Data Orchestration Supports the Next Advance in AI (WP025)

Achronix Data Acceleration

April 13, 2021 **White Paper**

Executive Summary

Artificial intelligence (AI) and machine learning (ML) technologies now power a rapidly expanding range of product and applications from deeply embedded systems to hyperscale data-center deployments. Although there is a huge degree of diversity in the hardware designs supporting these applications, all require hardware acceleration.

Deep-learning technology demands numerous tensor arithmetic operations. To support real-time execution, memory and processor performance must meet far higher targets than is possible with standard software-driven architectures. This need leads to the use of designs based on dedicated hardware accelerators performing parallelized and heavily pipelined tensor-arithmetic operations. To avoid pipeline stalls, data must be in the right place, at the right time and in the right format. Dedicated data orchestration hardware avoids the need for accelerator pipelines to stall, allowing operation at peak efficiency.

Data orchestration encompasses the pre- and post-processing operations that ensure the data seen by a machine learning engine arrives at an optimal speed and in the most suitable form for efficient processing. Operations range from resource management and utilization planning through I/O adaptation, transcoding, conversion and sensor fusion to data compaction and rearrangement within shared memory arrays. How these features are deployed will depend on the performance and cost requirements of the target application but for most use-cases a programmable-logic platform optimized for data ingestion, transformation and movement provides the best data-orchestration strategy for ML accelerators.

Introduction

Deep learning places a high degree of strain on computing hardware. The shift towards dedicated accelerators has provided a way for silicon technology to keep pace with AI development but these units on their own are not enough to satisfy the demand for higher performance at lower cost.

IC vendors and systems houses have, understandably, focused on the raw performance of their matrix and tensor processing arrays. At peak throughput, these architectures can achieve performance levels easily measured in teraoperations per second (TOPS), even for systems aimed at use in edge computing. Though understandable, the focus on peak TOPS carries the risk of underutilized hardware if there are delays caused by data being unavailable or in need of conversion into the correct format for each model layer.

Systems must compensate for network and storage delays and ensure data elements are correctly formatted and located to stream at a consistent rate into and out of the AI accelerator. Data orchestration provides the means to ensure correct data format and position on each clock cycle and, thus, maximize system throughput.

Because of the complexity of typical AI implementations, there are numerous tasks that must be handled by a data-orchestration engine, whether located in a data center, an edge-computing environment, or real-time embedded applications such as automated driver assistance system (ADAS) designs. Tasks include:

- Data manipulation
- Scheduling and load balancing across multiple vector units
- Packet inspection for data corruption such as that caused by a faulty sensor

Though it is possible to implement these functions by adding data-control and exception-handling hardware to the core processing array, the sheer variety of operations that might be needed, and the need for flexibility as AI models evolve, makes hardwiring the functions into the core accelerator silicon a potentially expensive short-term option. For example, in some environments encryption support is rapidly becoming a requirement to ensure high data security but differing levels of encryption may be used depending on the application sensitivity of the data in each layer. Fixed-architecture solutions run the risk of being unable to adapt to changing requirements.

One possible approach is to use a programmable microprocessor to control the flow of data through an accelerator. The problem with this approach is that software execution is simply unable to keep pace with the demands of accelerator hardware. A more hardware-centric response to data orchestration is needed, making it possible to have the accelerator design focus purely on core pipeline efficiency. An external data-orchestration can handle all of the memory and I/O management ensuring the flow of operands and weights is uninterrupted. As the data-orchestration engine must handle revisions and changes to application and model design, hardwired logic is not a suitable approach. Programmable logic allows modifications and avoids the risk of the data-orchestration engine becoming outdated.

In principle, the FPGA's combination of distributed memories, arithmetic units and lookup tables provide a mixture of features highly suited to the real-time reorganization, remapping and memory management of the streaming data required by AI-driven applications. FPGAs allow the creation of customized hardware circuits supporting the dense data streams of deeply pipelined AI accelerators while enabling users to change implementations to suit new architectures as required. However, the performance requirements of data orchestration call for new approaches to FPGA design.

Data Orchestration Use-cases

There are many different types of data-orchestration architectures in use-cases found in data-center, edge-computing and embedded-systems deployments. For example, in the data-center environment, multiple accelerators may be deployed on a single model, with their data throughput managed by one or more data-orchestration engines.

Inferencing systems require data orchestration to ensure maximum utility of each worker engine to avoid bottlenecks and ensure that incoming data samples are processed as quickly as possible. Distributed training adds a requirement for rapid neuron weight updates. Those updates must be distributed as quickly as possible to other workers handling relevant model parts to avoid stalling. Data-orchestration logic in an FPGA allows handling a wide range of weight-distribution and synchronization protocols to support efficient operation, while relieving much of the burden of data organizing from the accelerator itself. The following figure shows a possible implementation, with one FPGA managing multiple AI engines on the same board. With a suitable low-overheard communications protocol, the individual ML ASICs need no memory controller. Instead, the data-orchestration engine organizes all weights and data elements in local memory and simply feeds them in the correct order to each of the ASICs that it manages. The result is high performance at lower overall cost by reducing duplicated memory and interface logic.

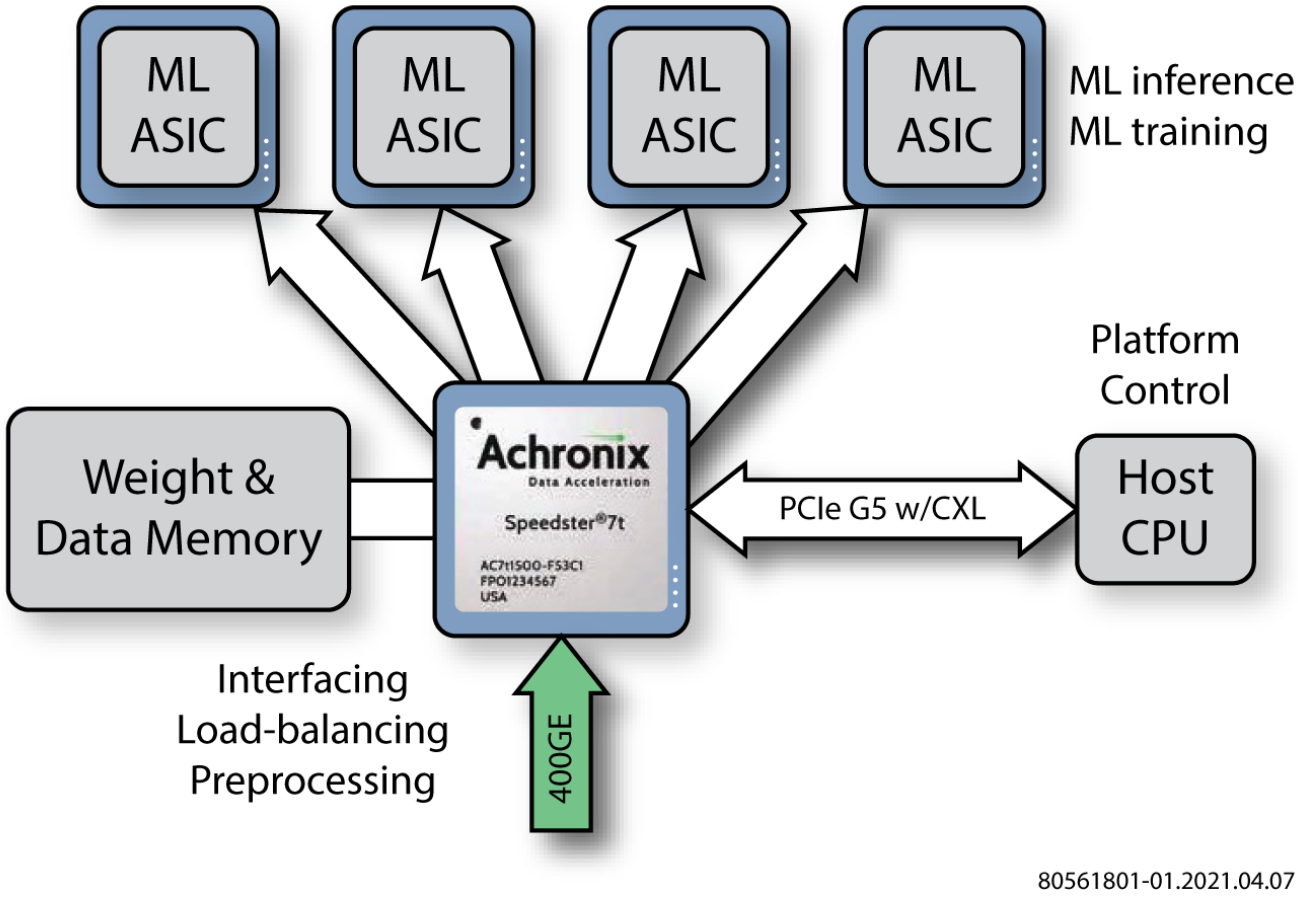

Figure 1: ***Data Orchestration Can Provide Load Balancing and Other Data-forwarding Functions at Speed for Parallelized AI Implementations***

With data orchestration, the hardware can further improve performance without increasing cost. One option takes advantage of compression of network or system bus data avoiding the use of more expensive interconnects. The logic-level programmability of FPGAs allows compression and decompression of data through network interfaces. Data-orchestration hardware also allows using forward-error correction protocols to ensure valid data is fed at full pipeline speeds. In most designs, corruption events will typically be rare but recovery is expensive for heavily pipelined accelerator designs if external support for correction is not available.

Figure 2 shows that there are numerous ways a data orchestration engine can optimize data flow and presentation to ML engines. For example, the format and structure of individual data elements provide a major opportunity to exploit the benefits of data orchestration since source data generally must be presented in a format that is amenable to feature extraction by the deep neural network (DNN).

In image recognition and classification applications, pixel data is often channelized so that each color plane can be processed separately before the results are aggregated by pooling layers that extract shape and other high-level information. The channelization assists the recognition of edges and other features that might not be easily recognized with combined RGB representations. More extensive transformations are performed in speech and language processing. Data is often mapped into a form more readily handled by the DNN. Rather than processing ASCII or Unicode characters directly, words and subwords to be processed are converted for the models into vectors and one-hot representations. Similarly, speech data might not be presented in the form of raw time-domain samples but, instead, converted into a joint time-frequency representation allowing important features to be more readily recognized by the early DNN layers.

Though the data transformations could be performed by arithmetic cores within an AI accelerator, it is likely not a good fit for the tensor engines. The nature of the reformatting makes it a suitable candidate for processing by an FPGA-based module. FPGAs enable the transforms to be performed effectively at wire speed without the delays incurred by software running on a general-purpose processor.

In real-time and embedded applications involving sensors, further benefits exist from preprocessing the data. For example, though DNNs can be trained to remove the effects of noise and changes in environmental conditions, their reliability can be improved using front-end signal processing to denoise or normalize data. In an automotive driver assistance system (ADAS) implementation, the camera systems must handle changes in lighting conditions. Frequently, a high level of dynamic range in the sensor can be exploited by using brightness and contrast adjustments. An FPGA can perform the necessary manipulation to feed the DNN a stream of pixels with less variability.

Sensor fusion, an increasingly important aspect of ADAS design, helps to improve the performance of the end system. Because environmental conditions can make single-sensor data hard to interpret, the AI model must efficiently take inputs from numerous different sensor types, including cameras, LIDAR and radar.

Format conversion is vital. For example, LIDAR provides depth information for objects in a cartesian space while radar operates on a polar coordinate system. Many models perform sensor fusion more readily by converting one coordinate space into the other. Similarly, image data from multiple cameras must be stitched together and transformed using projections that deliver the most useful information to the AI model.

Lower-level conversions are also needed. Vehicle OEMs buy sensor modules from a variety of suppliers, each of which interpret the connectivity communications standards in their own way. This calls for functions to parse data packets sent by these sensors over the in-vehicle network and convert the data into a standard format that the DNN can handle. Because of security concerns, modules must also authenticate themselves to the ADAS unit and, in some cases, send encrypted data. Data-orchestration silicon allows offloading the decryption and format conversion from the AI accelerator engines.

Further optimizations are possible by using front-end signal-processing functions implemented in a data-orchestration subsystem to strip out unnecessary data. For example, sensors handling inputs from microphones and other 1D sensors can remove silence or low-level background noises and reduce the number of video frames delivered when a vehicle is stationary, reducing the load on the AI engines.

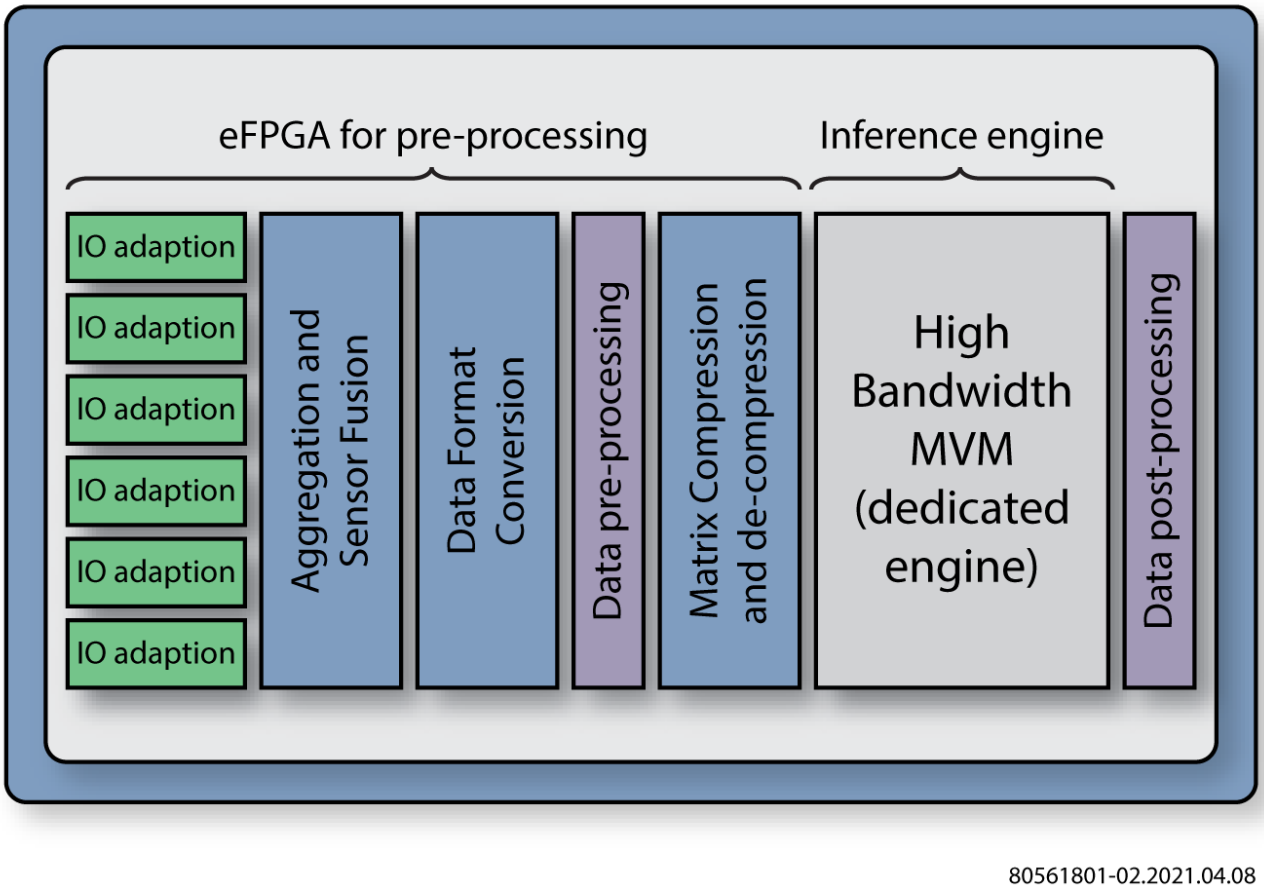

Figure 2: ***Data Orchestration Provides a Number of Options for Accelerating AI Functions***

An architecture optimized for data orchestration

Though the combination of configurable interconnect and programmable logic within an FPGA lends itself to data-orchestration tasks, FPGA architectures are not created equal. How they handle the demand for high-bandwidth data is key. Traditionally, FPGAs were not expected to act as a core element of the datapath but primarily provided control-plane assistance to processors that interact with memory and I/O. Data orchestration requires ingesting, transforming and managing data elements on behalf of processor and accelerator cores, putting an enormous strain on traditional FPGA architectures.

To support the bandwidth requirements of data orchestration, conventional FPGAs require extremely wide buses to handle the multiple streams passing through PCI Express and multigigabit Ethernet interfaces. For example, to support transfers of over 400Gb/s Ethernet data, a designer must route, using a programmable interconnect, a bus that is some 2048 bits wide to reliably meet timing, which would typically entail a clock operating on the order of several hundred megahertz. Such wide interconnects are very difficult to route due to congestion and timing closure issues with such large structures. The interconnects can consume hundreds of thousands of lookup tables (LUTs) that cannot be used to perform data orchestration or format conversion.

The Achronix Speedster7t FPGA overcomes issues faced by conventional FPGAs in part by its adoption of a dedicated 2D network-on-chip (NoC), an interconnect that can, with multiple parallel operations in different parts of the network achieve aggregate bandwidths as high as 20Tb/s. Not only does the 2D NoC provide a huge upgrade in speed relative to the FPGA fabric interconnect, the 2D NoC is able to move massive quantities of data between multiple PCIe Gen5, 400Gbps Ethernet ports and GDDR6 memory interfaces at higher speeds without consuming any of the FPGA's programmable resources.

In the Speedster7t FPGA, the NoC provides a 2D interconnect across the surface of the FPGA. It delivers data packets to soft cores anywhere within the device using dedicated network access points (NAPs). Each NAP provides access to a block of programmable logic or a hardware resource within the FPGA through an industry standard AXI port structure. There are independent NAPs for east-west and north-south traffic, providing additional flexibility and performance for logic that accesses the 2D NoC. This directional split helps optimize the latency of transfers that start and finish on the same 2D NoC path. Routing onto an orthogonal 2D NoC path adds a small, deterministic delay.

An important capability offered by the 2D NoC is Packet Mode, which is designed to make it easier to rearrange data arriving on high-bandwidth ports such as Ethernet into multiple data streams. Packet Mode makes possible the separation of packets arriving on a 200Gb/s or 400Gb/s Ethernet port and to deliver them to different soft cores. This packet separation is shown in the following figure with successive packets routed to different parts of the FPGA. As a result, Packet Mode enables easy creation of load-balancing architectures that would be difficult using a traditional FPGA.

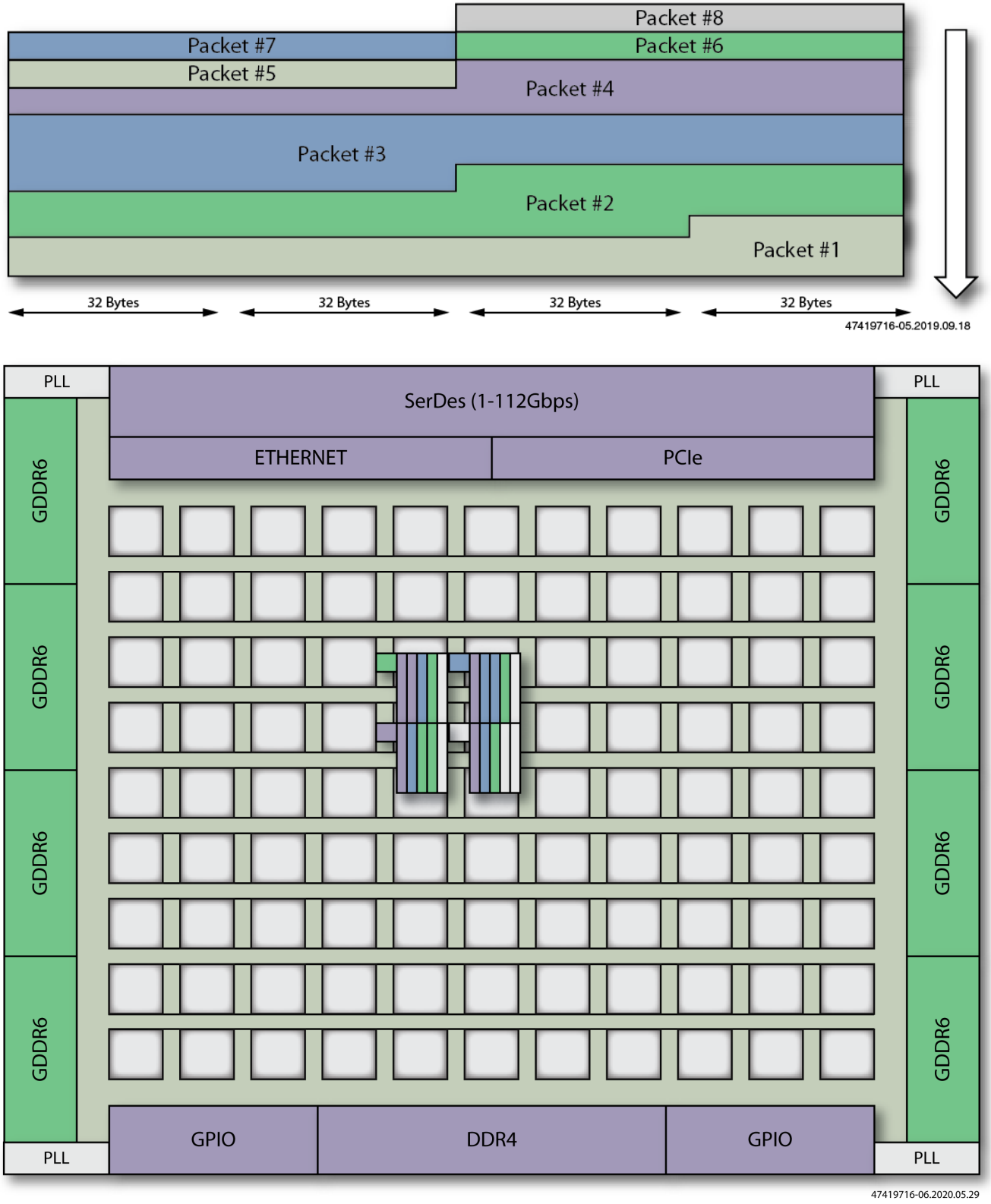

Figure 3: ***The NoC's Packet Mode Supports the Automated Distribution of Network Payloads to Different Parts of the Fabric***

As a further benefit, the 2D NoC supports partial reconfiguration far more easily: each logic block within the 2D array acts as an isolatable resource that can swap in new functionality without affecting any other logic blocks. This feature is further enhanced by virtualization and translation logic implemented by the 2D NoC and NAP controllers. An address translation table acts in a manner similar to the memory-management unit in a microprocessor to prevent tasks from interfering with each other's data. Address-translation tables in the NAP mean each soft core can access the same virtual address range but access external physical memory in completely different ranges. Access-protection bits provide further security, preventing cores from accessing protected address ranges. This level of protection is likely to become important in a range of AI-based applications where data-orchestration and other programmable-logic functions are implemented by different teams before being integrated into the end product.

As well as highly flexible data routing, data orchestration demands the application of fast arithmetic functions to augment the core AI accelerators. The Speedster7t FPGA deploys an array of machine-learning processor (MLP) blocks. Each MLP is a highly configurable, compute-intensive block, that can be arranged as up to 32 multipliers delivering performance of up to 60 TOPS. The MLP supports integer formats from 4 to 24 bits and various floating-point modes including direct support for Tensorflow's bfloat16 format and block floating-point (BFP) format. The surrounding programmable logic fabric provides many ways to optimize data flow taking full advantage of the data reuse and throughput opportunities offered by the MLP.

Given the wide range of environments in which data-orchestration hardware makes sense, a clear requirement for flexible deployment exists. Data-center applications may call for the use of one or more discrete, high-capacity devices such as the Speedster7t FPGA to route and preprocess traffic for multiple ML engines on a single board or distributed within a tray or rack. For edge-computing applications where size, power and cost, are major constraints, there are clear arguments for an SoC solution. Achronix is the only company that offers embedded FPGA (eFPGA) IP technology in addition to discrete FPGAs and is, therefore, in the unique position to support cost-reduction programs where the programmable-logic and interconnect functions can be incorporated into an SoC as shown in the following figure. Speedcore eFPGA IP uses the same technology as found in the Speedster7t FPGA, allowing for a seamless conversion from a Speedster7t FPGA to an ASIC with a Speedcore block. Customers can expect to get up to a 50% power reduction and up to a 90% unit cost reduction when using Speedcore IP to convert Speedster7t FPGAs to ASICs.

A further option is to employ chiplets in a multichip module. This provides the benefits of high-speed interconnects between the co-packaged FPGA-based data-orchestration modules and the ML engines. Achronix supports all of these implementation choices.

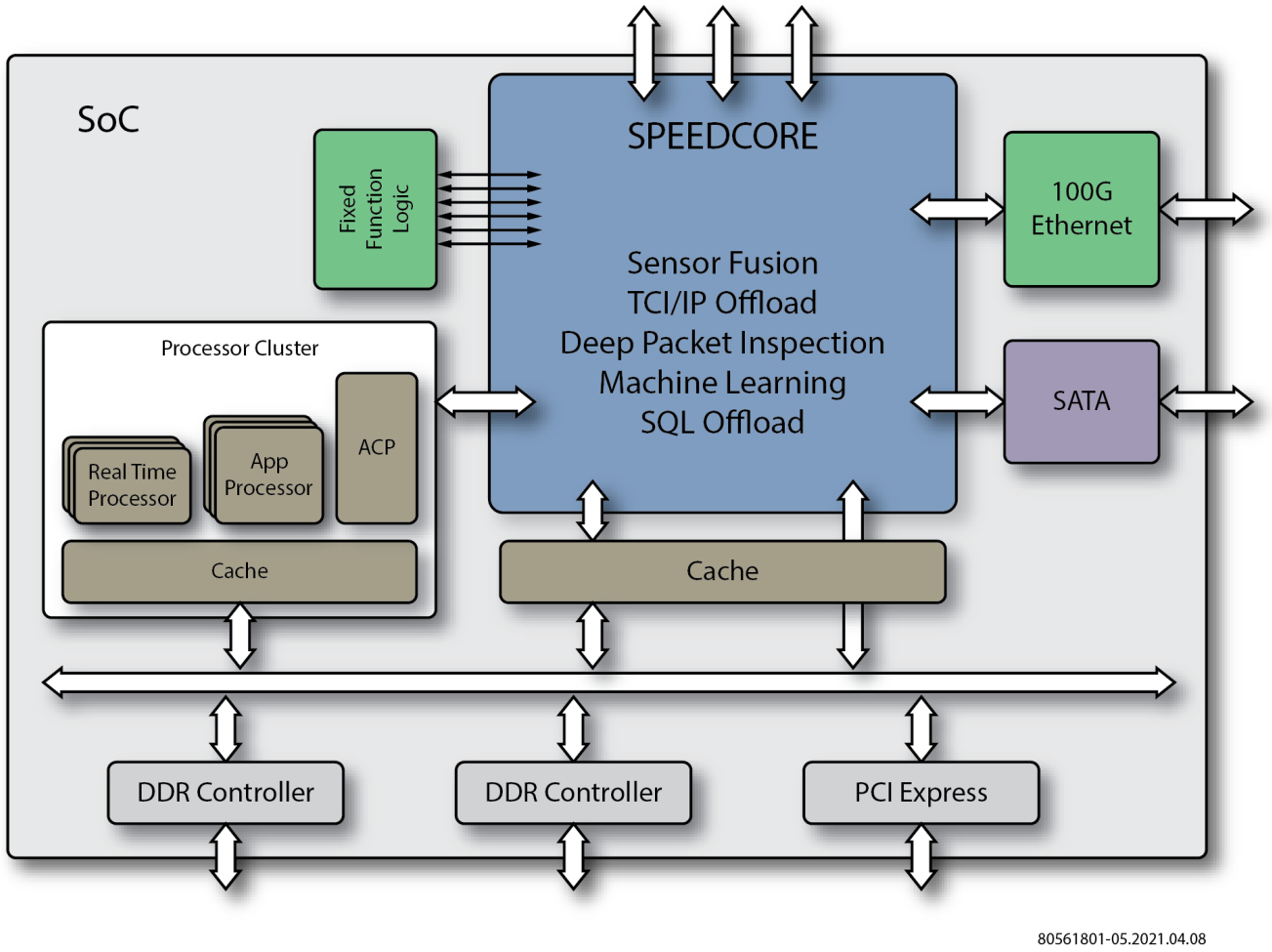

Figure 4: ***Embedded FPGA Technology Provides the Ability to Integrate Data Orchestration into Accelerator Silicon***

Conclusion

The rapid evolution of deep learning is placing immense pressure on the hardware architectures needed to implement the technology at scale. Although intense focus on peak TOPS scores driven by the realization that performance is an absolute requirement, intelligent data orchestration and management strategies provide a means to deliver systems that are highly cost and energy efficient.

Data orchestration encompasses a number of pre- and post-processing operations ensuring the data seen by an ML engine arrives at an optimal speed and in the most suitable form for efficient processing. Operations range from resource management and use planning through I/O adaptation, transcoding, conversion and sensor fusion to data compaction and rearrangement within shared memory arrays. Some orchestration engines use a subset of these features depending on the core requirements of the target ML architecture.

The Achronix Speedster7t FPGA architecture provides a highly flexible platform for these data-orchestration strategies. The FPGA, with its high throughput, low latency and extreme flexibility, delivers the data in a form that enables even highly specialized accelerators to adapt to changing requirements. Further, the Speedster7t FPGA's extensive logic and arithmetic capacity coupled with high-throughput interconnects allows the holistic design of front-end signal conditioning and back-end machine learning to maximize overall efficiency.

Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

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

Notice of Disclaimer