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.
Common Clock - Interface is synchronous to clock driven by ATE

Embedded Clock - Clock is embedded in the interface
- While ATE may drive the system clock needed by UUT, Clock for UUT could come from other source.
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></th>
<th>9</th>
<th>13</th>
<th>21</th>
<th>26</th>
<th>37</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>400</th>
<th>800</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI/SO+TAP+CLK</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chains</td>
<td>20</td>
<td>37</td>
<td>50</td>
<td>100</td>
<td>200</td>
<td>400</td>
<td>800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tester Chan</td>
<td>9</td>
<td>13</td>
<td>21</td>
<td>26</td>
<td>37</td>
<td>56</td>
<td>106</td>
<td>206</td>
<td>406</td>
<td>806</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

1/13/2014
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></th>
<th>200</th>
<th>50</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1.85</td>
</tr>
<tr>
<td></td>
<td>3.70</td>
<td>0.93</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.25</td>
<td>5</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>40</td>
<td>80</td>
<td>40</td>
</tr>
</tbody>
</table>

- **6.5G w/ 8b/10b**: 5.07, 10.14, 20.28, 25.35, 40.56
- **6.5G w/ 64/66b**: 6.18, 12.35, 24.71, 30.88, 49.42
- **11G w/64/66b**: 10.45, 20.91, 41.81, 52.27, 83.63
- **16G SATA 3.2/PCIe 4.0**: 15.44, 30.88, 61.76, 77.19, 123.51
- **25G w/ 128/130b**: 24.12, 48.25, 96.49, 120.62, 192.98

<table>
<thead>
<tr>
<th></th>
<th>200</th>
<th>50</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
<td>13</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>37</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>400</td>
<td>800</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></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>chains</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>10</td>
<td>X</td>
<td>25</td>
<td>50</td>
<td>100</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td>tester chan</td>
<td>9</td>
<td>13</td>
<td>21</td>
<td>26</td>
<td>37</td>
<td>56</td>
<td>106</td>
<td>206</td>
<td>406</td>
<td>806</td>
</tr>
</tbody>
</table>
Common Scan Clock - Simplified block diagram

- ATPG
- STIL/WGL
- Tester Binary
- 1149.1
- PDL
- STIL/WGL
- STIL/WGL
- Tester Binary
- SysClock
- SI1
- SI2
- SI3
- SI4
- SI5
- SI6
- SO1
- SO2
- SO3
- SO4
- SO5
- SO6
- TDI
- TDO
- TCK
- TMS
- TAP
Embedded Clock - block diagram - Scan virtualized

- ATPG
- STIL/WGL
- P1149.10
- STIL/WGL
- PHY Encoding (8b/10b, etc)
- Tester Binary

1149.1
PDL

Tester Binary

SysClock

Packet Encode/Decode

TAP

SDI

S01

S02

S03

S04

S05

S06

SDO

TDO

TCK TMS

Sipo Clk

Piso Clk
Multi-core support is via scan architecture (See 1149.1 pg. 411)
Gate_WSP selects gating of 3 Target Cores.
Sel_WSP selects core to observe response
Multi-core support is via scan architecture (See 1149.1 pg. 411)

BSDL Description

package REG_1500_ASSM is
use STD_1149_1_2012.all;
end REG_1500_ASSM;

package body REG_1500_ASSM is
use STD_1149_1_2012.all;
use REG_1500.all;
use REG_1500S.all;

attribute REGISTER_MNEMONICS of REG_1500_ASSM : package is
"WSP  ( " &
"   None   (0B00) <Bypass all WSPs>, "&
"   WSP1   (0B01) <Observe WSP(1)>, "&
"   WSP2   (0B10) <Observe WSP(2)>, "&
"   WSP3   (0B11) <Observe WSP(3) > "&
" )," &

"BRDCST ( " &
"   None (OB000) <All WSP held>, "&
"   WSP1 (OB001) <Scan WSP(1) only>, "&
"   WSP2 (OB010) <Scan WSP(2) only>, "&
"   WSP3 (OB011) <Scan WSP(3) only>, "&
"   1AND2 (OB110) <Scan just WSP(1) and WSP(2)> , "&
"   ALLWSP (OB111) <Scan all WSPs > "&
" )"&;

Attribute REGISTER_MNEMONICS of REG_1500_ASSM : package is
"WSP  ( " &
"   None   (0B00) <Bypass all WSPs>, "&
"   WSP1   (0B01) <Observe WSP(1)>, "&
"   WSP2   (0B10) <Observe WSP(2)>, "&
"   WSP3   (0B11) <Observe WSP(3) > "&
" )," &

"BRDCST ( " &
"   None (OB000) <All WSP held>, "&
"   WSP1 (OB001) <Scan WSP(1) only>, "&
"   WSP2 (OB010) <Scan WSP(2) only>, "&
"   WSP3 (OB011) <Scan WSP(3) only>, "&
"   1AND2 (OB110) <Scan just WSP(1) and WSP(2)> , "&
"   ALLWSP (OB111) <Scan all WSPs > "&
" )"&;

end REG_1500_ASSM;
BSDL Attributes & PDL

Need to communicate: Sysclck(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 characters
Outoforder - No

Power descriptions also possible in 1149.1-2013

1/13/2014

1/13/2014
BSDL Attributes & PDL

# MyCorp_SERDES.pdl
iPDLLLevel 0 -version STD_1149_1_2013
iProcGroup MyCorp_SERDES

iProc init_setup_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
Flow

- **PWR UP**
  - Pre-P1149.10 Test
  - **TAP Pins?**
    - Yes: **init_setup_1149_10**
    - No: P1149.10 based Test
  - P1149.10 based Test
- PWR DWN

Test SERDES or Test P1149.10 logic through TAP or other user defined means.
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?

Packet Decoder/Encoder with Distribution Matrix
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

SCAN - Interleaved IR/DR Scan Packet

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

<table>
<thead>
<tr>
<th>SOF</th>
<th>CONFIG</th>
<th>TARGET ID</th>
<th>CRC32</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x01</td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SOF = Start of Frame
EOF = 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

<table>
<thead>
<tr>
<th>SOF</th>
<th>TARGET</th>
<th>TARGET ID</th>
<th>CRC32</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SOF = Start of Frame
EOF = 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.

<table>
<thead>
<tr>
<th>TMS</th>
<th>TRST*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SOF</th>
<th>RESET</th>
<th>TYPE</th>
<th>CRC32</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03</td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SOF = Start of Frame
EOF = 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

\[
\begin{array}{cccc}
\text{SOF} & \text{RAW} & \text{CRC32} & \text{EOF} \\
0x04 & & & \\
\end{array}
\]

SOF = Start of Frame
EOF = End of Frame
Send data to IR or DR scan chain(s) as needed.

ICSU - IR Scan, Capture, Shift, Update

One-hot chain select is large for large # scan chains  
Reserve 0 for IR Scan?  400 scan chains = 50 bytes of chain select

<table>
<thead>
<tr>
<th>SOF</th>
<th>SCAN</th>
<th>ICSU</th>
<th># Data Words</th>
<th>Cycle Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ICSU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0x07

<table>
<thead>
<tr>
<th>#chain Sel</th>
<th>chain Select</th>
<th>PAYLOAD</th>
<th>CRC32</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Interleaved SCAN Packet Format - dealing with WSPs of different lengths
WSP0 = 32bits  WSP3 = 33bits  ICSU = Instruction, Capture, Shift, Update

Capture-shift

<table>
<thead>
<tr>
<th>8bits</th>
<th>4bits</th>
<th>4bits</th>
<th>8bits</th>
<th>8bits</th>
<th>16 bits</th>
<th>2x16</th>
<th>2x16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOF</td>
<td>SCAN</td>
<td>ID</td>
<td>ICSU</td>
<td>Grp</td>
<td>#WSP Sel</td>
<td>WSP Select</td>
<td># Data Bytes</td>
</tr>
<tr>
<td>0x06</td>
<td>0b0001</td>
<td>0b0110</td>
<td></td>
<td>0x01</td>
<td>0x1</td>
<td>0x0009</td>
<td>0x0008</td>
</tr>
</tbody>
</table>

... WSP0  WSP3  WSP0  WSP3  WSP0  WSP3  WSP0  WSP3

... CRC32  EOF

Shift

<table>
<thead>
<tr>
<th>2x16</th>
<th>2x16</th>
<th>2x16</th>
</tr>
</thead>
<tbody>
<tr>
<td># x16</td>
<td># Data Words</td>
<td>Cycle Cnt</td>
</tr>
<tr>
<td>0x06</td>
<td>0b10 0b010</td>
<td>0x01 0x1</td>
</tr>
</tbody>
</table>

... WSP3  CRC32  EOF

Update

<table>
<thead>
<tr>
<th>2x16</th>
<th>2x16</th>
</tr>
</thead>
<tbody>
<tr>
<td># x16</td>
<td>Cycle Cnt</td>
</tr>
<tr>
<td>0x06</td>
<td>0b11 0b010</td>
</tr>
</tbody>
</table>

... CRC32  EOF

High Speed TAG
Potential Out-bound Packets

TARGETR - Target Packet Response

CONFIGR - CONFIG Packet Response

RAWR - RAW Packet response

RESETR - Reset Packet response

SCANR - Interleaved Scan Packet Response

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.

SOF CONFIGR TARGET ID CRC32 EOF

0x81 0x01

SOF = Start of Frame
EOF = End of Frame
TARGETR

TARGETR - Response packet

<table>
<thead>
<tr>
<th>SOF</th>
<th>TARGETR</th>
<th>TARGET ID</th>
<th>CRC32</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x82</td>
<td></td>
<td>0x01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SOF = Start of Frame
EOF = End of Frame
RESETR - Issue reset* Response

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

| TMS | 0x00 | TRST* | 0x01 |

<table>
<thead>
<tr>
<th>SOF</th>
<th>RESETR</th>
<th>TYPE</th>
<th>CRC32</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x83</td>
<td></td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SOF = Start of Frame
EOF = End of Frame
Send data to IR or DR scan chain(s) as needed.

This may need to be condensed? If return scan data is not same as in-scan data, what difficulty exists for packet encoder to determine #datawords and #cycle count?

1 byte

<table>
<thead>
<tr>
<th>SOF</th>
<th>SCANR</th>
<th>#chain Sel</th>
<th>chain Select</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0x87

4 bytes

<table>
<thead>
<tr>
<th># Data Words</th>
<th>Cycle Count</th>
<th>PAYLOAD</th>
<th>CRC32</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

### Control symbols

<table>
<thead>
<tr>
<th>Input</th>
<th>DEC</th>
<th>HEX</th>
<th>HGF EDCBA</th>
<th>abcdei fghj</th>
<th>abcdei fghj</th>
</tr>
</thead>
<tbody>
<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>Tool</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

01 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

10 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</td>
<td>D0  D1  D2  D3  D4  D5  D6  D7</td>
</tr>
<tr>
<td>ZZZZ/ZZZZ</td>
<td>1</td>
<td>0x1e Z0  Z1  Z2  Z3  Z4  Z5  Z6  Z6</td>
</tr>
<tr>
<td>ZZZZ/SDDD</td>
<td>1</td>
<td>0x33 Z0  Z1  Z2  Z3  D5  D6  D7</td>
</tr>
<tr>
<td>SDDD/DDDD</td>
<td>1</td>
<td>0x78 D1  D2  D3  D4  D5  D6  D7</td>
</tr>
<tr>
<td>TZZZ/ZZZZ</td>
<td>1</td>
<td>0x87 Z1  Z2  Z3  Z4  Z5  Z6  Z7</td>
</tr>
<tr>
<td>DTZZ/ZZZZ</td>
<td>1</td>
<td>0x99 D0  Z2  Z3  Z4  Z5  Z6  Z7</td>
</tr>
<tr>
<td>DDTZ/ZZZZ</td>
<td>1</td>
<td>0xaa D0  D1  Z3  Z4  Z5  Z6  Z7</td>
</tr>
<tr>
<td>DDDT/ZZZZ</td>
<td>1</td>
<td>0xb4 D0  D1  D2  Z4  Z5  Z6  Z7</td>
</tr>
<tr>
<td>DDDD/TZZZ</td>
<td>1</td>
<td>0xcc D0  D1  D2  D3  Z5  Z6  Z7</td>
</tr>
<tr>
<td>DDDD/DTZZ</td>
<td>1</td>
<td>0xd2 D0  D1  D2  D3  D4  Z6  Z7</td>
</tr>
<tr>
<td>DDDD/DDTZ</td>
<td>1</td>
<td>0xe1 D0  D1  D2  D3  D4  D5  Z7</td>
</tr>
<tr>
<td>DDDD/DDDT</td>
<td>1</td>
<td>0xff D0  D1  D2  D3  D4  D5  D6</td>
</tr>
</tbody>
</table>
Control Characters for P1149.10 Unencoded/Scrambled
128/130 encoding should be similar to 64/66

<table>
<thead>
<tr>
<th>Data</th>
<th>K</th>
<th>8b/10b</th>
<th>64/66</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b01 00-0xff</td>
</tr>
<tr>
<td>Character</td>
<td></td>
<td>8b/10b</td>
<td>00-0xff 0xXX 0xXX 0xXX</td>
</tr>
<tr>
<td>K27.7 S (Start)</td>
<td>1</td>
<td>0xFB</td>
<td>0b10 0xFB 0xXX 0xXX 0xXX</td>
</tr>
<tr>
<td>K29.9 T (Terminate)</td>
<td>1</td>
<td>0xFD</td>
<td>0b10 0xFD 0xXX 0xXX 0xXX</td>
</tr>
<tr>
<td>K28.5 K (IDLE)</td>
<td>1</td>
<td>0xBC</td>
<td>0b10 0xBC 0xXX 0xXX 0xXX</td>
</tr>
<tr>
<td>K30.7 E (ERROR)</td>
<td>1</td>
<td>0xFE</td>
<td>0b10 0xFE 0xXX 0xXX 0xXX</td>
</tr>
<tr>
<td>K28.3 A (Stop)</td>
<td>1</td>
<td>0x7C</td>
<td>0b10 0x7C 0xXX 0xXX 0xXX</td>
</tr>
<tr>
<td>K28.0 R (Resume)</td>
<td>1</td>
<td>0x1C</td>
<td>0b10 0x1C 0xXX 0xXX 0xXX</td>
</tr>
</tbody>
</table>