Intel FPGA Streaming Video Protocol Specification
## Contents

About the Intel FPGA Streaming Video Protocol.................................................................3
  AXI4-Stream Protocol Signals..........................................................................................3
  Ready and Valid Behavior...............................................................................................4
  Data Exchange..............................................................................................................4
Intel FPGA Streaming Video Protocol..................................................................................5
  TDATA Format..............................................................................................................7
  Reset.............................................................................................................................8
  Intel FPGA Streaming Video Errors............................................................................8

Document Revision History for Intel FPGA Streaming Video Protocol.........................9
About the Intel FPGA Streaming Video Protocol

This protocol allows frequencies of 600 MHz for video processing pipelines supporting up to four pixels per clock cycle in parallel for Intel® Agilex™ FPGAs. Processing video streams with four pixels in parallel at 600 MHz allows 8K video processing at 60 frames per second. 8 pixels in parallel allows 8Kp120 or 16Kp30. On slower devices 6 pixels in parallel operating at 450 MHz also allows 8K60 video processing.

An AMBA AXI4-Stream protocol underpins this architecture, which meets the needs of video processing IP, allows easy interoperability with the latest Intel FPGA IP, and other third-party IP and allows a high degree of hyper-pipelining.

The AMBA AXI4-Stream protocol is natively supported in Platform Designer, allowing you to easily make connections between components.

All of the information regarding the size of each video frame, color space, interlacing, and any control of video processing IPs is communicated via the Avalon memory-mapped control interface. Only the AXI4-Stream bus contains video data.

The protocol allows interfaces to Intel FPGA video IPs or other AXI4-Stream compliant third-party video IPs.

Related Information
AMBA 4 AXI4-Stream Protocol

AXI4-Stream Protocol Signals

The Intel FPGA streaming video protocol allows you to transfer a variety of video data following the AXI4-Stream protocol signals.

Table 1. AXI4-Stream Protocol Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDATA</td>
<td>Set TDATA width according to need. The minimum allowable width of TDATA on all IP interfaces is 8 bits. The width of TDATA is byte aligned (i.e. multiple of 8). Systems that require smaller TDATA interfaces must pad their data. The TKEEP and TSTRB signals are unused, so every byte of TDATA is valid (no empty pixels). You deduce the exact length of a video packet from the image width. For other data, the components using the data must use the same convention for unused bytes.</td>
</tr>
<tr>
<td>TUSER[n:0]</td>
<td>Sized as 1 bit per byte of TDATA. The LSB of TUSER is strobed high for the first pixel of the first of a frame (or interlaced field) of video. The other bits of TUSER are unused.</td>
</tr>
<tr>
<td>TLAST</td>
<td>The protocol transmits each line of video as an AXI4-S packet. TLAST strobes high to indicate the last beat of the packet. When transmitting only one pixel per clock TLAST is the last pixel of the line.</td>
</tr>
<tr>
<td>TREady and TVALID</td>
<td>Indicate the valid cycles on the bus.</td>
</tr>
</tbody>
</table>
Ready and Valid Behavior

The AXI specification states: a source is not permitted to wait until \texttt{TREADY} is asserted before asserting \texttt{TVALID}. Intel mandates this behaviour for the external interfaces of video and vision processing IP components. Video and vision processing IP sinks must raise \texttt{TREADY} independently of whether the input \texttt{TVALID} is asserted. This behavior ensures that if a third party IP drives a video and vision processing IP sink and the third party IP does not respect this AXI rule for sources, the video pipe still operates correctly.

Related Information

AMBA 4 AXI4-Stream Protocol

Data Exchange

The \texttt{TVALID} and \texttt{TREADY} handshake determines when information passes across the AXI4-Stream. A two-way flow control mechanism enables both the input and output interfaces to control the rate at which the data and control information transmits across the interface. For a transfer to occur, assert both the \texttt{TVALID} and \texttt{TREADY} signals. You can assert either \texttt{TVALID} or \texttt{TREADY} first or both in the same clock cycle. An output interface must not wait until you assert \texttt{TREADY} before asserting \texttt{TVALID}. When you assert \texttt{TVALID}, keep it asserted until the handshake occurs. Other signals (\texttt{TDATA}, \texttt{TUSER}) must remain stable. An input interface can wait for \texttt{TVALID} to be asserted before asserting the corresponding \texttt{TREADY}. If an input interface asserts \texttt{TREADY}, it can deassert \texttt{TREADY} before \texttt{TVALID} is asserted.

A \texttt{TREADY} and \texttt{TVALID} handshake can occur when:

- \texttt{TVALID} is asserted before \texttt{TREADY}
- \texttt{TREADY} is asserted before \texttt{TVALID}
- \texttt{TREADY} and \texttt{TVALID} asserted at the same time

The figures do not show \texttt{TUSER} or \texttt{TLAST} but these signals also follow the same rules for validity as shown for \texttt{TDATA}.

Figure 1. **AXI4-Stream Handshake: TVALID before TREADY**

The figure shows the output interface presents the data and asserts the \texttt{TVALID} signal active-high. When the output interface asserts \texttt{TVALID}, the data from the output interface must remain unchanged until the input interface drives the \texttt{TREADY} signal active-high, indicating that it can accept the data. In this case, transfer takes place when the input interface asserts \texttt{TREADY} active-high.
Figure 2. **AXI4-Stream Handshake: TREADY before TVALID**
The figure shows the input interface drives TREADY active-high before the data and control information is valid. The destination can accept the data and control information in a single clock cycle. In this case, transfer takes place when the output interface asserts TVALID active-high.

![AXI4-Stream Handshake: TREADY before TVALID](image)

Figure 3. **AXI4-Stream Handshake: TVALID and TREADY asserted at the same time**
The figure shows that both, output and input interfaces assert TVALID and TREADY active-high in the same clock cycle, respectively. In this case, data transfer takes place in the same cycle.

![AXI4-Stream Handshake: TVALID and TREADY asserted at the same time](image)

**Intel FPGA Streaming Video Protocol**

The protocol runs on top of the AXI4-Stream wire-level protocol, with extensions for transporting video data.

The **TDATA** width depends on the color depth, color space, and number of pixels in parallel.

The width of a pixel, in bytes, is the number of bits per color sample or color depth (BPS), multiplied by the number of color channels per pixel (NC), rounded to the next byte and at least two bytes.

- **TDATA** byte width = max(2, ceil ((BPS × NC)/8)) × pixels in parallel

The protocol moves color planes in parallel, with up to 8 pixels in parallel in one beat of data. You can specify any integer number of pixels.

The protocol supports video frames (or fields) as small as 1 pixel by 1 pixel and as large as 65536 by 65536 pixels. The protocol does not support zero-pixel lines or frames.
Figure 4. Data packet – first line of video (1x30bit pixel interface)

The TUSER bus indicates the start of each frame. The TUSER bus must be TDATA/8 bits wide (and at least 2 bits wide to match the minimum TDATA width of 16 bits). The start-of-frame flag is bit 0 of the TUSER bus when driving the first pixel of a line. Drive this flag to 1 on the first pixel of a line to indicate that the line is the first line of a frame. Keep the TUSER bus at 0 in all other circumstances.

TUSER is 4 bits wide, as the 30 bits of pixel data occupy 4 bytes of TDATA. In the first beat of the transfer, TUSER indicates a new frame of video by asserting TUSER[0] to 1 (other bits, e.g. TUSER[3:1], can take any value). TLAST indicates the last pixel in each line.

Figure 5. Data Packet – last line of video

You cannot tell which is the last line of video of the frame. TLAST indicates the last set of pixels for that line (D7 in this figure).

Figure 6. Example of TUSER and TLAST transactions

The figure shows an example of how TUSER and TLAST signals identify the start of frame and end of line within a video frame.

TDATA Packing

Adjust pixel bits across the AXI bytes. When a pixel does not perfectly fill a given number of bytes, pad MSBs with undefined data. For less than 16 bits of pixel data (for example single pixel 10 bit mono video), pad bits from the MSB of pixel data to bit 15 with undefined data.
Figure 7. **Data Packet – color plane packing**

The figure shows the color ordering for both RGB data and YcbCr. Pixels in parallel = 1; NC = 3, BPS = 10

![Color plane packing](image)

Cr by definition is the red-chroma information of a YUV stream, which aligns to the red channel information for RGB video. With this ordering, if a YUV image displays accidentally on an RGB monitor, the colors appear almost correctly.

The luma component aligns with the RGB green, as it is the main luma contributor for some algorithms.

Often in math algorithms green is considered the primary luma contributor when determining brightness of an RGB stream, so keeping these busses aligned is critical.

Multiple pixels are packed with byte alignment at pixel boundaries.

Figure 8. **Data Packet – pixel packing example**

Four 30-bit pixels packed into 16 bytes

![Pixel packing example](image)

**TDATA Format**

You provide and receive video data to the IP in RGB and CMYK format via the AXI4-Stream video. The IP packs pixels across the data bus bytes. When a pixel does not perfectly fill a given number of bytes, the IP pads MSBs with undefined data.

Figure 9. **TDATA RGB Layout: One pixel per clock configuration**

![TDATA RGB Layout](image)

The IPs also support multiple pixels in parallel. Multiple pixels pack with byte alignment at pixel boundaries. The following figures show how pixels pack on the TDATA bus for two or four pixels in parallel, respectively.

Figure 10. **TDATA RGB Layout: Two Pixels per Clock Configuration**

![TDATA RGB Layout](image)
Reset

Video and vision processing IP components employ a synchronous reset and system resets must have a minimum duration of 256 clock cycles. In accordance with the AXI specification, all \texttt{TVALID} signals from components drive low during reset and for at least one cycle after reset is deasserted.

Intel FPGA Streaming Video Errors

Errors can be low-level violations or high-level streaming errors.

AXI4-Stream low-level protocol violations (such as \texttt{TLAST} stuck at zero \texttt{TVALID} or \texttt{TREADY} handshake fault) result in system failure and possibly lockup.

High level streaming errors can be:

- Early and late \texttt{TLAST} signalling at the end of packets on data packets, where a line is shorter or longer than expected
- Early or late \texttt{TUSER[0]} marking a start of frame, where a frame contains less or more lines than expected

These high-level errors are not protocol violations and are expected in a running system. Therefore, components can accept invalid video fields, but they may also generate them without breaking the specification.
## Document Revision History for Intel FPGA Streaming Video Protocol Specification

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021.08.02</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>