IEEE P1149.10

High Speed Test Access Port and Distribution Architecture

CJ Clark, Intellitech
Scope

This standard defines a high speed test access port for delivery of test data, a packet format for describing the test payload and a distribution architecture for converting the test data to/from on-chip test structures.

The standard re-uses existing High Speed I/O (HSIO) known in the industry for the High-Speed Test Access Port. The HSIO connects to an on-chip distribution architecture through a common interface. The scope includes the distribution architecture test logic and packet decoder logic. The objective of the distribution architecture and packet decoder is that it can be readily re-used with different Integrated Circuits (ICs) that host different HSIO technology such that the standard addresses as large a part of the industry as possible.

The scope includes IEEE 1149.1 Boundary Scan Description Language (BSDL) and Procedural Description Language (PDL) documentation which can be used for configuring a mission mode HSIO to a test mode compatible with the High Speed Test Access Port (HSTAP). The same BSDL and PDL can then be used to deliver high-speed data to the on-chip test structures.
Disclaimer:

This is an organizing/teaching presentation to communicate at least one technical approach to realizing the standard. Nothing in the presentation has been voted on by the working group. Alternative technical solution presentations are encouraged.
Traditional Common Clock

- N scan channels
- 2N touch downs
- Skew challenges
- Tester resource

Derived Clock

- N scan channels
- 5 Touch downs
- Scales with new PHY
- Bond multiple channels
IEEE P1149.10

Define packet encoder/decoder & distribution
Matrix (shaded area)

Define packets for packet decoder
Enable packet formats to validate link
during testing (CRC32, running disparity, etc)

Define vendor documentation for
Enabling P1149.10 interface, enabling
Loopback and documenting requirements
of interface (system clock, diff swing,
Encoding (8b/10b, 64/66b, 128/130b etc)

Testing or implementation of SERDES BIST
Is not an objective of the standard.
Testing P1149.10 interface is not an
objective
Common Clock = 80G/sec = 400 chains, 806 pins and 200Mhz operation
Embedded Clock SERDES = 80G/sec = User Defined chains, 37 pins and user defined Mhz

**Bandwidth**

<table>
<thead>
<tr>
<th>Tester Chan</th>
<th>9</th>
<th>13</th>
<th>21</th>
<th>26</th>
<th>37</th>
<th>56</th>
<th>106</th>
<th>206</th>
<th>406</th>
<th>806</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI/SO+TAP+CLK</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chains</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SERDES = 4 pins + 4 pin TAP + clock. Channel bonding is used to have 2, 4 and 8 Serdes lanes. 5 SERDES lanes just included for comparison purposes. Bandwidth adjusted for encoding bit loss and 2% overhead of packet

11/9/2015
Observations

80G bandwidth - 806 pins running 400 scan chains at 200Mhz

P1149.10 - More flexible with number of internal scan-chains and scan-chain architecture. Channel bonding can be used for higher bandwidth or multi-site testing. Users can make their own economic choices.

Embedded clock SERDES is scalable. More bandwidth coming.

Common Clock - reaching ceiling - needs more pins and higher clock rates.

Normalizing for ATE pins Embedded Clock is 5-10x bandwidth than common clock.

- Common Clock 100 pins = 10G @200mhz
  SERDES 10G bandwidth with 9 pins
  11 sites = 99pins = 110G total bandwidth
CEI-25G  2013  25G/sec  16G/sec SATA 3.2 now - PCIe 4.0 - coming

Credo, Snowbush, Avago all with 25G SERDES IP in 2013 - Others?
Some with 28G-  Xilinx Virtex 7 HT (shipping) GTZ run to 28G/sec

2 CEI-25G SERDES will yield 50G bandwidth

Need  240 pins at 200mhz to equal one CEI-25G interface link

**Bandwidth Comparison**

- 50Mhz Scan
- 100Mhz Scan
- 200Mhz - Scan
- 6.5G - 8b/10b
- 6.5G - 64/66b
- 11G - 64/66b
- 16G SATA 3.2/PCIe 4.0
- 25G - 128/130b
Cost Trade-offs

Probe Card:

P1149.10 requires Gigabit probe card ( 'increased' cost ) and handler infrastructure.

Compare with common clock requires increased cost probe card (800+ contacts for single location).

Handler

Requires gigabit connections but less of them. Could use (Coax however SATA, CAT-6A, Fiber are also low cost choices).

Silicon

re-using mission mode SERDES so not expecting vendor to implement SERDES just for P1149.10. P1149.10 does not preclude dedicated SERDES either.

No use

P1149.10 does not mandate it is used during wafer test. So one may use P1149.10 in later stage processes. Instrument access at benchtop, characterization and FPGA configuration.
## The numbers

<table>
<thead>
<tr>
<th>Chains</th>
<th>Tester Chan</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>6.5G w/ 8b/10b</td>
<td>5.07</td>
</tr>
<tr>
<td>6.5G w/ 64/66b</td>
<td>6.18</td>
</tr>
<tr>
<td>11G w/ 64/66b</td>
<td>10.45</td>
</tr>
<tr>
<td>16G SATA 3.2/PCIe 4.0</td>
<td>15.44</td>
</tr>
<tr>
<td>25G w/ 128/130b</td>
<td>24.12</td>
</tr>
<tr>
<td>SI/SO+TAP+CLK</td>
<td>X</td>
</tr>
<tr>
<td>Chains</td>
<td>X</td>
</tr>
<tr>
<td>Tester Chan</td>
<td>9</td>
</tr>
</tbody>
</table>
Embedded Clock - block diagram - Scan virtualized

ATPG

STIL/WGL

P1149.10

Packets

STIL/WGL

PHY Encoding (8b/10b, etc)

Tester Binary

Tester Binary

SysClock

TDI

TDO

SISO Clk

PISO Clk

Packet Encode/Decode

TAP

TCK TMS
Multi-core support is via scan architecture (See 1149.1 pg. 411)
Gate_CH selects gating of 3 Target Cores.
Sel_CH selects core to observe response

Note that in P1149.10 multiple SO can be returned to packet encoder
Multi-core support is via scan architecture (See 1149.1 pg. 411)

BSDL Description
**BSDL Attributes & PDL**

Need to communicate: Sysclk(s) frequency, DiffSwing, encoding

How to get SERDES into P1149.10 Mode

Anything needed shall be communicated

In 1149.1-2013

Attribute SYS_CLOCK_REQUIREMENTS of MyChip : entity IS
"(SysClk, 198.5e6, 201.5e6, 1149_10_Enable) ";

Pin, Min F, Max F, Instruction, Instruction

Possible Addition:

Attribute PHY_1149_10 of MyChip : entity IS
"(SATA_TXP, SATA_RXP, 500E-3, 800E-3, 8B10B) ";

TX Rep Port, RX Rep Port, Min V, Max V , Encoding

(Needs to support multiple pin pairs)

Need to enumerate encodings understood by this standard
(8b10b, 64/66b, 128/130b, others?)


Group definitions - define scan chain groups 12/2/2013

SOF/EOF/IDLE/XON/XOFF/Reset characters

Power descriptions also possible in 1149.1-2013
BSDL Attributes & PDL

# MyCorp_SERDES.pdl
iPDLLevel 0 -version STD_1149_1_2013
iProcGroup MyCorp_SERDES

iProc enable_1149_10 {} {
  iWrite PLL 125Mhz ;# @40bits = 5G
  iWrite mode 1149_10
  iWrite encoding 8b10b
  iWrite TX_Swing 800mv
  iWrite Power ON;# disable pwrdn
  iApply
}

See 1149.1 PDL Tutorial for information of how this gets Translated to ATE patterns
Mission mode

PWR UP

Test?

Yes

Pre-P1149.10 Test

TAP Pins?

Yes

TAP

init_setup_1149_10

No

No

Compliance Pins

Yes

Assert compliance pattern

Dedicated

mission

No

No

No

Compliance Character?

Yes

Send Compliance Character

P1149.10 based Test

PWR DWN

Test SERDES or Test P1149.10 logic through TAP or other user defined means
Approach with Instruction Register accessible

8b/10b, 64/66b, 128/129b encoding
Required in order to guarantee there are sufficient 1s/0s in the data stream such that the data and clock can be recovered at receiver.
Alternatives - Remove packet protocol
- Assume 8b/10b encoding is managed in PHY
- Match scan-chains to width of parallel side of PHY
  - 40 bit wide PHY then 40 scan-chains

Scan-chains driven by SIPO Clock.

For OIF CEI-25G-LR and 802.3bj 25G/sec rate with 64 bit wide PHY yields 390mhz SIPO scan clock.

Instruction register, device_id and all chains required to run at same clock rate
For efficiency, scan-chains count must be multiple of PHY width. DFT dependent on mission mode PHY characteristics.
Loss of board/3D/MCM use of P1149.10 Tx to P1149.10 Rx daisy chain
Alternatives - Remove packet protocol

- Could throw away two words of three to get lower clock rate

Cost of probe card etc for 11Gbit high-speed interface but yielding Only 3.6G/sec bandwidth.
Alternatives - Remove packet protocol

- Could multiplex 40 bit scan chains

DFT of scan-chains relies on PHY width. Must be multiples of PHY width
Or loss of bandwidth.

41 scan-chains would cut bandwidth to IC as 41/80. 11G interface would
Have a bandwidth of 5.6G/sec

Assumes a VLSI test with large numbers of concurrent active scan-chains
Instrument networks, init_data, boundary registers may have less need for
large interleaved data
Alternatives - Remove packet protocol

Chains could be going to MISR - how to communicate the bits are MISR values coming in 40 bit words?

Assumes scan-chain/channels only for Wafer based test. Sets as a priority Over other needs such as in-the-field test/configuration.
Can this approach work with minimal logic?
Potential In-bound Packets

Packet descriptions shown without bit level encoding

CONFIG - Enable uninitialized P1149.10 interface to be enumerated to 'n'

TARGET <n> - specify where packets go

RESET - Assert reset* or TRST* (internally different signals)

RAW - Enable Interface in a RAW data mode (suitable for BER testing)
   Data is not processed by packet processor subsequently and all RX data is sent to TX

CH-SELECT - Specifies Channels and groups participating in subsequent SCAN

SCAN - Interleaved IR/DR Scan Packet

BOND - Bond multiple lanes together for increased bandwidth

All "R" response packets are ignored and forwarded to TX

11/9/2015
Generic Format of Packet

- **SOP**
- **CMD**
- **PAYLOAD**
- **CRC32**
- **EOP x 4**

Time 0
CONFIG

CONFIG - Enable uninitialized P1149.10 interface to be enumerated to 'n'

Device powers up with TARGET_ID of 0000. All packets received when ID is 0000 are processed. Once TARGET_ID is set, only packets following a TARGET packet with TARGET_ID of same will be processed.

1 byte 1 byte 2 bytes 4 bytes 4 bytes
SOP CONFIG TARGET_ID CRC32 EOP x 4

0x01

SOP = Start of Frame
EOP = End of Frame
TARGET <n>

TARGET - Sets device to listen for subsequent packets

If TARGET_ID matches TARGET_ID of device, it
   a) sends TARGETR response packet
   b) accepts incoming packets and sends "R" responses on TX
      until next TARGET_ID
If TARGET_ID does not match TARGET_ID of device
   a) device forwards all packets received to TX, examining each
      for TARGET_ID packet

1 byte  1 byte  2 bytes  4 bytes  4 bytes

| SOP | 0x02 | TARGET | TARGET_ID | CRC32 | EOP x 4 |

SOP = Start of Frame
EOP = End of Frame
RESET - Issue reset*

RESET - Issues test logic reset equivalent to going to TLR in state machine

Assert RESET* - Device determines necessary length of time/TCK cycles needed. ATE should not care.

\[
\begin{array}{c|c|c}
\text{TMS} & 0x00 & \text{TRST*} & 0x01 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
\text{1 byte} & \text{1 byte} & \text{2 bytes} & \text{4 bytes} & \text{4 bytes} \\
\text{SOP} & \text{RESET} & \text{TYPE} & \text{CRC32} & \text{EOP x 4} \\
0x03 & & & & \\
\end{array}
\]

TYPE is 16 bits

SOP = Start of Frame

EOP = End of Frame
RAW - Put 1149.10 interface in raw data mode

RAW - enable BER testing of P1149.10 interface without packet decode
- Requires 1149.1 access to reset or power-cycle

1 byte 1 byte 2 bytes 4 bytes 4 bytes 4 bytes

<table>
<thead>
<tr>
<th>SOP</th>
<th>RAW</th>
<th>0x0000</th>
<th>COUNT</th>
<th>CRC32</th>
<th>EOP x 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0x04</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SOP = Start of Packet
EOP = End of Packet
CH-SELECT - Specify group and channels for SCAN

1 byte 1 byte 2 bytes 2 bytes
SOP CH-SELECT SCAN_GROUP #Ch-Select

0x05

# x 2 bytes 2 bytes 4 bytes 4 bytes
Channel-Select 0x0000 CRC32 EOP x 4

2* ((#Ch-Select -1) mod 2)
SCAN Packet

Send data to IR or DR scan chain(s) as needed.

ICSU - IR Scan, Capture, Shift, Update

<table>
<thead>
<tr>
<th>1 byte</th>
<th>1 byte</th>
<th>1 byte</th>
<th>1 byte</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>SCAN</td>
<td>ID</td>
<td>0b000</td>
<td>ICSU</td>
</tr>
<tr>
<td>0x06</td>
<td></td>
<td></td>
<td></td>
<td>#Payload-Frames</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4 bytes</th>
<th># x 4 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle-Count</td>
<td>PAYLOAD</td>
<td>CRC32</td>
<td>EOP x 4</td>
</tr>
</tbody>
</table>
Interleaved SCAN Packet Format - dealing with CHs of different lengths

CH0 = 32bits  CH3 = 33bits  ICSU = Instruction, Capture, Shift, Update
(Interleave does not need to be transmitted but it is 4 bits in this case)

Capture-shift

<table>
<thead>
<tr>
<th>1 byte</th>
<th>1 byte</th>
<th>2 bytes</th>
<th>2 bytes</th>
<th>2 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>CH-SELECT</td>
<td>Scan_Group</td>
<td>#CH-Select</td>
<td>Channel-Select</td>
<td>CRC32</td>
<td>EOP x 4</td>
</tr>
<tr>
<td>0x05</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0009</td>
<td>0x0001</td>
<td>0x0002</td>
<td>0x0020</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1 byte</th>
<th>1 byte</th>
<th>1 byte</th>
<th>1 byte</th>
<th>4 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>SCAN</td>
<td>ID</td>
<td>ICSU</td>
<td>#Payload-Frames</td>
<td>Cycle-Count</td>
</tr>
<tr>
<td>0x06</td>
<td>0x01</td>
<td>0b0000</td>
<td>0b0110</td>
<td>0x0000_0002</td>
<td>0x0000_0020</td>
</tr>
</tbody>
</table>

... 4bits 4bits 4bits 4bits 4bits 4bits 4bits 4bits CH0 CH3 CH0 CH3 CH0 CH3 CH0 CH3 ... QRST VXYZ ... 4bits 4bits 4bits 4bits 4bits 4bits 4bits 4bits CH0 CH3 CH0 CH3 CH0 CH3 CH0 CH3 ... ABCD MNOP ... 4 bytes 4 bytes CRC32 EOP x 4
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>0x05</td>
<td>CH-SELECT</td>
<td>0x0001</td>
</tr>
<tr>
<td>Scan_Group</td>
<td>0x0001</td>
<td>#CH-Select</td>
<td>0x0008</td>
</tr>
<tr>
<td>Channel-Select</td>
<td></td>
<td>CRC32</td>
<td>EOP x 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Field</th>
<th>Value</th>
<th>Field</th>
<th>Value</th>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>0x06</td>
<td>SCAN</td>
<td>0x02</td>
<td>ID</td>
<td>0b0000</td>
<td>ICSU</td>
<td>0b0010</td>
</tr>
<tr>
<td>#Payload-Frames</td>
<td></td>
<td>Cycle-Count</td>
<td>0x0000_0001</td>
<td></td>
<td></td>
<td>0x0000_0001</td>
<td></td>
</tr>
<tr>
<td>CH3</td>
<td>0bXXXL</td>
<td>PAD</td>
<td>0x000000</td>
<td>CRC32</td>
<td></td>
<td>EOP x 4</td>
<td></td>
</tr>
</tbody>
</table>

Shift

11/9/2015
### Packet Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>1 byte</td>
<td>0x05</td>
</tr>
<tr>
<td>CH-SELECT</td>
<td>1 byte</td>
<td>0x0001</td>
</tr>
<tr>
<td>Scan_Group</td>
<td>2 bytes</td>
<td>0x0001</td>
</tr>
<tr>
<td>#CH-Select</td>
<td>2 bytes</td>
<td>0x0009</td>
</tr>
<tr>
<td>Channel-Select</td>
<td>2 bytes</td>
<td>0x0000_0000</td>
</tr>
<tr>
<td>CRC32</td>
<td>4 bytes</td>
<td>0x0000_0000</td>
</tr>
<tr>
<td>EOP x 4</td>
<td>4 bytes</td>
<td>0x0000_0000</td>
</tr>
</tbody>
</table>

### Packet Example

```
0x05 0x0001 0x0001 0x0009
0x06 0x03 0x00 0x0b0001
```

### Additional Fields

- **#Payload-Frames**
- **Cycle-Count**
- **CRC32**
- **EOP**

Update
Interleaved SCAN Packet Format - dealing with CHs of different lengths
CH0 = 32bits  CH3 = 33bits   ICSU = Instruction, Capture, Shift, Update
(Interleave does not need to be transmitted but it is 8 bits in this case)
SO3

SI1
SI2

CAP
SHFT
UPD

ON0

Decompress

compactor

Core1

SO4

SI1
SI2

CAP
SHFT
UPD

ON1

Decompress

compactor

Core2

SI5

SO5

11 = broadcast, 01 = Core1, 10 = Core2

SO4 has to equal SO3
Interleaved SCAN Packet Format - CH1 = 5 bits CH2 = 5 bits CH3 = 4 bits CH4 = 4 bits, CH5 = 2 bits

Capture-shift

<table>
<thead>
<tr>
<th>SOP</th>
<th>SCAN</th>
<th>ID</th>
<th>0b00</th>
<th>ICSU</th>
<th>Scan_Group</th>
<th>#CH-Select</th>
<th>Channel-Select</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x05</td>
<td>0x01</td>
<td>0b0111</td>
<td>0x01</td>
<td>0x1</td>
<td>0b010000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... 2x16 2x16

<table>
<thead>
<tr>
<th>#Payload-Frames</th>
<th>Cycle-Count</th>
<th>PAD</th>
<th>CH5</th>
<th>CRC32</th>
<th>EOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0002</td>
<td>0x0002</td>
<td>0b11</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... 2x16 2x16

<table>
<thead>
<tr>
<th>SOP</th>
<th>SCAN</th>
<th>ID</th>
<th>ICSU</th>
<th>Group</th>
<th>#CH-Select</th>
<th>Channel-Select</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x05</td>
<td>0b10</td>
<td>0b111</td>
<td>0x01</td>
<td>0x1</td>
<td>0b000011</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... 2x16 2x16

<table>
<thead>
<tr>
<th>#Payload-Frames</th>
<th>Cycle-Count</th>
<th>CH2</th>
<th>CH1</th>
<th>CH2</th>
<th>CH1</th>
<th>CRC32</th>
<th>EOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0002</td>
<td>0x0005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scan Response

1 byte 1 byte 1 byte 1 byte 4 bytes
SOP SCANR ID 0b0000 ICSU #Payload-Frames

4 bytes

4 bytes 4 bytes
Cycle-Count PAYLOAD CRC32 EOP x 4
Channel Bonding packet

1 byte 1 byte 1 byte 1 byte 4 bytes 4 bytes

<table>
<thead>
<tr>
<th>SOP</th>
<th>BOND</th>
<th>0x00</th>
<th>LANE</th>
<th>CRC32</th>
<th>EOP x 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Out-bound Packets

TARGETR - Target Packet Response

CONFIGR - CONFIG Packet Response

RAWR - RAW Packet response

RESETR - Reset Packet response

CH-SELECTR - CH-SELECT Response

SCANR - Interleaved Scan Packet Response

BOND - Bond multiple lanes together for increased bandwidth

IDLE <n> - Tell ATE to insert N IDLE packets *(is this needed?)*

All inbound packets following a TARGET for an alternative device
CONFIGR

CONFIGR - Response to CONFIG command

Device powers up with TARGET_ID of 0000. All packets received when ID is 0000 are processed. Once TARGET_ID is set, only packets following a TARGET packet with TARGET_ID of same will be processed.

<table>
<thead>
<tr>
<th>1 byte</th>
<th>1 byte</th>
<th>2 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SOP</strong></td>
<td><strong>CONFIGR</strong></td>
<td><strong>TARGET_ID</strong></td>
<td><strong>CRC32</strong></td>
<td><strong>EOP x 4</strong></td>
</tr>
<tr>
<td>0x81</td>
<td>0x0000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SOP = Start of Frame
EOP = End of Frame
TARGETR

TARGETR - Response packet

<table>
<thead>
<tr>
<th>1 byte</th>
<th>1 byte</th>
<th>2 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>TARGETR</td>
<td>TARGET_ID</td>
<td>CRC32</td>
<td>EOP x 4</td>
</tr>
<tr>
<td>0x82</td>
<td>0x0000</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

SOP = Start of Frame
EOP = End of Frame
RESETR - Issue reset* Response

RESETR - Issues test logic reset equivalent to going to TLR in state machine

<table>
<thead>
<tr>
<th>1 byte</th>
<th>1 byte</th>
<th>2 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>RESETR</td>
<td>TYPE</td>
<td>CRC32</td>
<td>EOP x 4</td>
</tr>
<tr>
<td>0x83</td>
<td>0x0000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TYPE = TMS 0x00 or TRST* 0x01

SOP = Start of Frame
EOP = End of Frame
RAW response

<table>
<thead>
<tr>
<th>1 byte</th>
<th>1 byte</th>
<th>2 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>RAWR</td>
<td>0x0000</td>
<td>COUNT</td>
<td>CRC32</td>
<td>EOP x 4</td>
</tr>
<tr>
<td>0x84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# CH-SELECT response

<table>
<thead>
<tr>
<th>1 byte</th>
<th>1 byte</th>
<th>2 bytes</th>
<th>2 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>CH-SELECTR</td>
<td>SCAN_GROUP</td>
<td>#Ch-Select</td>
</tr>
<tr>
<td>0x85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># x 2 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Channel-Select</td>
<td>0x0000</td>
<td>CRC32</td>
<td>EOP x 4</td>
</tr>
</tbody>
</table>
SCANR Packet

Respond to IR or DR scan chain(s) as needed.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>1 byte</td>
<td>1 byte</td>
<td>1 byte</td>
<td>4 bytes</td>
<td></td>
</tr>
<tr>
<td>SOP</td>
<td>SCANR</td>
<td>ID</td>
<td>0xb0000</td>
<td>ICSU</td>
<td></td>
</tr>
<tr>
<td>0x86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 bytes</td>
<td># x 4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cycle-Count</td>
<td>PAYLOAD</td>
<td>CRC32</td>
<td>EOP x 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scan_Group</td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#CH-Select</td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOP</td>
<td>0x86</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCANR</td>
<td>0x02</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>0b0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICSU</td>
<td>0b0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Payload-Frames</td>
<td>0x0000_0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cycle-Count</td>
<td>0x0000_0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH3</td>
<td>0bXXXR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAD</td>
<td>0b0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOP x 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1 byte  1 byte  2 byte  4 bytes  4 bytes
SOP    BONDR    LANE    CRC32    EOP x 4

0x87
Special Characters

K28.5 IDLE character

IDLE characters can appear anywhere in the transmitted data stream. Receivers filter out IDLE characters.
### Special Characters in 8b/10b

<table>
<thead>
<tr>
<th>Input</th>
<th>DEC</th>
<th>HEX</th>
<th>Control symbols</th>
<th>RD = -1</th>
<th>RD = +1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DEC</td>
<td>HEX</td>
<td>HGF EDCBA</td>
<td>abcdei fghj</td>
<td>abcdei fghj</td>
</tr>
<tr>
<td>K.28.0</td>
<td>28</td>
<td>1C</td>
<td>000 11100</td>
<td>001111 0100</td>
<td>110000 1011</td>
</tr>
<tr>
<td>K.28.1 †</td>
<td>60</td>
<td>3C</td>
<td>001 11100</td>
<td>001111 1001</td>
<td>110000 0110</td>
</tr>
<tr>
<td>K.28.2</td>
<td>92</td>
<td>5C</td>
<td>010 11100</td>
<td>001111 0101</td>
<td>110000 1010</td>
</tr>
<tr>
<td>K.28.3</td>
<td>124</td>
<td>7C</td>
<td>011 11100</td>
<td>001111 0011</td>
<td>110000 1100</td>
</tr>
<tr>
<td>K.28.4</td>
<td>156</td>
<td>9C</td>
<td>100 11100</td>
<td>001111 0010</td>
<td>110000 1101</td>
</tr>
<tr>
<td>K.28.5 †</td>
<td>188</td>
<td>BC</td>
<td>101 11100</td>
<td>001111 1010</td>
<td>110000 0101</td>
</tr>
<tr>
<td>K.28.6</td>
<td>220</td>
<td>DC</td>
<td>110 11100</td>
<td>001111 0110</td>
<td>110000 1001</td>
</tr>
<tr>
<td>K.28.7 ‡</td>
<td>252</td>
<td>FC</td>
<td>111 11100</td>
<td>001111 1000</td>
<td>110000 0111</td>
</tr>
<tr>
<td>K.23.7</td>
<td>247</td>
<td>F7</td>
<td>111 10111</td>
<td>111010 1000</td>
<td>000101 0111</td>
</tr>
<tr>
<td>K.27.7</td>
<td>251</td>
<td>FB</td>
<td>111 11011</td>
<td>110110 1000</td>
<td>001001 0111</td>
</tr>
<tr>
<td>K.29.7</td>
<td>253</td>
<td>FD</td>
<td>111 11101</td>
<td>101110 1000</td>
<td>010001 0111</td>
</tr>
<tr>
<td>K.30.7</td>
<td>254</td>
<td>FE</td>
<td>111 11110</td>
<td>011110 1000</td>
<td>100001 0111</td>
</tr>
</tbody>
</table>
- **XAUI defines the following tools (characters):**

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>data (d)</td>
<td>Dxx.y</td>
</tr>
<tr>
<td>align (A)</td>
<td>K28.3</td>
</tr>
<tr>
<td>sync (K)</td>
<td>K28.5</td>
</tr>
<tr>
<td>skip (R)</td>
<td>K28.0</td>
</tr>
<tr>
<td>start (S)</td>
<td>K27.7</td>
</tr>
<tr>
<td>terminate (T)</td>
<td>K29.7</td>
</tr>
<tr>
<td>error (E)</td>
<td>K30.7</td>
</tr>
<tr>
<td>ordered set (O)*</td>
<td>K28.2</td>
</tr>
</tbody>
</table>

* Proposed to support FC Ordered Sets
Code Overview

Data Codewords have “01” sync preamble

64 bit data field (scrambled)

Mixed Data/Control frames are identified with a “10” sync preamble. Both the coded 56-bit payload and TYPE field are scrambled

8-bit TYPE combined 56 bit data/control field (scrambled)

00, 11 preambles are considered code errors and cause the packet to be invalidated by forcing an error (E) symbol on the HARI output
## Code Summary

<table>
<thead>
<tr>
<th>Hari Pattern</th>
<th>Sync</th>
<th>Bit fields 0-63</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDDD/DDDD</td>
<td>0 1</td>
<td>D0 D1 D2 D3 D4 D5 D6 D7</td>
</tr>
<tr>
<td>ZZZZ/ZZZZ</td>
<td>1 0</td>
<td>0x1e Z0 Z1 Z2 Z3 Z4 Z5 Z6 Z6</td>
</tr>
<tr>
<td>ZZZZ/SDDD</td>
<td>1 0</td>
<td>0x33 Z0 Z1 Z2 Z3 Z4 D5 D6 D7</td>
</tr>
<tr>
<td>SDDD/DDDD</td>
<td>1 0</td>
<td>0x78 D1 D2 D3 D4 D5 D6 D7</td>
</tr>
<tr>
<td>TZZZ/ZZZZ</td>
<td>1 0</td>
<td>0x87 Z1 Z2 Z3 Z4 Z5 Z6 Z7</td>
</tr>
<tr>
<td>DTZZ/ZZZZ</td>
<td>1 0</td>
<td>0x99 D0 Z2 Z3 Z4 Z5 Z6 Z7</td>
</tr>
<tr>
<td>DDTZ/ZZZZ</td>
<td>1 0</td>
<td>0xaa D0 D1 Z3 Z4 Z5 Z6 Z7</td>
</tr>
<tr>
<td>DDDT/ZZZZ</td>
<td>1 0</td>
<td>0xb4 D0 D1 D2 Z4 Z5 Z6 Z7</td>
</tr>
<tr>
<td>DDDD/TZZZ</td>
<td>1 0</td>
<td>0xcc D0 D1 D2 D3 Z5 Z6 Z7</td>
</tr>
<tr>
<td>DDDD/DTZZ</td>
<td>1 0</td>
<td>0xd2 D0 D1 D2 D3 D4 Z6 Z7</td>
</tr>
<tr>
<td>DDDD/DDT2</td>
<td>1 0</td>
<td>0xe1 D0 D1 D2 D3 D4 D5 Z7</td>
</tr>
<tr>
<td>DDDD/DDDT</td>
<td>1 0</td>
<td>0xff D0 D1 D2 D3 D4 D5 D6</td>
</tr>
</tbody>
</table>
Control Characters for P1149.10 Unencoded/UnScrambled
128/130 encoding should be similar to 64/66
64/67 (Interlaken) same as 64/66 with 3 bit encoding (bit 66 indicating invert)

<table>
<thead>
<tr>
<th>Control Character</th>
<th>8b/10b</th>
<th>64/66</th>
</tr>
</thead>
<tbody>
<tr>
<td>S (Start of Frame)</td>
<td>1 0xFB</td>
<td>0b10 0xFB 0xXX 0xXX</td>
</tr>
<tr>
<td>T (Terminate/EOF)</td>
<td>1 0xFD</td>
<td>0b10 0xFD 0xXX 0xXX</td>
</tr>
<tr>
<td>K (IDLE)</td>
<td>1 0xBC</td>
<td>0b10 0xBC 0xXX 0xXX</td>
</tr>
<tr>
<td>E (ERROR)</td>
<td>1 0xFE</td>
<td>0b10 0xFE 0xXX 0xXX</td>
</tr>
<tr>
<td>A (Stop)</td>
<td>1 0x7C</td>
<td>0b10 0x7C 0xXX 0xXX</td>
</tr>
<tr>
<td>R (Resume)</td>
<td>1 0x1C</td>
<td>0b10 0x1C 0xXX 0xXX</td>
</tr>
<tr>
<td>O (Clear)</td>
<td>1 0x5C</td>
<td>0b10 0x5C 0xXX 0xXX</td>
</tr>
</tbody>
</table>

Reset can be used to reset IC receive channel when errors or other occur.
It can also be used to get out of 'raw' mode.
Attribute PHY_1149_10 of MyChip : entity IS
"(SATA_TXP, SATA_RXP, 3.125E9, 500E-3, 800E-3, 8B_10B) ";

TX Rep Port, RX Rep Port, Min V, Max V, Encoding

<phy description> := attribute PHY_1149_10 of <entity> IS <phy_string> <semicolon>

<entity> := <component name> - defined in 1149.1 already

<phy_string> := <left paren><phy_list><right paren>
   {<comma> <left paren><phy_list><right paren> }
<phy_list> := <tx> <comma> <rx> <comma><rate><comma><min swing><comma>
   <max swing><comma><encoding>

<tx> := <representative port> | <portID>
.rx> := <representative port> | <portID>
<rate>:= <real> (bits/sec)
<min swing> := <real> (in Volts)
<max swing> := <real>
<encoding> := NONE | 8B_10B | 64B_66B | 64B_67B | 128B_130B | <mnemonic_identifier>
Attribute Packet_Map of MyChip : entity IS
"(Interleave_Size: 8 ), " &
"(Scan_Group 1 : 1 to 16 )," &
"(Scan_Group 2 : 17 to 32, 33 )";

Communicates the interleave size of the packet and the mapping of the groups chosen by the designer of the decoder/encoder in the IC.
Attribute CONTROL_CHARS of MyChip : entity IS
"(SOP: 0xFB ), " &
"(EOP: 0xFD )," &
"(IDLE: 0xBC)," &
"(ERROR: 0xFE )," &
"(XOFF: 0x7C )," &
"(XON: 0x1C)," &
"(COMPLIANCE: 0x33)," &
"(CLEAR: 0x5C);"

<control_character_description> := attribute CONTROL_CHARS of <target>
  <colon> entity IS <control_string> <semicolon>
<control_string> := <left paren><control_char> <colon> <hex pattern> <right paren>
{<comma> <left paren><control_char> <colon> <hex pattern> <right paren> }
<control_char> := SOP | EOP | IDLE | ERROR | XOFF | XON | CLEAR | COMPLIANCE

A semantic rule in the draft would require all <control char> to be defined
All <control char> should be passed on via the TX pins when the P1149.10 Interface does not match the current TARGET_ID.
Attribute CONTROL_CHARS of MyChip: entity IS

"(SOP : 0x2_FB_XX_XX_XX_XX_XX_XX_XX_XX )," &
"(EOP : 0x2_FD_XX_XX_XX_XX_XX_XX_XX_XX )," &
"(IDLE : 0x2_BC_XX_XX_XX_XX_XX_XX_XX_XX ),"&
"(ERROR : 0x2_FE_XX_XX_XX_XX_XX_XX_XX_XX ),"&
"(XOFF : 0x2_7C_XX_XX_XX_XX_XX_XX_XX_XX ),"&
"(XON : 0x2_1C_XX_XX_XX_XX_XX_XX_XX_XX ),"&
"(RESET : 0x2_5C_XX_XX_XX_XX_XX_XX_XX_XX )";
Channel-Select = 0b1111

CH1: AEI
CH2: BFJ
CH3: CGK
CH4: DHL

DATA_SIZE = 4
INTERLEAVE_SIZE = 1
Cycle Count = 3

2nd Data byte

1st Data byte

4th Data byte

3rd Data byte

SI0  A   E   I  →  SO0
SI1  B   F   J  →  SO1
SI2  C   G   K  →  SO2
SI3  D   H   L  →  SO3

LSB (transmitted first)
Channel-Select = 0b1111

CH1: AEI  DATA_SIZE = 4
CH2: BFJ  INTERLEAVE_SIZE = 1
CH3: CGK  Cycle Count = 3
CH4: DHL

LSB (transmitted first)

2nd Data byte

4th Data byte

1st Data byte

3rd Data byte
Channel-Select = 0b0111

CH1: ALI
CH2: BMJ
CH3: CNK
X = PAD/DON’T CARE

DATA_SIZE = 4
INTERLEAVE_SIZE = 1
Cycle Count = 3

LSB (transmitted first)
<table>
<thead>
<tr>
<th>Hex</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x7C</td>
<td>0b01111100</td>
</tr>
<tr>
<td>0x03</td>
<td>0b00000011</td>
</tr>
<tr>
<td>0x00</td>
<td>0b00000000</td>
</tr>
<tr>
<td>0x18</td>
<td>0b00011000</td>
</tr>
<tr>
<td>0x11</td>
<td>0b00010001</td>
</tr>
<tr>
<td>0x77</td>
<td>0b01110111</td>
</tr>
<tr>
<td>0x98</td>
<td>0b10011000</td>
</tr>
<tr>
<td>0x1C</td>
<td>0b00011100</td>
</tr>
</tbody>
</table>