBus protocol for test pattern transmission (as an example) is shown, with its protocol signals, DBUS_REQ, DBUS_GRT and DBUS_DAS, namely, Request, Grant and clock/strobe. Bus controller ‘knows’ that it is in CDC test mode through the data package header. The Bus Controller must be able to recognize a test header even when defects in the communication channel introduce errors in the header (this feature is present in our Bus Controller).

IV. EXPERIMENTAL RESULTS

In order to assess the effectiveness of the proposed methodology, extensive simulations and a comprehensive hardware test and validation procedure have been carried out.

The System under Test (SUT) corresponds to the architecture presented in Figure 1, but with 9 FPGAs and 3 buses (the complete DAE), the GBus of 64 bit wide and the DBus with 2 channels of 19 bit wide each. A Test Bus and a Test Connector are also available. In this case study, the hardware required by the CDC tester for a 19 lines bus, implemented in Virtex 3000 FPGA, is: 129 slices of the total of 14336 (0,9%), in which, contains 203 FFs of a total of 28672 (0,7%) and 240 LUTs of 28672 (0,8%) plus 2 BRAMs in a total of 96 (2,1%). For a 64 bit Bus, it would require 3 to 4 times the number of slices and a total of 4 BRAMs. For our system, all 3 testers occupy 6% of the total slices and 8% of the BRAMs.

A Simulation Procedure

For simulation purposes, the structure in Figure 1 has been inputted in ModelSim [16] simulator. Different transceivers have been modelled with different delays (between 1-8 ns) values. Each CDC tester compares the received data with its own and provides the result of the comparison.

Simulation results have been obtained with the entire system behaving in CDC test scenario. In this scenario, one module at a time broadcasts a test vector and all the remaining modules connected to that communication channel act as receivers.

The bus has been emulated according to its protocol. Faults, namely, stuck at, delay faults and glitches have been injected using the ModelSim fault set or programmed within the test bench. Stuck at faults and glitches can be introduced on the fly. The impact of opens and shorts is modelled by stuck-at faults. Coupling faults describe also shorts.

The following models have been used to emulate the physical failures: a) Short – a weak line is driven by the strong line (in the short); b) Open – weak signal keeper or stuck at; c) Glitches – the (chosen as) aggressor line(s) is (are) monitored. When an ‘edge’ occurs, a glitch is generated into the (chosen as) victim line.

Different aspects of the effectiveness of the methodology are highlighted by the simulation results in Figure 7 through Figure 9. In these figures, /clk is the receiver clock. /vg_addr and /vg_data are the generated test vector address and data, respectively. /vg_start and /vg_term indicates the beginning and the end of test vector generation. /erraddr is the transmitter domain ID and the vector address that generated the error. /errvecreg is the present error, /errvecacc is the error vector accumulator contents. /tempstatus when active indicates that no error has been detected since the last test header. /status indicates the error free domains. /testID identifies the transmitting domain. /Bus_data and /Bus_Das retain the explanation provided for Figure 5. /dataact and /datacury indicate new data at the CDC Interface output. /datain is data at the CDC Interface output. data_inp, data_inn, datap and datan are CDC Interface internal registers.

In Figure 7 the correctness of the test procedure is illustrated. A fault is injected in 4th line. The fault crosses the domains boundaries and is detected by the tester and accumulated. In Figure 7, the effect of a glitch in the Bus clock line is illustrated. The glitch, acting as clock pulse

![Figure 5 - DBus test pattern transmission protocol](image_url)
overwrites the present test vector with the next one. This can be observed from the resulting error. In terms of diagnosis, this type of signature (an early vector) may suggest the occurrence of a glitch in the clock line.

In Figure 9 and Figure 9, the answers of the different clock domains to the test vectors are considered. The signatures of the stuck-at and coupling faults are depicted. These figures highlight the diagnosis that can be made by looking into the historical of the detected errors. In fact, we can notice that the short between two lines is located only in specific areas of a test set (see Figure 9). On the other hand, a stuck-at fault generates errors throughout the entire test set (see Figure 9). This observation allows us to infer what type of fault has originated this faulty behaviour. In the real world, the content of the RAM is used for this diagnosis.

B Hardware Test: Setup and Results

Beyond simulation, the effectiveness of the methodology and the correct behaviour of the CDC tester and the synchronization structures need to be ascertained in a hardware implementation. A comprehensive test of the communication structure, namely the buses, the transceivers, the FPGA’s I/O and the Clock domains’ CDC Interfaces have been carried out. The test procedure follows the one described in Section IIB.

With CDC test, it has been possible to detect (and correct) in the physical prototype, the following faults in the communication channels: a) in the transceivers: bad soldering, dead transceivers and shorts between read/write lines of the transceivers; b) shorts between data lines; shorts between connector pins; and d) glitches in the bus clock line. These results show that the proposed CDC tester modules embedded in the synchronous domains of a GALS system, are able to carry out the test of the entire system communication infrastructure including the CDC Interface. These results also demonstrate that the CDC Interfaces are effective in restoring the synchronism among clock domains.

V. CONCLUSIONS

In this paper, an efficient built-in CDC test and diagnosis methodology for GALS systems and CDC tester architecture have been proposed. One CDC tester is embedded in each locally synchronous domain, using the local CDC Interface for accessing the communication channels. Thus, each

![Figure 7 – Test simulation results: test vector crosses CDC Interface and error is detected](image)

![Figure 7 – Test simulation results: glitch induced in the Bus clock line → CDC interface error](image)
asynchronous test controller is managed in the synchronous domain. T&D functionality is embedded in each FPGA as any other functionality, although being executed in a test scenario. This approach provides a simple, low-cost solution to T&D in GALS systems, dealing with the CDC problem.

The methodology and the CDC tester architecture have been validated in a complex case study, namely, the DAE multi-board (5), multi-bus (3), multi-FPGA (9) (Xilinx™ xc2v4000-4bf957) electronic system. Results of simulation and hardware validation prove that the CDC T&D methodology allows the detection and diagnosis of the most likely hardware static failures, namely, bad soldering, dead components, shorts between read/write lines of the transceivers, shorts between data lines and shorts between connector pins.

It has been also possible to detect dynamic failures, namely, glitches in the bus clock line. It has also been proved that the CDC T&D methodology allows the diagnosis of static and dynamic faults. In the examples, opens, shorts and glitches have been illustrated.

Acknowledgements - We thank the cooperation of our colleagues: F. Piedade, N. Pimenta and P. Neves from INOV, and R. Bugalho, P. Rodrigues and A. Trindade from LIP.

REFERENCES