Eight Benefits of Using an FPGA with an On-chip High-Speed Network (WP020)

Achronix
Data Acceleration

December 05, 2019 **White Paper**

Introduction

Since the initial introduction of FPGAs decades ago, each new architecture has continued to employ a bit-wise routing structure. While this approach has been successful, the rise of high-speed communication standards has required ever increasing on-chip bus widths to be able to support these new data rates. A consequence of this limitation is that designers often spend much of their development time trying to achieve timing closure, sacrificing performance in order to place and route their design.

Traditional FPGA routing is based on many individual segments that run horizontally and vertically throughout the FPGA, with switch boxes at the intersections of the horizontal and vertical routes to enable paths to be connected. A path from any source to any destination on the FPGA can be made with these segments and switch boxes. This uniform structure of FPGA routing enables extreme flexibility in implementing any logic function, for any data path width within the FPGA fabric.

While the bit-wise routing in FPGAs is very flexible, it has the downside in that each segment adds delay to any given signal path. Signals that need to span long distances in the FPGA will incur the delays of each of the connecting segments, slowing the performance of the function. Another challenge with bit-wise routing is congestion, which requires signal paths to detour around the congestion, which can incur more delays and causes the performance to degrade further.

Achronix saw this challenge as an opportunity to develop a new architecture to eliminate traditional FPGA design challenges and increase system performance. Achronix's solution was to create a revolutionary 2D high-speed network on chip (NoC) on top of the traditional segmented FPGA routing structure for its new Speedster7t FPGA family. The Speedster7t NoC connects to all of the on-chip high-speed interfaces: multiple ports of 400G Ethernet, PCIe Gen5, GDDR6 and DDR4/5.

The interior of the NoC consists of an array of rows and columns that distribute network traffic horizontally and vertically throughout the FPGA fabric. Master and slave NoC access points (NAPs) at the location where each row and column of the NoC cross. These NAPs can be a source or destination between the NoC and the programmable fabric.

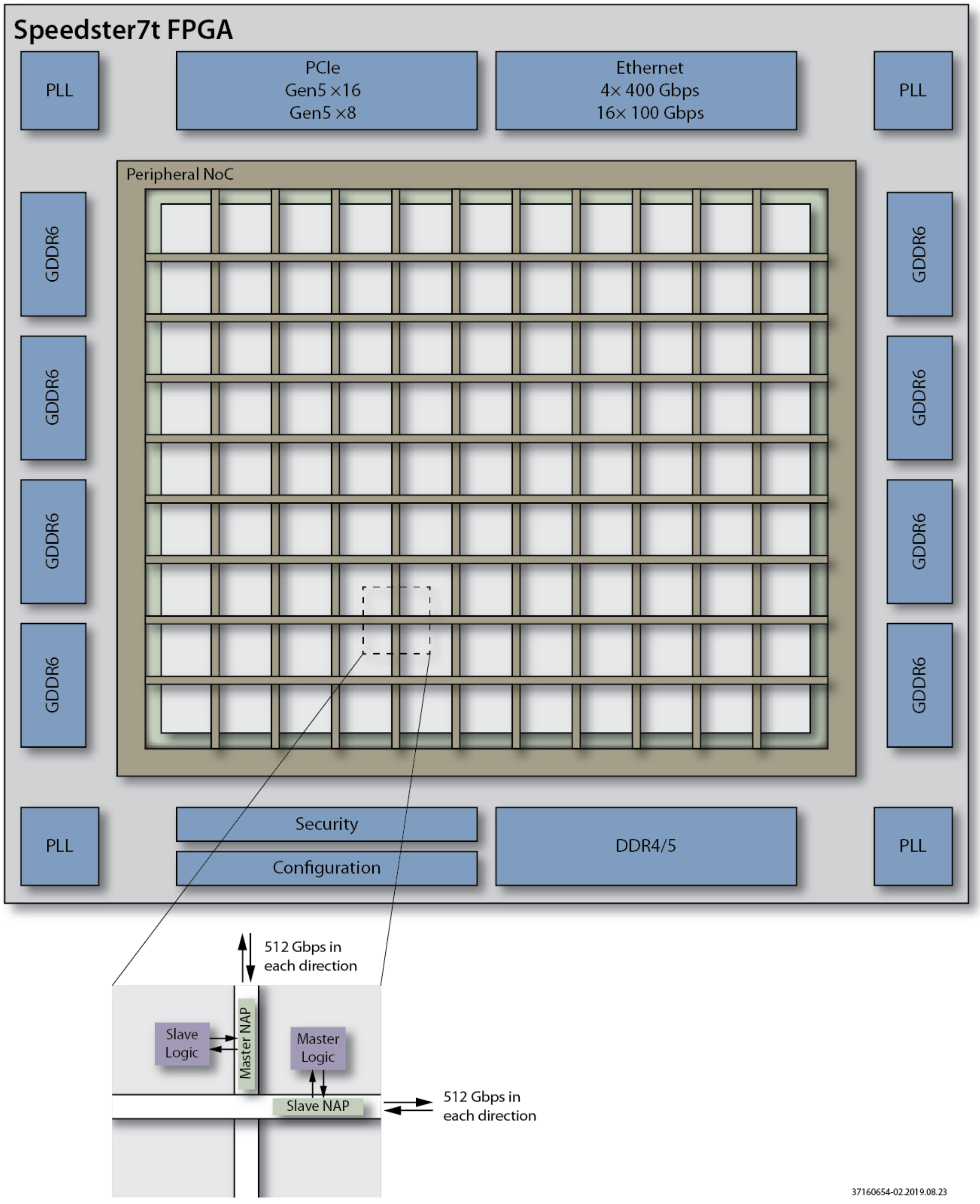

Figure 1: ***Speedster7t NoC and Interfaces***

The Speedster7t NoC might appear to only help with routing buses inside the FPGA; however, this new architecture enables significantly improved designer productivity, new design capabilities and the ability to handle intensive data processing applications with ease. The eight most significant productivity-enhancing, design-changing and performance-delivering use cases are described below.

Streamlined High-Speed Data Distribution Throughout the Entire FPGA Fabric

In traditional FPGA architectures, writing/reading to/from off-chip memory connected an FPGA to/from an external high-speed data source, requires the data to travel through a long, segmented routing path within the FPGA fabric. Not only does the restriction limit bandwidth, it can also consume routing resources needed for the user design hosted within the fabric, creating challenges for the FPGA designer to close timing, especially as device utilization increases with other logic functions.

Using the Speedster7t NoC, transmitting data from an external source into the FPGA and to memory is significantly easier than in traditional FPGA architectures. The Speedster7t NoC augments the conventional programmable interconnect within the FPGA array where the NoC acts like a superhighway network superimposed on a city street system. While the Speedster7t FPGAs' conventional, programmable-interconnect matrix continues to work well for slower, local data traffic, the NoC handles the more challenging, high-speed data flows.

Each row or column in the NoC is implemented as two 256-bit, unidirectional data channels operating at a fixed clock rate of 2 Ghz. Rows have east/west channels and columns have north/south channels, allowing each NoC row or column to handle 512 Gbps worth of data traffic in each direction simultaneously. In aggregate, these conduits can pipe a truly massive amount of data throughout the FPGA array by writing simple Verilog or VHDL code that allows the FPGA to communicate with the NAP and connect onto the NoC superhighway.

The figure below shows transactions between various points in the NoC. The logic at points 1 and 2 each have instantiated a horizontal NAP. The NAPs can both send and receive data, but each individual data stream is just in one direction. Similarly, the logic at points 3 and 4 both instantiate a vertical NAP and can send data streams between each other.

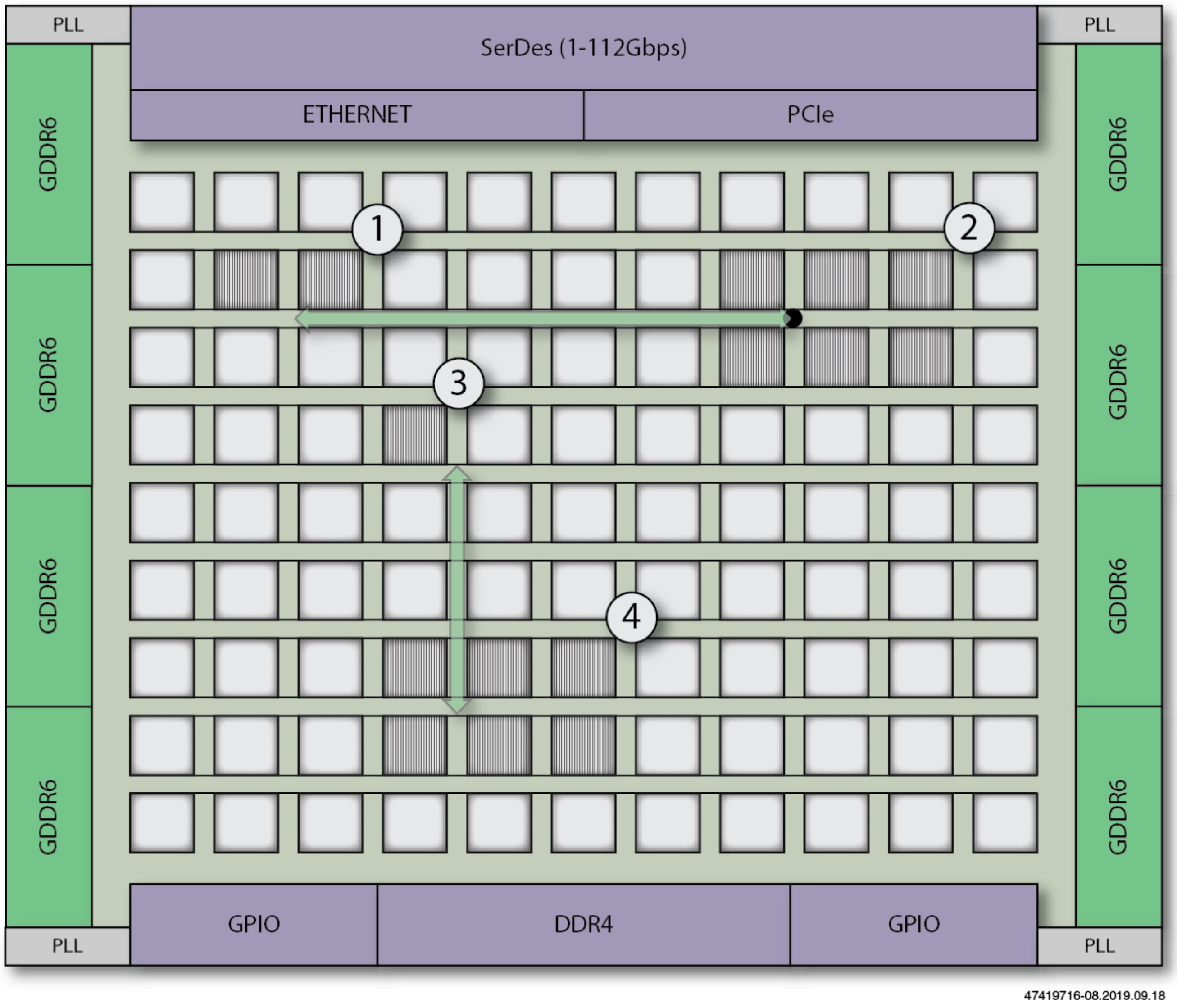

Figure 2: ***Data Streaming on NoC Across Device Fabric***

Automatically Connect PCIe Interfaces to Memory

In modern FPGAs, designers who need to connect high-speed interfaces to read/write memory devices connected to the FPGA must consider delays created by connecting through logic and routing within the device as well as the placement of the input and output signals. Building a simple memory interface often consumes significant time during the design cycle just for basic interface functionality.

In the Speedster7t architecture, connecting the embedded PCIe Gen5 interface to connected GDDR6 or DDR4 memory is handled automatically by the peripheral NoC and does not require designers to write any RTL to establish these connections. Since the NoC is connected to all of the peripheral IP interfaces, designers have significant flexibility in how to connect PCIe to any of the GDDR6 or DDR4 memory interfaces. In the example below, the NoC is able to provide enough bandwidth to sustain PCIe Gen 5 traffic connecting to any two channels of GDDR6 memory. This high-bandwidth connection is achieved without consuming any FPGA fabric resources and nearly zero design time. The user only needs to enable PCIe and GDDR6 interfaces in order to send transactions on the NoC.

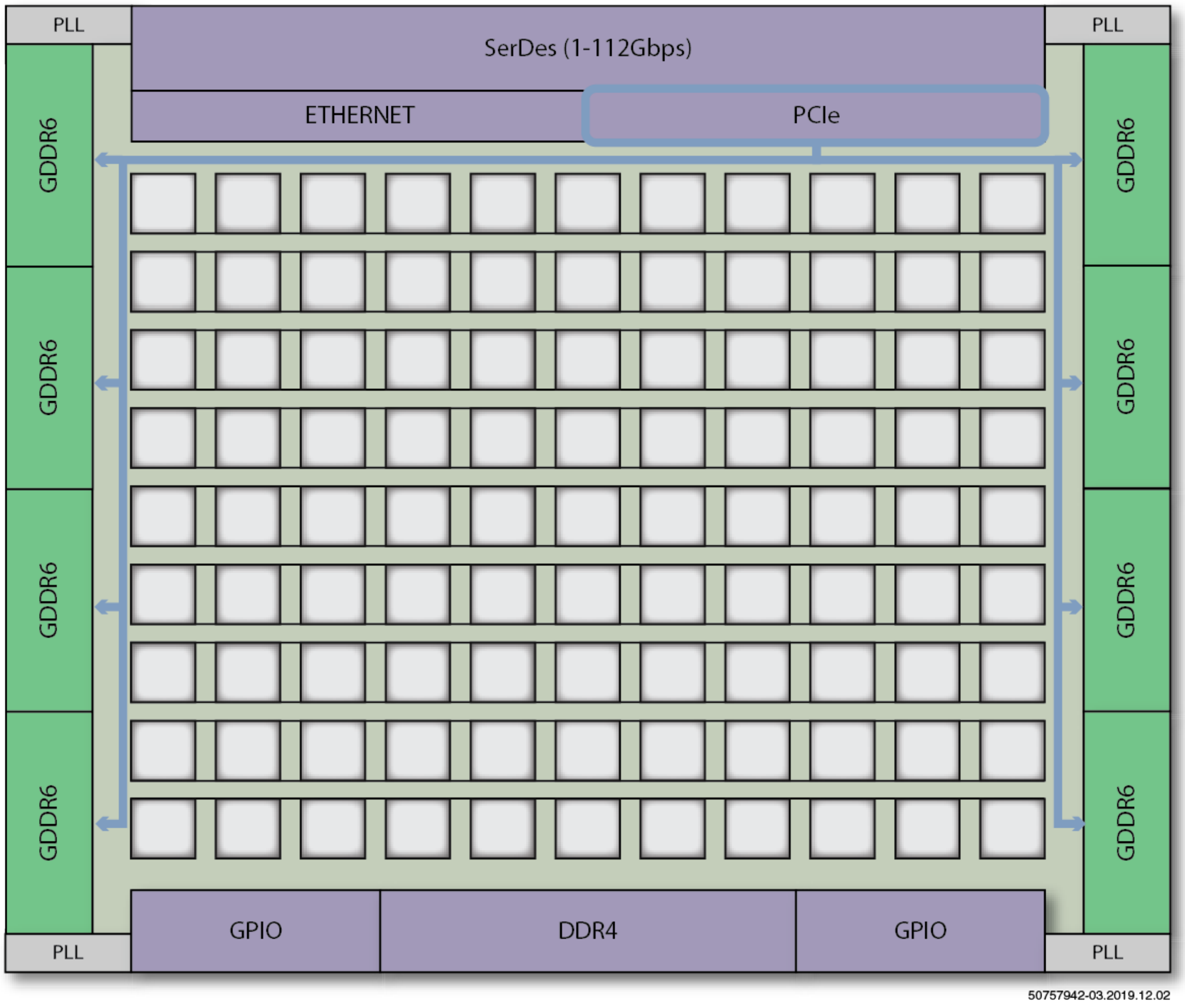

Figure 3: ***PCIe Connected Directly to GDDR6 Interface***

Enable Secure Partial Reconfiguration on Independent FPGA Fabric Blocks

Speedster7t FPGAs, like other SRAM-based FPGAs must be configured at power-up. Speedster7t FPGAs have an on-chip FPGA configuration unit (FCU) that manages the FPGA's initial configuration and any subsequent partial reconfiguration. The FCU is also connected to the NoC offering new levels of flexibility when configuring the FPGA. Using the NoC to transmit configuration bitstreams to the Speedster7t FCU allows for new ways to approach FPGA configuration that were not previously available.

The Speedster7t NoC is available for certain read/write transactions prior to device configuration: PCIe to GDDR6, PCIe to DDR4, and lastly PCIe to FCU. Once the PCIe interface is set up, the FPGA can receive configuration bitstreams via the PCIe interface destined for the FCU to configure the rest of the device. Once at the FCU, the configuration bitstream is written to FPGA fabric to configure the device. After the device is configured, designers have the flexibility to reconfigure portions of the FPGA (partial reconfiguration) to add new functionality or increase acceleration performance without shutting down the FPGA. New partial reconfiguration bitstreams can be sent through the PCIe interface to the FCU to reconfigure any part of the device. When part of the device is reconfigured, any data coming into or out of the newly configured region can be easily accessed in the Speedster7t1500 device by instantiating a NAP in the desired region to communicate with the NoC. The NoC removes the complexity of traditional FPGA partial reconfiguration in that the user does not have to worry about routing around existing logic functions and impacting performance or not being able to access certain device pins due to existing logic in that region. This capability saves designers time and provides increased flexibility when using partial reconfiguration.

Additionally, partial reconfiguration allows designers to adapt the logic within a device as workloads change. For example, if the FPGA is performing a compression algorithm on incoming data and that compression is no longer needed, the host CPU can tell the FPGA to reconfigure, loading a new design optimized to handle the next workload. Partial reconfiguration can be done independently at the fabric cluster level, all while the device is still operational. A clever use case is to develop a self-aware FPGA that through the use of a soft CPU which monitors device operations to initiate a partial reconfiguration in real time to turn off logic to save power, or to add more accelerator blocks in the FPGA fabric to temporarily handle a large increase in incoming data. These capabilities offer designers more configuration flexibility than ever before.

Easily Supports Hardware Virtualization

The Speedster7t NoC provides designers the unique ability to create virtualized secure hardware within a single FPGA by leveraging the NAP and its AXI interface. Connecting a programmable-logic design directly to the NoC requires little more than instantiating a NAP and its AXI4 interface into the logic design. Each NAP also has an associated address translation table (ATT) that converts a logical addresses at the NAP to the physical address on the NoC. The NAPs' ATTs permit programmable-logic blocks to use local addresses while mapping NoC-directed transactions to addresses as assigned by the NoC's global memory map. This remapping feature can be used in many ways. For example, it can be used to allow all identical copies of an acceleration engine to use virtual, zero-based addressing, while sending traffic from each acceleration engine to a different physical memory location.

Each ATT entry also contains an access-protection bit to preclude that node from accessing proscribed address ranges. This feature provides a significant inter-process security mechanism that prevents multiple simultaneous applications or multiple tasks running on one Speedster7t FPGA from interfering with memory blocks assigned to other applications or tasks. This security mechanism also helps prevent system crashes due to unintended, accidental, or even intentional memory-address conflicts. Additionally, designers can prevent logic functions from accessing entire memory devices using this scheme.

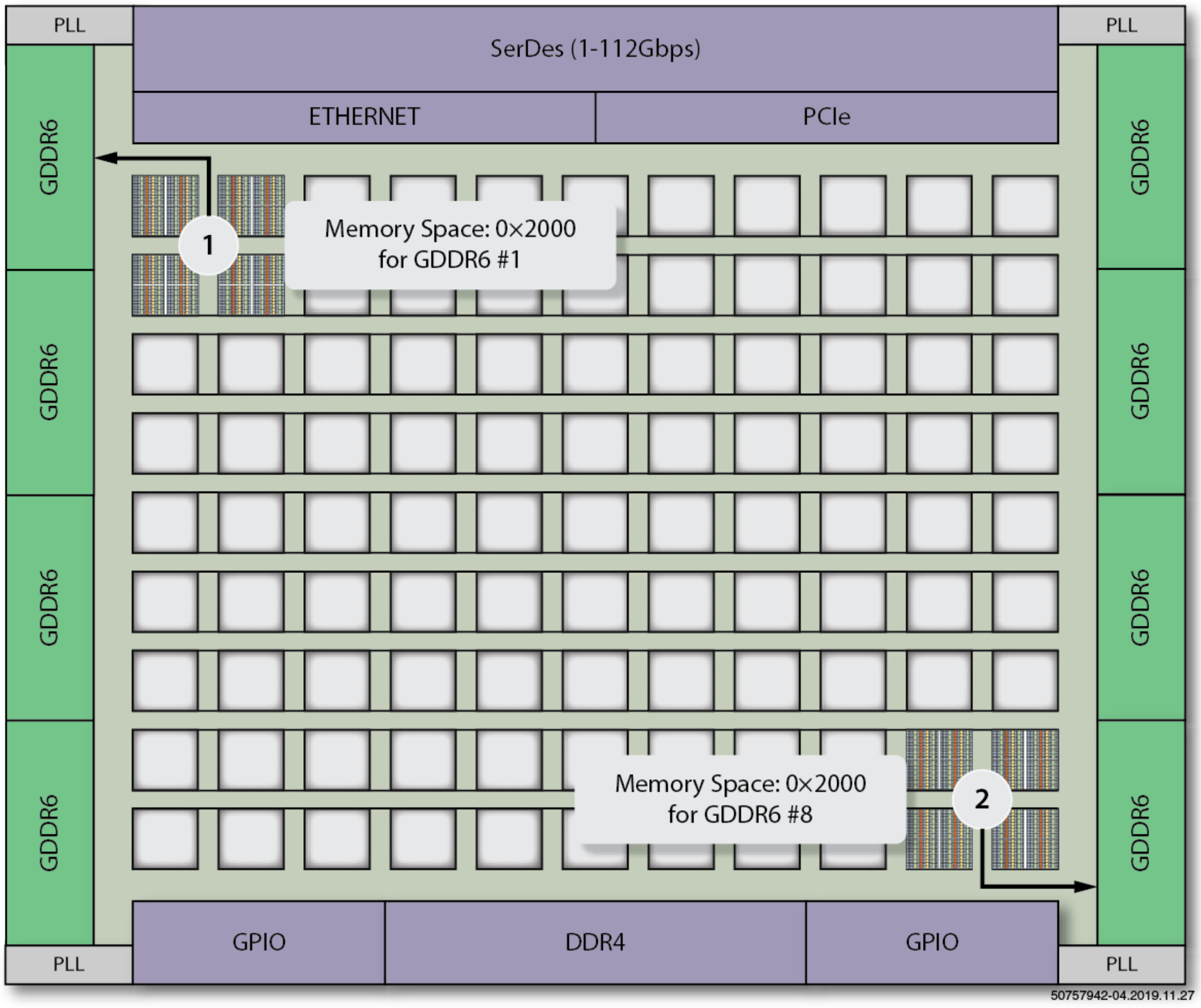

Figure 4: ***Hardware Virtualization using the Speedster7t NoC***

Simplified Team-Based Design

Team-based design for FPGAs is not a new concept, but the underlying architecture and routing dependencies on other portions of the FPGA made implementing this simple concept quite challenging. Once one portion of the design was finishing by a team, a second team working a new portion of the design often ran into challenges when trying to access resources on the other side of the device that required routing through the portion of the design already completed. Also, changing the region or size of the FPGA where a portion of a design is routed could cause ripple impact to all other FPGA design blocks.

Using the Speedster7t NoC, design blocks can be mapped to any portion of the FPGA and undergo changes to resource allocations without impacting timing, placement or routing of other FPGA blocks. Team-based design is made possible because all of the NAPs in a device allow each deign block unrestricted access to the NoC for communication. Therefore, if a certain portion of a design grows in size, as long as there are enough FPGA resources available, the data flows are automatically managed by the NoC, freeing designers from the worry about meeting timing and possible follow-on impacts to other parts of the design being worked on by other team members.

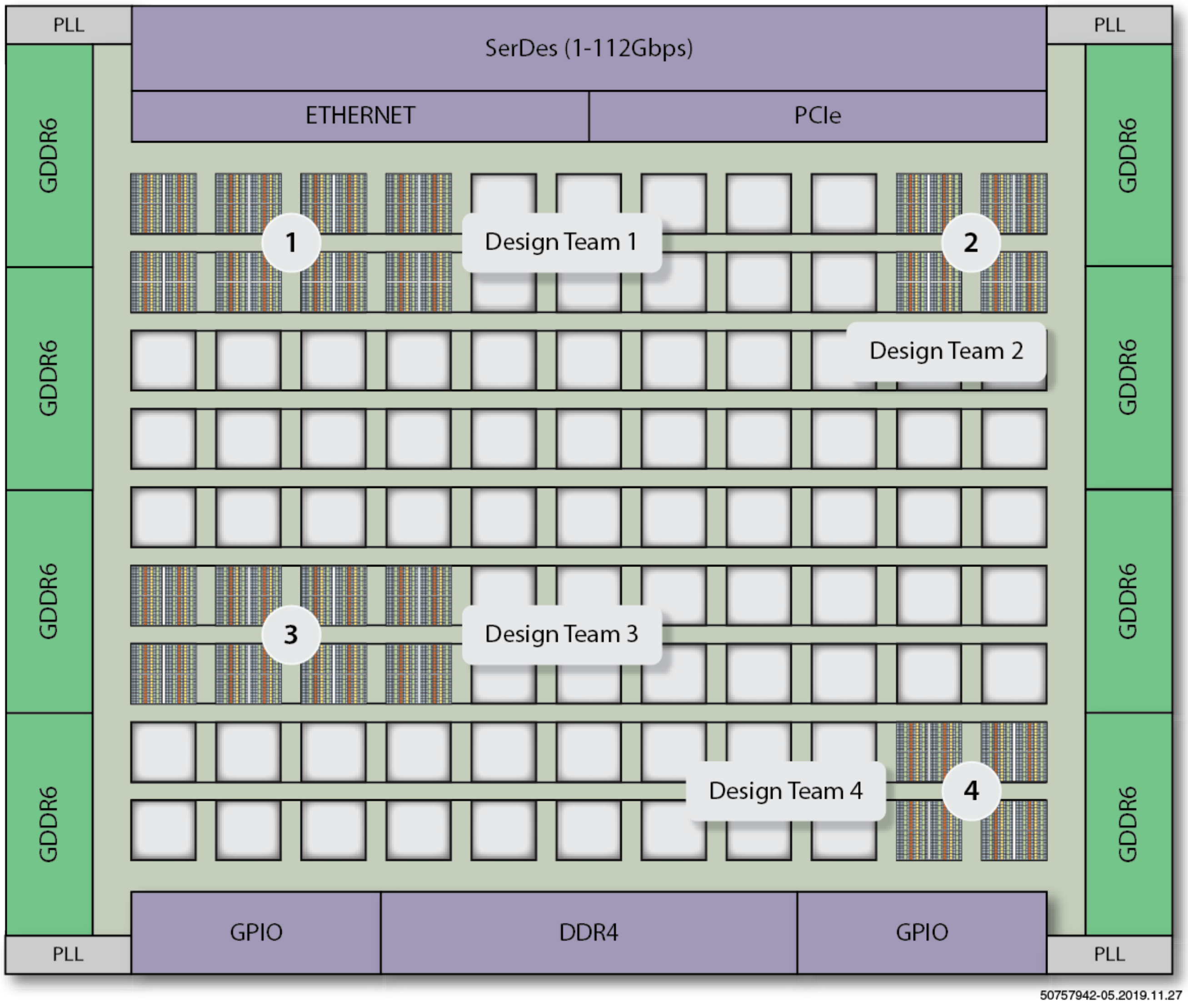

Figure 5: ***Multiple Design Teams Targeting the Same FPGA***

Faster Design Through Independent Interface and Logic Verification

Another unique feature of the Speedster7t NoC is the ability to allow designers to configure and validate I/O connections independently from the user logic. For example, one design team can validate a PCIe-to-GDDR6 interface while another design team can independently validate internal logic functions. This separation is made possible because the peripheral portion of the NoC connects the PCIe, GDDR6, DDR4 and FCU without consuming any FPGA resources. These connections can be tested without any HDL code, which allows for simultaneous and independent validation of interfaces and logic. This capability eliminates the dependencies between validation steps and allows for overall faster validation than is possible with traditional FPGA architectures.

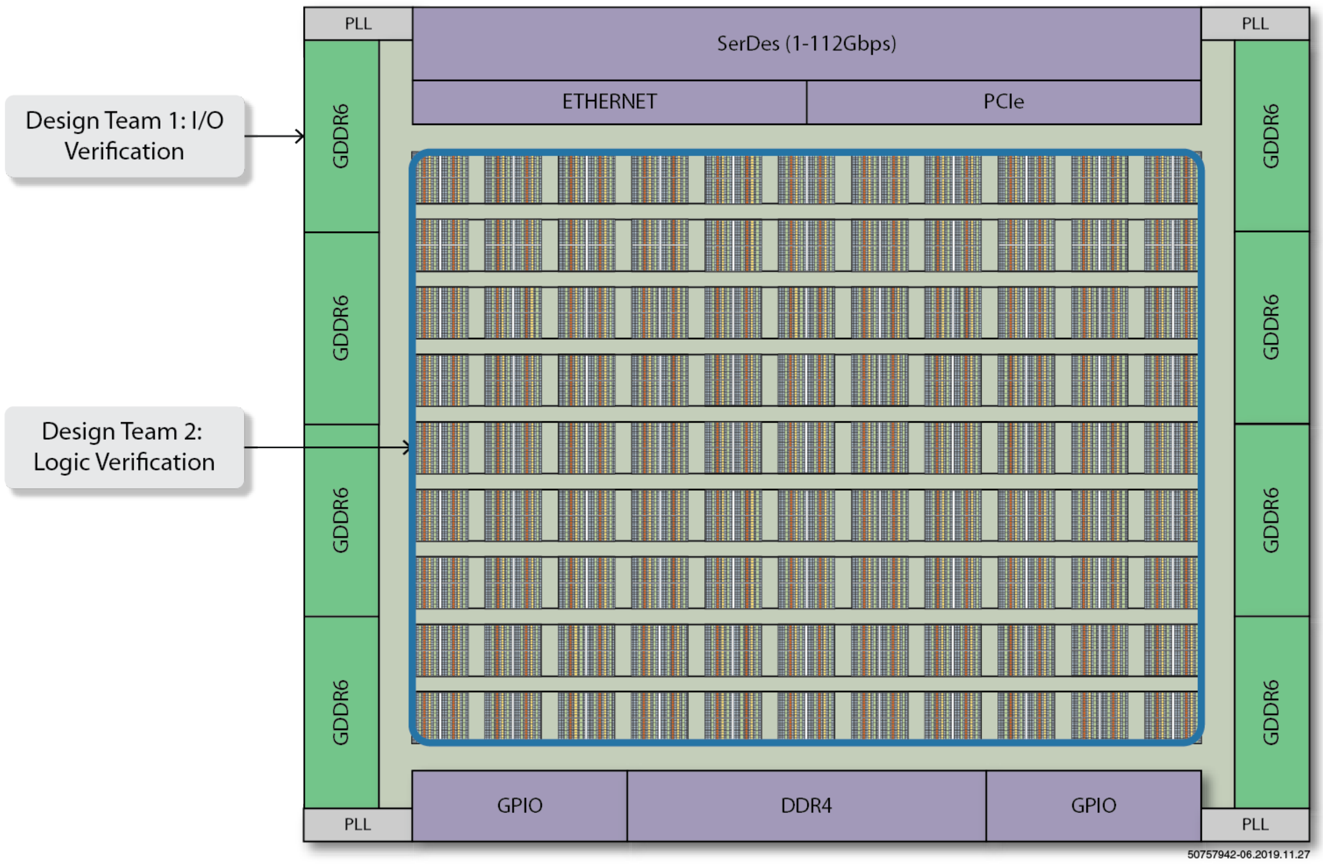

Figure 6: ***Independent I/O and Logic Verification***

Streamlined 400 Gbps Ethernet Applications with Packet Mode

The challenge with implementing high-speed 400 Gbps Ethernet data paths in FPGAs is to find a bus size that meets that performance capability of the FPGA. For 400G Ethernet, the only viable choices for full bandwidth operation are a 1,024-bit bus running at 724 MHz or a 2,048-bit bus running at 642 MHz. Such wide buses are difficult to route because they consume a vast amounts of logic resources within the FPGA fabric and create timing closure challenges at the required rates even in the most advanced FPGAs.

However, in the Speedster7t architecture, designers can use a new processing mode called packet mode where an incoming Ethernet stream is rearranged into four narrower 32-byte packets or four independent 256-bit buses running at 506 MHz. The advantages of this mode include fewer wasted bytes when the packet ends as well as data can be streamed in parallel without having to wait for the first packet to finish before starting the second packet transmission. The Speedster7t FPGA architecture is designed to enable packet mode by connecting the Ethernet MAC directly to specific NoC columns and in turn, from the NoC columns to the fabric using user-instantiated NAPs. Using the NoC column, data can be sent anywhere along that column into the FPGA fabric for further processing. Packet mode is configured using the ACE design tools which greatly simplifies the user design and enables higher productivity when handling 400G bps Ethernet data streams.

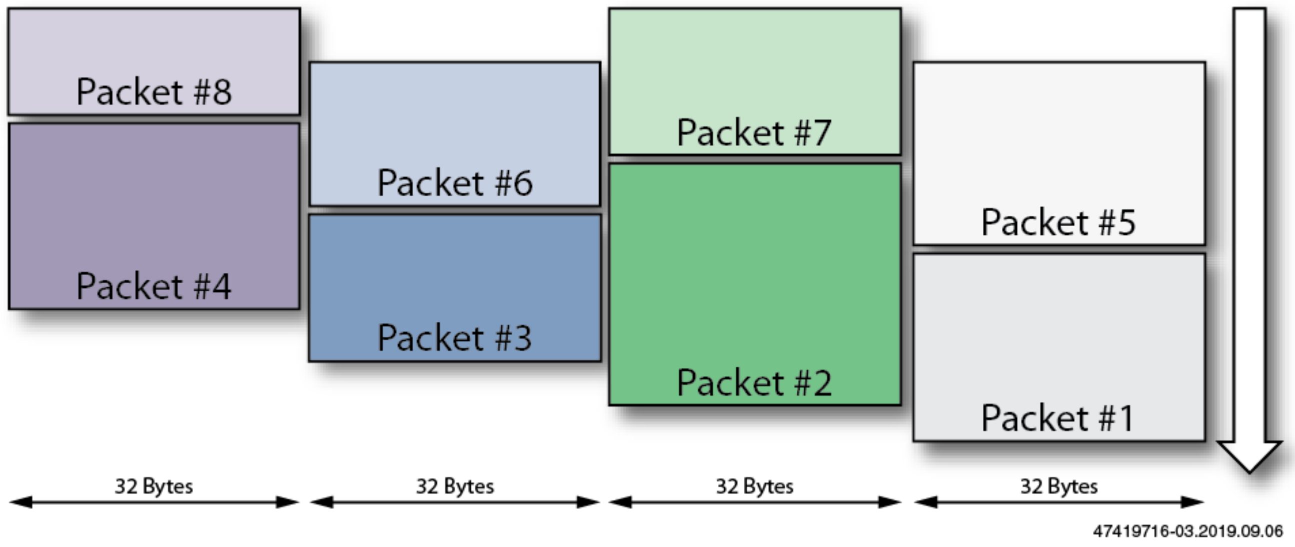

Figure 7: ***Data Bus Rearrangement for Packet Mode***

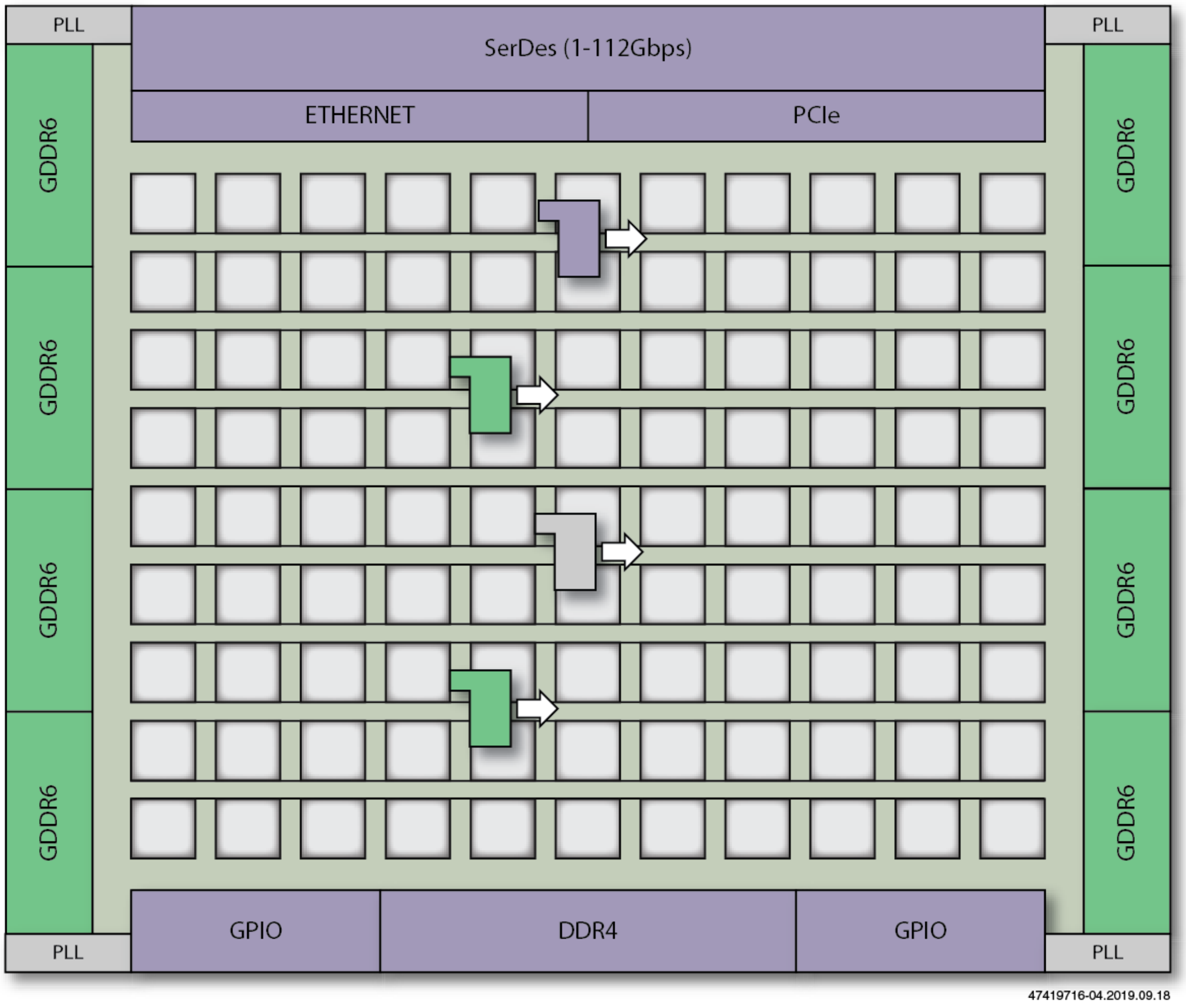

Figure 8: ***400 Gbps Ethernet Using Packet Mode***

Reduced Logic Utilization and Increased Overall FPGA Performance

The Speedster7t NoC allows for much more flexibility and a simpler design approach compared to what was previously available with traditional FPGAs. An underlying benefit is that a NoC automatically reduces the amount of logic needed for a given design. Instead of using the FPGA fabric for inter-block routing, a design can use the NoC. The complexity of connecting design elements into the Speedster7t NoC is automatically managed by the ACE design tools so designers can realize productivity without needing to write HDL code. This approach simplifies the time consuming challenge of achieving timing closure without degrading overall application performance due to routing congestion within the FPGA fabric. The NoC also enables higher device utilization without sacrificing performance in the FPGA and can significantly increase the quantity of LUTs available for computation.

To highlight this benefit, a sample design featuring a 2D input image convolution was created. Each uses the Speedster7t machine-learning processor (MLP) and BRAM blocks with each MLP performing 12 int8 multiplications in a single cycle. Forty 2D convolution blocks were chained together to utilize nearly all of the available BRAM and MLP resources within the device. In total, the 40 instances of the 2D convolution example design were running in parallel and utilized 94% of the MLPs, 97% of the BRAM but just 8% of the LUTs. The remaining 92% of the available LUTs can still be used for other functions.

As more instances were placed into the device, the F_{MAX} for individual blocks did not degrade. The design was able to maintain performance since data coming into and out of each 2D convolution block has direct access to GDDR6 memory from the NAP connected to the NoC versus needing to be routed through the FPGA fabric.

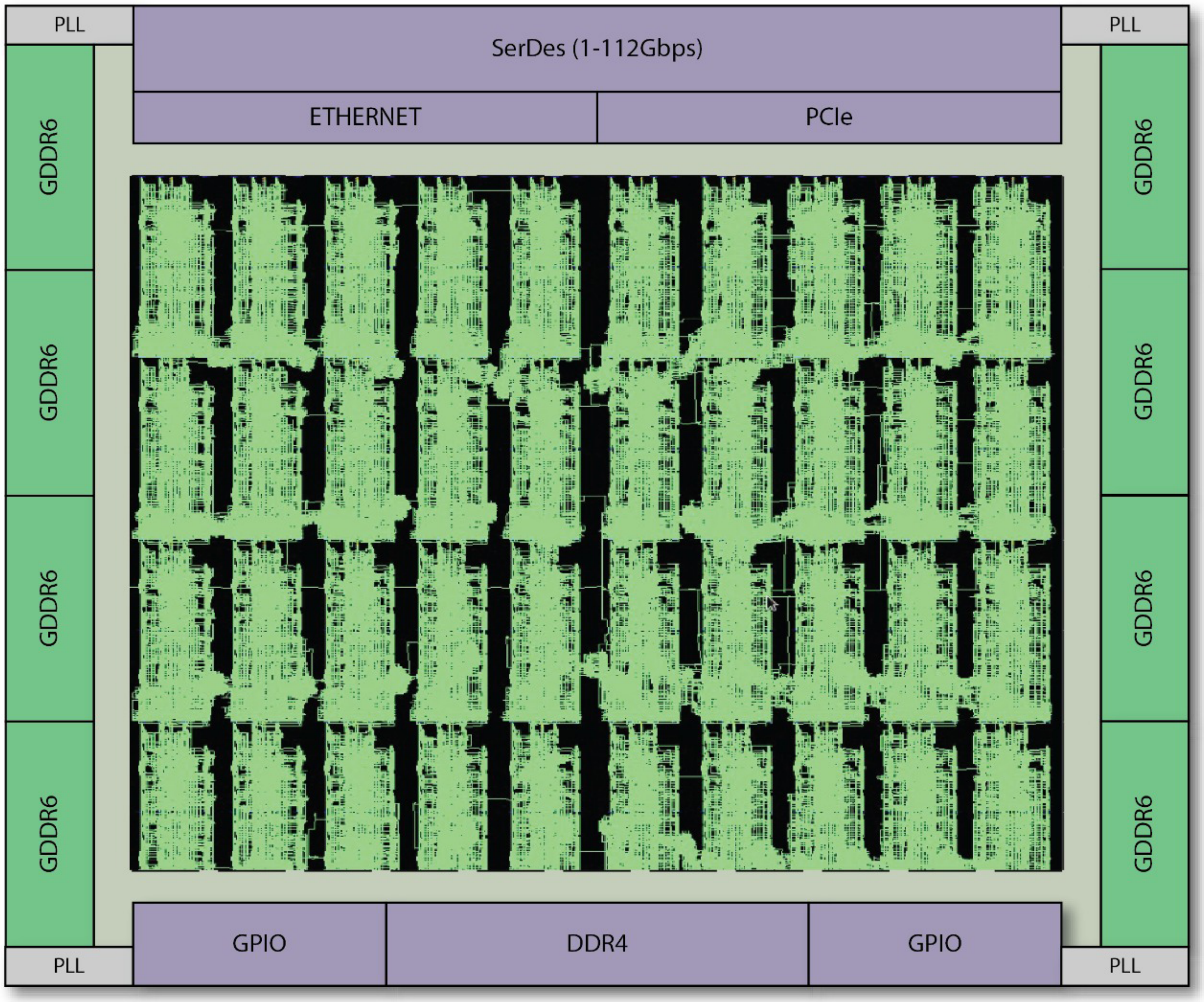

Figure 9: ***A Speedster7t Device with 40 Instances of a 2D Convolution Block***

Conclusion

The Speedster7t NoC enables a fundamental shift in the FPGA design process. Achronix is the first FPGA company to implement a 2D NoC which connects all of the system interfaces and the FPGA fabric. This new architecture allows Achronix FPGAs to be uniquely suitable for high bandwidth applications while significantly improving designer productivity. Because the NoC manages all of the networking functions between data accelerators designed into the FPGA and the high-speed data interfaces, designers need only design their data accelerators and connect them to a NAP primitive. ACE and the NoC take care of everything else. By using the NoC, FPGA designers will benefit from:

- Streamlined High-Speed Data Distribution Throughout the Entire FPGA Fabric
- Automatically Connect PCIe Interfaces to Memory
- Enable Secure Partial Reconfiguration on Independent FPGA Fabric Blocks
- Easily Supports Hardware Virtualization
- Simplified Team-Based Design
- Faster Design Through Independent Interface and Logic Verification
- Streamlined 400 Gbps Ethernet Applications with Packet Mode
- Reduced Logic Utilization and Increased Overall FPGA Performance

To learn more about the Speedster7t FPGA, visit www.achronix.com/speedster7t.

Achronix®

Data Acceleration

Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

Website: www.achronix.com
E-mail : info@achronix.com