Microsemi®

# Use of the IGLOO2 High-Performance Memory Subsystem in Innovative Bridging Applications

June 2013

## Abstract

Microsemi IGLOO®2 devices include an innovative High-Performance Memory Subsystem (HPMS) that provides dedicated functions (like DDR controllers, embedded non-volatile and SRAM memory, DMA, and an efficient AHB Matrix) to speed common system operations typical of innovative bridging applications. When the advanced features of the HPMS are combined with a wealth of FPGA fabric (including advanced DSP and embedded memory blocks), and up to 16 lanes of multi-protocol SERDES, impressive external bandwidth can be matched with extensive internal data-processing and data-management capabilities. This allows even the most bandwidth hungry bridging applications to run continually at full speed while keeping power, cost, and board space within even the stingiest of budgets. This white paper will demonstrate the needs and requirements of systems with advanced bridging functions to directly illustrate the advantages of IGLOO2 devices in these types of applications.

## Introduction

The number of interface standards a typical embedded system is faced with can be daunting. High-performance serial standards are pervasive in networking and communications systems, due to their improved performance, reduced power, and smaller pin-count. Lower performance serial standards are now the interconnect method of choice for everything from flash memories to wireless transceiver modules. Even high-performance Analog to Digital and Digital to Analog converters are moving to high-speed serial interface standards to improve performance, reduce power, and minimize board space. The task of interconnecting this motley assortment of standards typically is assigned to an interface bridge device that can connect to, store data from, and at times apply significant processing power to the data being moved from place to place. Additionally, there may be concerns about security, and if so, it would be important to implement security features to thwart malicious intrusion.

This type of bridging application has become the new ‘glue logic’ required to patch together the various functions within an embedded system. FPGAs have been the traditional ‘go to’ device for ‘glue logic’ applications and bridging applications are indeed a natural fit for FPGAs. Not all FPGAs have been constructed with bridging applications in mind, however, and many of the key requirements of bridging for embedded systems may get overlooked. However, the Microsemi IGLOO2 family has been architected, from the ground up, with bridging in mind. This white paper will identify the key requirements of embedded bridging functions and will show how the synergistic set of features available in IGLOO2 devices deliver the best-in-class solution to modern bridging oriented designs.

## An Overview of Typical Bridging Requirements

The main task of a bridging device is to connect many different standard interfaces: high-speed serial, low-speed serial, and perhaps even some ‘old school’ parallel interfaces, so that data can seamlessly move between them. Clearly, a large number of IOs are required with significant flexibility in implementing a wide range of industry standards— both serial and parallel in nature. High-speed SERDES needs to be present to support Ethernet, PCIe, and other established standards. For the lower speed interfaces, a wide variety of IO standards (LVTTL, LVCMOS, HSTL, and others) are required along with controllers for common protocols like SPI and $I^2C$.

Data storage is also a key requirement for bridging applications and in particular when many interfaces can run simultaneously, adequate memory performance can be critical in implementing an efficient bridge. Typically, a hierarchy of memory blocks are required with many small memories needed for elastic buffers, larger memories needed for storage of data packets, and larger memories, perhaps off-chip, for storing the largest data structures that may require additional data processing.

The data processing capacity of the bridge needs to be a key consideration as well. If there isn't enough processing power available on-chip to handle the high bandwidth data served up by the SERDES ports, the bridge may not be able to achieve the efficiency levels required of the application. Flexible data processing, perhaps in a distributed manner so independent data streams can be processed in parallel, may be the most efficient approach in a typical bridging application.

In many embedded applications security is a key concern.  In these cases, the embedded system needs to be protected from theft by securing the code and FPGA bitstreams (Design IP) using robust encryption functions. Protected key storage is critical in creating a robust protection scheme, so advanced methods should be used to keep security keys hidden from the most sophisticated attacks. The use of some higher level security features like anti-tampering, secure boot, secure updates, and applications level encryption/ decryption should also be considered if the embedded system can be compromised and significant losses result. Ideally, the bridge device would provide these security features at no additional cost, perhaps by integrating the required security features into the device that implements the bridge functions.

The above key functions—high-speed flexible IOs, a hierarchy of memory, distributed high-bandwidth computation, and built-in security, must also work together seamlessly or efficiency will suffer. The use of standard internal bus interfaces, for example, will make it easy to add building blocks as needed without negatively impacting precious design time and internal resources. On-chip resources to manage the bus accesses from the key functions, using a priority system geared towards bridging applications, is another example of possible device synergy. Finally, an easy to use high-level design flow that supports ‘drop-in’ placement of functional units, automatic connections to the desire busses, and guided configuration to create ‘correct by construction’ designs, would provide significant time to market advantages.

## Example Design: Control Plane Bridge Implementation

A common embedded bridging example can be found in control plane applications. An architecture level block diagram of a typical control plane is shown in Figure 1, below. Typically, a high-speed port like 1Gbps Ethernet (GbE), needs to be connected to a bridge that consolidates and manages traffic to a variety of other ports. PCIe, Serial RapidIO, or 10/100 Ethernet are common standards that might coexist on a typical control plane. The data from the GbE port would be separated into multiple packets and routed to the selected destination port. Traffic from the other ports would be combined and sent to the GbE port, based on the priority requirements of each channel. High priority channels, perhaps responsible for alarms or other time critical events, would receive higher priority than normal operational traffic.

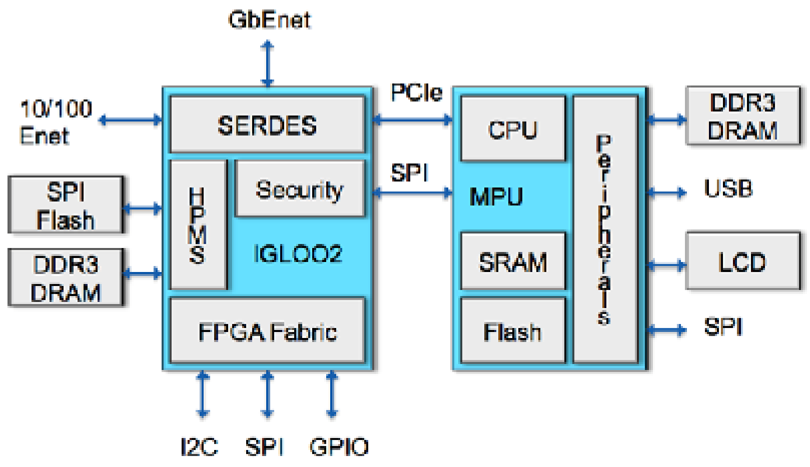

*Figure 1:* **Block Diagram of Control Plane Bridge Using IGLOO2**

In many cases, a large memory is needed to buffer the high-speed traffic from the GbE port, so an external DDR memory can be included for this purpose. Several internal buffer memories, perhaps one or two for each port, will be used as elastic buffers so that more efficient burst transactions can take place to and from DDR memory. Internal buffering might also be used when algorithms with significant computational capability require the storage of intermediate results. Packet processing, compression or encryption/ decryption functions could use the computational capabilities of the FPGA fabric to accelerate data processing requirements.

The bridge might also combine additional management functions used to monitor and control key chassis elements such as power, temperature, humidity, status displays, or to implement fault logging. These lower speed functions might use I$^2$C, SPI or even general purpose IOs to connect with the control plane bridge. Assuming security is important, the bridge device might also be responsible for creating the root-of-trust needed to protect the system from malicious intrusion. Security keys can be kept safe within the IGLOO2 device and a secure boot process, using an external SPI memory to store encrypted and protected MCU code images, can be implemented. This approach can create a robust security ‘barrier’ that provides the level of protection appropriate for an embedded system that manages a variety of sensitive information.

Now that the high level architecture has been described for a typical embedded bridge, we will look at the architecture and features of the Microsemi IGLOO2 FPGA family. Features that fit closely with design requirements will be described and the benefits that they bring to an example design will be highlighted. The wealth of features included in the IGLOO2 FPGA family and their close fit to the requirements of bridging applications will demonstrate that IGLOO2 devices are the best-in-class FPGA solution for embedded bridge designs.

## IGLOO2 Architecture Overview and Feature Fit for the Example Design

A block diagram showing the key elements of the IGLOO2 architecture is shown in Figure 2 below. The major functional blocks are the High-Performance Memory Subsystem (HPMS), the Multi Protocol 5G SERDES blocks, the FPGA Fabric, the Multi-Standard GPIO, and the Security Subsystem. In bridging applications all these functions work together, synergistically, but it is useful to first understand the key elements within each block and how they operate.

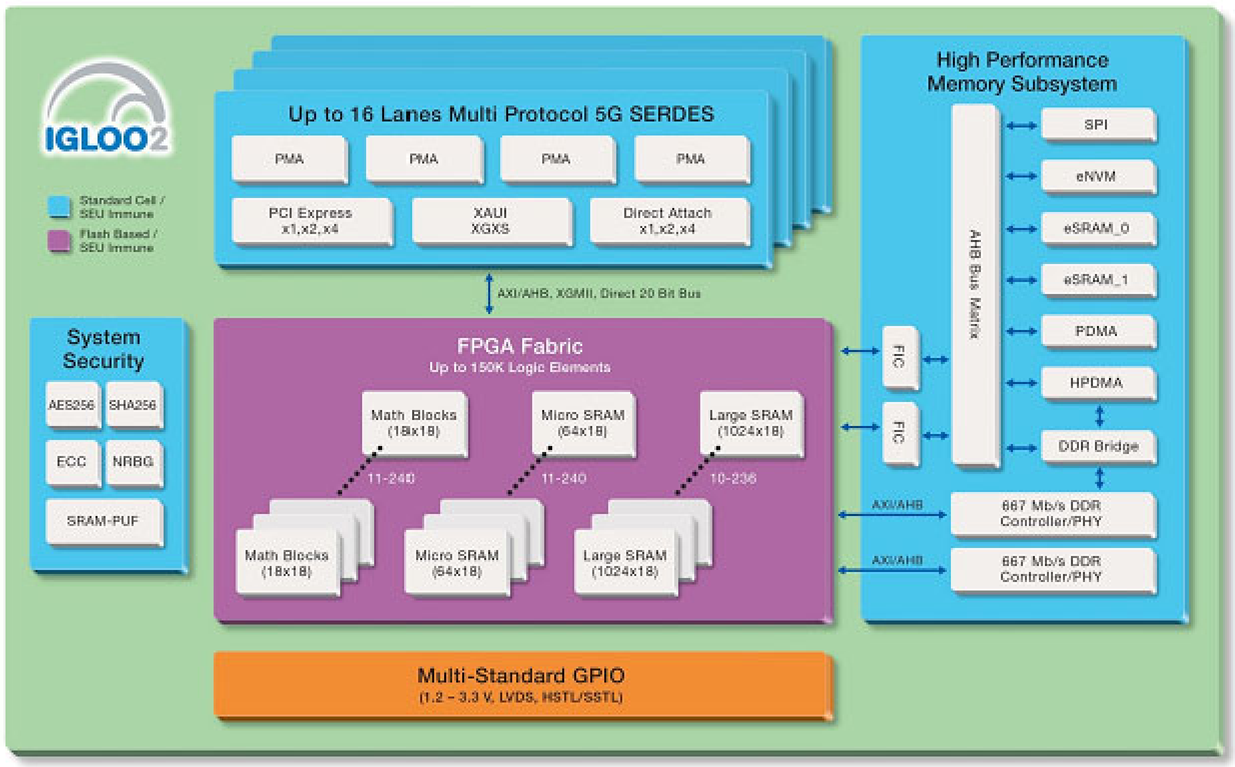

*Figure 2:* **IGLOO2 Architecture Block Diagram**

## IGLOO2 High-Performance Memory Subsystem

The IGLOO2 family combines several key memory storage and data transfer functions within the High-Performance Memory Subsystem. By combining these functions into a common block, IGLOO2 makes it easier to implement data oriented management, buffering and transfer functions—like video frame buffers, TCP/IP buffers, or PCIe Packets, without sacrificing the FPGA fabric for these common functions (A typical fabric implementation would require over 18K Logic Elements which could be a significant fraction of the entire user logic in some devices). The 'hardened' blocks of the HPMS results in a more integrated, lower power, and lower cost implementation in bridging oriented applications. A block diagram of the key elements in the HPMS is shown in Figure 3 below.

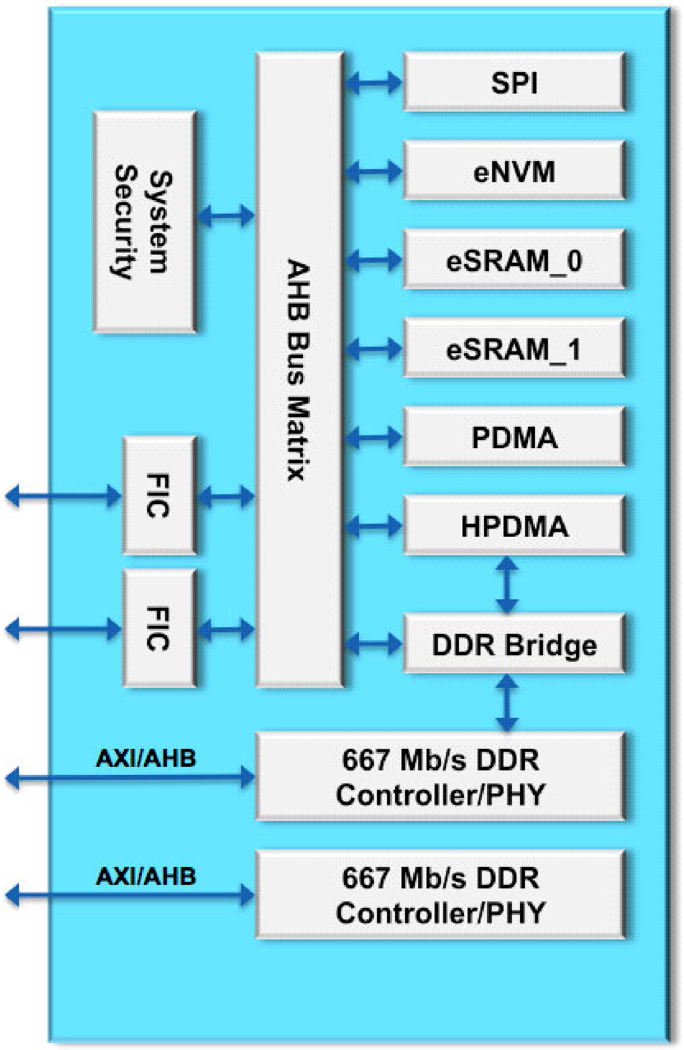

*Figure 3:* **IGLOO2 High-Performance Memory Subsystem Block Diagram**

## AHB Bus Matrix

The AHB bus matrix efficiently manages the interconnections between the various blocks within the HPMS. The bus matrix is the ‘traffic cop’ for the HPMS and is also useful in bridging applications where multiple competing resources need to share a common interconnection facility. The IGLOO2 AHB bus matrix has been architected with common bridging functions in mind. The customized set of interconnects used reduces the size of the AHB bus matrix over a fully interconnected implementation (a full switch) while preserving the high-performance connections needed to the key resources. For example, the HPDMA controller can act as a master with direct access to eSRAM and the DDR bridge for improved data transfer bandwidth.

## Fabric Interface Controller

The two Fabric Interface Controllers (FICs) provide connectivity between the FPGA fabric and the HPMS blocks making it easy to connect the HPMS to application specific functions within the FPGA fabric. The AXI4 or AHB-Lite FIC interfaces  make it easy to connect the HPMS to custom peripherals, hardware accelerators, or other bridge specific functions within the FPGA fabric. The FIC blocks create the common interface used by FPGA fabric-based functions and simplify the connection and management of these user created blocks.  Additionally, a wide variety of ‘pre-implemented’ functions, intellectual property (IP) cores, are available from Microsemi or third party partners that can simply ‘drop-in’ to the FPGA fabric and connect seamlessly to the FIC with just a few clicks of the mouse.

## SPI Controller

Serial peripheral Interface (SPI) is a synchronous serial data protocol that enables the microprocessor or microcontroller and peripheral devices to communicate with each other. In addition, the SPI controller can interface with large SPI flash and EEPROM devices and includes a hardware-based slave protocol engine. The IGLOO2 SPI controller can be an important element in creating efficient bridging applications. It not only provides a standard method to attach lower speed devices or large external memories (as a master) but can also provide slave connectivity when bridging functions are configured or managed by a host using an ‘out of band’ protocol, separate from the higher performance bridged interfaces. The SPI controller is easily configured within the IGLOO2 design flow using step-by-step guides that produce ‘correct-by-construction’ outputs used by the remainder of the FPGA design flow.

## eNVM Controller

The embedded Non-Volatile Memory (eNVM) controller simplifies reading and writing to the memory and also implements read protection and write protection for specific memory regions, so the designer can secure important data from unauthorized access. The availability of large amounts (128 KB) of eNVM enables the storage of large data tables, constants used in complex DSP algorithms, secure boot code, or translation matrices. In many bridging applications, it is convenient to have large non-volatile tables that are used to store routing tables and quality of service logs that must survive a power cycling event. On-chip storage can also be important in applications where security is important and the transfer of off-chip data could be subject to unauthorized ‘snooping’ or intrusion. The IGLOO2 eNVM block is ideal for these types of uses as well as a host of other possibilities.

## eSRAM

The two embedded SRAM blocks (eSRAM) in the HPMS are accessed using an interface controller that provides single error correction and dual error detection (SECDED) protection. The availability of two large blocks of eSRAM, accesses via the AHB matrix simplifies the implementation of buffers common to many bridging applications. For example, one buffer can be used to store data from the SERDES port while the other is used to store data from the external DDR memory (perhaps with both using DMA controller). This dramatically improves overall system bandwidth and significantly simplifies data buffer management. The IGLOO2 eSRAM block can also be used for storage of intermediate data for complex data processing algorithms, like Digital Signal Processing, video conversion, data encryption/decryption, data compression, or communications packet processing. The memory controller associated with the eSRAM makes accessing the memory easy and details like SECDED operations are transparent to the user.

## DMA Controllers

The Peripheral Direct Memory Access (PDMA) controller manages data transfers from HPMS peripherals-to-memory, memory-to-peripherals, memory-to-memory, and with the FPGA fabric via the two FIC blocks. In many bridging applications, the PDMA can be used to manage the data transfers to and from the lower performance communications channels, like SPI. Additionally, data transfers from the FPGA fabric-based custom interfaces and algorithmic accelerators can be accessed via the FIC block using the PDMA.  Use of the PDMA is particularly beneficial when managing data buffers that bridge low-speed peripheral ports with higher bandwidth connections to DDR memory or high-speed SERDES ports. Using the fixed function, PDMA dramatically improves transfer efficiency without requiring the creation of a DMA capability from additional FPGA fabric.

The High-Performance DMA controller provides fast data transfer between the eSRAM and eNVM blocks and the DDR Memory controller associated with the HPMS DDR bridge. One of the most common use models for HPDMA operation, is for paging data transfers from external DDR memory to local eSRAM. Buffers for communications bridging functions are commonly located in eSRAM memory and when high-performance transfers are required to external memory, the HPDMA controller is an optimal implementation. Local eSRAM pages can also be used for intermediate storage locations for data processing, perhaps using internal FPGA fabric hardware accelerators or DSP functions, significantly speeding operations when compared to multiple reads and writes targeted directly to and from external DDR memory.

## DDR Bridge

The DDR bridge manages multiple AHB bus masters accessing a DDR controller and optimizes the read and writes operations to a single external DDR memory using  read and write combining buffers and a round robin arbiter. Typical bridge applications require multiple buffers with data transfer requirements that are easily grouped into bursts. The DDR bridge round robin arbitration scheme is well suited for managing multiple buffers and is particularly efficient when high-speed SERDES ports on on-chip computations are both requesting DDR memory bandwidth. The ability of the DDR bridge to optimize memory accesses means that multiple processes can have memory bandwidth available when it is most needed.

## DDR Controllers

The DDR controllers have an integrated PHY and arbitration logic to simplify the control of external LPDDR1, DDR2, and DDR3 memory devices. The DDR controllers operate at up to 667 Mbps (333.33 MHz DDR) performance and support up to 4 GB of memory. Most IGLOO2 designs, and in particular bridging applications, will utilize external memory, so the inclusion of dedicated DDR memory controller functions with flexible connections to the rest of the HPMS is much more cost and power efficient. The IGLOO2 DDR controller includes bandwidth-enhancing features like dynamic scheduling, read priority and write combining, that improves memory bandwidth by combining memory accesses into efficient transactions instead of implementing lower bandwidth individual accesses.  Additionally, use of tightly coupled HPDMA, via the DDR bridge, can improve data transfer performance to external DDR memory so that high bandwidth communications ports, like those implemented using high-speed SERDES, can operate at full speed. This is just one example of the synergy provided by the integrated and optimized IGLOO2 HPMS.

## System Security Block

The IGLOO2 system security block manages all the programming and security related functions included on IGLOO2 devices. The best-in-class security features of IGLOO2 FPGAs include encrypted user bitstream-key loading, certificate of conformance support, X.509 compliant digital certificate management, elliptical curvecryptography  support, an Advanced Encryption Standard (AES) engine, a SHA-256 engine, a Non-Deterministic Random Bit Generator (NRBG), Differential Power Analysis (DPA) countermeasures, anti-tampering countermeasures, single-use debug passcodes, SRAM-PUF, device level zeroization, and FlashLock® protection of bitstream decryption keys, and security settings. Some of the higher-level functions (like zeroization, AES, SRAM-PUF, NRBG, and SHA-256) are easily included in user designs as security services accessible via the security block.

The use of dedicated ‘hard coded’ functions reduces the overhead for implementing secure systems in IGLOO2 devices. The above security services complement the inherently secure nature of IGLOO2 devices. Because the configuration bit stream resides internal to the device in non-volatile flash memory, it is much more secure than other FPGA configuration implementations. Additionally, programming bit streams are also encrypted and protected so that IGLOO2 base designs are free from reverse engineering, cloning, or overbuilding, even if devices are programmed by an external contract manufacturer. In most IGLOO2 applications, this level of protection comes for free since no FPGA fabric is required to implement these functions. Security typically ‘comes along for the ride’ when an IGLOO2 FPGA is used to implement a non-security related section of an embedded system, like a communications bridge.

The System Security block is also responsible for controlling the IGLOO2 Flash*Freeze low power mode. The Flash*Freeze mode is an innovative low power feature available on IGLOO2 devices that allows the entire device to go into a very low power state. Because IGLOO2 FPGAs use non-volatile technology, configuration data is not lost in this low power state and the device can ‘wake-up’ without needing to reconfigure. This feature can dramatically reduce power consumption when device operations are only required periodically, perhaps only after a ‘wake-up’ command from an external device. Some bridge applications may not need to be active 100% of the time (perhaps because sensor reading are only needed a few times a second) and in these cases, Flash*Freeze can offer a significant power savings.

## FPGA Fabric

The ability to customize user logic and efficiently and easily integrate it with the other key blocks is perhaps one of the most important capabilities of the IGLOO2 family. IGLOO2 devices are optimized for advanced security, high-reliability, and low power consumption applications; several key features of the embedded FPGA fabric contribute to the best-in-class results the IGLOO2 family delivers. IGLOO2 FPGA fabric is composed of five key building blocks: the logic module, the large SRAM, the micro SRAM, the Math block, and the routing resources that connect everything together. These low level elements can be used to construct the wide range of functions required in embedded systems designs.

The tightly coupled interface between the HPMS and the FPGA fabric, via the FIC blocks, makes it very easy to add fabric-based peripherals and hardware accelerators. The Math blocks and block memories within the fabric support the design of the wide range of data storage, data movement, and data processing subsystems common in bridging designs.  Many useful bridging related IP cores are available in the IGLOO2 library and can be added to a design with just a few clicks of the mouse.

## SERDES Block

The IGLOO2 high-speed Serializer Deserializer (SERDES) interface block supports multiple high-speed serial protocols using a SERDES transceiver of up to 5 Gbps, supporting Peripheral Component Interconnect Express (PCI Express®), 10 G Attachment Unit Interface (XAUI), and Serial Gigabit Media Independent Interface (SGMII). In addition, many user defined high serial protocols can beimplemented in the IGLOO2 fabric using the External Physical Coding Sublayer (EPCS) interface. The SERDES block is configurable to support single or multiple serial protocol modes of operation. Some of the key features of the SERDES interface include:

- Support for four SERDES/PCS lanes per interface or 16 in total.
- The XAUI/XGXS extension allows implementation of a 10 Gbps (XGMII) Ethernet PHY interface by connecting the Ethernet MAC fabric interface through an appropriate soft IP block in the fabric.
- PCIe endpoint supports PCIe Base Specification 1.1 for Gen1 and PCIe Base Specification 2.0 for Gen1 (2.5 Gbps) or Gen2 (5.0 Gbps) protocol with width of x1, x2 or x4. The application interface to the PCIe link is available through the FPGA and can be programmed to the AXI or AHB master and slave interfaces.

The SERDES interface is the enabler for the high-bandwidth interconnections to GbE and other serial ports. The addition of dedicated logic for the higher-level protocols commonly found in bridging functions, such as XAUI/XGXS and PCIe, supports more efficient implementations. These dedicated functions reduce the burden on the FPGA fabric, reduce power consumption, and simplify the integration process. The use of a common bus (AXI or AHB master or slave interfaces) makes it a simple task to interconnect SERDES-based functions with the High-Performance Memory System and/or fabric-based functions without sacrificing performance or functionality.  The availability of up to 16 lanes of 5 Gbps SERDES with modest (150K) LUT counts and eight lanes at even smaller (50K) LUT counts, is the best in the industry and ideal for the less logic intensive bridging class of designs.

## Multi-Standard IOs

The IGLOO2 Multi-Standard IOs (MSIOs) support a wide range of industry standards and have many advanced features to simplify connecting to off-chip resources. Single-ended standards supported on IGLOO2 include 3.3V LVTTL, LVCMOS (3.3, 2.5, 1.8, 1.5, and 1.2V) and 3.3V PCI. Voltage referenced standards include 1.5V HSTL, SSTL (2.5 and 1.8V). Differential standards include LVPECL, LVDS, RSDR, B-LVDS, and Mini-LVDS. Some of the programmable features available on the MSIOs include programmable input delay, weak pull-up/pull-down, drive strength, Schmidt trigger input and receiver.

IGLOO2 devices are the most IO-rich in the industry and at the 150K LE capacity level they have nearly twice the number of IOs as some competitors. This additional IO capability is ideally suited for bridging applications where wide high-speed busses or a large number of lower speed serial busses need to be implemented.

## Conclusion

Microsemi IGLOO2 FPGAs are the best-in-class FPGA implementation target for embedded bridging applications. The hardened implementations for the High-Performance Memory Subsystem, the 16 lane multi-protocol 5G SERDES block, the System Security block, and the multi-standard GPIO block all work together to deliver the highest on- and off-chip data bandwidth possible. The programmable fabric contains both small and large SRAMs blocks useful for temporary storage and elastic buffers used to manage data traffic efficiently. The FPGA fabric also contains sufficient logic elements and math blocks to implement the computational capability needed to keep up with the high bandwidth communications channels typically feeding a bridging function. The IGLOO2 on-chip flash-based configuration technology with bit stream encryption, offers a level of security no other FPGA can provide—and it comes for free. This protects the valuable IP within the IGLOO2 device and can be extended to help protect the rest of an embedded bridge design as well.

**Microsemi Corporate Headquarters**
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

55900156-0/6.13