

NON-DETERMINISTIC DUT BEHAVIOR DURING FUNCTIONAL TESTING OF HIGH SPEED SERIAL BUSES: CHALLENGES AND SOLUTIONS

By: Jonathan Hops, Brian Swing, Brian Phelps, Bruce Sudweeks, John Pane, James Kinslow
Semiconductor Test, Teradyne Inc., Boston, MA

Abstract

The characteristics defining non-determinism for PCI Express busses are explored. The RapidIO® bus is used as a point of comparison. ATE architecture is proposed to significantly reduce the yield and throughput impact of random output.

Introduction

According to the International Technology Roadmap for Semiconductors, the penetration of high-speed serial interfaces into new designs of semiconductor devices is increasing dramatically¹. The throughput on these ports is expected to increase from 3.2 Gbps to 6 Gbps in 2006². Additionally, it is noted that loop-back techniques are unlikely to meet the product quality objectives on these ports. Therefore, one can conclude that functional testing of these interfaces will be a mandatory component of final production testing. However, with functional testing comes the problem of non-deterministic data (NDD) that is present in the device transmit data stream. Unless avoided, NDD can cause production yields to drop significantly. Designing production tests that avoid NDD increases the cost of development time considerably. If NDD cannot be avoided and yield remains high, the only available technique for handling NDD is with post-processing, which can significantly degrade production throughput. Therefore, finding an approach to NDD that is integrated with the HW and SW of the ATE can save time as well as increase yields and throughput.

This paper describes the non-deterministic data problem in functional testing of a Device Under Test (DUT). It also highlights the unique challenges and opportunities presented by High Speed Serial (HSS) busses, with a focus on PCI Express. Although parallel busses are not specifically addressed in this paper, the non-determinism in a RapidIO device implementation is explored to help understand the scope of the NDD problem. Lastly, a specific architecture is explored and proposed for real-time pass/fail analysis of HSS data streams in the ATE environment.

Background

Non-deterministic DUT behaviour occurs when the response of a device to input is different from the predicted response, but remains valid for a known good part. The non-deterministic data (NDD) that the DUT produces complies with the device specification and the protocol of the bus being tested, but the order and content of the information on the bus is different than was originally simulated by the designer.

Common causes of NDD are:

1. Unknown or random initial states
2. Unknown or random lock time for internal PLLs or other dynamic circuits on the DUT.
3. Different time domains on the DUT leading to race conditions at the DUT transmitter.
4. The physical layer encoding or processing steps that inherently add non-determinism.

Parallel bus protocols and serial bus protocols both organize information into packets. A packet is a formatted stream of bits that represents control commands or data payloads that are communicated from the transmitter of one device to the receiver of a second device. During inactive periods, a device is transmitting special alignment or idle packets; the intent of these packets is to maintain bit, signal, and packet alignment, but the receiver requires no other action. Three types of non-determinism influence the resulting data stream from the device. The NDD types are:

Type 1: Cycle slipping at initialization: Where random, or random length inactive periods occur at the beginning of bus transmission. For HSS embedded clock applications, this time period also includes the time needed for the receiver to recover the clock from the data stream.

Type 2: Cycle slipping in the middle of transmission: Where random, or random length inactive periods occur in the middle of a transmission, in between control and data payload packets.

Type 3: Out-of-order of packets: Where control or data payload packets differ in the order they are transmitted from one test run to the next. Often times, out-of-order packets can also cause data in the protocol's data link or physical layer to be different. Therefore, both out-of-order and changing data needs to be dealt with in the same solution.

All three types of NDD can be present at the same time. This presents unique challenges to automatic test equipment. Traditional ATE relies on the deterministic comparison of expected data at expected timing, to actual data and actual timing. Today, there are only two ways to maintain a high yield when non-determinism is present - either avoid it by changing the test program and test conditions, or capture the DUT response data and post process. Both of these techniques can cause time-to-market risks and reduce production throughput. If NDD is ignored, when an unexpected packet sequence is allowed to cause a DUT to fail a functional test, then a reduction in production yield may occur.

Key Challenges

There are 3 key challenges that are shared between parallel and HSS busses, relating to non-determinism:

1. Identifying the start of a packet sequence (NDD Type 1)
2. Finding the right expect data to compare to the actual data from the device (NDD Type 2 and 3)
3. Doing 1 and 2, in real-time, across all lanes of the bus, in a way that is consistent with the protocol.

However, HSS protocols do not have identical challenges presented by parallel protocols. With parallel protocols, it is possible to program a match loop to handle Type 1 NDD. In such a case, the match loops prevent the vector data from advancing until an unexpected vector occurs. With HSS protocols however, this isn't possible since the idle period of a bus has random synchronization packets inserted. HSS busses also differ from parallel busses in the areas of lane interdependence and running disparity. Both of these issues are discussed in detail, below.

Lane Interdependence

A parallel bus spreads all the bits associated with a single byte of a packet across the data pins of a bus. Real-time analysis of the received protocol would have to take into account all data pins before making any determination on whether that byte was appropriate or not. The RapidIO® High Speed Parallel Bus running at 1 Gbps per lane would need to process data at 8 Gbps in order to determine pass, fail, and identify the next vector³.

In this example, the RapidIO® bus has 8 transmit channels. The IDLE packet signifies the inactive period of the bus, which is a 32-bit packet. In the presence of Type 2 NDD, IDLE packets will be inserted into or deleted from the transmit stream at random times, relative to the expected data stream and can be ignored by the receiver. In the ATE environment, it would take 4 vector cycles transmitted by the DUT and received by tester. However, to recognize this as an IDLE packet, all 32 bits in these four cycles would need to be processed. Thus,

determining whether to ignore 4 cycles of data on one lane requires information from all the other lanes associated with the bus. An algorithm to handle Type 2 NDD would have to buffer and process all the data on the bus as a group. Thus, parallel busses have lane interdependence.

Serial busses do not have the same lane interdependency. For example, we can refer to the PCI Express specification⁴. A PCI Express bus with 8 lanes transmitting data at 2.5 GHz must process the received data at an effective rate of 20 Gbps. Unlike a parallel bus, the data transmitted on this bus does not need to be reconstructed on a bit-by-bit basis - it can be reconstructed on a byte-by-byte basis. Due to the 8-Bit/10-Bit (8B/10B) encoding and the serial nature of the bus, each byte is fully contained on a single lane. Thus, identifying protocol elements of the transmission can be accomplished at the slower rate of 2.5 Gbps, if we can ensure that the elements considered essential are lane independent. Note that this is less than the effective rate of data necessary for processing on the parallel bus.

Again, if we focus on PCI Express with Type 2 NDD, then we can see that the inactive period of the bus on each lane can be identified without looking to other lanes of data for information. The Logical Idle state is identified by the transmission of the D0.0 10-bit symbol interspersed with the SKIP ordered set - a 40-bit packet. According to the protocol, all lanes transmit the same information during this period. Assuming that data scrambling is turned off for testing, only 6 10-bit symbols will be transmitted on each lane while the bus is inactive (D0.0, COM, SKP - of each disparity). To identify this inactive period all needed data is contained on the lane being processed. Thus, from the perspective of Type 2 NDD, the PCI Express transmit lanes are independent.

Running Disparity

A challenge unique to the HSS protocols is the running disparity. In an 8B/10B protocol, each 8-bit byte can be translated into 2 10-bit symbols. One of the goals of the DUT transmitter is to keep the number of 1's and 0's balanced in the data stream. The running disparity is the determination of whether more 1's have been transmitted in the last symbol (typically referred to as positive disparity), or more 0's have been transmitted (typically referred to as negative disparity); if 1's and 0's are balanced, the running disparity remains unchanged. The transmitter decides which of the 2 10-bit symbols to send out next based on the running disparity resulting from the prior symbol.

Non-determinism in the area of running disparity can be a side effect of one of the other types of non-determinism. An inserted packet, in the case of Type 3 non-determinism, can change the running disparity of the data as it is transmitted. In such an instance, this would cause

functional failures to be identified by the ATE that should really be ignored. Therefore, any solutions for NDD transmissions for serial busses need to take the running disparity into account.

ATE Users Survey

A small, informal survey was conducted with ATE users to try and determine the frequency and types of non-determinism they expect in their devices, across a number of different bus protocols (PCI-Express, Serial ATA, XAUI, and Serial RIO).

Out of the 7 user groups interviewed:

- 1 out of 7 expects NO NDD on their devices
- 6 out of 7 expect Type 1 NDD on their devices
- 4 out of 7 expect Type 2 NDD or think it may be a risk
- 5 out of 7 are NOT concerned about Type 3 NDD at all.

Of the remaining 2 user groups that are at least a little bit concerned with Type 3 NDD, only one of them is somewhat concerned, the other one “can’t say it won’t happen”.

After reviewing a variety of different bus types available on existing DUTs, the RapidIO® bus was chosen for detailed analysis. The RapidIO® bus is a parallel bus rather than a HSS bus, but the analysis is applicable to HSS since there is currently a Serial RapidIO® specification, which is based on the parallel version. The device was expected to contain Type 1, Type 2 and Type 3 NDD influences as the frequency changed over its allowable range. In fact, in a shmoo of voltage versus frequency, the DUT did exhibit NDD behaviour. The shmoo indicated:

- 189 passing conditions (after post processing the actual DUT response data vs. expect data)
- Out of the passing conditions, there were 25 unique packet sequences – including IDLE Packets.
- According the definitions of NDD presented earlier, the 25 sequences indicated the presence of Type 1 and Type 2 NDD. However, no Type 3 NDD was encountered.

From the data gathered to date, it can be concluded that a solution for HSS non-determination should be concerned with Type 1 and Type 2 NDD; however, there is little expectation of the occurrence of Type 3 NDD in HSS data streams. The solution must also cover a variety of bus protocols in order to handle the variety of emerging DUT types.

The following section summarizes the requirements that must be met by an ATE solution to non-determinism in HSS protocols.

Solution Requirements

The ATE solution to NDD for High Speed Serial Busses must address the following requirements:

1. Recover the strobe clock from the data stream (clock data recovery, part of Type 1 NDD)
2. Adjust to the non-deterministic beginning of a bus transmission relative to the expected data (the rest of Type 1 NDD).
3. Handle the cycle slippage in the middle of transmission due to inactive periods of the bus (Type 2 NDD).
4. Account for running disparity non-determinism.
5. Make pattern burst pass/fail determination in real-time – without time consuming post-processing.
6. Handle multiple HSS protocols.
7. It would be an added advantage, albeit a second priority, to satisfy the additional requirement of handling Type 3 NDD.

Solution for Type 1 NDD on HSS

The proposed solution to Type 1 NDD for HSS busses is described in this section¹. The discussion is focused on 1 lane of data, taking advantage of the lane independence mentioned earlier. The processing steps accomplished by the Type 1 NDD Alignment block are:

1. Receiver extracts the embedded clock signal from the data being transmitted.
2. Receiver achieves bit alignment.
3. Receiver achieves 10-Bit symbol alignment to the transmitted 8B/10B encoded symbols.
4. Receiver finds the beginning of functional data – a Data Link Layer Packet for link configuration or some other functional starting point, like the of a Start of Transaction Layer Packet.

Step 1 and 2 are accomplished by the Clock Data Recovery circuitry in the ATE pin electronics. So, that leaves steps 3 and 4 to describe.

In accomplishing these steps, the receiver must be careful to avoid aliasing to a data stream that is not 10-Bit symbol aligned. Aliasing, in this context, means that while looking for a specific 8B/10B encoded symbol sequence, an identical bit sequence could be found that was not 10-Bit Aligned. Referring Figure 1, a DUT transmits a PCI Express data pattern. The expected symbols are the STP- symbol, followed by the D11.6-symbol. However, the DUT is actually transmitting D25.6-; D11.6-; D17.2- symbols, in that order. If the receiver is not already aligned to where the 10-bit boundary is, then the STP-, D11.6- sequence can be mistakenly found in the transmitted data at a location that is not 10-Bit aligned.

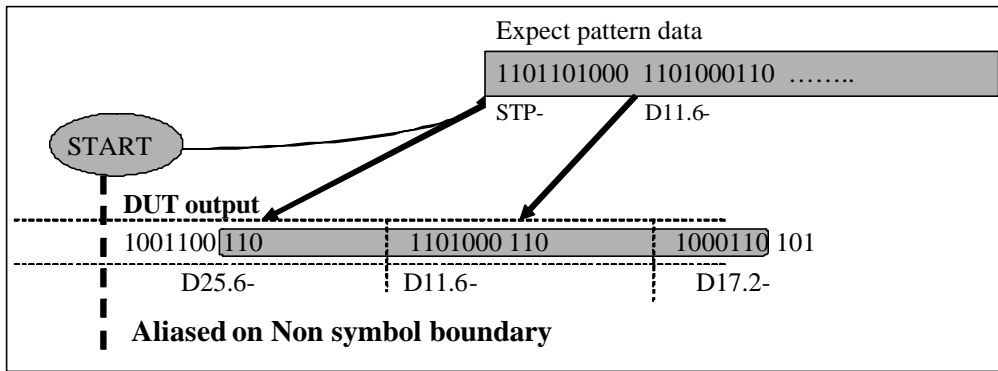


Figure 1. Aliasing in 8B/10B Symbol Sequence

In performing this analysis, it was determined that 2 common symbols could be used for 10-Bit Symbol alignment that:

- a. Do not alias in any combination of 2 other symbols
- b. Are very likely to occur during the initial sequence of symbols

These are the FTS and the COM symbols.

It is therefore feasible to break up Step 3 and Step 4 - first achieve 10-Bit Alignment, then look for the beginning of the real data link layer or transaction layer protocol packets

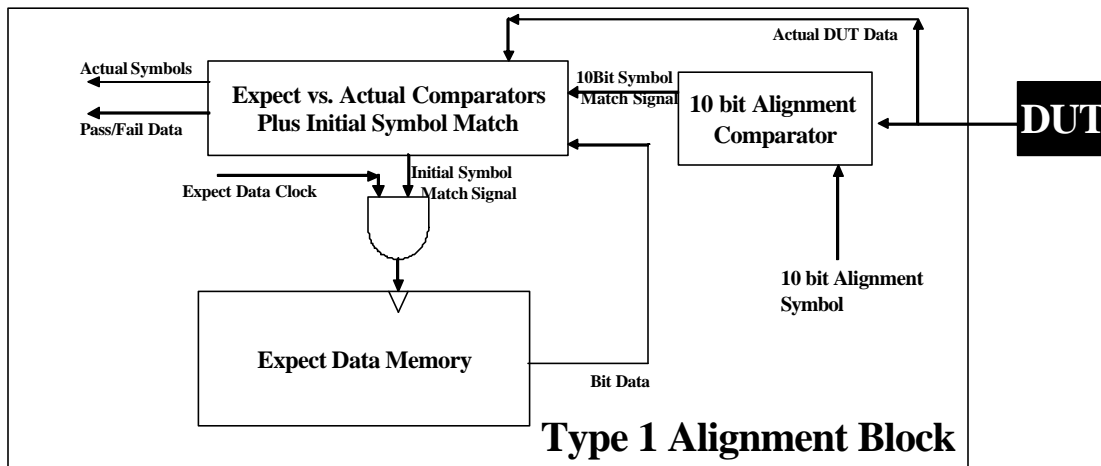


Figure 2. Type 1 Alignment Block.

The above figure (Figure 2.) illustrates how step 3 and 4 can be accomplished for HSS busses in the presence of Type 1 NDD^v. On the right is the actual DUT data, after clock data recovery and bit alignment. This data first goes to the 10-Bit Alignment Comparator to be compared against the 10-Bit Alignment Symbol that had been pre-loaded. If the actual DUT data matches the pre-loaded data, then the 10Bit Symbol

Match Signal is activated and the next stage of comparators are enabled. The purpose of this next block is twofold:

1. Compare the expect symbols vs. the actual symbols
2. Determine if the initial symbol match signal should be asserted.

The initial symbol match signal is asserted when the first set of expected symbols from the expect data memory matches actual DUT data after the 10Bit symbol match has occurred. Since this search is done on a symbol-by-symbol basis, rather than bit-by-bit, no aliasing occurs. The initial symbol match signal also gates whether or not the expect data memory is clocked to the next set of vector data, whose rate is controlled by the expect data clock. Finally, actual symbols and pass/fail data are sent off to the other parts of the receiving subsystem.

It is important to note, that another responsibility of the block labelled “Expect vs. Actual Comparators Plus Initial Symbol Match” is to compare not only the initial expected symbols, but also the opposite running disparity, so that disparity non-determinism can be accounted for.

Solution for Type 2 NDD on HSS

The proposed solution to Type 2 NDD for HSS busses is described in this section¹. The processing steps accomplished by the Type 2 NDD Processing block are:

1. Receiver identifies the symbols associated with the inactive bus periods.
2. Receiver determines whether the symbols should be transmitted to the Signature Generators
3. Receiver translates running disparity (if desired).
4. Receiver calculates a Signature based on the symbols it receives.
5. Determine pass or fail based on the allowable signatures for the burst.

The Signature Generator and the Symbol Map are two components to this portion of the solution.

Signature Generator

A signature corresponds to a calculated checksum based on an acceptable sequence of packet data received on a DUT transmit lane. A signature generator such as a CRC arithmetic register, or linear feedback shift register can be used for this application. Since a signature generator is sensitive both to the order and content of the data it receives, a different signature would normally be produced each time a differently sized inactive bus period is encountered. In order for the signature generator to be useful in the presence of Type 2 NDD, we must have a way to selectively turn on and off the flow of symbols to the signature generator. This is where the symbol map comes in.

Symbol Map

The Symbol Map has 2 functions:

1. To selectively enable or disable 10-Bit Symbols from being forwarded to the Signature Generators
2. To allow translation of 10-Bit symbols from one running disparity to another.

As shown in Figure 3, the Symbol Map has only one input – a 10-Bit Symbol. As output, it has 2 items: a Signature Generator Enable and an output 10-Bit Symbol. Only if the Enable is “yes” will the output 10-Bit Symbol be clocked through the Signature Generation circuitry. Note that the Actual DUT Data represented by symbols D0.0+, K28.0+, and K28.5- are not enabled in the map, and so they are effectively absent from the data that goes to the Signature Generator.

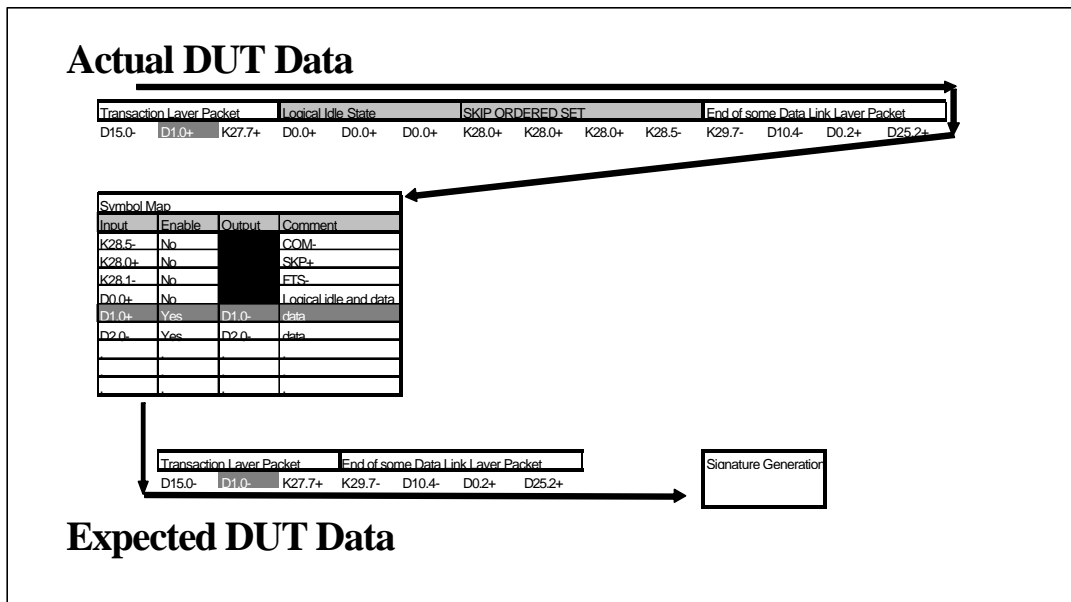


Figure 3. Symbol Map

Also note in figure 3, that the D1.0+ symbol (highlighted), appears in the Actual DUT Data. However, as it flows through the Symbol Map, it is translated to D1.0-. It is the D1.0- Symbol that is processed rather than the D1.0+. In this way, running disparity non-determinism in the Actual DUT Data can be eliminated as an influence on the resulting signature.

The Symbol Map also provides a facility to handle multiple HSS protocols. Since different protocols may have different 10-Bit Symbols representing the inactive period of their busses, these can be entered into the Symbol Map and disabled individually for each bus-and-lane combination.

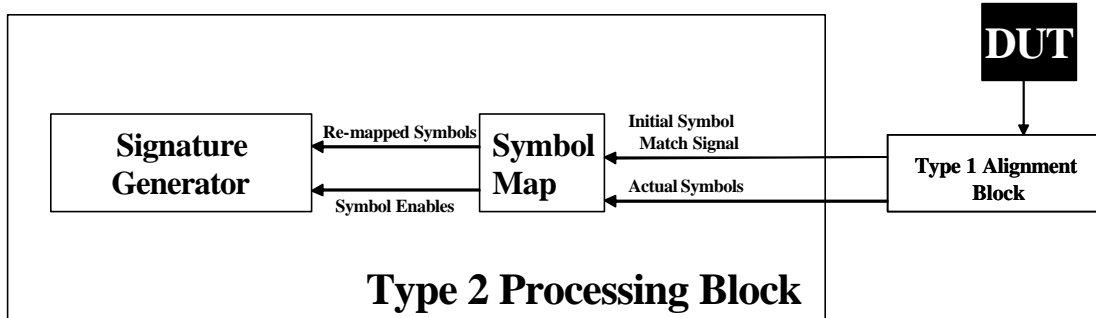


Figure 4. Type 2 Processing Block

The proposed solution for HSS busses in the presence of Type 2 NDD is depicted in Figure 4, above¹. Coming in on the right, The Actual DUT data flows into the Type 1 Alignment Block. [For details, refer to the previous section]. The pertinent outputs of this block for Type 2 NDD are the Initial Symbol Match Signal and the Actual DUT Symbols. The Symbol Map then processes the symbols according to the control settings that were loaded and outputs the two signals described above – the Symbol Enables and the newly mapped Symbols (referred to as “Re-mapped Symbols” in the figure).

Because Type 1 Alignment occurs prior to the start of the Signature Generators, and because the Symbol Map can inhibit the inactive bus period from affecting the signature, it is expected that this architecture will produce only 1 valid signature per lane per pattern burst for the high speed serial busses investigated. The real-time analysis of pass/fail is then reduced to the process of validating whether the single signature appears on the valid signature list for the burst. The initial valid signature per lane per burst can either be calculated by offline simulation or through a post processing routine run on the tester.

Type 3 NDD solutions for HSS

As stated above, Type 3 NDD is characterized by out-of-order data packets, which may also lead to slightly different header data being transmitted in data link layer and transaction layer packets. Although there doesn’t appear to be a substantial customer base for addressing this requirement, we have analysed the solution for Type 2 NDD to see if it can work for Type 3.

If we assume the same architecture as described in the previous section, it is clear that a burst which produces a different order of packets from a prior burst will produce different signatures per lane (per burst) than the ones generated by the prior burst. Therefore, the number of signatures for a given burst will be dependent on the number of unique packet sequences the burst can generate. However, since inserted inactive periods between packets will have no effect on the signature, this number of sequences should be far less than if the inactive periods could not be ignored.

With the current design, the test time increase due to Type 3 NDD in the production environment will be limited to searching a longer list of valid signatures per lane per burst. Although the search could be sped up via an implementation in hardware, no change to the architecture is recommended until there is more demand for fast Type 3 NDD functional testing.

This still leaves the question of how to get a good enough set of valid signatures in the signature list so that production yield is increased. The proposed architecture limits the solution to post processing of actual DUT data. Post processing can be done in the engineering lab, but it will be time consuming. Device engineering labs will need to balance the amount of online tester time needed to generate a list of valid signatures with the projected yield increase.

Summary

Non-deterministic data on High Speed Serial busses can have a detrimental effect to device yield, throughput, and time-to-market. Three types of NDD have been described, accounting for variability in the DUT response

at the beginning and in the middle of a pattern burst. The unique challenges of testing HSS Busses influenced by NDD have been reviewed in the proposed architecture. As devices implement more of these busses at higher speeds, we will be able to have a better understanding of the frequency of the different types of non-determinism

being experienced. It is therefore recommended that a more detailed survey of non-deterministic device behaviour be conducted as the next generation of devices comes to production. Today's information indicates that the solutions presented here will meet the yield and throughput demands of known devices.

Endnotes

¹Similar solutions are described in pending patent applications owned by Teradyne, Inc

References:

- [1] International Roadmap Committee, *International Technology Roadmap for Semiconductors 2003, Executive Summary*, 2003
- [2] International Roadmap Committee, *International Technology Roadmap for Semiconductors 2003, Test and Test Equipment*, 2003
- [3] RapidIO Trade Association, *RapidIO® Interconnect Specification, Rev 1.1, 3/2001*, 2001
- [4] PCI-SIG, *PCI Express Base Specification, Rev 1.0 7/22/2002*, 2002

RapidIO® is a registered trademark of the RapidIO Trade Association.