1. **AXI Stream**

- The AXI-Stream protocol is used as a standard interface to exchange data between connected components. AXI-Stream is a point-to-point protocol, connecting a single Transmitter and a single Receiver.

- Some data stream forms that AXI Stream use: Byte stream (transmitting data bytes and null bytes), Continuous aligned stream (CAS) (transmitting data bytes), Continuous unaligned stream (CUS) (no position bytes between the first data byte and the last data byte of each packets) and Sparse stream (transmitting data bytes and position bytes, most of them are data bytes, the position in packets are fixed).

- **Packets** are a group of bytes that are transported together across an AXI-Stream interface. A packet can consist of a single transfer or multiple transfers. Interconnect components can use packets to deal more efficiently with a stream in packet-sized groups. A packet is similar to an AXI burst.

1. **AXI-Stream signals:**

Required:

- TVALID: source from Transmitter

Optional:

- TREADY: highly recommended, source from Receiver to accept transfer

- TDATA: provide data that is passing across the interface, source from Transmitter

- TKEEP, TSTRB: byte qualifier for data in TDATA

- TLAST: boundary of a packet, source from Transmitter

- TID: stream identifier, source from Transmitter

- TDEST: provide routing info

- TUSER: any sideband user-defined data along with data stream

- ACLK: global clock signal, active high

- ARESETn: global reset signal

*\*A transfer takes place when both TVALID and TREADY are asserted.*

1. **Handshake signaling between TVALID and TREADY:**

2 way flow control mechanism enables Transmitter and Receiver to control the rate at which the data and control information is transmitted across the interface.

1. **Byte location:**

- Byte n is in transfer t: t = INT(n/w) = floor(n/w)

- Position b of n: b = n-(t\*w)

- Index in TDATA: TDATA[(8b+7):8b]

Where w is width of data bus

1. **Adaptation:**

An adaptable data width can be configured to match the system in which a component is integrated (merged, packed, downsizing or upsizing).

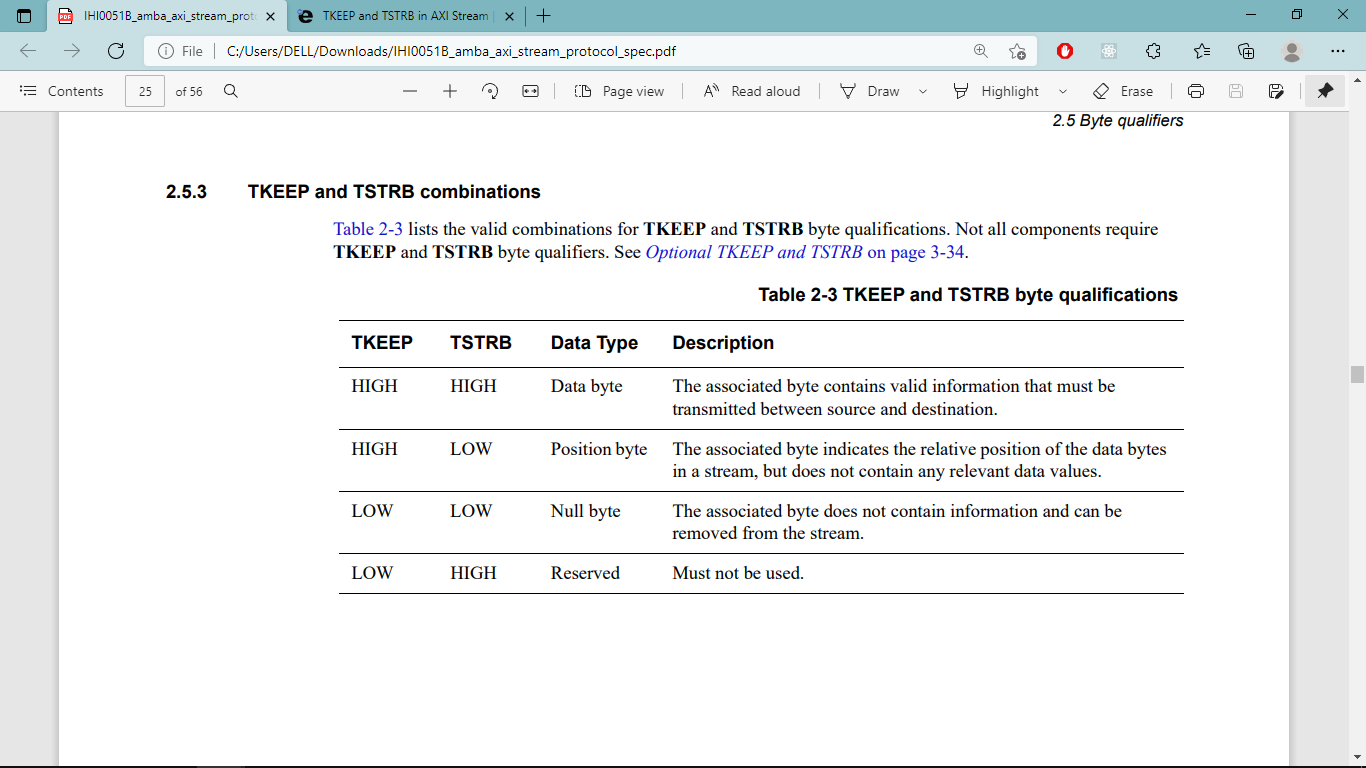
1. **Byte qualifiers:**

- TKEEP: When TKEEP is asserted, it indicates that the associated byte must be transmitted to the destination. When TKEEP is deasserted, it indicates a null byte that can be removed from the stream. A transfer is permitted with all TKEEP bits deasserted.

- TSTRB: When TSTRB is asserted, it indicates that the associated byte contains valid information and is a data byte. When TSTRB is deasserted, it indicates that the associated byte does not contain valid information and is a position byte.

\* TKEEP[x] is associated with TDATA[(8x+7):8x]

\* TSTRB[x] is associated with TDATA[(8x+7):8x



1. **Boundary of packets:**

- When asserted, TLAST indicates packet boundary or an efficient point to make an arbitration change on a shared link. When deasserted, TLAST indicates that another transfer can follow. This means it is acceptable to delay the current transfer for the purpose of upsizing, downsizing, or merging.

- The start of a packet is determined as:

- The first occurrence of a TID and TDEST pair after reset

- The first transfer after the end of the preceding packet for any unique set of TID and TDEST values

- In case lack of data or position bytes, TLAST is used to Indicate the end of a packet when there are no more data or position bytes to transmit; The transfer can be merged with earlier one if both have same TID and TDEST or if earlier transfer have TLAST deasserted.

1. **Source and destination signaling:**

- TID Provides a stream identifier that can be used to differentiate between multiple streams of data that are being transferred across the same interface.

- TDEST Provides coarse routing information for the data stream.

- Transfers that have the same TID and TDEST values are from the same stream. Merging of transfers belonging to different streams is not permitted.

1. **Clock and Reset:**

- A single clock signal, ACLK, is used by each component. Input signals are sampled on the rising edge and output signal changes must occur after the rising edge of ACLK.

- ARESETn is a single, active-LOW reset signal. The reset signal can be asserted asynchronously, but deassertion **must be synchronous after** the rising edge of ACLK.

- A Transmitter interface must only begin driving TVALID at a first rising ACLK edge **after a rising edge at which ARESETn is asserted HIGH**.

1. **User signaling:**

- Sideband signaling can be used for data byte-based, transfer-based, packet-based, or frame-based information

- Specific uses:

- Marking the location or type of special data items

- Providing ancillary information that must accompany the data, such as control signals, and flags

- Identifying segments of a packet

- The location of the User signals for byte x: TUSER[((x\*m)+(m-1)):(x\*m)]

Where w is width of the interface

m is number of User signals

u is number of User bits (u = m\*w)

x = 0 -> w-1

- The transfer of TUSER bits is not required or guaranteed when the associated TKEEP is deasserted LOW

1. **Compatibility:**

- Transmitter compatibility: The data width of the interfaces must be the same for a Transmitter and Receiver interface to be compatible. Any Transmitter component that supports TREADY can be made interface compatible with any Receiver component that supports the full feature set.

- Receiver compatibility:

- Data width of the two interfaces must be the same

- Receivers are not required to support null or position bytes

- Interconnect compatibility: The interconnect is required to reliably transport all data bytes and position bytes from the Transmitter to the Receiver.

- Continuous packets:

- **True** Only continuous packets are supported

\* Transfer interleaving for different packets must not occur. This means that when TLAST is LOW, TID and TDEST must not change in the next transfer

\* TSTRB is not present. Position bytes are not supported

\* There must not be null bytes within a packet. This means:

+ If TLAST is LOW, all bits of TKEEP must be HIGH.

+ If TLAST is HIGH, null bytes are only permitted in the bytes above all the data bytes

- **False** Packets are not constrained to be continuous. If Continuous\_Packets is not declared, it is considered to be False

1. **Transfer Interleaving and Ordering:**
2. **Transfer interleaving:**

- It’s the process of interleaving transfers from different streams on a transfer-by-transfer basis. A Transmitter can interleave streams at source and an interconnect can interleave streams from multiple Transmitters at a point of convergence. Receivers can be designed to accept any number of interleaved streams or a limited number of interleaved streams.

- A Receiver, with the Continuous\_Packets property set to True, cannot accept interleaved streams

- One of the following techniques must be used to ensure a Receiver capabilities are not exceeded:

- The Receiver can be accessed by just one Transmitter, which must not exceed the interleaving capability of the Receiver

- The Receiver is accessed by multiple Transmitters, each of which does not interleave packets

- The Receiver is accessed by multiple Transmitters and a higher-level control mechanism ensures that the overall interleaving capability of the Receiver is not exceeded

1. **Transfer ordering:**

- The AXI-Stream protocol requires that all transfers remain ordered. The reordering of transfers is not permitted

- The stream interleaving observed by a Receiver is not increased.

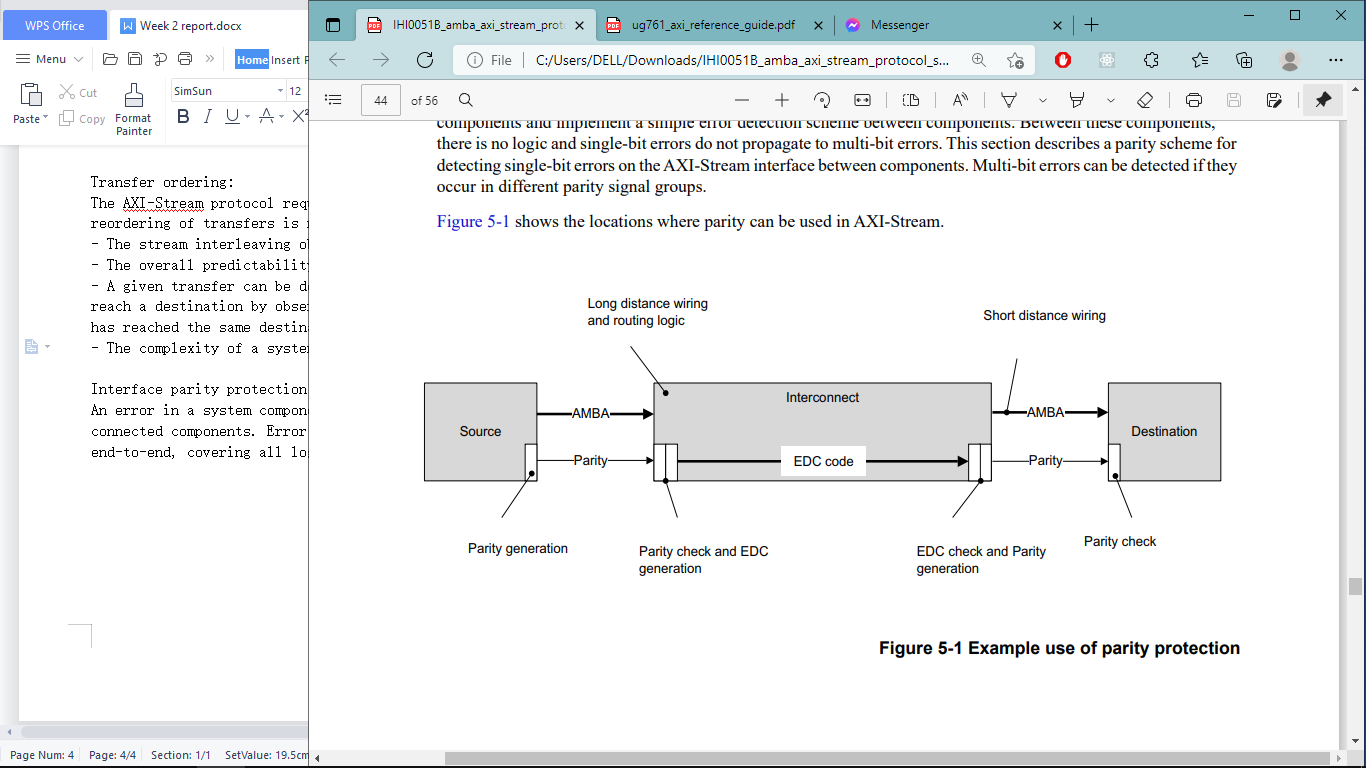
- The overall predictability of the system is improved

- A given transfer can be determined, independent of the TID of the transfers, to reach a destination by observing that a later transfer from the same Transmitter has reached the same destination

- The complexity of a system is reduced

1. **Interface parity protection:**

- An error in a system component can propagate and cause numerous errors across connected components. Error Detection and Correction (EDC) is required to operate end-to-end, covering all logic and wires from source to destination



1. **Configuration:**

- The EDC scheme is defined by the Check\_Type property. The following Check\_Type values are defined:

- **False** There are no checking signals on the AXI-Stream interface

- **Odd\_Parity\_Byte\_All** Odd parity checking is included for all signals. Each bit of the parity signal covers up to 8 bits.

- Some common attributes to check:

- Odd parity is used. Odd parity means that there is always an odd number of bits asserted across the interface signal and check signal

- Each parity check bit covers no more than 8 bits of payload. This limitation assumes that there is a maximum of three logic levels available in the timing allowance for generating each parity bit

- For a check signal that is wider than 1 bit:

\* Check bit [n] corresponds to [(8n+7):8n] in the associated signal

\* If the associated signal is not an integer number of bytes, the most significant bit of the check signal covers fewer than 8-bits in the most significant portion of the associated signal.

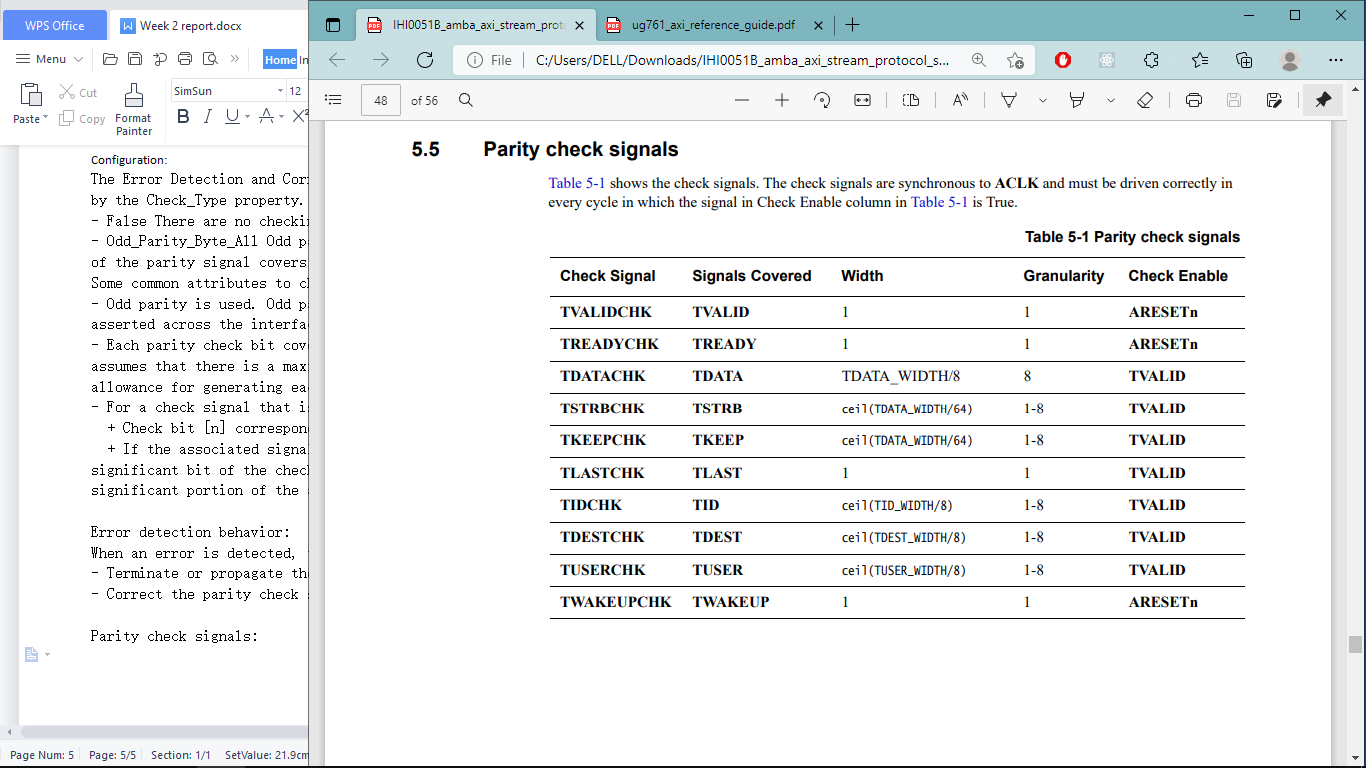
1. **Error detection behavior:**

- When an error is detected, the Receiver can:

- Terminate or propagate the transfer

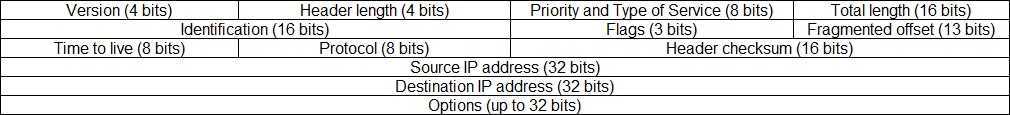
- Correct the parity check signal or propagate the error

1. **Parity check signals:**



1. **IP Header**

- An **IP header** is a prefix to an IP packet that contains information about the IP version, length of the packet, protocol, source and destination IP addresses, etc. It consists of the following fields:



**- Version:** the version of the IP protocol.

**- Header length:** the length of the header in 32-bit words. The minumum value is 20 bytes, and the maximum value is 60 bytes.

**- Priority and Type of Service:** specifies how the datagram should be handled. The first 3 bits are the priority bits.

**- Total length:** the length of the entire packet (header + data). The minimum length is 20 bytes, and the maximum is 65,535 bytes.

**- Identification:** used to differentiate fragmented packets from different datagrams.

**- Flags:** used to control or identify fragments.

**- Fragmented offset:** used for fragmentation and reassembly if the packet is too large to put in a frame.

**- Time to live:** limits a datagram’s lifetime. If the packet doesn’t get to its destination before the TTL expires, it is discarded.

**- Protocol:** defines the protocol used in the data portion of the IP datagram. For example, TCP is represented by the number 6 and UDP by 17.

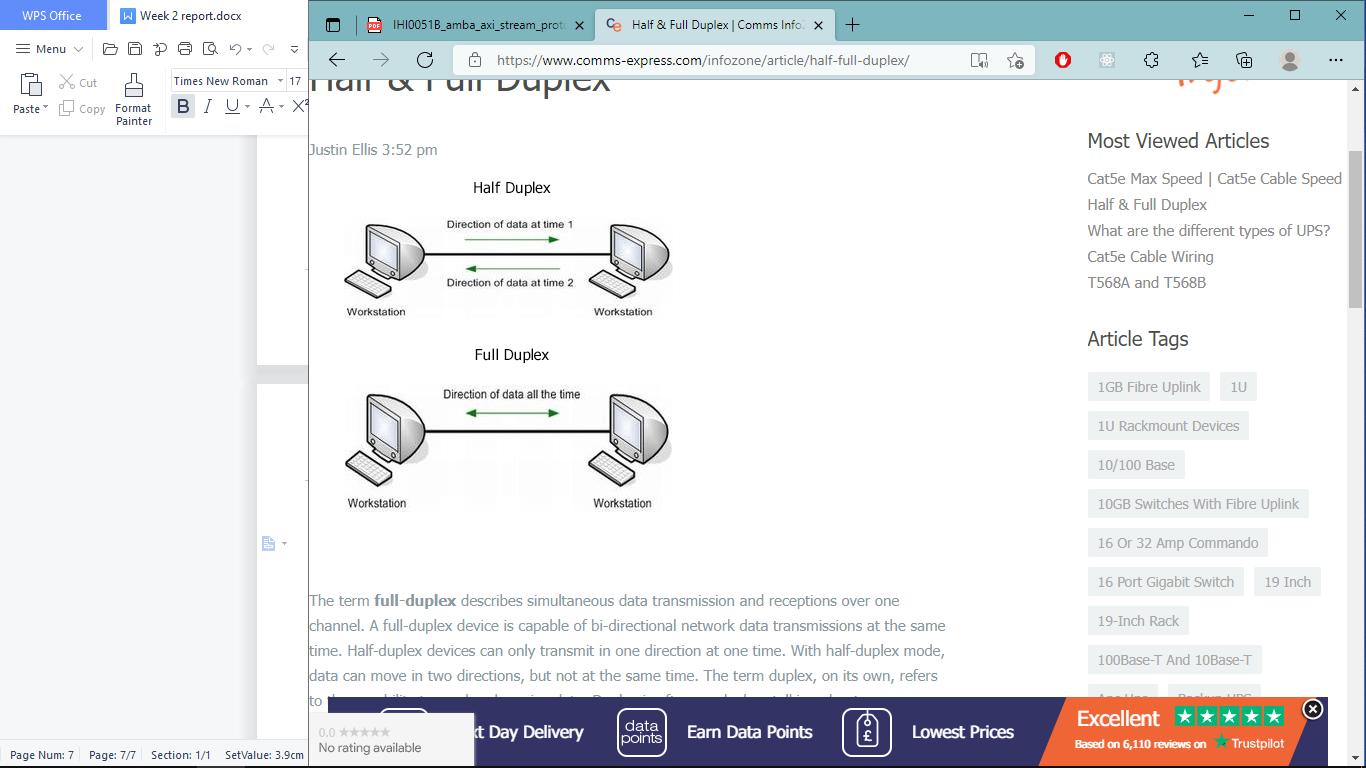
**- Header checksum:** used for error-checking of the header. If a packet arrives at a router and the router calculates a different checksum than the one specified in this field, the packet will be discarded.

**- Source IP address:** the IP address of the host that sent the packet.

**- Destination IP address:** the IP address of the host that should receive the packet.

**- Options:** used for network testing, debugging, security, and more. This field is usually empty.

1. **Full Duplex**



**What is Full-Duplex:**

- The term ****full-duplex**** describes simultaneous data transmission and receptions over one channel. In full-duplex mode, both stations can transmit and receive simultaneously. In full\_duplex mode, signals going in one direction share the capacity of the link with signals going in another direction, this sharing can occur in two ways:

- Either the link must contain two physically separate transmission paths, one for sending and the other for receiving.

- Or the capacity is divided between signals travelling in both directions.

**How it works:**

- ****Full/half-duplex device**** allows users to choose either full or half-duplex modes. With half-or-full duplex devices, such as [modems](https://www.comms-express.com/categories/routers-modems/" \t "https://www.comms-express.com/infozone/article/half-full-duplex/_blank), a switch will be set to either full or half mode. When set to a half-duplex system, connections alternate use of the communication channel and hardware can determine the time each data link in the system is allotted for transmissions.

- In full duplex mode, Sender can send the data and also can receive the data simultaneously.Telephones are common examples of full-duplex devices. They allow both people to hear each other at the same time. In the computer world, most [network](https://techterms.com/definition/network) protocols are duplex, enabling hardware devices to send data back and forth simultaneously. For example, two computers connected via an [Ethernet](https://techterms.com/definition/ethernet) cable can send and receive data at the same time. Wireless networks also support full-duplex communication. Additionally, modern [I/O](https://techterms.com/definition/io) standards, such as [USB](https://techterms.com/definition/usb) and [Thunderbolt](https://techterms.com/definition/thunderbolt), are full-duplex.

**Why is Full Duplex beneficial:**

**-** Full duplex provides better performance than simplex and half duplex mode.

- Full-duplex mode is used when communication in both directions is required all the time. The capacity of the channel, however, must be divided between the two directions.   
Example: Telephone Network in which there is communication between two persons by a telephone line, through which both can talk and listen at the same time.