WP150 (v1.0) July 20, 2001

# Solving the Challenges for Terabit Networking and Beyond

Author: Dean Eden

## The Challenges

In today's world of modular networking and telecommunications design, it is becoming increasingly difficult to keep alignment with the many different and often changing interfaces, both inter-board and intra-board. Each manufacturer has their own spin on the way in which devices are connected. To satisfy the needs of our customers, we must be able to support all their interface requirements. For us to be able to make products for many customers, we must adopt a modular approach to the design. This modularity is the one issue that drives the major problem of shifting our bits from one modular interface to another.

Everyone strives to find a complete, all-encompassing solution set that can be used to build any customer requirement. However, this leads to diversity, not in the blocks which are needed, but in the different ways to implement each block. Figure 1 shows a typical modular terabit network node (the number of user interfaces has been reduced in this example for clarity). The diverse interface types are typical of such a system with a common interface that is required across the passive backplane to the switching or routing core devices.

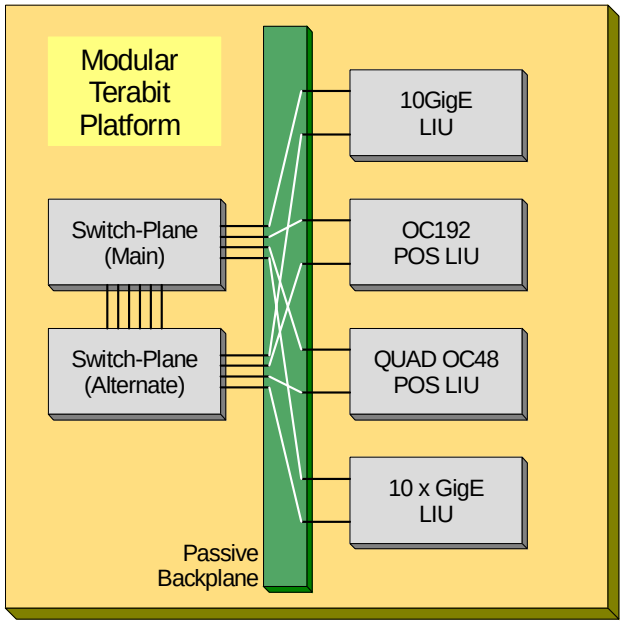


*Figure 1:* **Modular Terabit Network Node**

Each of the interfaces to the outside world has its own requirements for interconnections on the interface card. As yet, no true convergence has been reached for the disparate worlds of the long haul, mixed traffic, transmission system (which came from the traditional telecoms world), and the more immediate wireless data world of the networking industry. The resulting dilemma is that, while it is now possible to carry all types of traffic on each connection technology, there are still individual interface issues.

## Interfacing Dilemmas

There are many interface types; some designed for inter-chip and some for inter-board communications, some requiring hardware solutions, and some extending to higher levels of complexity requiring software interpretation and control. How can we, as designers, possibly hope to provide products that will not be out of date next week?

One approach is to pick a methodology that suits today's application and hope that it will last long enough to provide the product with at least a reasonable life span. This has an obvious disadvantage when we consider that the bandwidth needs of our customers seem to increase exponentially with time. In fact, this is precisely the reason why we have so many apparent standards, and a new one seems to emerge almost every week. Most engineers would agree that the time taken to ratify any interface standard and gain general acceptance is prohibitive; by the time it is issued, it will be behind the technology advances which have been made.

What are the alternatives? We must be able to support our customers thirst for ever-increasing bandwidth and, at the same time, produce our products in a timely and efficient manner. Our solution options today are limited to the choices we make when proposing our new products. We must choose a device vendor who has a chip-set that closely matches our design requirements, not only in functionality, but also in the interconnection methodology, because we must interface to everything else in our system.

Here lies the problem; there is no vendor who sells a set of standard chips for all possible combinations of real world interface types which can satisfy all customers, all the time. Everyone must make a compromise somewhere in his or her system, and this compromise will almost always fall at the interface between the various and disparate peripheral devices in any system. A typical choice of interface will be made by default, based on the device used as the backbone or core of our system element. For example, in a Terabit Router, with many different user interface types, this choice will be made by the core switch or routing element interface specification. This brings up the next challenge in our scenario; how do we connect many high-bandwidth devices to the same switching or routing fabric across a backplane?

Increasing bandwidth has already dictated that interconnection methods for high-speed interface must be serial, not parallel; there simply is not enough space for all the required pins in a parallel system. A 64-bit parallel system for transferring 10 Gbits of data between a peripheral and a switch fabric must be clocked at 156 MHz, and would need hundreds of termination and matching components to make the connection electrically safe and compliant with today's stringent EMI laws. Multiply this issue by the number of user interface cards in any system and the physical connection problems alone make the task impossible.

The difficulty at this time is that we have many existing and necessary devices using parallel technology to connect with other parts of the system. However, this occurs at the circuit card intra-connection level only and will continue to do so for several years to come. If the serial interconnection is limited to the parts of the system which need it, then a reasonable compromise is reached between the desired functionality and the reduction of backplane and switch fabric interconnection complexity.

How do we provide a standard solution for our interconnection methodology while retaining the flexibility to connect disparate interfaces together? As many companies have discovered, FPGAs provide the ideal fabric to solve this problem. Today, with the increasing bandwidth and fast I/O capabilities of FPGAs, this required flexibility is a reality.

## Solution Strategies

Knowing that FPGA technology can solve many of the potential problems when designing an overall system architecture, the designer does not have to worry about the details of individual card-level interconnectivity issues. Once an overall system architecture has been decided upon, it is now possible to design the generic backplane-level interface which will be included on every modular user interface card, and then provide specific designs for the interconnection of the various application specific devices. In some cases, it will be possible to replace some of these devices by absorbing the functionality into the FPGA along with the backplane interface.

The following diagram shows the example network node with a greater level of detail, illustrating where FPGAs can potentially be used to solve difficult integration issues. As can be

seen from the examples, and the even greater detail shown in subsequent sections, each of the different user interfaces has a similar overall design concept on which the individual functionality requirements are built. In addition, although the examples shown in this document show the FPGA connected directly to the switching or routing fabric using direct multi-gigabit links, it is possible to make this connection via the FPGA through a proprietary or standard third-party backplane connection system. For a third-party backplane system, the FPGA provides the interfaces between this system and the diverse user ports.

As can be seen in the diagram, an FPGA has many uses within this architecture, that is, more density of complexity is achievable in a single device. The greatest benefit is reprogrammability, which allows the user or network provider to change or enhance the capabilities of the FPGA to extend the life of any product. The use of Xilinx IRL technology provides the means by which the routing table could be changed, or the protocol filters can be updated, to suit the latest specification or standards change — and these change can be made remotely over any network.

Designers who use FPGAs can release their products earlier, because they do not have to wait for standards to solidify. A good example is the ever-changing specification for multi-gigabit ethernet aggregation in to 10GigE links. There are several proprietary systems in existence and products are already on the market using these systems. At present, you need to install the same equipment at each transmission link. However, an FPGA can be remotely reprogrammed to match the far-end system, whether it uses a proprietary interface or meets the evolving IEEE802.3ad specification, thus allowing the end product to change as needed.

Note that some of the detail of the cards shown in Figure 2 has been omitted to improve clarity.

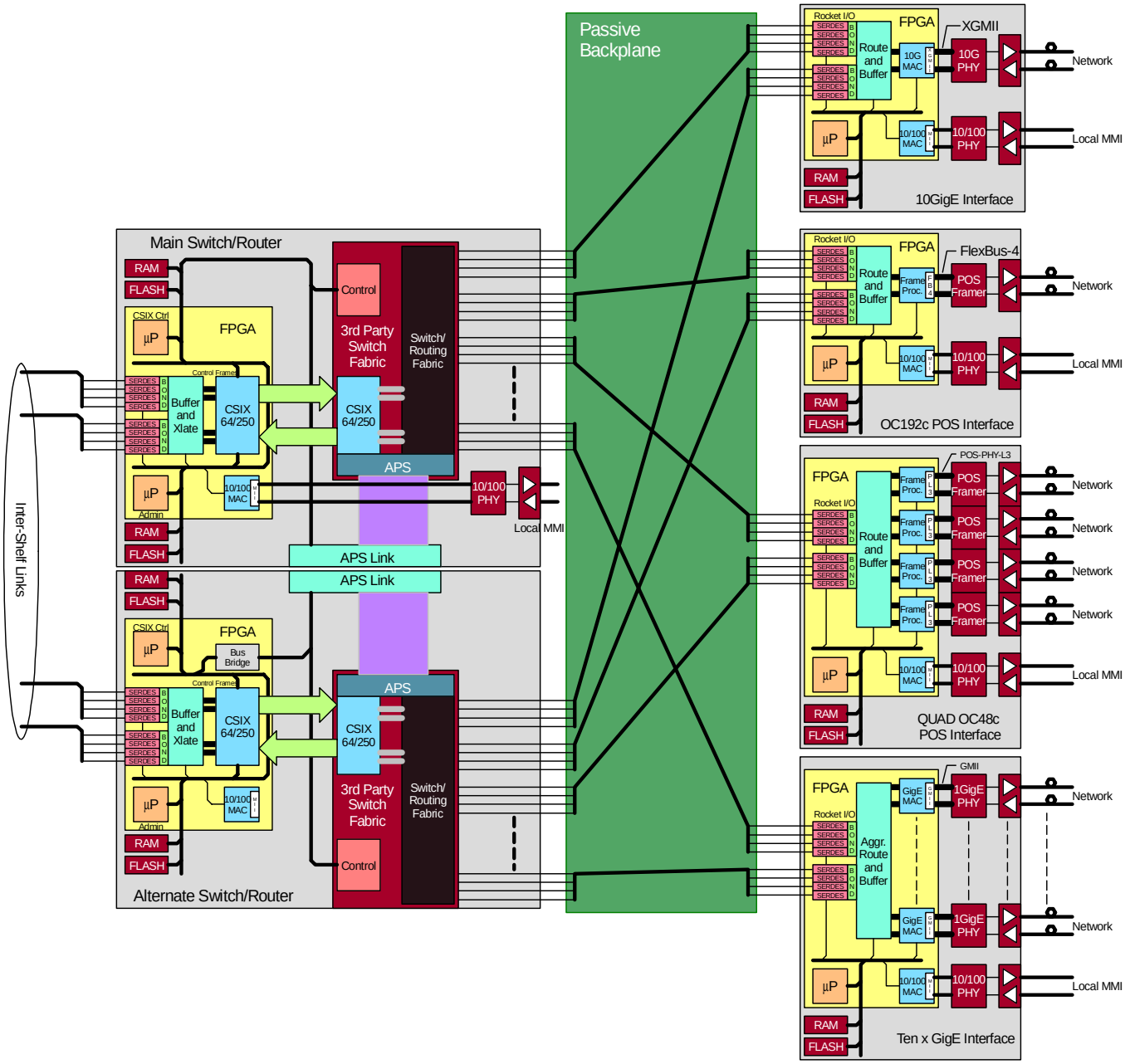


*Figure 2:* **Detailed Block Diagram**

As can be seen from the diagram, using the FPGA to provide the common core of every interface allows the designer to choose specific interface devices for the user connections and adapt these to his chosen internal modular architecture as independent blocks. Changes to the FPGA design then permit changes to the specific physical interface devices (should they become obsolete or unavailable) without complete system redesign.

## Xilinx FPGA: The Configurable Solution

As discussed in previously, a fairly generic FPGA design can be used throughout the system. Figure 3 shows the elements of this design, together with the options required for the different parts of the system.

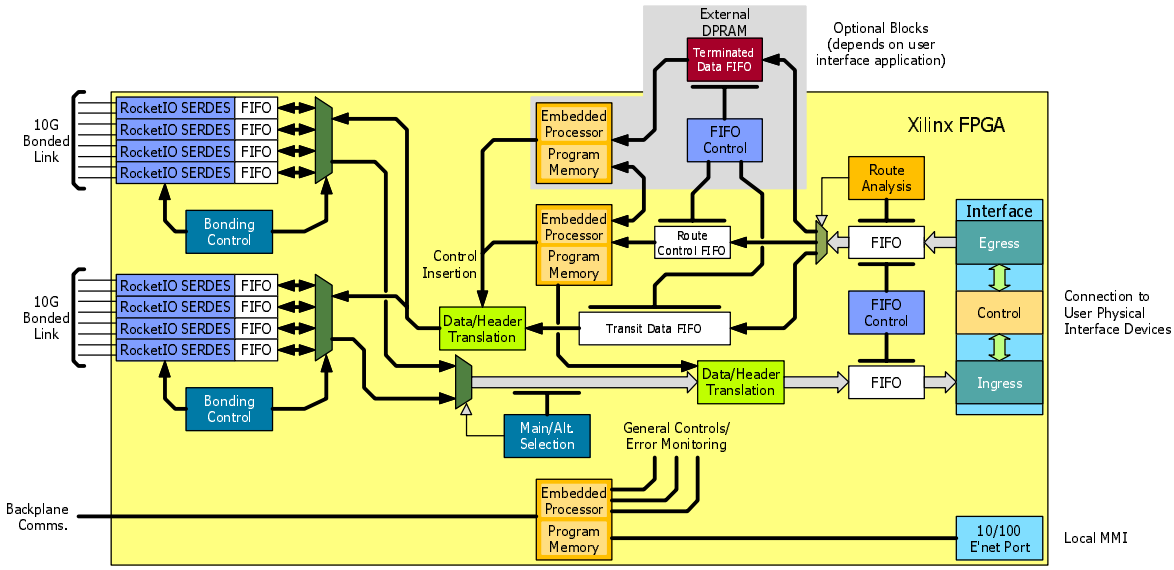


*Figure 3:* **Generic FPGA Block Diagram**

The generic design of this FPGA allows for a 1+1 protected 10-Gbit backplane interface and a 10 Gbps physical interface block which is custom designed to suit each particular PHY ASSP. Between these two extremities, a buffering and message processing system is used to provide data path switching and control.

The backplane interface uses four bonded 3.125 Gbps serial LVDS links connected directly to the FPGA SERDES blocks to provide an aggregate bandwidth of more than 10 Gbps. If using a commercially available backplane chip set solution, these SERDES devices can be replaced by a suitable high-speed parallel interface to this system, such as a CSIX type interface. In either case, the internal interface from the buffering and control subsystem is similar.

The buffering and control subsystem shown in Figure 3 can provide three separate paths for incoming frames or data. Standard user data passes through the system via the simplest path, being only buffered, for rate adaption purposes, on the way through. Control data is routed to a dedicated processing sub-block. More complex functionality requiring the termination of some frames, for example, if connected to a CSIX managed interface, is allowed through the use of another separate dedicated processing sub-block which has extra data buffering capability externally to the FPGA. For systems with multiple user interfaces, such as those with many to one aggregation at the interface card (OC48 to OC192), several user interface blocks would be placed on the FPGA with an aggregetion and segmentation function added to the buffering sub-system.

The custom user interface is designed to suit the particular device chosen to fulfill the physical interface role. Connecting to the user network via a block in the FPGA ensures that, should the physical interface device need to be changed, due to obsolesence or specification change, a minimum amount of change is required for the remainder of the subsystem.

## Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 07/20/01 | 1.0 | Initial Xilinx release. |