

Eliminate Packet Buffering Busywork

The Packet Queue IP core simplifies packet buffering and aggregation, letting you focus on high-level system design.

by Russ Nelson
Senior Design Engineer
Xilinx, Inc.
russ.nelson@xilinx.com

Stacey Secatch
Staff Design Engineer
Xilinx, Inc.
stacey.secatch@xilinx.com

United Parcel Service and FedEx are arguably two of the most sophisticated package delivery services in the world, but they still do not build their own trucks. They know that their strategic advantage lies in figuring out how to move packages around the world efficiently.

Take a lesson from these two successful companies – let the new Xilinx® Packet Queue LogiCORE™ IP be the delivery vehicle for your packets so that you can focus on your strategic advantage instead of wasting time and money on the mechanics of packet buffering.

Packet Queue joins FIFO Generator and Block Memory Generator in an impressive

portfolio of Xilinx no-cost memory cores. Packet Queue buffers packetized data, multiplexes one or more channels together, provides advanced features such as retransmission, discarding of unwanted or corrupted packets, and includes interfaces for complete scheduling control.

Using Xilinx CORE Generator™ software (as shown in Figure 1), you can configure and generate an optimized, pre-engineered solution to your protocol bridging, aggregation, or packet-buffering application. And because Packet Queue provides an optimized and supported solution at no cost, you will realize real savings in terms of NRE and time to market. In this article, we'll highlight Packet Queue's features and show how it can enhance two sample networking designs.

Channelized Data Transfer

Seldom does the output from a system look identical to the input. This is particularly the case in networking applications, where data is often encapsulated and re-encapsulated in different protocols

as it travels through the network. Packet Queue is particularly suited for these systems. It supports as many as 32 input channels, transparently performing data-width conversion while simultaneously migrating data from each independent input clock domain to the output clock domain. Packet Queue's sideband data feature is also extremely versatile, enabling unusual data widths and additional control or status signaling.

Shared Memory Architecture

Packet Queue reduces your design cost, not only as a no-cost core but also by reducing FPGA resources versus traditional implementations. Packet buffering memory is segmented and allocated to packets dynamically as they are written to the core. This enables sharing of memory between data from all channels. You can therefore configure the overall size of the core to represent the peak requirements of the system, as opposed to the sum of the peak requirements of each channel.

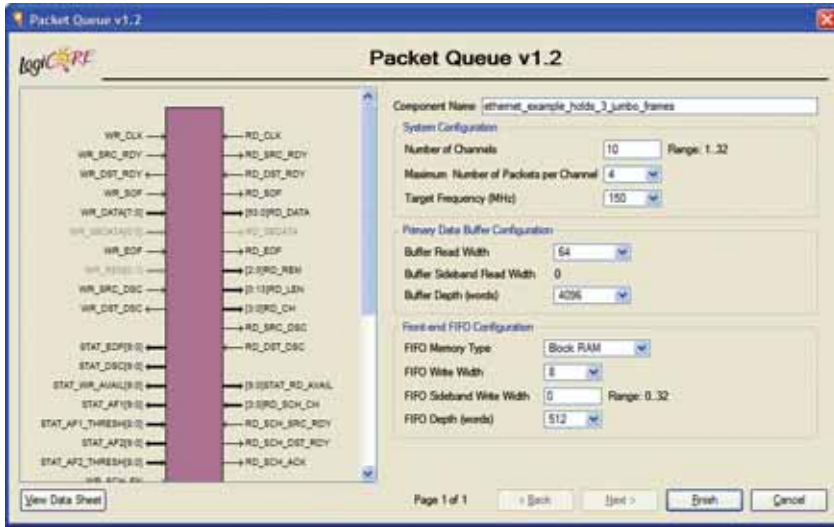


Figure 1 – Packet Queue GUI showing potential configuration for Ethernet example.

Overall memory needs will vary greatly from design to design. Packet Queue enables you to pick the solution that is right for your system’s specifications, resulting in memory savings that translate directly to a lower unit cost for you.

Retransmit and Discard

If your application requires the ability to recover from receiving corrupted packets or to resend packets corrupted during transmission, Packet Queue provides simple yet powerful options. Packet Queue’s input logic supports the discard of an incoming packet at any point before packet completion by asserting a single signal. This functionality can also be used to discard unwanted packets, such as those with invalid cyclic redundancy checks (CRCs). Similarly, you can interrupt a packet being transmitted with a single signal, enabling retransmission at a later time. Together, these two simple yet powerful features allow Packet Queue to easily interface with unreliable links.

Complete Scheduling Control

The most powerful feature of Packet Queue isn’t something included in the core – it’s something *not* included in the core. As a designer of a networking system, your competitive advantage lies in your scheduling algorithms. Packet Queue provides status and control ports for directing multiplexing

between data channels, letting you decide how to transfer data based on the requirements of your system – whether that means round robin, weighted round robin, fixed priority, or anything else you can design. For example, you can choose to give priority to channels that require a higher quality of service (QoS) to reduce queuing latency.

Aggregation Example

Our first example application is that of an Ethernet bridge, between ten 1-Gb ports and a single 10-Gb port, as shown in Figure 2. Packet Queue can assist you with aggregation. Several of its features come into play here:

- Multiple channels – this configuration uses 10 of Packet Queue’s 32 available input channels. You can also configure more input ports for oversubscription of the 10-Gb link.
- Clock conversion – the Xilinx 1-Gb Ethernet MAC LogiCORE solution (GEMAC) provides data at a frequency of 125 MHz. The Xilinx 10-Gb Ethernet MAC LogiCORE solution (10GEMAC) expects data at 156.25 MHz. Packet Queue seamlessly resynchronizes data from one clock domain to the other, allowing each input GEMAC core to exist in its own clock domain.
- Width conversion – the GEMAC core provides 8-bit data, while the 10GEMAC core expects 64-bit data.

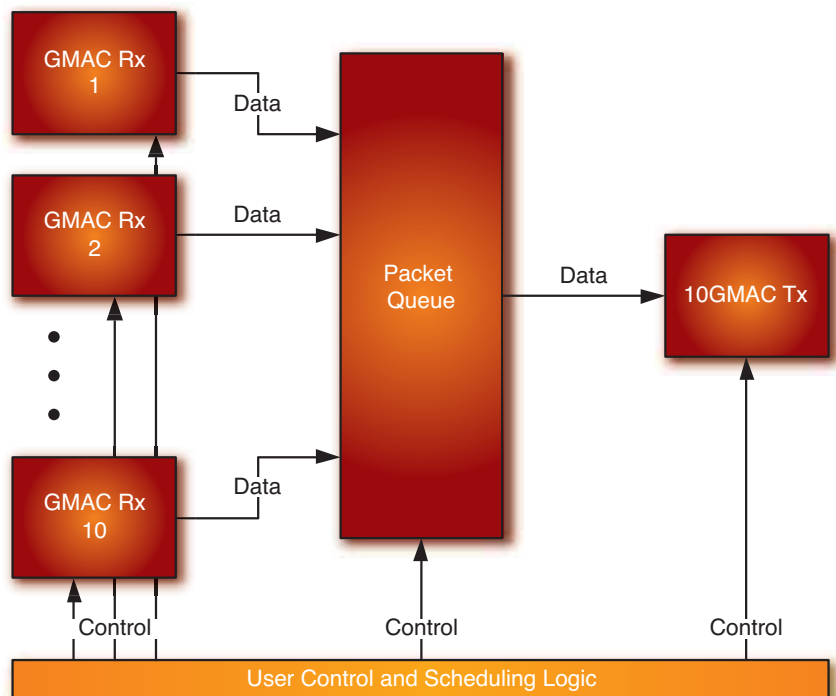


Figure 2 – Packet Queue aggregation of 10 Ethernet channels

Packet Queue automatically converts between the different widths.

- Scheduling control – Packet Queue’s versatile scheduling interfaces enable you to implement any algorithm you desire to control data flow through the core. A simple system might use a round-robin scheme, while a more complex implementation could include QoS, giving priority to particular ports based on the type of data expected.
- Shared memory architecture – the dynamic memory allocation used by Packet Queue is particularly suited to Ethernet because it supports variable frame sizes, including jumbo frames. Because memory is shared between channels, the total memory needed is reduced, while still handling bursty data. Simply determine the peak outstanding data and apply it to the shared memory size instead of the size of each channel. For example, the core as configured in the GUI shown in Figure 1 supports as many as three stored jumbo frames at any given time, in addition to smaller standard frames.

Buffering Example

Our second example application, shown in Figure 3, uses Packet Queue to provide

transmit buffering and retry capabilities for the Xilinx Serial RapidIO LogiCORE solution (SRIO). The RapidIO protocol allows for as many as 31 outstanding (unacknowledged) packets, with the possibility of retransmitting any or all of the outstanding packets. Packet Queue is ideally suited to provide this amount of buffering, as well as providing other features to make the integration as simple and seamless as possible. These features include:

- LocalLink Interfaces – both Packet Queue and SRIO use the Xilinx LocalLink standard for their data interfaces. This enables you to connect the two cores’ data interfaces together without any glue logic. No glue logic means faster development and less time spent debugging.
- Retransmit packets – Packet Queue’s retransmit capability meshes quite well with the requirements of RapidIO. The SRIO core provides two interfaces through which packet retransmit requests are made. When there is a problem with the packet currently on the transmit interface, SRIO asserts its “discontinue” signal and the packet is retransmitted by the Packet Queue without any further intervention by your control logic. When retransmission of

more than one packet is required, your control logic then translates the request for all of the failed packets currently located in the Packet Queue and schedules them for resend as normal. Following packet acknowledgment, your logic then instructs Packet Queue to free the memory that was allocated.

- Sideband data and channelization – RapidIO optionally allows packets with different priorities. Packet Queue supports this in either of two ways, giving you the flexibility to pick the implementation that best suits your system requirements. A simpler way is to configure the core with a single bit of sideband data that is mapped to the SRIO core’s critical request input. Doing so causes the critical request flag to pass through the Packet Queue along with the associated packet.

The more complex (but more powerful) method for implementing priorities is to use multiple Packet Queue channels, with prioritized and critical request packets written to separate channels. This strategy allows higher priority packets to pass lower priority packets in the Packet Queue, but requires you to implement more complex scheduling and retransmit control logic.

Conclusion

The Xilinx Packet Queue LogiCORE IP is a simple yet powerful solution for protocol bridging, aggregation, and packet buffering systems. In addition to the examples we’ve discussed, Packet Queue is great for bridging SPI-3 to SPI-4.2, SPI-4.2 to XAUI, PCI to PCI Express, and countless other protocol combinations. As networking protocols multiply and networks carry more diverse types of data, the need for these types of systems will only grow. Packet Queue provides a cost-effective solution for your designs and lets you deliver your product to your customers earlier.

For more information about the Packet Queue, visit www.xilinx.com/systemio/packet_queue. Feedback and suggestions for new features are always welcome. To contact the Packet Queue team, e-mail packet_queue@xilinx.com.

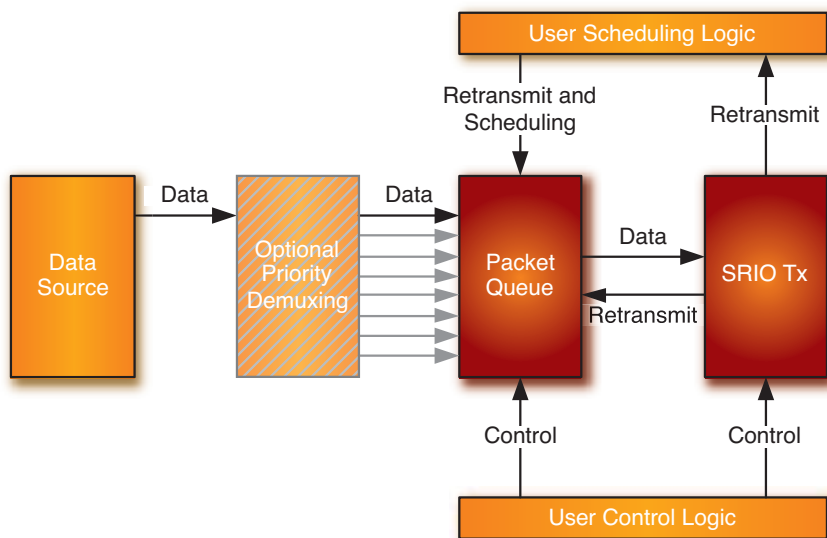


Figure 3 – Serial RapidIO transmit buffering with optional priority reordering