A Training on High-Speed I/O Interfaces: Interlaken Protocol

FTF-NET-F0154

Feras Hamdan

A P R . 2 0 1 4
Bibliography

• Senior Application, R&D, Engineer,
• T4240 Product SME Application Engineer.
• Involved with Interlaken bring up group activities, as full functionality has not been yet fully tested.
• Currently writing a new T4240 Interlaken Application Note.
Agenda

• Interlaken, Introduction.
• Interlaken Protocol Layer.
• Interlaken Framing Layer.
• Interlaken Look Aside Specification.
• Freescale T4240 Interlaken Module.
• References.
• Questions and answers.
Interlaken, Introduction

1. Glossary
2. Interlaken Prior Rivals
3. SPI 4.2 features.
4. XAUI features.
5. Interlaken history
6. Interlaken definition
<table>
<thead>
<tr>
<th><strong>Definitions and Key Variables</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>stMax</strong></td>
<td>Maximum size of a data burst (multiple of 64 bytes)</td>
</tr>
<tr>
<td><strong>BurstShort</strong></td>
<td>Minimum interval between Burst Control Words (minimum value of 32 bytes, incrementing by 8 bytes)</td>
</tr>
<tr>
<td><strong>BurstMin</strong></td>
<td>Parameter to specify the smallest end-of-packet burst</td>
</tr>
<tr>
<td><strong>MetaFrameLength</strong></td>
<td>The quantity of data sent on each lane including one Synchronization Word, one Scrambler State Word, one Diagnostic Word, one or more Skip Words, and the data payload</td>
</tr>
<tr>
<td><strong>Word</strong></td>
<td>An 8-byte quantity, and the fundamental unit of data and control information that is transferred across the interface</td>
</tr>
<tr>
<td><strong>Block Type</strong></td>
<td>The first six bits of each Control Word, used to distinguish different types of Control Words: bits [63:58] for Synchronization, Skip, Scrambler State, and Diagnostic Words, and bit [63] for Burst/Idle Words)</td>
</tr>
<tr>
<td><strong>Burst Control Word</strong></td>
<td>A Control Word with bit 63 = ‘1’ and Type = ‘1’</td>
</tr>
<tr>
<td><strong>Idle Control Word</strong></td>
<td>A Control Word with bit 63 = ‘1’ and Type = ‘0’</td>
</tr>
<tr>
<td><strong>Synchronization Word</strong></td>
<td>A Control Word with Block Type = 0b011110 sent out on all lanes simultaneously with a periodicity of MetaFrameLength, used to synchronize the scrambler and perform lane alignment</td>
</tr>
<tr>
<td><strong>Scrambler State Word</strong></td>
<td>A Control Word with Block Type = 0b001010, sent immediately after the Synchronization Word, used to transmit the current scrambler state to the receiver</td>
</tr>
<tr>
<td><strong>Skip Word</strong></td>
<td>A Control Word with Block Type = 0b000111, used to provide clock compensation for repeater functions</td>
</tr>
<tr>
<td><strong>Diagnostic Word</strong></td>
<td>A Control Word with Block Type = 0b011001, sent immediately preceding the Synchronization Word, used to communicate a per-lane error diagnostic and optional per-lane status</td>
</tr>
<tr>
<td><strong>Lane Skew Tolerance</strong></td>
<td>107 UI</td>
</tr>
</tbody>
</table>
2- Interlaken Prior Rivals, XAUI & SPI 4.2

• The two dominant high-speed chip-to-chip interface protocols for networking applications

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAUI</td>
<td>SPI 4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>narrow 4-lane interface</td>
<td>as a packet-based interface it lacks channelization</td>
<td>channelization</td>
<td>excessive width of the interface limits its scalability,</td>
</tr>
<tr>
<td>long reach</td>
<td>Lacks flow control</td>
<td>programmable burst sizes</td>
<td>the source-synchronous nature of the protocol reduces its effective reach</td>
</tr>
<tr>
<td>suits a variety of implementations: FR4 on PCB, backplanes, and cable</td>
<td>offer only fixed configurations</td>
<td>per-channel backpressure</td>
<td>offer only fixed configurations</td>
</tr>
</tbody>
</table>

![Diagram showing XAUI and SPI 4.2 interfaces](image-url)
3- SPI 4.2 features “Wikipedia”

- **SPI-4.2** is a version of the System Packet Interface published by the Optical Internetworking Forum.
- used in systems that **supports** OC-192 SONET interfaces and is sometimes used in 10 Gigabit Ethernet based systems
- **SPI-4** is an interface for **packet and cell transfer** between a physical layer (PHY) device and a link layer device, like NPU, for aggregate bandwidths of OC-192 Asynchronous Transfer Mode (ATM) and Packet over SONET/SDH (POS), as well as 10 Gigabit Ethernet applications
- SPI-4 uses two types of **transfers** data and control governed by RCTL signal.
- **The physical layer** of SPI-4.2 is very similar to the HyperTransport 1.x interface, although the logical layers are very different.
- The interface consists of (per direction):
  - Sixteen **LVDS pairs** for the data path
  - One LVDS pair for control
  - One LVDS pair for clock at half of the data rate
  - Two FIFO status lines running at 1/8 of the data rate
  - One status clock
    - **(TDCLK, DAT[15:0], TCTL)** and **(RDCLK, RDAT[15:0], RCTL)**.
    - **(TSCLK, TSTAT[1:0])** and **(RSCLK, RSTAT[1:0])**
- The clocking is **source-synchronous** and operates around **700 MHz**.
4- XAUI features “Wikipedia” & “IEEE”

- **XAUI**, the initials of “X for 10 Attachment Unit Interface”, is a standard for extending the XGMII (10 Gigabit Media Independent Interface) between the MAC and PHY layer of 10 Gigabit Ethernet, it is supposed to extend the operational distance between MAC and PHY of the XGMII and to reduce the number of interface signals.
- primarily intended as a chip-to-chip interface implemented with traces on a printed circuit board, allows distances up to approximately 50 cm.
- Independent transmit and receive data paths implemented over Four SerDes lanes, Utilization of 8B/10B coding, conveying the XGMII 32-bit data and control
- Each of the four XGMII lanes is transmitted across one of the four XAUI lanes. (4x32 XGMII -> 4 x 1 SerDes)
- The source XGXS converts XGMII Idle control characters (interframe) into an 8B/10B code sequence.
- The destination XGXS recovers clock and data from each XAUI lane and deskews the four XAUI lanes into the single-clock XGMII.
- The destination XGXS adds to or deletes from the interframe as needed for clock rate disparity compensation prior to converting the interframe code sequence back into XGMII Idle control characters.
5- Interlaken History

- It was invented by Cisco and Cortina in 2006, now a standards consortium which now has many members like Altera, Xilinx, Broadcom (Netlogic/RMI), Cavium, Cortina and Ezchip.

- Goal was to have a viable alternative that address the gap between standards for (parallel backplane, look aside interfaces) and the deficiencies of Ethernet as a viable alternative

- An alliance was formed in 5th November 2007.

- Interlaken V1.2, 7th October 2008, Intended to be a backplane or “Streaming Interface”

- Interlaken LA V1.1, December 2008, Intended to allow NPU’s to connect serially to TCAM, as a packetized interface scalable from 3Gbps to 100Gbps

- It leverages the same underlying PHY, Data Link Layer.

- Interlaken Interoperability Recommendations Revision 1.7 issued in 23rd of May 2013 In order to facilitate interoperability between devices from different vendors for typical applications at data transfer rates of 10 Gb/s, 25 Gb/s, 50 Gb/s,100 Gb/s and 400 Gb/s.
6- Interlaken Definition

• A royalty-free, narrow, high-speed channelized chip-to-chip interconnect protocol for high-bandwidth, 10/100G Ethernet and beyond, and reliable packet transfers.

• It is based on the channelization, 256 channels and up to extended 64k ones, and per-channel flow control features of SPI-4.2, while reducing the number of IC I/O pins by using high-speed SerDes technology.

• Bundles of serial links, although independent from the number of SerDes lanes and SerDes rates, create a logical connection between chips with backpressure capability, out band and in band simple xon/xoff, and data-integrity protection to boost the performance of communications equipment.

• A simple control word structure to delineate packets, similar in function to SPI4.2

• A continuous Meta Frame of programmable frequency to guarantee lane alignment, synchronize the scrambler, perform clock compensation, and indicate lane health.

• 64B/67B data encoding and scrambling with performance that scales with the number of lanes.
Interlaken Protocol Layer

1. Interlaken Rev 1.2 October 7, 2008
2. Transmission Format
3. Data Transmission Procedures
4. Burst Structure
5. Control Word Format
6. Flow Control
There are two fundamental structures {or layers} that define the Interlaken Protocol:

- **Data transmission format** {protocol layer}
- **Meta Frame Format** {Framing Layer}

Data sent across the interface is segmented into **bursts**, which are subsets of the original **packet** data.

Each burst is bounded by **two control words**, one before and one after, and sub-fields within these control words affect either the data following or preceding them for functions like start-of-packet, end-of-packet, error detection, and others.

Each burst is associated with a **logical channel**, which can represent a physical networking port in the system or some other logically connected stream of data.

Packet data is transmitted sequentially by means of one or more bursts, and the **size** of the bursts is a **configurable parameter**.

By segmenting the data into bursts, the interface allows the **interleaving** of data transmissions from different channels for **low-latency** operation.

The **Meta Frame** is defined to support the transmission of the data over a **SerDes** infrastructure.

Metaframe includes set of **four unique control words**, which are defined to provide (1) lane **alignment**, (2) **scrambler initialization**, (3) **clock compensation**, and (4) **diagnostic** functions.

The **Meta Frame** runs **in-band** with the data transmissions, using the specific formatting of the control words to distinguish it from the data.
2- Transmission Format {protocol layer}

- Data is transmitted across the Interlaken interface via a configurable **number of SerDes lanes, 1++**.
- The fundamental unit of data sent across the interface is an **8-byte word** with 64B/67B encoding per lane.
- Data and control words are **striped across the lanes sequentially**, beginning with lane 0, ending at lane M.
- Transport is accomplished via two fundamental word types: **Data Words and Burst/Idle Control Words**, which are distinguished via the 64B/67B framing bits.

The Framing Layer introduces four additional Control Words,
3- Data Transmission Procedures

- The bandwidth of the Interlaken interface is divided into **data bursts from the supported channels**
- Data packets are transferred across the interface by means of **one or more bursts**, with the bursts **delineated by means of one or more Control Words**
- The scheduling logic in the transmitting device is **free to choose the order in which channels are serviced**, subject to the constraint of the flow control state
- Bursts are transmitted on each channel until the packet is completely transferred, at which point a new packet transfer on that channel may begin
4- Burst Structure

1- **BurstMax**: The maximum size of a data burst (a multiple of 64 bytes)

2- **BurstShort**: The minimum size of a data burst (a minimum of 32 bytes, with 8-byte increments). It also guarantees a minimum separation between successive Burst Control Words. The minimum BurstShort interval is 32 bytes, with larger values possible in increments of 8 bytes.

3- **BurstMin**: Defined to be a multiple of 32 bytes, for scheduling algorithms, subject to the constraints that:

   \[ \text{BurstShort} \leq \text{BurstMin} \leq \text{BurstMax}/2 \]

   - **BurstShort** is enforced by adding extra Idle Control Words before the next Burst Control Word
   - Idle Control Word 1 indicates EOP and the appropriate size for the Last Data Word, and the CRC24 of Idle ControlWord 1 covers both the Last Data Word and Idle Control Word 1
## 5- Control Word Format

### Idle/Burst Control Word

<table>
<thead>
<tr>
<th>Bit 66</th>
<th>Inversion Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>‘10’ Framing</td>
</tr>
<tr>
<td>64</td>
<td>‘1’ Control</td>
</tr>
<tr>
<td>63</td>
<td>Type</td>
</tr>
<tr>
<td>62</td>
<td>SOP</td>
</tr>
<tr>
<td>61</td>
<td>EOP_Format</td>
</tr>
<tr>
<td>60</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>Reset Calendar</td>
</tr>
<tr>
<td>56</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td></td>
</tr>
</tbody>
</table>

**In-Band Flow Control**

| 40   |                |
| 39   |                |

**Channel Number**

| 32   |                |
| 31   |                |

**Multiple-Use:**
- Flow Control or
- Channel Number
- Extension

**CRC24**

\[ x^{24} + x^{21} + x^{20} + x^{17} + x^{15} + x^{11} + x^{9} + x^{8} + x^{6} + x^{5} + x + 1 \]

### Framing Layer Control Word

<table>
<thead>
<tr>
<th>Bit 66</th>
<th>Inversion Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>‘10’ Framing</td>
</tr>
<tr>
<td>64</td>
<td>‘0’ Control</td>
</tr>
<tr>
<td>63</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td></td>
</tr>
</tbody>
</table>

**Block Type specific format**

\[ 0 \]
# 5- Control Word Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Position</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inversion</td>
<td>66</td>
<td>Used to indicate whether bits [63:0] have been inversed to limit the running disparity; 1 = inverted, 0 = not inverted</td>
</tr>
<tr>
<td>Framing</td>
<td>65:64</td>
<td>64B/67B mechanism to distinguish control and data words; a ‘01’ indicates data, and a ‘10’ indicates control</td>
</tr>
<tr>
<td>Control</td>
<td>63</td>
<td>If set to ‘1’, this is an Idle or Burst Control Word; if ‘0’, this is a Framing Layer Control Word (see Section 5.4, Framing Layer, on page 26)</td>
</tr>
<tr>
<td>Type</td>
<td>62</td>
<td>If set to a ‘1’, the channel number and SOP fields are valid and a data burst follows this control word (a ‘Burst Control Word’); if set to a ‘0’, the channel number field and SOP fields are invalid and no data follows this control word (an ‘Idle Control Word’)</td>
</tr>
<tr>
<td>SOP</td>
<td>61</td>
<td>Start of Packet. If set to a ‘1’, the data burst following this control word represents the start of a data packet; if set to a ‘0’, a data burst that follows this control word is either the middle or end of a packet</td>
</tr>
<tr>
<td>EOP_Format</td>
<td>80:57</td>
<td>This field refers to the data burst preceding this control word. It is encoded as follows: ‘1xxx’ - End-of-Packet, with bits[59:57] defining the number of valid bytes in the last 8-byte word in the burst. Bits[59:57] are encoded such that ‘000’ means 8 bytes valid, ‘001’ means 1 byte valid, etc., with ‘111’ meaning 7 bytes valid; the valid bytes start with bit position [63:56]. ‘0000’ - no End-of-Packet, no ERR ‘0001’ - Error and End-of-Packet All other combinations are left undefined.</td>
</tr>
<tr>
<td>Reset Calendar</td>
<td>56</td>
<td>If set to a ‘1’, indicates that the in-band flow control status represents the beginning of the channel calendar</td>
</tr>
<tr>
<td>In-Band Flow Control</td>
<td>55:40</td>
<td>The 1-bit flow control status for the current 16 calendar entries; if set to a ‘1’ the channel or channels represented by the calendar entry is XON, if set to a ‘0’ the channel represented by the calendar entry is XOFF</td>
</tr>
<tr>
<td>Channel Number</td>
<td>39:32</td>
<td>The channel associated with the data burst following this control word; set to all zeroes for Idle Control Words</td>
</tr>
<tr>
<td>Multiple-Use</td>
<td>31:24</td>
<td>This field may serve multiple purposes, depending on the application. If additional channels beyond 256 are required, these 8 bits may be used as a Channel Number Extension, representing the 8 least significant bits of the Channel Number. If additional in-band flow control bits are desired, these bits may be used to represent the flow control status for the 8 calendar entries following the 16 calendar entries represented in bits[55:40]. These bits may also be reserved for application-specific purposes beyond the scope of this specification.</td>
</tr>
<tr>
<td>CRC24</td>
<td>23:0</td>
<td>A CRC error check that covers the previous data burst (if any) and this control word</td>
</tr>
</tbody>
</table>
6- Flow Control

A key feature of Interlaken is the ability to communicate per-channel backpressure:

1. an out-of-band flow control interface
2. in-band channel, flow control information uses a simple on-off mechanism to signal permission to transmit on a particular channel.

![Flow Control Diagram](image)
Interlaken Framing Layer

1- Overview
2- 64B/67B Encoding
3- Synchronous scrambler
4- Lane Alignment
5- Diagnostic
6- Clock compensation & Skip
7- Rate Matching
8- Error Conditions
9- Test Patterns
Framing Layer Overview

Overview of Framing Layer

<table>
<thead>
<tr>
<th>Function</th>
<th>Purposes</th>
</tr>
</thead>
<tbody>
<tr>
<td>64B/67B Encoding</td>
<td>Distinguish 8-byte word boundaries;</td>
</tr>
<tr>
<td></td>
<td>Distinguish control and data words;</td>
</tr>
<tr>
<td></td>
<td>Bound the baseline wander</td>
</tr>
<tr>
<td>Synchronous Scrambler</td>
<td>Guarantee bit transition density;</td>
</tr>
<tr>
<td></td>
<td>Eliminate error multiplication</td>
</tr>
<tr>
<td>Lane Alignment</td>
<td>Align all the lanes within a bundle</td>
</tr>
<tr>
<td>Diagnostic</td>
<td>Provide diagnostics and optional per-lane status messaging</td>
</tr>
<tr>
<td>Skip</td>
<td>Compensate for clock differential in an electrical repeater</td>
</tr>
<tr>
<td>Rate Matching</td>
<td>Optimize receiver design by matching the data rate of the interlaked with the data rate of the downstream services</td>
</tr>
</tbody>
</table>

- The framing layer uses Framing Layer Control
- Bits [63:58] are used to distinguish this type of control word, with bit [63] = 0, and bits [62:58] indicating the Block Type
2- 64B/67B Encoding

1- Delineate word boundaries
2- Provide randomness to the EMI generated by the electrical transitions,
3- Allow for clock recovery
4- Maintain DC balance.
5- The encoding protocol selected for Interlaken is a modification of the
   64B/66B used for the IEEE 802.3ae 10 Gigabit Ethernet specification
6- Interlaken inverts the sense of the bits in each transmitted word such
   that the running disparity always stays within a +/- 96-bit bound.
7- Each lane of the bundle maintains a running count of the disparity
   a ‘1’ bit increments the disparity by one, and a ‘0’ bit decrements the
   disparity by one
8- Before transmission, the disparity of the new word is calculated and then
   compared to the current running disparity
9- If the new word and the existing disparity both have the same sign, the bits within the new word are inverted.
10- The 64B/67B encoding creates an overhead of 4.5%.
3- Synchronous Scrambler

- Interlaken employs an independent synchronous scrambler on each lane of the interface.
- The synchronous scrambler does not feed the input data back upon itself; rather each bit is XOR’d with the current state of the scrambler, so no error multiplication may occur.
- The scrambler polynomial is: \( x^{58} + x^{39} + 1 \)
- The scrambler polynomial is activated after device reset, and the transmitter never resets it.
- The current scrambler state is sent to the receiver to allow it to decode the data which follows.
- The scrambler advances and rolls over indefinitely during interface operation, with the exception that it does not advance during the transmission of the unscrambled Synchronization and Scrambler State Words.
- Implementations should initially reset the scrambler to different non zero values on each lane.
- Interlaken synchronizes via the combination of a unique 64-bit Synchronization Word and a Scrambler State Word that are transmitted consecutively as part of the Meta Frame.
- **Because Interlaken** provides for the addition or removal of a Skip Word to manage clock compensation in an electrical repeater, the repeater may need to adjust the position of the Synchronization Word relative to how it was originally transmitted.
- To allow for synchronization at the start of operation and after errors, the Synchronization and Scrambler State Words are transmitted unscrambled.
4- Lane Alignment

- Once the word boundaries are identified and the scrambler properly reset, the lanes of the bundle must be aligned.
- Interlaken guarantees that Synchronization Words are sent across the interface at a fixed frequency to regularly align the datapath SerDes lanes.
- To achieve alignment, the Synchronization Word is transmitted simultaneously across all lanes.
- The receiver identifies synchronization words, measures the skew between them across the lanes of the bundle, and adjusts its internal skew compensation logic accordingly.
- The transmission frequency of Synchronization Words is defined by the MetaFrameLength.

![Interlaken Lane Alignment Segmentation (4-Lane Example)](image)

### Skew Budget

<table>
<thead>
<tr>
<th>Skew Source</th>
<th>Budget (UI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMA Tx</td>
<td>67</td>
</tr>
<tr>
<td>PCB and Medium</td>
<td>40</td>
</tr>
<tr>
<td>Total</td>
<td>107</td>
</tr>
</tbody>
</table>

Interlaken is specified to tolerate a worst-case skew between the individual lanes of a multi-lane interface of 107 UI (unit intervals).

PMA Tx: worst-case implementation would use a 67-bit wide interface into each SerDes lane; if so, the maximum skew between two blocks of SerDes using different PLLs.

PCB & Medium: This parameter should scale with SerDes rate. As the highest performance SerDes used for Interlaken in the medium term is expected to be 6.375 Gbps, old data, or twice the XAUI frequency, then this requirement is double the 20 UI requirement for XAUI, or 40 UI.
5- Lane Diagnostics

- The Diagnostic Word is identified with the Block Type value of 0b011001.
- There are two functions assigned to the Diagnostic Word - a lane Status Message and per-laneCRC32 error detection.
- The Status Message is carried in bits [33:32] of the Diagnostic Word. The format of the message consists of a Status Bit 1 representing the health of this lane, and Status Bit 0 representing the health of the entire interface. A '1' is defined to mean a healthy condition, and a '0' to indicate a problem.
- CRC32 It is calculated over all the words transmitted within the Meta Frame, before scrambling and inversion, except for the 64B/67B framing bits, but including bits [63:0] of the Diagnostic Word itself, with the CRC32 field padded to all zeros.
- The 58-bit scrambler state within the Scrambler State Word is also treated as all zeroes when computing the CRC32. The CRC32 polynomial is: \(x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}+x^{9}+x^{8}+x^{6}+1\)
- Diagnostic Words are counted as part of the MetaFrame_Length just as Synchronization, Scrambler State, Skip, Data, and Burst/Idle Control Words.

![Diagnostic Word Diagram](image1)

![CRC32 Calculation Illustration](image2)
6- Clock Compensation & Skip

- Skip Word enables clock compensation for a repeater function. It is mandatory for receivers to correctly identify and remove them from the received data.

- There can be a slight difference in clock rate on each side of the repeater, and to bridge this gap it is necessary to periodically remove a Skip Word if the second clock is slower than the first, or to add a Skip Word if the second clock is faster than the first.

- A single Skip Word is defined as a required part of the Meta Frame, but additional Skip Words may be added at any point in the Meta Frame, except between the Diagnostic, Synchronization, and Scrambler State Words.
6- Clock Compensation & Skip

- To maintain consistent Meta Frame formatting, in this case the Diagnostic, Synchronization, and Scrambler State Words must be deleted.

- If a Diagnostic Word is deleted, the repeater should transmit a low-pass filtered version of the status message to avoid eliminating any transitory status information.

- If CLK_B in the above example is instead faster than CLK_A, the opposite approach is required - an additional Skip Word is added, and the last Payload Word of the current Meta Frame is shifted into the next Meta Frame.

- Eventually this process requires that a Diagnostic Word be added; in this case the status message should retain the same information as the immediate prior status message.

- If the repeater determines that it needs to discard a word due to a clock difference on only a subset of all the lanes, it shall still discard all the words across the interface simultaneously, not just on the affected lane(s).

- Using a MetaFrameLength of 2K words, at most sixteen bytes, is sent every 16KB, or at a ratio of 1:1,024. A 100ppm differential in clock frequency represents a ratio of 1:10,000, so this Meta Frame frequency meets this compensation requirement 😊. Notice the overhead of the metaframe is small, 4 x 8 /(2K x 8) =0.2%.

- Note that the MetaFrameLength may also be set shorter to enable quicker lane alignment or a smaller quantity of data over which the diagnostic RC is calculate.
7- Rate Matching

- Interlaken defines a mandatory rate matching function for:
  1. Translation applications between Interlaken and an existing protocol such as SPI4.2. So as to reduce the expensive buffering function cost in the bridging device.
  2. To prevent the receiver buffering capacity reduction when the data rate can be guaranteed to be less than the maximum achievable rate.
- Interlaken provides rate matching by offering the ability to insert Idle Control Words into the datapath at a defined frequency to limit the bandwidth of data transferred across the interface.
- The rate matching logic controls the throughput of the interface as a whole, rather than individual channels.

Freescale implementation:
- Credit = Delta x Div(# LA core clock cycles, TIME) - Div(# transmitted bytes, 1 or GRANULARITY if EOP)
- Max Credit = BurstLimit, for example BURSTMAX = 256 B, BurstLimit = 512B, MAX = 512B-256B-96B = 160B
- Delta = Δ <= Max Credit
- \( T_n = \frac{\text{TIME}}{\text{LA-CORE_FREQ}} \)
- Goal Byte rate = \( R_g = \frac{\Delta}{T} = \Delta \times \text{LA-CORE_FREQ} / \text{TIME}. \)
- Line Byte rate = \( R_L \) is the effective transmitted data rate at the output of SerDes lane, given the insertion of idles or output blanking.
- \( C_n \) is the credit value at a timer expiration event when counting down from TIME to zero at LAC clock rate.
- \( C_n = C_{n-1} - \Delta \times \text{step}(C_{n-1}) + R_L T_n \), on the average \( C = 0 \) then \( \Delta = R_L T_n \), or per expected \( R_g = R_L = \frac{\Delta}{T_n} \), notice step(-n)=0, step(0)=step(+n) =1.
- Idle Words must be sent between data bursts, so the rate matching logic shall have a worst-case latency of BurstMax before it can act.
8- Error Conditions

- The Receive SerDes Loses Lock
- The Receive Logic Loses Word Boundary Sync
- Bad Scrambler State
- Lane Alignment Fails
- Burst CRC24 Errors.
- Flow Control Errors
- Unknown Control Word Types
- Bad 64B/67B Code words
- Diagnostic CRC32 Errors

---

### Statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX_Packets</td>
<td>Number of packets received (per channel)</td>
</tr>
<tr>
<td>RX_Bytes</td>
<td>Number of bytes received (per channel)</td>
</tr>
<tr>
<td>TX_Packets</td>
<td>Number of packets transmitted (per channel)</td>
</tr>
<tr>
<td>TX_Bytes</td>
<td>Number of bytes transmitted (per channel)</td>
</tr>
<tr>
<td>RX_Bad_Packets</td>
<td>Number of packets that are errored (i.e., bad CRC, FIFO overflow, ERR bit set, etc.; per channel)</td>
</tr>
<tr>
<td>RX_FIFO_Overflow</td>
<td>Number of packets dropped due to receive FIFO overflow (per channel)</td>
</tr>
<tr>
<td>RX_CRC_Error</td>
<td>Number of bursts with a detected CRC error</td>
</tr>
<tr>
<td>RX_FC_Error</td>
<td>Number of errors detected on the out-of-band flow control interface</td>
</tr>
<tr>
<td>RX_BurstMax_Error</td>
<td>Number of bursts received longer than the BurstMax parameter</td>
</tr>
<tr>
<td>RX_Alignment_Error</td>
<td>Number of alignment sequences received in error (i.e., those that violate the current alignment)</td>
</tr>
<tr>
<td>RX_Alignment_Failure</td>
<td>Number of times alignment was lost (after four consecutive RX_alignment_errors)</td>
</tr>
<tr>
<td>RX_Word.Sync_Error</td>
<td>Number of times a lane lost word boundary synchronization (per lane)</td>
</tr>
<tr>
<td>RX_CDR_Error</td>
<td>Number of times a lane lost clock-data-recovery (per lane)</td>
</tr>
<tr>
<td>RX_Lane_CRC_Error</td>
<td>Number of errors in the lane CRC (per lane)</td>
</tr>
<tr>
<td>RX_Bad_Control_Error</td>
<td>Number of words received with Control Word framing ('x10') that don’t match one of the defined Control Words</td>
</tr>
</tbody>
</table>
3. Test Patterns

- They consist of **two types**: a programmable pattern and a PRBS31/23/7 pattern generator and checker.
- The programmable pattern generator must be capable of storing a set of patterns and repetition values like PatternA * RepetitionA times, followed by PatternB * RepetitionB times.
- The Interlaken controller should support **a minimum of two programmable patterns**, with the pattern length defined by SerDes requirements, and **a minimum 8-bit repetition register per pattern**.
- A programmable pattern check may also optionally be provided, dependent on the particular SerDes test requirements.
- The PRBS pattern generator polynomials supported are:

<table>
<thead>
<tr>
<th>PRBS Polynomials</th>
<th>Name</th>
<th>Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRBS31</td>
<td>$x^2 + x + 1$</td>
<td></td>
</tr>
<tr>
<td>PRBS23</td>
<td>$x^7 + x^6 + 1$</td>
<td></td>
</tr>
<tr>
<td>PRBS7</td>
<td>$x^7 + x^6 + 1$</td>
<td></td>
</tr>
</tbody>
</table>
Interlaken Look Aside Specification

1- Interlaken LA 1.1 protocol definition.
2- Interlaken LA Modifications.
3- Interlaken LA Control Word.
4- Interlaken LA Control Word.
5- Notes.
1- Interlaken Look-Aside 1.1 Dec. 2008 Protocol Definition

- It is suitable for short, transaction-related transfers
- It is not directly compatible with Interlaken and can be considered a different operational mode
- Example of Look-Aside devices: Search engines, Policing engines, Value-add memories, Queuing and scheduling engines.

Interlaken Look-Aside Protocol Modifications

1. Modifications to the control word, including fewer channels and including a protocol type field
2. Mandating the use of packet mode rather than segment mode
3. Mandating the use of single-burst packets for short packets
4. Reducing BurstShort from 32 bytes to as low as 8 bytes
5. Mandating the use of a common reference clock and only one skip word
6. It does not mandate any particular number of lanes for an implementation since the wide variety of LA applications come with their own specific bandwidth requirements.

Interlaken Look-Aside Requirements

- Interlaken Look-Aside receivers must be able to receive messages sent with BurstShort = 16 bytes, but it can receive messages with BurstShort = 8 bytes, if required by the application.

- Interlaken Look-Aside transmitters must be able to transmit messages with BurstShort = 16 bytes. This must be the default condition, also should implement a user-selectable operational mode whereby they can send messages with BurstShort = 8 bytes
2- Interlaken Look-Aside Protocol Modifications

1. Modifications to the control word, including **fewer channels** and including a **protocol type field**

2. Mandating the use of **packet mode** rather than **segment mode**

3. Mandating the use of **single-burst** packets for **short packets**

4. Reducing BurstShort from 32 bytes to as low as 8 bytes

5. Mandating the use of a **common reference** clock and only one skip word

6. It does not mandate any particular number of lanes for an implementation since the wide variety of LA applications come with their own specific bandwidth requirements.

Dual-Mode Requirements

- To support both Interlaken and Interlaken Look-Aside dynamically on a single interface by interpreting the Protocol Type field in the control word:
  - **Receivers** must support the **smallest expected BurstShort** (8, 16 bytes) since the standards Interlaken short is 32 bytes
  - **Transmitters** must use a BurstShort of 16 or 8 bytes for Interlaken Look-Aside frames with 32 bytes in Normal Interlaken
3- Interlaken Look-Aside Control Word

- It contains **a larger number of application-specific bits**. This allows applications to **place user data in the control word**, providing high throughput for the short packets common to look-aside interfaces.

- The Interlaken Look-Aside control word differs from the Interlaken Control Word in the following ways:
  - The most significant channel number bit now carries a **protocol identifier field**
  - The number of **channels** is reduced to **2**, and the otherwise unused channel bits now carry arbitrary user data
  - The **number of flow control bits** is reduced to **two**, and the reset calendar bit is eliminated. These otherwise unused bits now carry arbitrary user data

The figure above shows the message rate for Interlaken and Interlaken Look-Aside when the BurstShort is configured for 8 bytes, and 12 lanes of 6.25 Gbps are aggregated to form a single interface.
<table>
<thead>
<tr>
<th><strong>Framing</strong></th>
<th>66</th>
<th>bits [63:0] have been inverted to limit the running disparity; 1 = inverted, 0 = not inverted.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Control</strong></td>
<td>63</td>
<td>If ‘1’, this is an Idle or Burst Control Word; if ‘0’, this is a Framing Layer Control Word.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>62</td>
<td>If set to ‘1’, the channel number and SOP fields are valid and a data burst follows this control word (a ‘Burst Control Word’); if set to a ‘0’, the channel number field and SOP fields are invalid and no data follows this control word (an ‘Idle Control Word’).</td>
</tr>
<tr>
<td><strong>SOP</strong></td>
<td>61</td>
<td>Start of Packet. If set to a ‘1’, the data burst following this control word represents the start of a data packet; if set to a ‘0’, a data burst that follows this control word is either the middle or end of packet.</td>
</tr>
<tr>
<td><strong>EOP_Format</strong></td>
<td>60:57</td>
<td>‘1xxx’ – End-of-Packet, with bits [59:57] defining the number of valid bytes in the last 8-byte word in the burst. Bits [59:57] are encoded such that ‘000’ means 8 bytes valid, ‘001’ means 1 byte valid, etc., with ‘111’ meaning 7 bytes valid; the valid bytes start with bit position [63:56]. ‘0000’ – no End-of-Packet, no ERR ‘0001’ – Error and End-of-Packet All other combinations are left undefined.</td>
</tr>
<tr>
<td><strong>Application Specific 0</strong></td>
<td>56:42</td>
<td>Application specific data associated with the newly started packet. This field is valid only in Start-of-Packet control word. It is recommended that packet processors and co-processors use this field to carry data to reduce the required number of data words.</td>
</tr>
<tr>
<td><strong>Channel 1 XON</strong></td>
<td>41</td>
<td>Flow control indication for channel 1; 1 = XON, 0 = XOFF.</td>
</tr>
<tr>
<td><strong>Channel 0 XON</strong></td>
<td>40</td>
<td>Flow control indication for channel 0; 1 = XON, 0 = XOFF.</td>
</tr>
<tr>
<td><strong>Protocol Type</strong></td>
<td>39</td>
<td>0 = Interlaken Control Word; 1 = Interlaken Look-Aside Control Word. “It was part of channel number”</td>
</tr>
<tr>
<td><strong>Application Specific 1</strong></td>
<td>38:33</td>
<td>This field is valid only in Start-of-Packet control word. It is recommended that packet processors and co-processors use this field to carry data to reduce the required number of data words.</td>
</tr>
<tr>
<td><strong>Channel Number</strong></td>
<td>32</td>
<td>Used to indicate one of two channel numbers associated with the data burst following the control word.</td>
</tr>
<tr>
<td><strong>Application Specific 2</strong></td>
<td>31:24</td>
<td>This field is valid only in Start-of-Packet control word. It is recommended that packet processors and co-processors use this field to carry data to reduce the required number of data words.</td>
</tr>
<tr>
<td><strong>CRC</strong></td>
<td>23:0</td>
<td>A CRC error check that covers the previous data burst, if any, and this control word.</td>
</tr>
</tbody>
</table>
5- Notes

- **Packet transfer mode considerations (low complexity)**
- Look-aside interfaces support only **packet mode** operation
- In a single segment data, control words (e.g., idles) are **not allowed to interrupt the packet before the EOP control word** terminates the packet, except metaframe where its boundary is ok to interrupt packet transfer
- **Multiplexing data packet traffic and co-processor traffic, dual mode**, on a single interface can be accomplished using the Protocol Type field.
- Because transactions are typically small in size (less than 1024 bits), **interleaving segments from multiple channels has little benefit**
- Maximum Burst size, **BurstMax = 256 Bytes**, can be larger optionally, with segmentation that complies with **BurstMax, MurstShort( 16 bytes)**, and **BurstMin**
- Both the data path device and the look-aside co-processor use the **identical reference clock, which ensures that there is no rate difference between the transmit and receive clocks**, thus there is **no need to send additional skip characters.**

- There is **no need to shift payload** data to maintain the constant length metaframe.

- An Interlaken Look-Aside **metaframe** consists of only **one skip** character, mandated by Interlaken, The **exception** to this section is for **dual-mode** Interlaken transmitters connected to dual-mode Interlaken receivers.
Freescale T4240 Interlaken Module

1- Interlaken Look-Aside Interface.
2- T4240 (LAC) Features
3- Look-Aside Controller Block Diagram.
4- Modes of Operation
5- Interlaken LA controller configuration registers
6- Initialization Information
7- TCAM Usage in Routing Example
8- CAM Theory of Operation
9- Ternary CAM Cell Example.
10- Interlaken Look-Aside TCAM Board
11- Renesas TCAM Riser card Schematics
1- Interlaken Look-Aside Interface

- T4240 as a data path processor, requiring millions of look-ups per second. Expected requirement in edge routers.

- For lowest latency, each (thread) in T4240 will have a portal into the Interlaken Controller, allowing multiple search requests and results to be returned concurrently.

- Interlaken LA is expected to gain traction as an interface to other low latency/minimal data exchange co-processors, such as Traffic Managers.

- PCIe and sRIO protocols are better for higher latency/high bandwidth applications.

- Quad channel implementation (2 standardized)
- Shared but reserved resources per command/response portal
- Prevents inter execution contexts from interfering with each other
- Command/response word defined by software
2- T4240 (LAC) Features:

- Supports Interlaken Look-Aside Protocol definition, rev. 1.1
- Supports 24 partitioned software portals
- Supports in-band per-channel flow control options, with simple xon/xoff semantics
- Freescale supports Interlaken x4 and x8, at 6.25 & 10.3125 Gbps rates
- Ability to disable the connection to individual SerDes lanes
- A continuous Meta Frame of programmable frequency to guarantee lane alignment, synchronize the scrambler, perform clock compensation, and indicate lane health
- 64B/67B data encoding and scrambling
- Programmable BURSTSHORT parameter of 8 or 16 bytes
- Error detection illegal burst sizes, bad 64/67 word type and CRC-24 error
- Error detection on Transmit command programming error
- Built-in statistics counters and error counters
- Dynamic power down of each software portal
3- Look-Aside Controller Block Diagram

- Request Table
  - Dispatch/Decoding
  - OB SW portal
    - OB Buf PTR Queue
      - transmit Buffer Manager
      - OB SRAM
      - ARB
      - Prefetch Buffer
      - Interlaken LA MAC
      - Serdes (TX, RX)
    - IB SW portal
      - IB Buf PTR Queue
        - IB SRAM
        - receive Buffer Manager
4- Modes of Operation

- T4240 LA controller can be either in Stashing mode or non stashing.
- The LAC programming model is based on big Endinan mode, meaning byte 0 on the most significant byte.
- In non Stashing mode software has to issue dcbf each time it reads SWPnRSR and RDY bit
5- interlaken LA controller configuration registers

- 4KBytes hypervisor space 0x0000-0x0FFF
- 4KBytes managing core space 0x1000-0x1FFF
- in compliant with trusted architecture ,LSRER, LBARE, LBAR, LLIODNRn, accessed exclusively in hypervisor mode, reserved in managing core mode.
- Statistics, Lane mapping, Interrupt, rate, metaframe, burst, FIFO, calendar, debug, pattern, Error, Capture Registers
- **LAC software portal memory, n= 0,1,2,3,....23 :**
  - SWPnTCR/ SWPnRCR—software portal 0 transmit/Receive command register
  - SWPnTER/SWPnRER—software portal 0 transmit/Receive error register
  - SWPnTDR/SWPnRDR0,1,2,3 —software portal 0,1,2,3 transmit/Receive data register 0,1,2,3
  - SWPnRSR—software portal receive status register
erlaken LA controller configuration registers/Detailed register descriptions

The LAC transmit status register reports LAC transmit portal status.

Address: 22_8000h base + F00h offset = 22_8F00h

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-7</td>
<td>Reserved</td>
</tr>
<tr>
<td>8-31</td>
<td>Transmit portal busy, each bit corresponds to the status of a single portal, i.e., bit 8 indicating port 23 status, port 9 indicating port 22 status, and so on.</td>
</tr>
<tr>
<td>0</td>
<td>Transmit portal is idle</td>
</tr>
<tr>
<td>1</td>
<td>Transmit portal is processing a packet.</td>
</tr>
</tbody>
</table>

Software Portal transmit command register (LAC_SP_SWPNnTCR)

This register provides transmit control packet content.

<table>
<thead>
<tr>
<th>Field Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>0x0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000</td>
</tr>
</tbody>
</table>

LAC_SP_SWPNnTCR field descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Data is ready to be sent on transmit side</td>
</tr>
<tr>
<td>0000</td>
<td>Data is not ready to be sent</td>
</tr>
<tr>
<td>0000</td>
<td>The data written to transmit data registers is ready to be sent on transmit side</td>
</tr>
<tr>
<td>0000</td>
<td>This field is reserved.</td>
</tr>
<tr>
<td>0000</td>
<td>Reserved.</td>
</tr>
<tr>
<td>0000</td>
<td>The channel number to be sent alone with the interlaken LA control word. Note bit 2 is reserved in 2 channel mode.</td>
</tr>
<tr>
<td>0000</td>
<td>This field is reserved.</td>
</tr>
<tr>
<td>0000</td>
<td>Reserved.</td>
</tr>
<tr>
<td>0000</td>
<td>This field refers to the highest order transmit data register that has been written prior to writing EOP to SWPNnTCR. End of packet with bits [10:15] indicating the number valid bytes in the highest order transmit data register. Bits[10:15] are encoded such that '000000' means all 64 bytes are valid, 000001 means 1 byte valid, etc.</td>
</tr>
<tr>
<td>0000</td>
<td>Reserved.</td>
</tr>
<tr>
<td>0000</td>
<td>Application specific data associated with newly started packet. This field corresponds to application specific 0 in Look-Aside control packet. Note that 6 bit of AppData[0-14] are reserved for PORT ID. The location of the PORT ID is determined according to LMR[PID,LOC] encoding.</td>
</tr>
<tr>
<td>0000</td>
<td>AppData[13:14] are ignored in 4-channel mode.</td>
</tr>
<tr>
<td>0000</td>
<td>Application specific data associated with newly started packet. This field corresponds to application specific 1 in Look-Aside control packet.</td>
</tr>
<tr>
<td>0000</td>
<td>AppData[13:14] are ignored in 4-channel mode.</td>
</tr>
<tr>
<td>0000</td>
<td>Application specific data associated with newly started packet. This field corresponds to application specific 2 in Look-Aside control packet.</td>
</tr>
<tr>
<td>0000</td>
<td>Reserved.</td>
</tr>
<tr>
<td>0000</td>
<td>Transmit data to be sent out on Interlaken links.</td>
</tr>
</tbody>
</table>

Software Portal receive data register 0, 1, 2, 3

This register contains data received on interlaken links. Note that the highest order SWPNnRDR register written may not have all the bytes valid if SWPNnRDR has both RDY and EOP Format set as 1xxxxx. Writes to this software portal register are simply dropped.

LAC_SP_SWPNnRDR 0, 1, 2, 3

<table>
<thead>
<tr>
<th>Field Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>0x0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000</td>
</tr>
</tbody>
</table>

LAC_SP_SWPNnRDR field descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-7</td>
<td>Reserved</td>
</tr>
<tr>
<td>8-31</td>
<td>Data received through Interlaken links.</td>
</tr>
</tbody>
</table>
### 21.4.7 Software Portal receive status register (LAC_SP_SWPnRSR)

All the bits except RDY in SWPnRSR are read-only. GSR, GSR_DONE, SR, OPCU/OPC and DRP can be cleared in through bits in SWPnRCR register.

#### LAC_SP_SWPnRSR field descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Data is ready to be received through software portal registers</td>
</tr>
<tr>
<td>0</td>
<td>Data is not ready to be received</td>
</tr>
<tr>
<td>1</td>
<td>The data written from data registers is ready to be received through software portal registers</td>
</tr>
<tr>
<td>1</td>
<td>This field is reserved.</td>
</tr>
<tr>
<td>2-9</td>
<td>The channel number received alone with the Interlaken LA control word. Note bit 2 is reserved in 2 channel mode.</td>
</tr>
<tr>
<td>4-5</td>
<td>This field is reserved.</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>Global reset is set by software setting LMR[GSR].</td>
</tr>
<tr>
<td>8-15</td>
<td>LAC has completed internal clean up responding to global soft reset request set through LMR[GSR].</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
</tr>
<tr>
<td>17</td>
<td>Outstanding packet counter under run. This is set when outstanding packet counter is cleared by the software, no TX command with GO bit set is sent, yet an Interlaken packet is received by this portal.</td>
</tr>
<tr>
<td>18-20</td>
<td>Reserved</td>
</tr>
<tr>
<td>21-23</td>
<td>Reserved</td>
</tr>
<tr>
<td>24</td>
<td>Packets have been dropped in the corresponding transmit software portal.</td>
</tr>
<tr>
<td>25-27</td>
<td>Reserved</td>
</tr>
<tr>
<td>28-31</td>
<td>Indicates which SWPnRDR registers are valid:</td>
</tr>
<tr>
<td>32-46</td>
<td>Application specific data associated with newly started packet. This field corresponds to application specific 6 in Look-Aside control packet. Note that 8 bits of AppData[0-14] are reserved for PORT ID, the location of the PORT ID is determined according to LMR[PID] encoding.</td>
</tr>
<tr>
<td>47-62</td>
<td>Application specific data associated with newly started packet. This field corresponds to application specific 1 in Look-Aside control packet.</td>
</tr>
<tr>
<td>53-60</td>
<td>Application specific data associated with newly started packet. This field corresponds to application specific 2 in Look-Aside control packet.</td>
</tr>
<tr>
<td>61-127</td>
<td>Reserved</td>
</tr>
<tr>
<td>128-511</td>
<td>Receive data byte 0-47</td>
</tr>
</tbody>
</table>

#### LAC_SP_SWPnRCR field descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Writing 1 to this bit indicating that software intends to clear SWPnRSR[GSR]</td>
</tr>
<tr>
<td>1</td>
<td>Writing 1 to this bit indicating that software intends to clear SWPnRSR[GSR_DONE]</td>
</tr>
<tr>
<td>2</td>
<td>Writing 1 to this bit indicating that software intends to clear SWPnRSR[SR]</td>
</tr>
<tr>
<td>3</td>
<td>Writing 1 to this bit indicating that software intends to clear SWPnRSR[OPCu] and SWPnRSR[OPC]</td>
</tr>
<tr>
<td>4</td>
<td>Writing 1 to this bit indicating that software intends to clear SWPnRSR[DRP]</td>
</tr>
<tr>
<td>5-11</td>
<td>This field is reserved.</td>
</tr>
</tbody>
</table>

#### LAC_LRSR field descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-7</td>
<td>This field is reserved.</td>
</tr>
<tr>
<td>8-31</td>
<td>Receive portal busy, each bit corresponds to a single portal status. i.e. bit 8 indicating port 23 status, port 9 indicating port 22 status, and so on.</td>
</tr>
<tr>
<td>0</td>
<td>Receive portal is idle</td>
</tr>
<tr>
<td>1</td>
<td>Receive portal is processing a packet.</td>
</tr>
</tbody>
</table>
6- Initialization Information

• Global register initialization
  1. Write to LBAER and LBAR to be the same as LAWs registers allocated to LAC memory space. Note that the memory map window has to be 128KB in size.
  2. Clear the error detect register by writing a 1 to the fields.
  3. Write to error interrupt enable register to enable corresponding error interrupt.
  4. Initialize MAC, these registers should not be changed after initialization:
     — 0x408, 0x608 - Lane Disable
     — 0x410, 0x610 - Config Register
     — 0x550-0x55C, 0x750-0x75C - Remap Lane Registers
     — 0x430, 0x630 - Burst Config Register
     — 0x434, 0x634 - MetaFrame Synchronization Register
     — 0x438 - Transmit Rate Limiter Register
     — 0x43C - Transmit Rate Limiter Register
  5. Write to LSRER to enable stashing if stashing is desired
  6. Write to LMR to setup NUM_PORT
  7. Write to LMR to enable MACE

• Software portal initialization
  1. Software issues write 1 to SWPnRCR[SRCLR] clear SWPnRSR[SR].
  2. Software keeps polling SWPnRSR, i.e. read and dcbf to get OPC, Outstanding Packet Counter, until it is 0.
  3. Software writes 0 and dcbf to SWPnRSR.
  4. Software writes 0 to SWPnTER to clear error bit.
  5. Software writes to SWPnTDR and SWPnTCR to start sending out transmit packets.
7- TCAM Usage in Routing Example

**Example Routing Table**

<table>
<thead>
<tr>
<th>Entry No.</th>
<th>Address (Binary)</th>
<th>Output Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101XX</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>0110X</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>011XX</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>10011</td>
<td>D</td>
</tr>
</tbody>
</table>

CAM-based implementation of the routing table of Table
8- CAM Theory of Operation

search data registers/drivers

ML sense amps (MLSAs)

(a)

(b)
9-Ternary CAM Cell Example

1. D and D on the two store elements are not complement to each other.
2. “X” is stored by setting both D and D equal to logic “1”, which disables both pull down paths and forces the cell to match regardless in the inputs.
Renesas 20-Mbit Standard TCAM R8A20410BG

Renesas 20-Mbit QUAD Search Full Ternary CAM is a high-performance search engine targeted for demanding network applications.

- High Speed Search Rate (Max. 360 Msps @ 80, 160, 360-bit)
- Flexible Search Key (40 L/H, 80, 160, 320, 480, 640-bit)
- 576 pin FCBGA, 27 mm × 27 mm body
10- Interlaken Look-Aside TCAM Board: Board Outline

Notes

• Requires I2C access (0x68..0x6F) for initialization
• Requires interrupt connections.

PCI Express Form Factor

- SGMII Ext. (non-standard)
- PEX 8x (standard)
11- Renesas TCAM Riser card Schematics
References
For More Information

- T4240 RM Chapter 22 Interlaken LA and Chapter 19, SerDes Module
- Interlaken_Alliance_Nov_5_PR_final
- Interlaken_Interop_Recommendations_v1.7
- Interlaken.Look_Aside_Protocol_Definition_v1.1
- Interlaken_Protocol_Definition_v1.2
- Interlaken_Protocol_Definition_v1.2 CHANGE_BARS
- Interlaken_Retransmit_Extension_v1.2
- Preliminary Renesas 5M-bit Serial TCAM4 Functional Specification
- Broadcom NL1100_NLP_USR_GUIDE
Any Interlaken Related Question?
Introducing The QorIQ LS2 Family

**Breakthrough, software-defined approach to advance the world’s new virtualized networks**

**New, high-performance architecture built with ease-of-use in mind**
Groundbreaking, flexible architecture that abstracts hardware complexity and enables customers to focus their resources on innovation at the application level.

**Optimized for software-defined networking applications**
Balanced integration of CPU performance with network I/O and C-programmable datapath acceleration that is right-sized (power/performance/cost) to deliver advanced SoC technology for the SDN era.

**Extending the industry’s broadest portfolio of 64-bit multicore SoCs**
Built on the ARM® Cortex®-A57 architecture with integrated L2 switch enabling interconnect and peripherals to provide a complete system-on-chip solution.
QorIQ LS2 Family
Key Features

SDN/NFV Switching

Data Center

Wireless Access

High performance cores with leading interconnect and memory bandwidth
- 8x ARM Cortex-A57 cores, 2.0GHz, 4MB L2 cache, w Neon SIMD
- 1MB L3 platform cache w/ECC
- 2x 64b DDR4 up to 2.4GT/s

A high performance datapath designed with software developers in mind
- New datapath hardware and abstracted acceleration that is called via standard Linux objects
- 40 Gbps Packet processing performance with 20Gbps acceleration (crypto, Pattern Match/RegEx, Data Compression)
- Management complex provides all init/setup/teardown tasks

Leading network I/O integration
- 8x1/10GbE + 8x1G, MACSec on up to 4x 1/10GbE
- Integrated L2 switching capability for cost savings
- 4 PCIe Gen3 controllers, 1 with SR-IOV support
- 2 x SATA 3.0, 2 x USB 3.0 with PHY

Unprecedented performance and ease of use for smarter, more capable networks
See the LS2 Family First in the Tech Lab!

4 new demos built on QorIQ LS2 processors:

- Performance Analysis Made Easy
- Leave the Packet Processing To Us
- Combining Ease of Use with Performance
- Tools for Every Step of Your Design