10 Gigabit Ethernet Consortium

10GBASE-R PCS Test Suite version 0.4

UNH-IOL — 121 Technology Drive, Suite 2 — Durham, NH 03824 — +1-603-862-0090
10 Gigabit Ethernet Consortium — 10geclab@iol.unh.edu — +1-603-862-0205

Vendor X
Company Name
Street Address
City, State, Zip

August 21, 2005
Report Rev. 1.0

Enclosed are the results from the Clause 49 10GBASE-R PCS testing performed on:

Device Under Test (DUT):	10 Gigabit Ethernet Switch
Hardware Version:	N/A
Firmware Version:	N/A
Software Version:	N/A
Miscellaneous:	Tested on port 1

The test suite referenced in this report is available at the UNH-IOL website:

ftp://ftp.iol.unh.edu/pub/10gec/testsuites/Clause_49_PCS_Test_Suite_v0.4.pdf

Issues Observed While Testing
Test 49.3.2: 64_GOOD – The DUT was observed to improperly acquire block_lock upon reception of only 63 consecutive valid sync headers.
Test 49.3.3: 16_BAD – The DUT was observed to require 17 invalid sync headers in a group of 64 blocks to lose block_lock.

For specific details regarding issues please see the corresponding test result.

Testing Completed 08/09/2005

John Q. Tester

John Q. Tester
johnqtester@iol.unh.edu
+1603-862-0205

Review Completed 08/10/2005

John Q. Reviewer

John Q. Reviewer
johnqreviewer@iol.unh.edu
+1-603-862-4342

Digital Signature Information

This document was created using an Adobe digital signature. A digital signature helps to ensure the authenticity of the document, but only in this digital format. For information on how to verify this document's integrity proceed to the following site:

http://www.iol.unh.edu/certifyDoc

If the document status still indicates "Validity of author NOT confirmed", then please contact the UNH-IOL to confirm the document's authenticity. To further validate the certificate integrity, Adobe 6.0 should report the following fingerprint information:

MD5 Fingerprint: DB27 087D 94C8 CB63 7679 50E1 2239 C564
SHA-1 Fingerprint: 5411 C271 9458 ECB2 F401 E0C9 0026 25C3 98D3 E8FE

Result Key

The following table contains possible results and their meanings:

Result	Interpretation
PASS	The Device Under Test (DUT) was observed to exhibit conformant behavior.
PASS with Comments	The DUT was observed to exhibit conformant behavior however an additional explanation of the situation is included, such as due to time limitations only a portion of the testing was performed.
FAIL	The DUT was observed to exhibit non-conformant behavior.
Warning	The DUT was observed to exhibit behavior that is not recommended.
Informative	Results are for informative purposes only and are not judged on a pass or fail basis.
Refer to Comments	From the observations, a valid pass or fail could not be determined. An additional explanation of the situation is included.
Not Applicable	The DUT does not support the technology required to perform these tests.
Not Available	Due to testing station or time limitations, the tests could not be performed.
Borderline	The observed values of the specified parameters are valid at one extreme, and invalid at the other.
Not Tested	Not tested due to the time constraints of the test period.

Test Setup

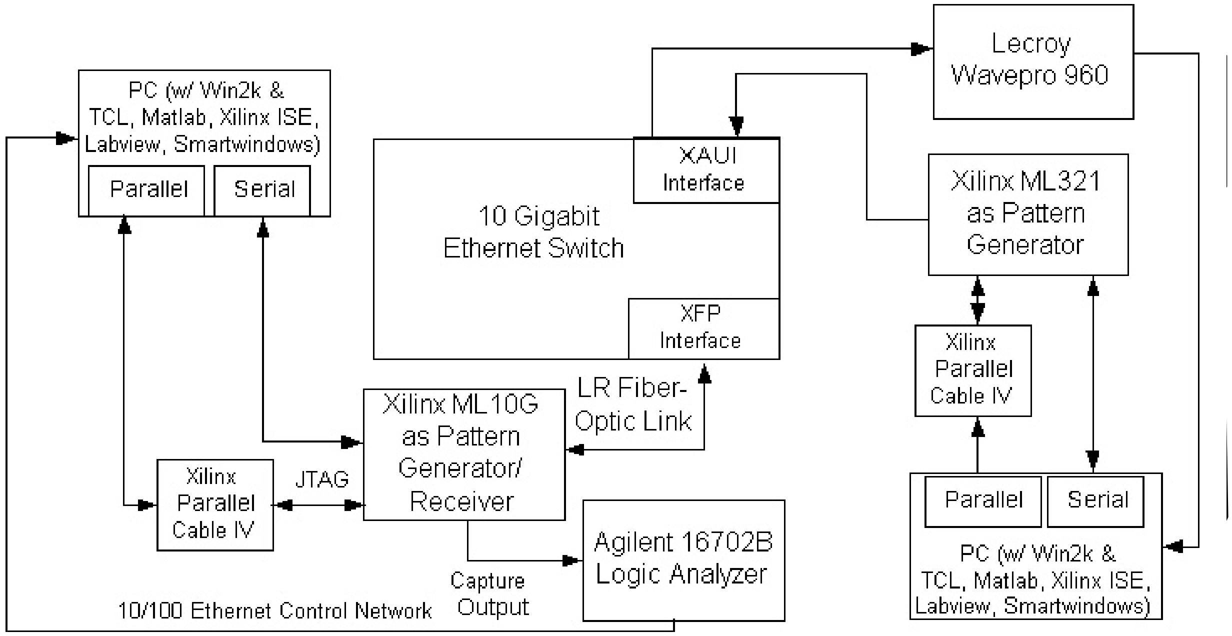
Lecroy Wavepro 960
PC (w/ Win2k & TCL, Matlab, Xilinx ISE, Labview, Smartwindows)
Parallel
Serial
XAUI Interface
10 Gigabit Ethernet Switch
Xilinx ML321 as Pattern Generator
XFP Interface
Xilinx Parallel Cable IV
LR Fiber-Optic Link
Xilinx ML10G as Pattern Generator/ Receiver
Xilinx Parallel Cable IV
JTAG
Parallel
Serial
Agilent 16702B Logic Analyzer
PC (w/ Win2k & TCL, Matlab, Xilinx ISE, Labview, Smartwindows)
Capture Output
10/100 Ethernet Control Network

Figure 1 - Test Configuration

Figure 1 above depicts the test setup employed throughout the testing process. The XFP interface and the XAUI interfaces of the Device Under Test (DUT) were used to provide access to the DUT in all test cases. Control access to the DUT was provided via an Ethernet connection. The test system consists of 2 PCs, an Agilent 16702B, a Wavepro 960 DSO, a Xilinx ML10G, and a Xilinx ML321.

Xilinx ML10G system:

- 10GBASE-R signaling is generated by the “ML10G” Testing Station. This system continuously sends valid 10GBASE-R Idle when not instructed to send a programmable transmit pattern. When the transmit pattern is sent, the Logic Analyzer is set to trigger on non-Idle code-blocks.
- Labview software controls the generation of the test vectors and programming of the ML10G. Labview software controls the downloading and analysis of the signaling captured on the Agilent 16702B.
- The 10GBASE-R signaling from the DUT is captured by the ML10G and then relayed to the Agilent 16702B.

Xilinx ML321 system

- XAUI signaling is generated by the “Xilinx ML321” Testing Station. This system continuously sends valid (though not truly randomized) XAUI Idle when not instructed to send a programmable transmit pattern. When the transmit pattern is sent, a trigger signal is sent to the DSO to capture the response of the DUT.
- Labview software controls the generation of the test vectors and programming of the ML321. Labview and Matlab software control the downloading and analysis of the signaling captured on the Lecroy WavePro 960.
- The 4-channel XAUI signaling from the DUT is captured single-endedly on a Lecroy WavePro 960. The positive signals (Tx+) are driven into the 50ohm terminations of the DSO, while the negative signals (Tx-) are terminated with 50ohm terminations.

The PCs are used for five purposes:

- via the network to control the DSO and download and process the waveforms in Matlab
- via the network to control the Logic Analyzer and download and process the captures signals in Labview
- via an Ethernet connection to control the DUT
- via the parallel ports to download the firmware for the Xilinx ML321 and Xilinx ML10G
- via the Serial ports to control transmissions from the Xilinx ML321 and Xilinx ML10G

Section1: Detailed Table of Results

GROUP 1: Scrambler/Descrambler and Bit Ordering

Test # and Label	Part(s)	Result(s)
Test 49.1.1: Transmit Scrambler	**a**	**PASS**
Comments on Test Procedure		
Purpose: To verify the transmit scrambler polynomial. Procedure: a. The Testing station is set up to capture and analyze a bit stream from the DUT. The DUT should use the defined scrambler polynomial.		
Comments on Test Results		
a. The DUT was observed to properly use the defined scrambler polynomial.		

Test # and Label	Part(s)	Result(s)
Test 49.1.2: Receive Descrambler	**a-b**	**PASS**
Comments on Test Procedure		
Purpose: To verify the receive descrambler polynomial. Procedure: a. The DUT is sent validly scrambled data. Management indications and transmissions from the DUT are observed. The DUT should be able to lock on to and recover validly scrambled data. b. The DUT is sent invalidly scrambled data that is otherwise valid. Management indications and transmissions from the DUT are observed. The DUT should not be able to lock on to and recover invalidly scrambled data.		
Comments on Test Results		
a. The DUT was observed to properly lock on to and recover validly scrambled data. b. The DUT was observed to properly not lock on to and recover invalidly scrambled data.		

Test # and Label	Part(s)	Result(s)
Test 49.1.3: Transmit bit ordering	**a**	**PASS**
Comments on Test Procedure		
Purpose: To verify the transmit bit ordering. Procedure: a. The Testing station is set up to capture and analyze a bit stream from the DUT. The bit ordering of the DUT should match with Figures 49-2 and 49-5.		
Comments on Test Results		
a. The DUT was observed to utilize the proper bit ordering.		

Test # and Label	Part(s)	Result(s)
Test 49.1.4: Receive bit ordering	**a**	**PASS**

Comments on Test Procedure

Purpose: To verify the receive bit ordering.

Procedure:

a. The DUT is sent a bit stream with the proper bit ordering. The DUT should be able to lock on to and recover data with the correct bit ordering.

b. The DUT is sent a bit stream with an improper bit ordering. The DUT should not be able to lock on to and recover data with an incorrect bit ordering.

Comments on Test Results

a. The DUT was observed to properly lock on to and recover data with the correct bit ordering.

b. The DUT was observed to properly not lock on to and recover data with the incorrect bit ordering.

GROUP 2: Coding Rules

Test # and Label	Part(s)	Result(s)
Test 49.2.1: 64B/66B Transmitter Block Encoder	**a**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT can properly transmit and encode valid 66-bit blocks.

Purpose:

a. The DUT is instructed to transmit one of the following blocks:

$D_0D_1D_2D_3/D_4D_5D_6D_7$	$O_0D_1D_2D_3/S_4D_5D_6D_7$	$T_0C_1C_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/T_4C_5C_6C_7$
$C_0C_1C_2C_3/C_4C_5C_6C_7$	$O_0D_1D_2D_3/O_4D_5D_6D_7$	$D_0T_1C_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4T_5C_6C_7$
$C_0C_1C_2C_3/O_4D_5D_6D_7$	$S_0D_1D_2D_3/D_4D_5D_6D_7$	$D_0D_1T_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4D_5T_6C_7$
$C_0C_1C_2C_3/S_4D_5D_6D_7$	$O_0D_1D_2D_3/C_4C_5C_6C_7$	$D_0D_1D_2T_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4D_5D_6T_7$

The DUT should properly transmit and encode all valid 66-bit blocks.

b. The DUT is instructed to transmit one of the following blocks:

Sync	Block type	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
10	0x1E	0x00	0x2D	0x33	0x4B	0x55	0x66	0x78	0x00
10	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E

The DUT should properly transmit and encode all valid control codes.

c. The DUT is instructed to transmit one of the following blocks:

Sync	Block type	D_1	D_2	D_3	O_1	O_4	D_5	D_6	D_7
10	0x55	0x00	0x00	0x01	0x0	0x0	0x00	0x00	0x01
10	0x55	0x00	0x00	0x01	0x0	0xF	0x00	0x00	0x01
10	0x55	0x00	0x00	0x00	0xF	0xF	0x00	0x00	0x0

The DUT should properly transmit and encode all valid O codes.

Comments on Test Results

a. The DUT was observed to properly encode and transmit all valid 66-bit blocks.
b. The DUT was observed to properly encode and transmit all valid control codes.
c. The DUT was observed to properly encode and transmit all valid O codes.

Test # and Label	Part(s)	Result(s)
Test 49.2.2: 64B/66B Transmitter Invalid Block Handling	**a**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT does not transmit invalid blocks or control codes.

Procedure:

a. The DUT is instructed to transmit one of the following blocks:

$C_0C_1C_2C_3/D_4D_5D_6D_7$	$C_0C_1C_2C_3/T_4C_5C_6C_7$	$D_0D_1D_2D_3/S_4D_5D_6D_7$	$D_0D_1D_2D_3/O_4D_5D_6D_7$
$T_0C_1C_2C_3/D_4D_5D_6D_7$	$T_0C_1C_2C_3/S_4D_5D_6D_7$	$T_0C_1C_2C_3/O_4D_5D_6D_7$	$S_0D_1D_2D_3/C_4C_5C_6C_7$
$S_0D_1D_2D_3/T_4C_5C_6C_7$	$S_0D_1D_2D_3/O_4D_5D_6D_7$	$O_0D_1D_2D_3/D_4D_5D_6D_7$	$O_0D_1D_2D_3/T_4C_5C_6C_7$
$T_0C_1C_2C_3/T_4C_5C_6C_7$	$S_0D_1D_2D_3/S_4D_5D_6D_7$		

The DUT should encode all invalid blocks as errors.

Comments on Test Results

a. The DUT was observed to properly encode all invalid blocks as errors.

Test # and Label	Part(s)	Result(s)
Test 49.2.3: 64B/66B Receiver Block Decoding and Control Code Mapping	**a-c**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT can properly receive and decode valid 66-bit blocks.

Procedure:

a. The testing station is instructed to transmit one of the following blocks to the DUT:

$D_0D_1D_2D_3/D_4D_5D_6D_7$	$O_0D_1D_2D_3/S_4D_5D_6D_7$	$T_0C_1C_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/T_4C_5C_6C_7$
$C_0C_1C_2C_3/C_4C_5C_6C_7$	$O_0D_1D_2D_3/O_4D_5D_6D_7$	$D_0T_1C_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4T_5C_6C_7$
$C_0C_1C_2C_3/O_4D_5D_6D_7$	$S_0D_1D_2D_3/D_4D_5D_6D_7$	$D_0D_1T_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4D_5T_6C_7$
$C_0C_1C_2C_3/S_4D_5D_6D_7$	$O_0D_1D_2D_3/C_4C_5C_6C_7$	$D_0D_1D_2T_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4D_5D_6T_7$

The DUT should properly receive and decode all valid 66-bit blocks.

b. The testing station is instructed to transmit one of the following blocks to the DUT:

Sync	Block type	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
10	0x1E	0x00	0x2D	0x33	0x4B	0x55	0x66	0x78	0x00
10	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E

c.

The DUT should properly receive and decode all valid control codes.

d. The testing station is instructed to transmit one of the following blocks to the DUT:

Sync	Block type	D_1	D_2	D_3	O_1	O_4	D_5	D_6	D_7
10	0x55	0x00	0x00	0x01	0x0	0x0	0x00	0x00	0x01
10	0x55	0x00	0x00	0x01	0x0	0xF	0x00	0x00	0x01
10	0x55	0x00	0x00	0x00	0xF	0xF	0x00	0x00	0x0

The DUT should properly receive and decode all valid O codes.

Comments on Test Results

a. The DUT was observed to properly decode and receive all valid 66-bit blocks.
b. The DUT was observed to properly decode and receive all valid control codes.
c. The DUT was observed to properly decode and receive all valid O codes.

Test # and Label	Part(s)	Result(s)
Test 49.2.4: 64B/66B Receiver Invalid Code Handling	**a - d**	**PASS**
	e	**Not Tested**

Comments on Test Procedure

Purpose: To verify that the DUT properly handles the reception of invalid codes and blocks.

Procedure:

a. The testing station is instructed to transmit a frame with one data block containing an invalid sync header. The DUT should replace the received blocks with EBLOCK_R<71:0>, and should not receive the frame.

b. The testing station is instructed to transmit otherwise valid idle blocks with the 0x1E block type field replaced with a reserved block type field. The DUT should replace the received blocks with EBLOCK_R<71:0>.

c. The testing station is instructed to transmit otherwise valid control blocks with the control code fields replaced with a reserved control field. The DUT should replace the received blocks with EBLOCK_R<71:0>.

d. Instruct the testing station to transmit continuous sequence ordered_sets to the DUT, using a 10GBASE-R O code of 0x0, and indicating remote fault. The DUT should replace the received blocks with EBLOCK_R<71:0>, and should not indicate reception of remote fault when receiving O codes that do not have a value of 0x0.

e. Instruct the testing station to transmit a block containing one of the following encodings:

$C_0C_1C_2C_3/D_4D_5D_6D_7$	$C_0C_1C_2C_3/T_4C_5C_6C_7$	$D_0D_1D_2D_3/S_4D_5D_6D_7$	$D_0D_1D_2D_3/O_4D_5D_6D_7$
$T_0C_1C_2C_3/D_4D_5D_6D_7$	$T_0C_1C_2C_3/S_4D_5D_6D_7$	$T_0C_1C_2C_3/O_4D_5D_6D_7$	$S_0D_1D_2D_3/C_4C_5C_6C_7$
$S_0D_1D_2D_3/T_4C_5C_6C_7$	$S_0D_1D_2D_3/O_4D_5D_6D_7$	$O_0D_1D_2D_3/D_4D_5D_6D_7$	$O_0D_1D_2D_3/T_4C_5C_6C_7$
$T_0C_1C_2C_3/T_4C_5C_6C_7$	$S_0D_1D_2D_3/S_4D_5D_6D_7$		

The DUT should replace the received blocks with EBLOCK_R<71:0>.

Comments on Test Results

a. The DUT was observed to properly replace the received blocks with EBLOCKs. If FASTSYNC was set to "fast 66B frame sync" the DUT was observed to transmit LF in response to the invalid sync header.

b. The DUT was observed to properly replace the received blocks with EBLOCKs.

c. The DUT was observed to properly replace the received blocks with EBLOCKs.

d. The DUT was observed to properly replace the received blocks with EBLOCKs and did not indicate the reception of a fault.

e. A method to transmit invalid blocks from the testing station was not available.

Test # and Label	Part(s)	Result(s)
Test 49.2.5: Idle Control Code Insertion/Deletion	**a**	**Not Performed**

Comments on Test Procedure

Purpose: To verify that the DUT properly inserts and delete idle

Idle control characters (/I/) are transmitted when idle control characters are received from the XGMII. Idle characters may be added or deleted by the PCS to adapt between clock rates. /I/ insertion and deletion shall occur in groups of 4. /I/s may be added following idle or ordered sets. They shall not be added while data is being received. When deleting /I/s, the first four characters after a /T/ shall not be deleted.

Comments on Test Results

This test is still under development and was not performed.

Test # and Label	Part(s)	Result(s)
Test 49.2.6: Sequence Ordered_set deletion	**a**	**Not Performed**

Comments on Test Procedure

Purpose: To verify that the DUT properly deletes ordered_sets.

Sequence ordered_sets may be deleted by the PCS to adapt between clock rates. Such deletion shall only occur when two consecutive sequence ordered sets have been received, and only one of the two ordered sets may be deleted.

Comments on Test Results

This test is still under development and was not performed.

GROUP 3: Lock State Machine

Test # and Label	Part(s)	Result(s)
Test 49.3.1: Identification of sync header	**a-b**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies valid and invalid sync headers.

Procedure:

a. The testing station is instructed to transmit a continuous stream of blocks with a sync header of either ‘01’ or ‘10’. The DUT should achieve block_lock while receiving an incoming bit stream with sync headers of ‘01’ or ‘10’.

b. The testing station is instructed to transmit a continuous stream of blocks with a sync header of either ‘00’ or ‘11’. The DUT should not achieve block_lock while receiving an incoming bit stream with sync headers of ‘00’ or ‘11’.

Comments on Test Results

a. The DUT was observed to properly achieve block_lock while receiving a continuous stream of blocks with sync headers of ‘01’ or ‘10’.

b. The DUT was observed to properly not achieve block_lock while receiving a continuous stream of blocks with sync headers of ‘00’ or ‘11’.

Test # and Label	Part(s)	Result(s)
Test 49.3.2: 64_GOOD	**a**	**FAIL**

Comments on Test Procedure

Purpose: To verify that the DUT needs to see 64 consecutive valid sync headers before achieving block_lock.

Procedure:

a. The testing station is instructed to transmit *n* blocks with a valid sync header followed by 1 block with an invalid sync header. The value of *n* is increased until the DUT is observed to achieve block_lock. The DUT should achieve block after receiving a continuous looping pattern of 64 or more valid sync headers, followed by a single invalid sync header.

Comments on Test Results

a. The DUT was observed to improperly achieve block_lock after receiving a continuous looping pattern of 63 blocks with a valid sync header followed by one block with an invalid sync header. The DUT properly did not achieve block_lock after receiving a continuous looping pattern of 62 blocks with a valid sync header followed by one block with an invalid sync header.

Test # and Label	Part(s)	Result(s)
Test 49.3.3: 16_BAD	**a**	**FAIL**

Comments on Test Procedure

Purpose: To verify that the DUT needs to see 16 out of 64 invalid sync headers before slipping.

Procedure:

a. The testing station is instructed to transmit a set of 64 blocks containing *n* blocks with invalid sync headers. The value of *n* is increased until the DUT is observed to lose block_lock. The DUT should lose block_lock after receiving 16 invalid sync headers out of a group of 64 sync headers.

Comments on Test Results

a. The DUT was observed to improperly maintain block_lock upon reception of 16 invalid sync headers in a group of 64 blocks. The DUT was observed to lose block_lock upon reception of 17 invalid sync headers in a group of 64 blocks.

GROUP 4: BER Monitor State Machine

Test # and Label	Part(s)	Result(s)
Test 49.4.1: Value of 125us_timer	**a**	**Not Performed**

Comments on Test Procedure

Purpose: To verify that the value of the 125us_timer is between 93.75µs and 126.25µs.

a. The 125us_timer described in Figure 49-13 must take on a value of 125µs +1%, – 25%. The timer is used to set the hi_ber flag if more than 16 invalid sync headers are found within the window of 125µs. The DUT is instructed to send 15 invalid sync headers to the DUT. The transmit station is then instructed to send 1 invalid sync header a specific amount of time later. Indications and transmissions from the DUT are observed. The DUT should set the high_ber flag if it receives 16 invalid sync headers within 93.75µs to 126.25µs

Comments on Test Results

This test is still under development and was not performed.

GROUP 5: Transmit State Machine

Test # and Label	Part(s)	Result(s)
Test 49.5.1: Identification of T_TYPE(C)	**a-c**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies the reception of T_TYPE(C) vectors.

Procedure:

a. The DUT is provided with the following XGMII bit streams to transmit:

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
0x07	0x07	0x07	0x07	0x07	0x07	0x07	0x07
0x1C	0x3C	0x7C	0xBC	0xDC	0xF7	0x07	0x07

The DUT should properly encode all 72-bit vectors.

b. The DUT is provided with the following XGMII bit streams to transmit:

C_0	C_1	C_2	C_3	O_4	D_5	D_6	D_7
0x07	0x07	0x07	0x07	0x9C	0x00	0x00	0x01
0x07	0x07	0x07	0x07	0x9C	0x00	0x00	0x02
0x07	0x07	0x07	0x07	0x9C	0x12	0x34	0x56
0x1C	0x3C	0x7C	0xBC	0x9C	0x00	0x00	0x01
0x07	0x07	0x07	0x07	0x5C	0x00	0x00	0x01

O_0	D_1	D_2	D_3	C_4	C_5	C_6	C_7
0x9C	0x00	0x00	0x01	0x07	0x07	0x07	0x07
0x9C	0x00	0x00	0x02	0x07	0x07	0x07	0x07
0x9C	0x12	0x34	0x56	0x07	0x07	0x07	0x07
0x9C	0x00	0x00	0x01	0x1C	0x3C	0x7C	0xBC
0x5C	0x00	0x00	0x01	0x07	0x07	0x07	0x07

The DUT should properly encode all 72-bit vectors.

c. The DUT is provided with the following XGMII bit streams to transmit:

O_0	D_1	D_2	D_3	O_4	D_5	D_6	D_7
0x9C	0x00	0x00	0x01	0x9C	0x00	0x00	0x01
0x9C	0x00	0x00	0x02	0x9C	0x00	0x00	0x02
0x9C	0x00	0x00	0x01	0x9C	0x00	0x00	0x02
0x9C	0x00	0x00	0x02	0x9C	0x00	0x00	0x01
0x9C	0x00	0x00	0x01	0x9C	0x12	0x34	0x56
0x9C	0x12	0x34	0x56	0x9C	0x00	0x00	0x01
0x9C	0x00	0x00	0x02	0x9C	0x12	0x34	0x56
0x9C	0x12	0x34	0x56	0x9C	0x00	0x00	0x02
0x5C	0x00	0x00	0x00	0x9C	0x00	0x00	0x01
0x9C	0x00	0x00	0x01	0x5C	0x00	0x00	0x00
0x5C	0x00	0x00	0x00	0x9C	0x00	0x00	0x02
0x9C	0x00	0x00	0x02	0x5C	0x00	0x00	0x00
0x9C	0x12	0x34	0x56	0x5C	0x00	0x00	0x00
0x5C	0x00	0x00	0x00	0x9C	0x12	0x34	0x56
0x5C	0x00	0x00	0x00	0x5C	0x00	0x00	0x00

The DUT should properly encode all 72-bit vectors.

<continued on next page>

Test # and Label	**Part(s)**	**Result(s)**
Test 49.5.1: Identification of T_TYPE(C)	**a-c**	**PASS**

Comments on Test Results

a. The DUT was observed to properly encode all 72-bit vectors.
b. The DUT was observed to properly encode all 72-bit vectors.
c. The DUT was observed to properly encode all 72-bit vectors.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.5.2: Identification of T_TYPE(S)	**a**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies the reception of T_TYPE(S) vectors.

Procedure:

a. The DUT is provided with the following XGMII bit streams to transmit:

S_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0xFB	0x55	0x55	0x55	0x55	0x55	0x55	0xD5

The DUT should properly encode all 72-bit vectors.

b. The DUT is provided with the following XGMII bit streams to transmit:

C_0	C_1	C_2	C_3	S_4	D_5	D_6	D_7
0x07	0x07	0x07	0x07	0xFB	0x55	0x55	0x55
0x1C	0x3C	0x7C	0xBC	0xFB	0x55	0x55	0x55

O_0	D_1	D_2	D_3	S_4	D_5	D_6	D_7
0x9C	0x00	0x00	0x01	0xFB	0x55	0x55	0x55
0x9C	0x00	0x00	0x02	0xFB	0x55	0x55	0x55
0x9C	0x12	0x34	0x56	0xFB	0x55	0x55	0x55
0x5C	0x00	0x00	0x00	0xFB	0x55	0x55	0x55

The DUT should properly encode all 72-bit vectors.

Comments on Test Results

a. The DUT was observed to properly encode all 72-bit vectors.
b. The DUT was observed to properly encode all 72-bit vectors.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.5.3: Identification of T_TYPE(T)	**a**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies the reception of T_TYPE(T) vectors.

Procedure:

1. The DUT is instructed to transmit the first of the following 72-bit vectors to the testing station, and using 0x07 for all control characters:

$T_0C_1C_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/T_4C_5C_6C_7$
$D_0T_1C_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4T_5C_6C_7$
$D_0D_1T_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4D_5T_6C_7$
$D_0D_1D_2T_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4D_5D_6T_7$

The DUT should properly encode all 72-bit vectors.

Comments on Test Results

a. The DUT was observed to properly encode all 72-bit vectors.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.5.4: Identification of T_TYPE(D)	**a**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies the reception of T_TYPE(D) vectors.

Procedure:

a. The DUT is instructed to transmit all valid 72-bit data blocks. The DUT should properly encode all 72-bit vectors.

Comments on Test Results

a. The DUT was observed to properly encode all 72-bit vectors.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.5.5: Identification of T_TYPE(E)	**a**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies the reception of T_TYPE(E) vectors.

Procedure:

a. The DUT is provided with the following XGMII bit streams to transmit:

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
0x07	0x07	0x07	0x07	0x07	0x07	0x07	0xFE
0x07	0x07	0x07	0x07	0x07	0x07	0xFE	0x07
0x07	0x07	0x07	0x07	0x07	0xFE	0x07	0x07
0x07	0x07	0x07	0x07	0xFE	0x07	0x07	0x07
0x07	0x07	0x07	0xFE	0x07	0x07	0x07	0x07
0x07	0x07	0xFE	0x07	0x07	0x07	0x07	0x07
0x07	0xFE	0x07	0x07	0x07	0x07	0x07	0x07
0xFE	0x07	0x07	0x07	0x07	0x07	0x07	0x07

The DUT should properly encode all 72-bit vectors as T_BLOCK_TYPE(E).

Comments on Test Results

a. The DUT was observed to properly encode all 72-bit vectors as T_BLOCK_TYPE(E).

Test # and Label	**Part(s)**	**Result(s)**
Test 49.5.6: TX_INIT state	**a**	**Not Tested**

Comments on Test Procedure

Purpose: To verify that the DUT properly behaves while in the TX_INIT state.

Procedure:

a. The DUT is instructed to reset. When being reset, the DUT should continuously transmit local fault blocks.

Comments on Test Results

a. This test was not performed during the testing period.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.5.7: TX_C state	**a-c**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly behaves while in the TX_C state.

Procedure:

a. The DUT is instructed to transmit continuous control blocks such that it enters the TX_C state. The DUT should properly encode all 72-bit vectors and remain in the TX_C state while transmitting C blocks.
b. The DUT is instructed to transmit an S control character while in the TX_C state. The DUT should properly encode all 72-bit vectors and transition to the TX_D state while transmitting an S block.
c. The DUT is instructed to transmit an S control character while in the TX_C state. This is repeated transmitting both D and T blocks. The DUT should properly encode all 72-bit vectors and transition to the TX_E state when transmitting E, D, or T blocks.

Comments on Test Results

a. The DUT was observed to properly remain in the TX_C state.
b. The DUT was observed to properly transition to the TX_D state.
c. The DUT was observed to properly transition to the TX_E state.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.5.8: TX_D state	**a-c**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly behaves while in the TX_D state.

Procedure:

a. The DUT is instructed to transmit a valid data block while in the TX_D state. The DUT should properly encode all 72-bit vectors and remain in the TX_D state.
b. The DUT is instructed to transmit a valid T block while in the TX_D state. The DUT should properly encode all 72-bit vectors and transition to the TX_T state.
c. The DUT is instructed to transmit a valid E block while in the TX_D state. This is repeated using valid S and C blocks. The DUT should properly encode all 72-bit vectors and transition to the TX_E state.

Comments on Test Results

a. The DUT was observed to properly remain in the TX_D state.
b. The DUT was observed to properly transition to the TX_T state.
c. The DUT was observed to properly transition to the TX_E state.

Test # and Label	Part(s)	Result(s)
Test 49.5.9: TX_T state	**a-c**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly behaves while in the TX_T state.

Procedure:

a. The DUT is instructed to transmit a valid C block while in the TX_T state. The DUT should properly encode all 72-bit vectors, and transition to the TX_C state.
b. The DUT is instructed to transmit a valid S block while in the TX_T state. The DUT should properly encode all 72-bit vectors, and transition to the TX_D state.
a. The DUT is instructed to transmit a valid E block while in the TX_T state. This is repeated using valid D and T blocks. The DUT should properly encode all 72-bit vectors, and transition to the TX_E state.

Comments on Test Results

a. The DUT was observed to properly transition to the TX_C state.
b. The DUT was observed to properly transition to the TX_D state.
c. The DUT was observed to properly transition to the TX_E state.

Test # and Label	Part(s)	Result(s)
Test 49.5.10: TX_E state	**a-d**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly behaves while in the TX_E state.

Procedure:

a. The DUT is instructed to transmit a valid D block while in the TX_E state. The DUT should properly encode all 72-bit vectors and transition to the TX_D state.
b. The DUT is instructed to transmit a valid C block while in the TX_E state. The DUT should properly encode all 72-bit vectors and transition to the TX_C state.
c. The DUT is instructed to transmit a valid T block while in the TX_E state The DUT should properly encode all 72-bit vectors and transition to the TX_T state.
d. The DUT is instructed to transmit a valid D block while in the TX_E state The DUT should properly encode all 72-bit vectors and remain in the TX_E state.

Comments on Test Results

a. The DUT was observed to properly transition to the TX_D state.
b. The DUT was observed to properly transition to the TX_C state.
c. The DUT was observed to properly transition to the TX_T state.
d. The DUT was observed to properly remain in the TX_E state.

GROUP 6: Receive State Machine

Test # and Label	Part(s)	Result(s)
Test 49.6.1: Identification of R_TYPE(C)	**a-c**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies the reception of R_TYPE(C) vectors.

Procedure:

a. Instruct the testing station to transmit a block containing one of the following encodings:

Sync	Block type	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
10	0x1E	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
10	0x1E	0x2D	0x33	0x4B	0x55	0x66	0x78	0x00	0x00

The DUT should properly decode all 66-bit vectors.

b. Instruct the testing station to transmit a block containing one of the following encodings:

Sync	Block type	C_0	C_1	C_2	C_3	O_4	D_5	D_6	D_7
10	0x2D	0x00	0x00	0x00	0x00	0x0	0x00	0x00	0x01
10	0x2D	0x00	0x00	0x00	0x00	0x0	0x00	0x00	0x02
10	0x2D	0x00	0x00	0x00	0x00	0x0	0x00	0x00	0x00
10	0x2D	0x00	0x00	0x00	0x00	0xF	0x00	0x00	0x00
10	0x2D	0x2D	0x33	0x4B	0x55	0x0	0x00	0x00	0x01

Sync	Block type	D_1	D_2	D_3	O_0	C_4	C_5	C_6	C_7
10	0x4B	0x00	0x00	0x01	0x0	0x00	0x00	0x00	0x00
10	0x4B	0x00	0x00	0x02	0x0	0x00	0x00	0x00	0x00
10	0x4B	0x12	0x34	0x56	0x0	0x00	0x00	0x00	0x00
10	0x4B	0x00	0x00	0x00	0xF	0x00	0x00	0x00	0x00
10	0x4B	0x00	0x00	0x01	0x0	0x2D	0x33	0x4B	0x55

The DUT should properly decode all 66-bit vectors.

c. Instruct the testing station to transmit a block containing one of the following encodings:

Sync	Block type	D_1	D_2	D_3	O_0	O_4	D_5	D_6	D_7
10	0x55	0x00	0x00	0x01	0x0	0x0	0x00	0x00	0x01
10	0x55	0x00	0x00	0x02	0x0	0x0	0x00	0x00	0x02
10	0x55	0x00	0x00	0x01	0x0	0x0	0x00	0x00	0x02
10	0x55	0x00	0x00	0x02	0x0	0x0	0x00	0x00	0x01
10	0x55	0x00	0x00	0x01	0x0	0x0	0x12	0x34	0x56
10	0x55	0x12	0x34	0x56	0x0	0x0	0x00	0x00	0x01
10	0x55	0x12	0x34	0x56	0x0	0x0	0x00	0x00	0x02
10	0x55	0x00	0x00	0x02	0x0	0x0	0x12	0x34	0x56
10	0x55	0x12	0x34	0x56	0xF	0x0	0x00	0x00	0x01
10	0x55	0x12	0x34	0x56	0xF	0x0	0x00	0x00	0x02
10	0x55	0x00	0x00	0x01	0x0	0xF	0x12	0x34	0x56
10	0x55	0x00	0x00	0x02	0x0	0xF	0x12	0x34	0x56
10	0x55	0x12	0x34	0x56	0xF	0xF	0x78	0x9A	0xBC
10	0x55	0x12	0x34	0x56	0x0	0x0	0x78	0x9A	0xBC

The DUT should properly decode all 66-bit vectors.

Comments on Test Results

a. The DUT was observed to properly decode all 66-bit vectors.

b. The DUT was observed to properly decode all 66-bit vectors.

c. The DUT was observed to properly decode all 66-bit vectors.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.6.2: Identification of R_TYPE(S)	**a-c**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies the reception of R_TYPE(S) vectors.

Procedure:

a. Instruct the testing station to transmit a block containing one of the following encodings:

Sync	Block type	C_0	C_1	C_2	C_3		D_5	D_6	D_7
10	0x33	0x00	0x00	0x00	0x00	0x0	0x55	0x55	0x55
10	0x33	0x2D	0x33	0x4B	0x55	0x0	0x55	0x55	0x55

The DUT should properly decode all 66-bit vectors and receive the frame.

b. Instruct the testing station to transmit a block containing one of the following encodings:

Sync	Block type	D_1	D_2	D_3	O_0		D_5	D_6	D_7
10	0x66	0x00	0x00	0x01	0x0	0x0	0x55	0x55	0x55
10	0x66	0x00	0x00	0x02	0x0	0x0	0x55	0x55	0x55
10	0x66	0x12	0x34	0x56	0x0	0x0	0x55	0x55	0x55
10	0x66	0x00	0x00	0x00	0xF	0x0	0x55	0x55	0x55

The DUT should properly decode all 66-bit vectors and receive the frame.

c. Instruct the testing station to transmit a block containing one of the following encodings:

Sync	Block type	D_1	D_2	D_3	D_4	D_5	D_6	D_7
10	0x78	0x55	0x55	0x55	0x55	0x55	0x55	0xD5

The DUT should properly decode all 66-bit vectors and receive the frame.

Comments on Test Results

a. The DUT was observed to properly decode all 66-bit vectors and receive the frame.
b. The DUT was observed to properly decode all 66-bit vectors and receive the frame.
c. The DUT was observed to properly decode all 66-bit vectors and receive the frame.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.6.3: Identification of R_TYPE(T)	**a**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies the reception of R_TYPE(T) vectors.

Procedure:

a. Instruct the testing station to transmit the following 66-bit vectors to the DUT, containing the appropriate block field types, and using 0x00 for all control characters:

$T_0C_1C_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/T_4C_5C_6C_7$
$D_0T_1C_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4T_5C_6C_7$
$D_0D_1T_2C_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4D_5T_6C_7$
$D_0D_1D_2T_3/C_4C_5C_6C_7$	$D_0D_1D_2D_3/D_4D_5D_6T_7$

The DUT should properly decode all 66-bit vectors and receive the frame.

Comments on Test Results

a. The DUT was observed to properly decode all 66-bit vectors and receive the frame.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.6.4: Identification of R_TYPE(D)	**a**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies the reception of R_TYPE(D) vectors.

Procedure:

a. The testing station is instructed to transmit data blocks to the DUT such that the DUT is placed in the RX_D state. All valid data blocks should be sent to the DUT. The DUT should properly decode all 66-bit vectors and receive the frames.

Comments on Test Results

a. The DUT was observed to properly decode all 66-bit vectors and receive the frames

Test # and Label	**Part(s)**	**Result(s)**
Test 49.6.5: Identification of R_TYPE(E)	**a**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly identifies the reception of R_TYPE(E) vectors.

Procedure:

a. Instruct the testing station to transmit the first of the following 66-bit vectors to the DUT:

Sync	Block type	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
10	0x1E	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x1E
10	0x1E	0x00	0x00	0x00	0x00	0x00	0x00	0x1E	0x00
10	0x1E	0x00	0x00	0x00	0x00	0x00	0x1E	0x00	0x00
10	0x1E	0x00	0x00	0x00	0x00	0x1E	0x00	0x00	0x00
10	0x1E	0x00	0x00	0x00	0x1E	0x00	0x00	0x00	0x00
10	0x1E	0x00	0x00	0x1E	0x00	0x00	0x00	0x00	0x00
10	0x1E	0x00	0x1E	0x00	0x00	0x00	0x00	0x00	0x00
10	0x1E	0x1E	0x00	0x00	0x00	0x00	0x00	0x00	0x00

The DUT should properly decode all 66-bit vectors as R_BLOCK_TYPE(E).

Comments on Test Results

a. The DUT was observed to properly decode all 66-bit vectors as R_BLOCK_TYPE(E).

Test # and Label	Part(s)	Result(s)
Test 49.6.6: RX_INIT state	a,b,d	PASS
	c	Not Tested

Comments on Test Procedure

Purpose: To verify that the DUT passes local fault across the XGMII when in the RX_INIT state.

When the DUT has been reset, is in a test mode, is receiving frequent errors, or does not have block lock, it will remain within the RX_INIT state. While in this state, the DUT will be sending local fault blocks up across the XGMII to signify that the receiver is not operational.

a. When the DUT does not have block_lock, it must pass continuous local fault across the XGMII.
b. When the DUT has hi_ber, it must pass continuous local fault across the XGMII.
c. When the reset signal is set, the DUT must pass continuous local fault across the XGMII.
d. When in the receive test mode, the DUT must pass continuous local fault across the XGMII.

Comments on Test Results

a. When the DUT does not have block_lock, it was observed to properly pass local_fault across the XGMII.
b. When the DUT has high_ber, it was observed to properly pass local_fault across the XGMII.
c. This test was not performed during the testing period.
d. When in the receive test mode, the DUT was observed to properly pass local_fault across the XGMII. The DUT was put into the receive test mode by setting bit 3.42.5.

Test # and Label	Part(s)	Result(s)
Test 49.6.7: RX_C state	a-c	PASS

Comments on Test Procedure

Purpose: To verify that the DUT properly behaves while in the RX_C state.

When the receiver of the DUT has achieved synchronization, and when valid C blocks are being observed at the 66-bit level, the DUT will enter the RX_C state in Figure 49-15. The DUT will remain in this state as long as valid idle or sequence ordered_sets are being received. Upon the reception of an S block, the DUT will transition to the RX_D state. Reception of E, D, or T blocks will force the DUT to the RX_E state.

a. The DUT should properly decode all 66-bit vectors and remain in the RX_C state while receiving C blocks.
b. The DUT should properly decode all 66-bit vectors and transition to the RX_D state while receiving an S block.
c. The DUT should properly decode all 66-bit vectors and transition to the RX_E state when receiving E, D, or T blocks.

Comments on Test Results

a. The DUT was observed to properly decode all 66-bit vectors and remain in the RX_C state while receiving C blocks.
b. The DUT was observed to properly decode all 66-bit vectors and transition to the RX_D state while receiving an S block.
c. The DUT was observed to properly decode all 66-bit vectors and transition to the RX_E state while receiving E, D, or T blocks.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.6.8: RX_D state	**a-d**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly behaves while in the RX_D state.

When the receiver of the DUT has achieved synchronization, and when valid C blocks are being observed at the 66-bit level, the DUT will enter the RX_D state in Figure 49-15 upon the reception of an S block. The DUT will remain in the RX_D state as long as valid D blocks are being received. The DUT will transition to the RX_T state upon reception of a T block, provided that the block following the T is either an S or C block. For all other combinations of received blocks, the DUT will transition to the RX_E state.

a. The DUT should properly decode all 66-bit vectors and remain in the RX_D state.
b. The DUT should properly decode all 66-bit vectors and transition to the RX_T state.
c. The DUT should properly decode all 66-bit vectors and transition to the RX_E state.
d. The DUT should properly decode all 66-bit vectors and transition to the RX_E state.

Comments on Test Results

a. The DUT was observed to properly decode all 66-bit vectors and remain in the RX_D state.
b. The DUT was observed to properly decode all 66-bit vectors and transition to the RX_T state.
c. The DUT was observed to properly decode all 66-bit vectors and transition to the RX_E state.
d. The DUT was observed to properly decode all 66-bit vectors and transition to the RX_E state.

Test # and Label	**Part(s)**	**Result(s)**
Test 49.6.9: RX_T state	**a-b**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly behaves while in the RX_T state.

When the receiver of the DUT has achieved synchronization, and when valid D blocks are being observed at the 66-bit level, the DUT will enter the RX_T state in Figure 49-15 upon the reception of an S block. The DUT will remain in the RX_D state as long as valid D blocks are being received. The DUT will transition to the RX_T state upon reception of a T block, provided that the block following the T is either an S or C block. Once the RX_T state is entered, the DUT will transition to either the RX_C or RX_D state, depending on the next code.

a. The DUT should properly decode all 66-bit vectors, and transition to the RX_C state.
b. The DUT should properly decode all 66-bit vectors, and transition to the RX_D state.

Comments on Test Results

a. The DUT was observed to properly decode all 66-bit vectors and transition to the RX_C state.
b. The DUT was observed to properly decode all 66-bit vectors and transition to the RX_D state.

Test # and Label	Part(s)	Result(s)
Test 49.6.10: RX_E state	**a-e**	**PASS**

Comments on Test Procedure

Purpose: To verify that the DUT properly behaves while in the RX_E state.

When the receiver of the DUT encounters some sort of error, it will transition, from any other state, to the RX_E state. While in this state, regardless of the 66-bit words coming into the receiver, error blocks will be decoded and passed up to the XGMII. If errors are received in the middle of a frame, the DUT may return to the RX_D state when it receives a valid D block, or transition to the RX_T state at the end of the frame. If valid C blocks are received, the DUT will transition to the RX_C state. The DUT will remain in the RX_E state if errors continue to be received or if an S block is encountered.

a. The DUT should properly decode all 66-bit vectors and transition to the RX_D state.
b. The DUT should properly decode all 66-bit vectors and transition to the RX_C state.
c. The DUT should properly decode all 66-bit vectors and transition to the RX_T state.
d. The DUT should properly decode all 66-bit vectors and remain in the RX_E state.
e. The DUT should properly decode all 66-bit vectors and remain in the RX_E state.

Comments on Test Results

a. The DUT was observed to properly decode all 66-bit vectors and transition to the RX_D state.
b. The DUT was observed to properly decode all 66-bit vectors and transition to the RX_C state.
c. The DUT was observed to properly decode all 66-bit vectors and transition to the RX_T state.
d. The DUT was observed to properly decode all 66-bit vectors and remain in the RX_E state.
e. The DUT was observed to properly decode all 66-bit vectors and remain in the RX_E state.

GROUP 7: Test Pattern Verification

Test # and Label	Part(s)	Result(s)
Test 49.7.1: Pseudo-random test pattern transmission	**a-b**	**Not Available**

Comments on Test Procedure

Purpose: To verify that the DUT generates the correct pseudo-random test patterns.

Clause 49 defines a pseudo-random test pattern generator and checker to be implemented in the PCS. This functionality allows the tester to not only verify BER over a channel, but also to force the DUT into transmitting the appropriate signals used for testing. When the PCS offers a direct connection to the PMA, the implementation of the generator and checker is mandatory. The jitter test pattern is generated by the scrambler, using specific seeds that are regularly loaded through the MDIO registers, and using one of two possible data patterns to be scrambled. Every 128 blocks, the scrambler is loaded with one of two seeds, or their inverses, in the following pattern: Seed A, Seed A Invert, or Seed B, Seed B Invert. The values of the seeds are defined in Clause 52, and shown in Table 52-20. The data patterns fed through the scrambler are either all zeros, or the encoding of two Local Fault ordered_sets, and the data pattern will be inverted when the inverted seeds are being used. Register 3.42 controls the different test patterns that can be transmitted and checked.

a. The DUT should properly generate Pseudo-Random Test Pattern 1.
b. The DUT should properly generate Pseudo-Random Test Pattern 2.

Comments on Test Results

a. This test was not completed during the testing period due to test station limitations.
b. This test was not completed during the testing period due to test station limitations.

Test # and Label	Part(s)	Result(s)
Test 49.7.2: PRBS31 test pattern transmission	**a**	**Not Available**

Comments on Test Procedure

Purpose: To verify that the DUT generates the correct PRBS31 test pattern.

Clause 49 defines an optional PRBS31 test pattern mode that can be implemented within the PCS for both transmission and checking of the pattern. When selected, the generator sends 16 bits of the test pattern at a time down to the PMA, bypassing the scrambler. The pattern generator implements an inverted version of the bit stream generated with the following polynomial: $G(x) = 1 + x^{28} + x^{31}$. Register 3.42 controls the different test patterns that can be transmitted and checked.

a. The DUT should properly generate the PRBS31 test pattern.

Comments on Test Results

a. This test was not completed during the testing period due to test station limitations.

Test # and Label	Part(s)	Result(s)
Test 49.7.3: Square wave test pattern transmission	**a**	**Not Available**

Comments on Test Procedure

Purpose: To verify that the DUT generates the correct square wave test pattern.

Certain PMD tests that need to be performed require that the PCS generate a square wave pattern. Such a pattern necessarily needs to bypass the encoder and scrambler, and does not contain any sync bits. When the square wave pattern is selected, the PCS will send a repeating pattern of *n* ones followed by *n* zeros where *n* may be any number between 4 and 11, inclusive. The value of *n* is and implementation choice, and may be a programmable value in some devices. Register 3.42 controls the different test patterns that can be transmitted and checked.

a. The DUT should properly generate the square wave test pattern for all values of *n*.

Comments on Test Results

a. This test was not completed during the testing period due to test station limitations.

Test # and Label	Part(s)	Result(s)
Test 49.7.4: Pseudo-random test pattern reception	**a,b**	**Not Available**

Comments on Test Procedure

Purpose: To verify that the DUT generates the correct pseudo-random test patterns.

Clause 49 defines a pseudo-random test pattern generator and checker to be implemented in the PCS. This functionality allows the tester to not only verify BER over a channel, but also to force the DUT into transmitting the appropriate signals used for testing. When the PCS offers a direct connection to the PMA, the implementation of the generator and checker is mandatory. The jitter test pattern is generated by the scrambler, using specific seeds that are regularly loaded through the MDIO registers, and using one of two possible data patterns to be scrambled. Every 128 blocks, the scrambler is loaded with one of two seeds, or their inverses, in the following pattern: Seed A, Seed A Invert, or Seed B, Seed B Invert. The values of the seeds are defined in Clause 52, and shown in Table 52-20. The data patterns fed through the scrambler are either all zeros, or the encoding of two Local Fault ordered_sets, and the data pattern will be inverted when the inverted seeds are being used. Register 3.42 controls the different test patterns that can be transmitted and checked.

The pseudo-random checker utilizes the lock state machine and descrambler as it would during normal operations. However, the hi_ber state machine is disabled when the test pattern mode is enabled. When the output of the descrambler is the data pattern or its inverse, a match is detected. Since the transmitter's scrambler is loaded with a seed value every 128 blocks and the receiver's descrambler is running normally, a mismatch will be detected once every 128 blocks in the absence of errors. Therefore, the receiver will count in 128-block windows and ignore the first block within a window that contains a mismatch. All subsequent mismatches will be counted as errors.

a. The DUT should lock on and receive Pseudo-Random Test Pattern 1, and all mismatches apart from the first in every 128-blocks, should be counted.
b. The DUT should lock on and receive Pseudo-Random Test Pattern 2, and all mismatches apart from the first in every 128-blocks, should be counted.

Comments on Test Results

a. This test was not completed during the testing period due to test station limitations.
b. This test was not completed during the testing period due to test station limitations.

Test # and Label	Part(s)	Result(s)
Test 49.7.5: PRBS31 test pattern reception	**a**	**Not Available**

Comments on Test Procedure

Purpose: To verify that the DUT generates the correct PRBS31 test pattern.

Clause 49 defines an optional PRBS31 test pattern mode that can be implemented within the PCS for both transmission and checking of the pattern. When selected, the generator sends 16 bits of the test pattern at a time down to the PMA, bypassing the scrambler. The pattern generator implements an inverted version of the bit stream generated with the following polynomial: $G(x) = 1 + x^{28} + x^{31}$. Register 3.42 controls the different test patterns that can be transmitted and checked. When the receive channel is operating in PRBS31 test mode, the PRBS31 pattern checker checks the bits received from the PMA, bypassing the descrambler, and checks them relative to the PRBS31 test pattern. The pattern error checker is self synchronizing, and compares each bit received to the result of the PRBS31 generator based on the prior 31 bits received. When no errors occur, the pattern error signal will be zero. When an isolated bit error occurs, the PRBS31 pattern error signal will go high three times, and the test-pattern error counter will increment once for each bit time that the PRBS31 pattern error signal is high.

a. The DUT should properly receive the PRBS31 test pattern, and properly count all errors.

Comments on Test Results

a. This test was not completed during the testing period due to test station limitations.