



A High-Speed Serial Connectivity Solution with Aurora IP

Aurora is a highly scalable protocol for applications requiring point-to-point connectivity.

by Mrinal J. Sarmah
Hardware Design Engineer
Xilinx, Inc.
mrinal.sarmah@xilinx.com

Hemanth Puttashamaiah
Hardware Design Engineer
Xilinx, Inc.
hemanth.puttashamaiah@xilinx.com

With advances in communication technology, you can achieve gigahertz data-transfer rates in serial links without having to make trade-offs in data integrity. The proliferation of serial connectivity can be attributed to its advantages over parallel communication, including:

- Improved system scalability
- More flexible, thinner cabling
- Increased throughput on the line with minimal additional resources
- More deterministic fault isolation
- Predictable and reliable signaling schemes
- Topologies that promise to scale to the needs of the end user
- Exceptional bandwidth per pin with very high skew immunity
- Reduced system costs because of smaller form factors, fewer PCB traces and layers, and lower pin/wire count

Although it offers real benefits, serial I/O has some negative attributes. Serial interfaces require high-bandwidth management inside the chip, special initialization and monitoring, bonding of lanes in an aggregated channel of multiple lanes, elastic buffers for data alignment, and de-skewing. Also, flow control is complex and you must maintain the correct balance between high-level features and total chip area.

Multi-Gigabit Transceivers

Following the industry-wide migration from parallel to serial interfaces, Xilinx introduced multi-gigabit transceivers (MGTs) to meet bandwidth requirements as high as 6.5 Gbps.

The common functional blocks in these transceivers are an 8B/10B encoder/decoder, transmit buffer, SERDES, receive buffer, loss-of-sync finite state machine (FSM), comma detection, and channel bonding logic. These transceivers have built-in clock data recovery (CDR) circuits that can perform at gigahertz rates. Built-in phase-locked loops (PLL) generate the fabric and transceiver clocks.

Transceivers have several advantages:

- With their self-synchronous timing models, they reduce the number of traces on the boards and eliminate clock-to-data skew
- Multiple MGTs can achieve higher bandwidths

- MGTs with point-to-point connections make switched fabric architectures possible
- The elastic buffers available in MGTs provide high skew tolerance for channel-bonded lanes

MGTs are designed for configurable support of multiple protocols; hence their control is fairly complex. When designing or integrating designs with a high-speed connectivity solution using MGTs, you will have to consider MGT initialization, alignment, channel bonding, idle sequence generation, link management, data delineation, clock skew, clock compensation, error detection, and data striping and destriping. Configuring transceivers for a particular application is challenging, as you are expected to tune more than 200 attributes.

The Aurora Solution

The Xilinx® Aurora protocol and its associated designs address these challenges by managing the MGT's control interface.

Aurora is free, small, scalable, and customizable. With low overhead, Aurora is a protocol-agnostic, lightweight, link-layer protocol that can be implemented in any silicon device/technology.

With Aurora, you can connect one or more MGTs to form a communication channel. The Aurora protocol defines the structure of data packets and procedures for

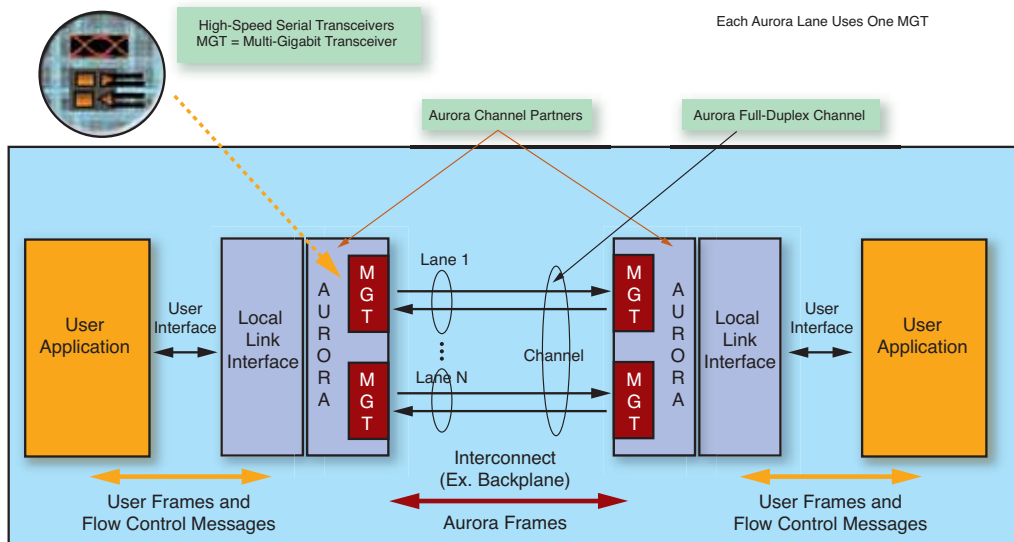


Figure 1 – Connectivity scenario using Aurora

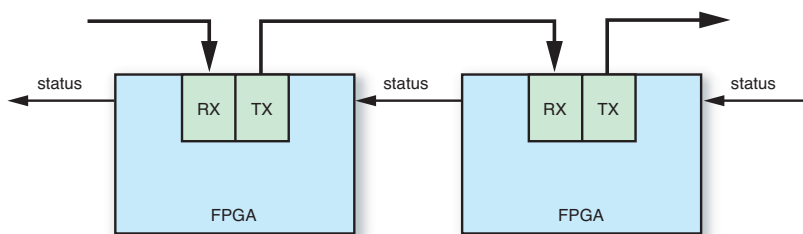


Figure 2 – Simplex token ring structure

flow control, data striping, error handling, and initialization to validate MGT links.

Aurora shrink-wraps MGTs by providing a transparent interface to them, allowing the upper layers of proprietary or industry-standard protocols such as Ethernet and TCP/IP to ride on top of it and provide easily access.

This easy-to-use pre-defined protocol needs very little time to integrate with existing user designs. Being lightweight, Aurora does not have an addressing scheme and cannot support switching. It does not define correction within data payloads.

Aurora is defined for physical and data-link layers in the Open Systems Interconnection (OSI) model and can easily integrate into existing networks.

A Typical Connectivity Scenario

Figure 1 is an overview of a typical Aurora application. The Aurora interface transfers data to and from user application functions through the user interface. The

user interface is not part of the Aurora protocol specification.

The Aurora protocol engine converts generic arbitrary-length data from the Xilinx LocalLink user interface to Aurora protocol-defined frames and transfers it across channel partners comprising one or more high-speed serial links. The number of links between channel partners is configurable and device-dependent.

Most Xilinx IPs are developed based on the legacy LocalLink interface. Any user interface designed for other LocalLink-based IPs can be directly plugged into Aurora. For more information on the LocalLink interface, visit www.xilinx.com/products/design_resources/conn_central/localink_member/sp006.pdf.

You can think of Aurora as a bridge between the LocalLink interface and the MGT.

Aurora's LocalLink interface is customizable for 2- or 4-byte data. Your selection of

a 2- or 4-byte interface should be based on the throughput requirements and latency involved. A 4-byte design has more latency than a 2-byte design, but offers better throughput and consumes less resources than an equivalent 2-byte design.

Aurora channels can function as uni-directional (simplex) or bi-directional (full-duplex). Full-duplex module initialization data comes from the channel partner through a reverse path, whereas in simplex the initialization happens through four side-band signals. Aurora is available in simplex TX only, RX only, or both. "Both" operation is like full duplex, except that the

transmit and receive sections communicate independently. You can use Aurora simplex in token-ring fashion. Figure 2 shows the ring structure of Aurora-S (simplex).

Data Flow in Aurora

Data is transferred between Aurora channel partners as frames. Data flow primarily comprises the transfer of protocol data units (PDUs) between the user application and the Aurora interface, as well as the transfer of channel PDUs between channel partners.

Soon after power-up, when the core comes out of all reset states, the transmitter sends initialization sequences. If the link is fine and the link partner recognizes these patterns, it sends acknowledgement patterns. After receiving sufficient acknowledgement patterns, the transmitter transits to a link-up state, indicating that the individual transceiver connection between channel partners is established. Once the link is up, the Aurora protocol engine moves to a channel verification stage for a single-lane channel, or a channel-bonding stage (preceeding a verification stage) for a multi-lane channel. When channel verification is complete, Aurora sends a channel-up signal, which is followed by actual data transmission. The link initialization process is shown in Figure 3.

Frame Types

As stated previously, data is sent over Aurora channels as frames. There are five frame types, listed in the order of their priority:

- Clock compensation (CC)
- Initialization sequences
- Native flow control (NFC)
- User flow control (UFC)
- Channel PDUs
- Idles

Aurora allows channel partners to use separate reference clocks by inserting CC sequences at regular intervals. It can accommodate differential clock rates up to 200 parts per million (ppm) between the transmitter and the receiver.

An Aurora frame sends data in a quantity of two bytes called a symbol. In case the number of bytes in the PDU is odd, it appends one extra byte called a pad. On the receive side, the pad is removed by the receive logic and data is presented to the LocalLink interface.

The transmit LocalLink frame is encapsulated in the Aurora frame, as shown in Figure 4. Aurora encapsulates frames with start-channel PDU (SCP) and end-channel PDU (ECP) symbols. On the receive side, the reverse process occurs. The transceivers are responsible for encoding/decoding the frame into 8B/10B characters, serialization, and de-serialization.

Flow Control

Aurora supports an optional flow control mechanism to prevent data loss caused by different source and sink rates between channel partners (Figure 5).

Native Flow Control

NFC is meant for data-link layer-rate control. NFC operation is governed by two NFC FSMs: RX and TX.

The RX NFC FSM monitors the state of the RX FIFO. When there is a risk of overflow, it generates NFC PDUs, requesting that the channel partner pause transmission of user PDUs for a specific time duration. The TX NFC state machine responds by waiting during the requested time, allowing the RX

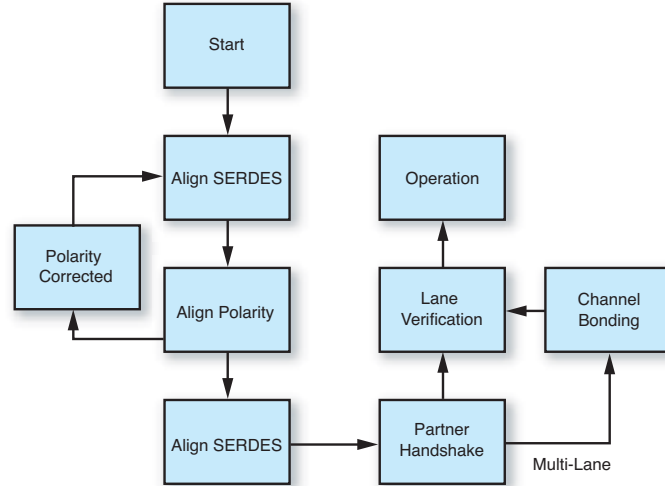


Figure 3 – Data flow in Aurora

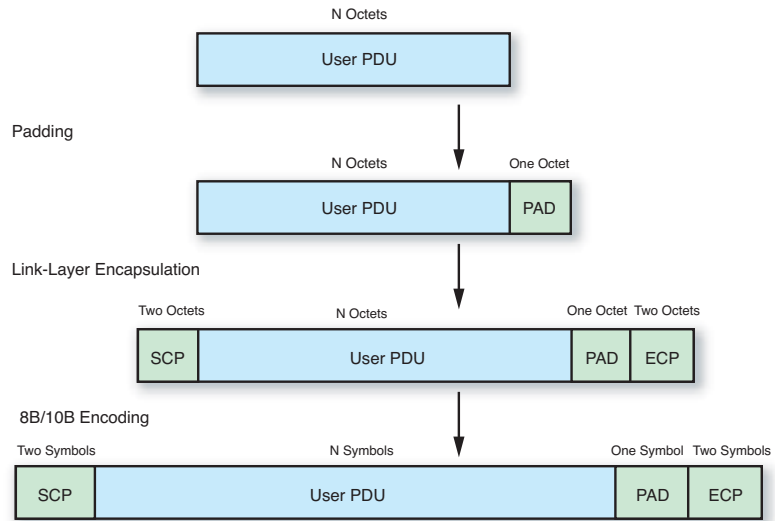


Figure 4 – Frame encapsulation in transmit Aurora

FIFO to come out of its overflow state.

While sending NFC requests, the TX NFC FSM should eliminate any round-trip delay. Ideally, NFC requests are sent before receive FIFO overflows to account for this delay. You can program the NFC pause value from 0 to 256; the maximum pause value is infinite. An NFC pause value is non-cumulative and new NFC request values override the old.

There are two NFC request types: immediate mode and completion mode. In immediate mode, an NFC request is processed while low-priority requests are halted. In completion mode, the current frame is

transmitted; only after completion of the transfer is the NFC request processed.

User Flow Control

UFC is used to implement user-defined flow control at any layer. The number of UFC messages can be 2 to 16 bytes in a frame. UFC messages are generated and interpreted by the user application.

Applications

Aurora is a simple, scalable open protocol for serial I/O. It can be implemented in FPGAs or ASICs. Aurora can be applied where an inexpensive, high-performance

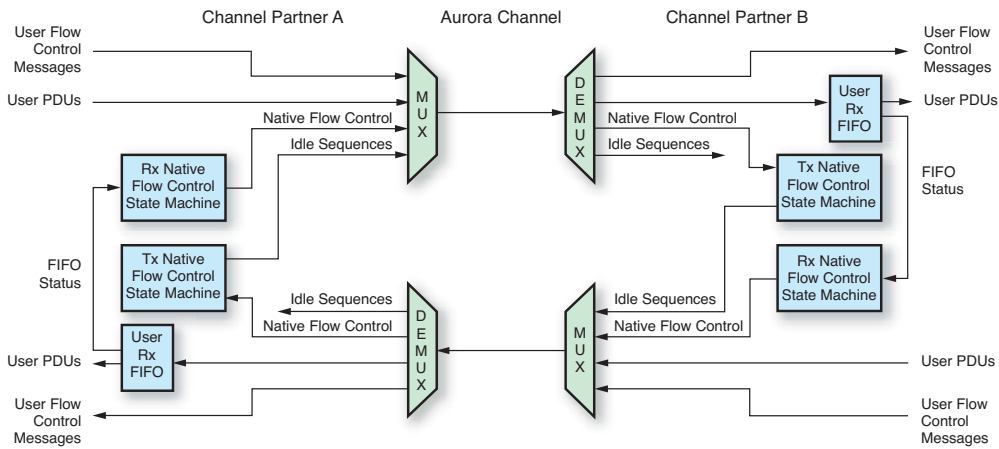


Figure 5 – NFC operation in Aurora

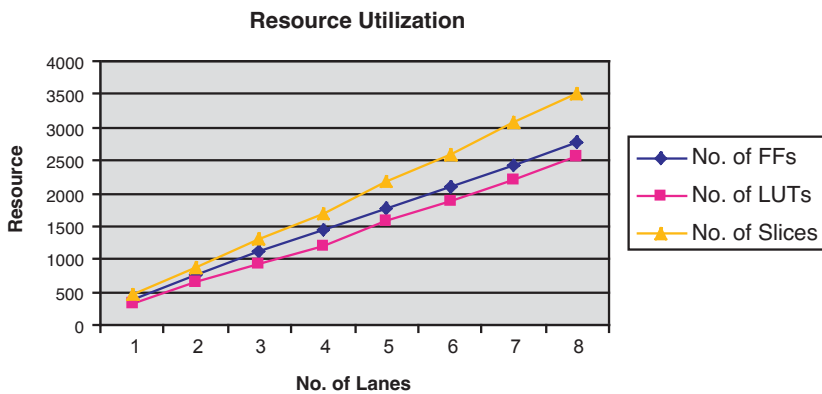


Figure 6 – Performance curves for various Aurora flavors

link layer with little resource consumption is required, saving many hours of work for users who were considering developing their own MGT protocols.

Aurora has a variety of applications, including:

- Video
- Medical
- Backplane
- Bridging
- Chip-to-chip and board-to-board communication
- Splitting functionality among several FPGAs
- Simplex communication

Aurora provides more throughput for fewer resources. Plus, it can adapt to customer needs, allowing additional func-

tions on top. Aurora simplex operation offers the most flexibility and the least resource use where users require high throughput in one direction. Simplex is ideal for non-data communication like streaming video.

Performance Statistics

Aurora is designed to use minimal FPGA resources. A single-lane 2-byte framing design consumes approximately 383 look-up tables and 374 flip-flops. The curves in Figure 6 show the resource utilization statistics for different lane configurations on a Virtex™-5 LXT device.

The latency for a single-lane 2-byte design is 10 cycles. Transceivers insert some inherent latency. Thus, the overall latency is approximately 29 cycles for an Aurora design on a RocketIO™ GTP transceiver.

You have the freedom to choose reference clocks from a range of values. The overall throughput depends on the reference clock value and the line rate chosen.

Aurora: CORE Generator IP

Aurora is released as a part of CORE Generator™ software, with about 10 configurable parameters. You can configure an Aurora core by selecting streaming/framing interface, simplex/full-duplex data flow, single/multiple MGTs, reference clock value, line rate, MGT location(s) based on number of MGTs selected, and reference clock source. The supported line rate can range from 0.1 Gbps to 6.5 Gbps depending on the Virtex device selected. The core comes with simulation scripts supporting MTI, NC-SIM, and VCS simulators and build scripts to ease design synthesis and bit generation.

Conclusion

Aurora is easy to use. The IP is efficient, scalable, and versatile enough to transport legacy protocols and implement proprietary protocols. Aurora is also backward-compatible: an Aurora design in a Virtex-5 device can talk to an Aurora design in a Virtex-4 device, making the IP independent of underlying MGT technologies. Xilinx IP and software tools allow you to take advantage of Aurora's improved feature set. One future enhancement currently in the research phase is the inclusion of a packet-retry feature to provide link reliability.

For more information, visit www.xilinx.com/aurora.

TAKE THE NEXT STEP (Digital Edition: www.xcellpublications.com/subscribe/)

- Download the Aurora protocol specification and bus functional model
- Order your free Aurora LogiCORE design
- Read about Aurora application examples