Interlaken is a royalty-free interconnect protocol that was developed by Cisco Systems and Cortina Systems in 2006. The full Interlaken protocol (described in the Interlaken Protocol Specification, v1.2) was designed to support chip-to-chip packet transfers in high-bandwidth networking equipment. A full Interlaken solution consists of the Interlaken Protocol Layer which runs on top of the Interlaken Framing Layer. The full Interlaken protocol is often viewed as too complex for designers who are looking for a lightweight, low-overhead way to connect multiple FPGAs, systems, or sensors. For these cases, the Interlaken Framing Layer gives us everything we need for an extremely scalable, high-bandwidth, low-overhead and lightweight FPGA interconnect that supports any FPGA device family all with a common user interface. The Interlaken Framing Layer is combined with the FPGA transceiver to form a complete IP core that we call the StreamDSP Interlaken-PHY IP Core.

It's important to note that the Interlaken-PHY IP core is NOT a full Interlaken Protocol implementation because it only implements the Framing Layer of the Interlaken specification. The Interlaken-PHY IP Core gives users an extremely easy and efficient way to achieve scalable, high-bandwidth, low-overhead connections between FPGAs, systems, and sensors. The Interlaken-PHY IP Core provides a simple, parallel, streaming data interface to the user and does not support data packetization, virtual channels, framing, flow control, or other features found in full-featured protocols. The user interfaces with customizable-depth FIFOs on the input and output of the core. On the transmit side, the user is responsible for not allowing the TX FIFO to empty by writing data to it at a rate greater than or equal to (Line_Rate/67). If there is no data to send, the controlIn input bit can be used to write “Empty” control words which can then be ignored at the destination. On the receive side, the user is responsible for not allowing the RX FIFO to overflow by reading it at a fast enough rate which also defined as greater than or equal to (Line_Rate/67). For a more detailed explanation of the user interface, please see the complete StreamDSP Interlaken-PHY User Guide.

StreamDSP is committed to performance, efficiency, and flexibility, and we’re continually updating the core to support new transceiver based devices offered by both Altera and Xilinx. A complete reference design is provided for each family, as well as a self-verifying testbench. In addition, our testing procedure includes exhaustive Altera <> Xilinx interoperability testing to ensure compatibility.

StreamDSP is committed to delivering the highest level of customer support to ensure a smooth system integration. We also offer IP core customization and FPGA design services.
IP Datasheet

Details

FPGA Family Support

**ALTERA/INTEL**
- Stratix-IV GX
- Stratix-V GX
- Arria-V GX
- Arria-10 GX
- Stratix-10 GX L-Tile
- Stratix-10 GX H-Tile

**Example Design**
- Altera Stratix-IV GX PCIe Dev Kit
- Altera Stratix-V GX PCIe Dev Kit
- Altera Arria-V GX PCIe Dev Kit
- Altera Arria-10 GX PCIe Dev Kit
- Intel Stratix-10 GX PCIe L-Tile Dev Kit
- Intel Stratix-10 GX PCIe H-Tile Dev Kit

**XILINX**
- Kintex-7
- Artix-7
- Virtex-6 GTX
- Virtex-7 GTX
- Virtex-7 GTH
- Kintex UltraScale
- Virtex UltraScale
- Virtex UltraScale+
- Zynq UltraScale+

**Example Design**
- Xilinx KC705 Evaluation Kit
- Xilinx AC701 Evaluation Kit
- Xilinx ML605 Evaluation Kit
- Xilinx VC707 Evaluation Kit
- Xilinx VC709 Evaluation Kit
- Xilinx KCU105 Evaluation Kit
- Xilinx VCU108 Evaluation Kit
- Xilinx VCU118 Evaluation Kit
- Xilinx ZCU102 Evaluation Kit

All deliveries include VHDL and Verilog simulation models, a self-checking testbench with simulation scripts, and ready-to-run design targeted at a popular development board for each family (listed above).

---

**Resource Usage**

<table>
<thead>
<tr>
<th>Registers</th>
<th>LUTs</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1803</td>
<td>1891</td>
<td>4 Blocks</td>
</tr>
</tbody>
</table>

* single-lane application. Resources scale linearly with number of lanes

**Throughput Examples**

1 lane @ 2.5 Gbps = 296 MB/s
4 lanes @ 5.0 Gbps = 2375 MB/s
10 lanes @ 14.1 Gbps = 16,743 MB/s

**Delivery Options**

HDL Language: Verilog
License Types:
- Netlist
- Source Code

* Free, supported evaluations available on request