WP492 (v1.0.1) June 13, 2017

Xilinx All Programmable Devices: A Superior Platform for Compute-Intensive Systems

By: Cathal Murphy and Yao Fu

Xilinx® All Programmable FPGAs and SoCs offer the most efficient, cost effective, lowest latency, design-flexible, and future-proofed compute platforms for an array of compute intensive workloads.

ABSTRACT

To keep pace with the growing demand for data processing, tomorrow's systems require a step change in compute capability. Yesterday's solutions (e.g., x86 processors) can no longer deliver the compute bandwidth required in an efficient, cost-effective manner, leaving system designers searching for a new compute platform.

FPGAs and GPUs are increasingly viewed by system designers as the compute platforms that can deliver on tomorrow's requirements.

This white paper analyzes GPUs and Xilinx FPGA and SoC devices in the context of providing the necessary compute efficiency and flexibility for this new era.

Introduction

Tomorrow's systems (e.g., Cloud Data Centers [DC] and autonomous vehicles) require a step change in compute capabilities to support an expanding set of workloads and their underlying, evolving algorithms[Ref 1]. For example, big data analytics, machine learning, vision processing, genomics, and advanced driver assistance systems (ADAS) sensor fusion workloads are all pushing compute boundaries beyond what existing systems (e.g., x86 based systems) can deliver in a cost effective and efficient manner.

System architects are searching for a new compute platform that can address these requirements. The platform needs to be flexible enough to integrate into existing infrastructure and to support the range of workloads and their evolving algorithms. Additionally, many of these systems must offer low and deterministic latency to support the fast response time required by real-time systems e.g., autonomous vehicles.

Graphics processing unit (GPU) vendors have aggressively positioned GPUs as the compute platform of choice for this new era, mainly on the back of success in high performance computing (HPC) with machine learning training. During this effort, GPU vendors have modified their architecture to target machine learning inference workloads.

However, fundamental GPU architecture limitations are continually overlooked by GPU vendors. These limitations significantly impact the GPU's ability to deliver on the necessary compute in a cost-effective and efficient manner at the system level. For example, in the cloud DC systems, demand for various workloads vary substantially throughout the day. Furthermore, the underlying algorithms of these workloads are evolving at a rapid pace. The limitations of the GPU architecture prevent many of today's workloads and tomorrow's evolved workloads from mapping to a GPU, resulting in idle or inefficient hardware. The Limitations of GPU Architecture section of this white paper explores these limitations in more detail.

Conversely, Xilinx FPGAs and SoCs have many key attributes that make them an ideal choice to address the challenges of tomorrow's system requirements. These unique attributes include:

- Massive compute power and efficiency for all data types
- Massive flexibility to maximize realizable compute and efficiency benefits for a large array of workloads
- I/O flexibility for easy integration into the system and higher efficiency
- Massive on-chip memory caches for increased efficiency and lowest latency

The Unique Benefits of Xilinx FPGAs and SoCs section of this white paper provides information about the benefits of Xilinx architecture, comparing and contrasting to the GPU architecture and its limitations.

GPU Origins and Target Workloads

GPUs origins are rooted in the PC era, with NVidia laying claim to the world's first GPU in 1999, but many other graphics devices preceded it[Ref 2]. GPUs have been designed from the ground up to offload/accelerate graphics tasks, e.g., shading and transforming of pixel arrays from the CPU, resulting in an architecture that is highly parallel with high throughput[Ref 3]. In essence, the GPU's main purpose has been to render high quality images for visual display units (VDUs).

Over the years, a small number of non-graphics but massively parallel and memory bound workloads have benefited from an implementation on a GPU rather than a CPU, e.g., medical imaging that requires large matrix math. GPU vendors recognized the opportunity to broaden the market for GPUs into non-graphics applications, leading to the development of non-graphics based programming languages for GPUs, e.g., OpenCL. These programming languages essentially turn a GPU into a general-purpose GPU (GPGPU).

Machine Learning

In more recent times, one of the workloads that has mapped well to GPU implementation is machine learning training. Leveraging GPUs has resulted in a substantial reduction in training time for deep neural networks.

GPU vendors are attempting to leverage their success in machine learning training to gain traction in machine learning inference (deployment of trained neural networks). As machine learning algorithms and the required data precisions evolve, GPU vendors have been tweaking their architectures to stay relevant. One example of tweaking is NVidia's support of INT8 in their Tesla P4 product. However, even lower precision, such as binary and ternary, is being explored by many users today[Ref 4]. To take advantage of such advancements in machine learning and other areas, GPU users must wait and buy new hardware when or if it becomes available. As described later in this white paper, there is no need to wait or buy new hardware for users of Xilinx's FPGAs and SoCs because of their inherent flexibility.

Machine learning is the basis of the GPU vendors' attempts to establish themselves as the compute platform of choice for this new era of compute. But to understand a GPU's suitability for tomorrow's systems, a more holistic, system-level analysis needs to be performed, taking into account the many limitations of the GPU architecture as well as how system requirements evolve over time.

Limitations of GPU Architecture

This section of the white paper delves into a typical GPU architecture to uncover its limitations and how they apply to various algorithms and workloads.

Arrays of SIMT ALUs

Figure 1 shows a typical block diagram for a GPU. At the heart of a general-purpose GPU's compute are large arrays of arithmetic logic units (ALUs) or cores. These ALUs are generally considered to be single instruction multiple thread (SIMT), which is similar to single instruction multiple data (SIMD).

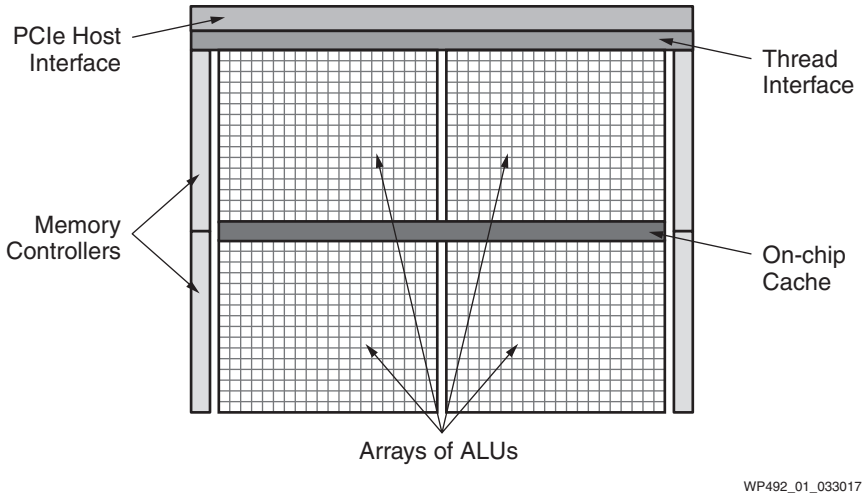

Figure 1: **GPU Block Diagram**

The basic principle is that workloads are broken up into thousands of parallel threads. A very large number of GPU threads are required to prevent ALUs from lying idle. These threads are then scheduled so that groups of ALUs are performing the same (single) instruction in parallel. Leveraging SIMT allows GPU vendors to realize area and power-efficient implementation relative to a CPU, because much of a core's resources can be shared with other cores in the same group.

However, it is clear that only certain workloads (or parts of a workload) can be mapped efficiently onto this massively parallel architecture[Ref 5]. If enough commonality or parallelism is not found among the threads that make up a workload (e.g., sequential or moderately parallel workloads), ALUs lie idle, resulting in reduced compute efficiency. In addition, threads that make up a workload are scheduled to maximize ALU utilization, introducing additional latency. Even with features such as independent thread scheduling, found in NVidia's Volta architecture, the underlying architecture remains SIMT, as does the need for massively parallel workloads.

For sequential, moderately parallel, or sparse workloads, the compute and efficiency offered by the GPU can even be lower than that offered by a CPU[Ref 6]. One quantitative example of this is in sparse matrix math implementations on a GPU; for a smaller number of non-zero elements, a GPU is below or on par with a CPU from a performance and efficiency perspective[Ref 7][Ref 8]. Interestingly, many researchers are investigating sparse convolution neural networks as a means to exploit the massive redundancy in many convolution neural networks[Ref 9]. This trend clearly

represents challenges for GPUs in machine-learning inference. Sparse matrix math is also a key element in big data analytics.[Ref 10]

Most workloads that contain massive amounts of parallel compute tasking also contain some sequential or moderately parallel elements, meaning a hybrid GPU-CPU system is required to meet system performance requirements[Ref 11]. Clearly, the need for a high-end CPU impacts the efficiency and cost-effectiveness of the platform, and the required communication between the CPU and GPU adds a potential bottleneck to the system.

One additional limitation of the SIMT/GPU architecture is that the ALU's functionality is governed by its fixed instruction set and data type support.

Discrete Data-Type Precision Support

System designers are exploring reduced-data-type precision as one of the means to deliver a step change in compute efficiency—without a significant loss of accuracy[Ref 12][Ref 13][Ref 14]. Machine learning inference is leading the charge to reduced precision, going first to FP16 followed by INT16 and INT8. Researchers are exploring further down the precision curve today, even as far as binary[Ref 4][Ref 15].

GPU ALUs typically offer native support for single-precision floating point (FP32) and, on occasion, double-precision floating point (FP64). FP32 has been the precision of choice for graphics workloads, while FP64 is often selected for some HPC applications. Precision below FP32 is not typically supported efficiently in a GPU. Therefore, moving to reduced precision on a standard GPU has little benefit other than reducing the required memory bandwidth.

GPUs typically offer some binary operation capability, but typically only 32 bitwise operations per ALU. That is a lot of complexity and area for a 32 binary operation. In the context of binarized neural networks, the algorithm requires an XNOR operation followed by a population count. NVidia GPUs are only capable of a population count operation every four cycles, which massively impacts the available binary compute[Ref 18].

As shown in Figure 2, in an effort to keep pace with developments in the machine learning inference space, GPU vendors have been making the necessary silicon changes to support a limited set of reduced precision data types, e.g., FP16 and INT8. For example, the NVidia GPUs on Tesla P4 and P40 cards support INT8, providing four INT8 operations per ALU/Cuda core.

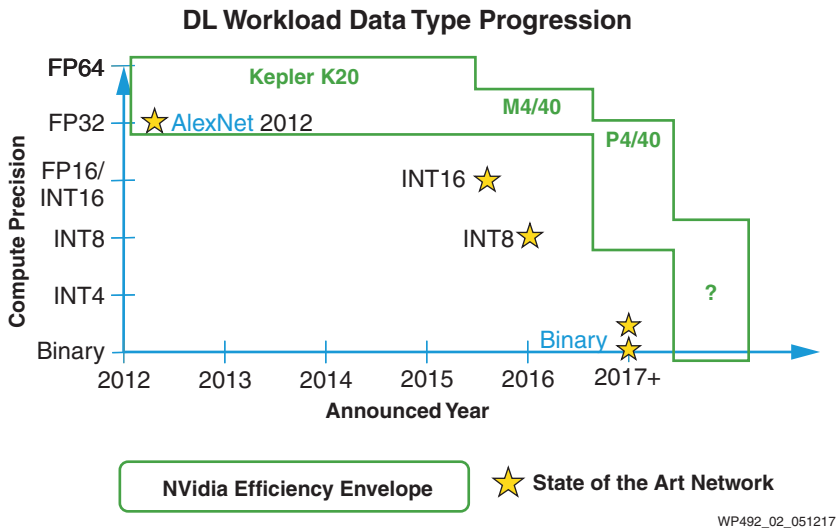

Figure 2: **NVidia Reduced Precision Support**

However, machine-learning inference benchmarks published by NVidia for GoogLeNet v1 inference on Tesla P40 show only a 3X improvement in efficiency for INT8 implementation vs. a FP32 implementation, illustrating the underlining challenges with squeezing reduced precision support into the GPU architecture and achieving efficient results[Ref 16].

As machine learning and other workloads move to lower and custom precisions, GPU vendors need to bring more new products to the market, and their existing users need to upgrade their platform to benefit from any advances in this space.

SW-Defined Datapath via Rigid Memory Hierarchy

Similar to a CPU, data flow in a GPU is defined by software and is governed by the GPU's rigid and complex memory hierarchy[Ref 17]. A typical GPU memory hierarchy is illustrated in Figure 3. Individual threads have their own memory space in the register file to store variables local to the thread. A small number of threads (in the same block) can communicate via shared memory; all threads can communicate via global or off-chip memory[Ref 18].

As noted in Figure 3, the energy and latency associated with memory access increases by over 100X and 80X respectively, as data traverses the memory hierarchy from the register file to global memory[Ref 15][Ref 17][Ref 19]. In addition, memory conflicts are inevitable, increasing latency and causing ALUs to lie idle, resulting in lost compute and efficiency.

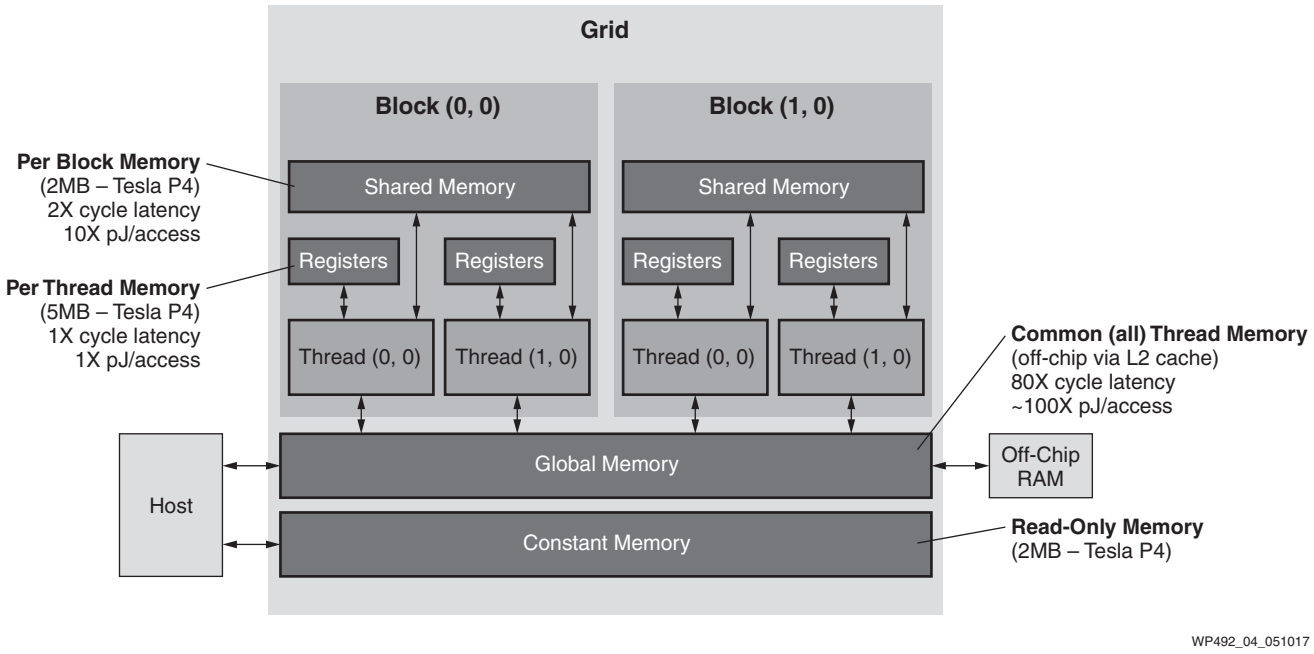

Figure 3: **Typical GPU Memory Hierarchy**

Therefore, if the compute and efficiency potential of the GPU is to be realized, a workload's data flow must map precisely to the GPU memory hierarchy. In reality, very few workloads have sufficient data locality patterns to enable efficient mapping to GPUs. For such workloads, the realizable compute and efficiency are substantially reduced, and the latency of the solution is increased when implemented on a GPU[Ref 19][Ref 20].

One clear quantifiable example of this data flow limitation is in machine learning inference. GPUs must batch heavily, e.g., by 128, to realize an efficient but higher latency solution. Ultimately, batching localizes the machine learning processing at the expense of latency[Ref 21]. This effect can clearly be seen in NVidia P40 benchmarks for GoogLeNet v1 inference. For GoogLeNet v1, the network is compute bound due to the P40 memory bandwidth, therefore, the memory bandwidth reduction associated with batching does not help much. However, the P40 clearly needs to batch by 128 to realize even 50% of the theoretical performance of the GPU, introducing massive latency into the system[Ref 16].

In some cases, data can be pre-processed by a CPU to assist the workload to map better to the GPU SIMT architecture and memory hierarchy—but at the cost of more CPU compute and power, eroding any benefits of the GPU[Ref 7].

Limited I/O Options

As described in the GPU Origins and Target Workloads section, GPUs have been designed as co-processors. To facilitate communication with the host, GPUs have traditionally had a hard PCIe® interface as well as some number of off-chip DRAM interfaces (e.g., GDDR5) only. In recent generations, some GPUs have adopted hard interfaces to enable GPU to GPU communication. A CPU is still required to interface with the network and allocate jobs to the GPU, adding additional power to the system and introducing a bottleneck because of the limited bandwidth available via PCIe. For example, NVidia's Tesla P40 supports PCIe 3.0 x16, giving it only 16GB/s of bandwidth.

GPU vendors have started to build small SoCs, e.g., NVidia Tegra X1, that offer integrated GPU compute, an ARM® processor, and some common automotive peripherals such as HDMI, MIPI, SIP, CAN, and basic Ethernet. These devices, however, only have a small quantity of compute, so they must rely on additional discrete GPUs to achieve the necessary compute. However, the interface to the discrete GPUs is extremely limited, e.g., Tegra X1 supports only PCIe 2.0 x4, resulting in a massive bottleneck. The power dissipation of the additional SoC further erodes the efficiency of the platform.

On-Chip Memory Resources

In addition to the negative impact from a latency, efficiency, and throughput perspective, off-chip memory supplies substantially lower bandwidth relative to the local/on-chip memory. Therefore, if a workload relies on off-chip memory, the memory bandwidth of the off-chip memory becomes a bottleneck and the compute resources lie idle, decreasing compute and efficiency offered by the GPU.

Therefore, it is advantageous to have large on-chip, low latency, and high bandwidth memory. Using machine learning inference again as an example, GoogLeNet requires a total memory footprint of 27.2MB for weights, assuming FP32 implementation, which no GPU can offer—meaning off-chip memory is needed[Ref 22]. In many cases, expensive high bandwidth memory (HBM) and batching is required to prevent cores lying idle. If devices with larger on-chip memories are chosen, the cost of HBM is removed together with the additional latency and power.

Power Envelope

GPU vendors typically design their cards and GPUs to fit into a 250W power envelope, relying on active thermal management to regulate temperature. In tackling the machine learning inference market, NVidia has built devices that fit within the 75W power envelope, e.g., Tesla M4 and P4. Even 75W can be well outside the allowable power and thermal envelope allowed at the system level. GPU absolute power consumption continues to represent a challenge to wide adoption of GPUs.

Functional Safety

GPU origins are in the consumer graphics and high-performance compute space where functional safety is not a requirement. As GPU vendors begin to target ADAS applications, functional safety is becoming a priority and a requirement. Devices need to be designed from the ground up to ensure they can achieve the required level of functional safety certification required to be considered in these ADAS applications. This is a long and involved learning curve for GPU vendors, requiring new tools and devices.

Origins of Xilinx FPGAs

In 1984, Xilinx invented the field programmable gate array (FPGA), enabling its users to program (and re-program) a nearly unlimited number of functions in a single device. Previously, system designers realized these functions with many generic discrete logic components or by building costly ASICs[Ref 23].

More than thirty years on, flexibility and programmability are still the pillars of Xilinx's All Programmable FPGAs and SoCs. Xilinx delivers programmable platforms that address the core needs of several end applications in the wired and wireless communication, cloud computing, medical, automotive, industrial, and aerospace and defense markets. All of these applications require significant compute, and many have very strict real-time requirements such as industrial automation and ADAS.

Traditionally, one of the challenges associated with using FPGAs has been the need to program them using a hardware description language (HDL), such as Verilog or VHDL. Recently, Xilinx developed SDSoC™ and SDAccel™ tools to unlock the many benefits of programmable devices to a wider range of users, e.g., software developers and system architects[Ref 24], and built additional acceleration stacks to speed up the system designer's efforts to realize the benefits of Xilinx devices[Ref 25].

Unique Benefits of Xilinx FPGAs and SoCs

Raw Compute Power

Contrary to the claims made by GPU advocates, a single Xilinx device delivers massive raw compute power, e.g., 38.3 INT8 TOP/s in a Virtex® UltraScale+™ XCVU13P FPGA. A state of the art NVidia Tesla P40 card offers a similar 40 INT8 TOP/s of raw compute power when running at its base frequency, but at more than 2X the power of the Xilinx based solution[Ref 26]. And the flexibility and on-chip memory found on Xilinx devices results in significantly higher compute capability for many workloads and applications (see Flexibility in All Programmable Devices and On-Chip Memory Resources).

In addition, the flexibility provided by Xilinx devices means that the full range of data type precisions—e.g., FP32, INT8, binary, and custom[Ref 27]—can be supported. For example, for binarized neural networks, Xilinx can deliver a staggering 500TOPs/s of binarized compute (assuming 2.5 LUTs per operation), equating to 25X more than a GPU is typically capable of. While some precisions map best to the DSP resources, others are best suited to the implementation in programmable logic, while others use a combination. This flexibility ensures that the device's compute and efficiency scales as precision is reduced, all the way down to binary operations.

A great deal of research in the machine learning space targets the optimum precision from a compute, accuracy, and efficiency perspective[Ref 28][Ref 29][Ref 30][Ref 31][Ref 32]. Regardless of where the optimum is for a given workload, Xilinx devices' compute power and efficiency scale, giving the full benefit of the move to reduced precision.

For a number of years, many users of FPGAs have implemented systolic array processing designs to achieve optimum performance for a number of workloads, including machine learning inference [Ref 33][Ref 34]. To ensure Xilinx FPGA and SoC users maximize the realizable compute and efficiency for such workloads on existing Xilinx devices, Xilinx has several resources available. These resources include INT8 optimization and the coupling of the DSP arrays to the most efficient memory hierarchy of block RAM and UltraRAM[Ref 27]. More information on these resources is available from your local Xilinx sales representative.

Interestingly, to improve the available compute and efficiency for today's deep learning workloads, NVidia has hardened similar functionality in the form of their Tensor Cores found in the Volta architecture. However, deep learning workloads evolve over time, so the Tensor Core architecture will likely need to change and GPU users will need to wait and then buy new GPU hardware.

Efficiency and Power

From a system level perspective, a compute platform must deliver the maximum compute for a given power or thermal envelope. To address this need, compute platforms need to:

- Fall within the allowable power envelope
- Maximize the realizable compute within that power budget

Xilinx offers a broad portfolio of All Programmable devices, enabling the user to choose the devices that map best to the power and thermal envelope available. In addition, Xilinx's UltraScale+ devices have a low voltage mode (V_{LOW}), enabling 30% lower power and 20% higher efficiency.

As shown in Table 1, Xilinx devices offer the most efficient general-purpose compute platform from a raw compute perspective for fixed precision data types. This is primarily due to the lower overhead associated with processing in Xilinx FPGA-based architecture. For example, GPUs require additional complexity around their compute resources to facilitate software programmability. For tensor operations for today's deep learning workloads, NVidia's Tesla V100, offers comparable efficiency to Xilinx FPGA and SoCs because of their hardened Tensor Cores. However, deep learning workloads are evolving at a rapid pace, so it is unclear how long NVidia's Tensor Cores will remain efficient for deep learning workloads. Clearly, the NVidia V100 is also challenged from an efficiency perspective for other general-purpose workloads.

Table 1: **Device Efficiency Assuming 90% Device Utilization and 80% Active Clock Cycles(1)**

Device	General Purpose Compute Efficiency	Tensor Operations Efficiency
NVidia Tesla P4	209 GOP/s/W(2) (INT8)	
NVidia Tesla P40	188 GOP/s/W (INT8)	
NVidia Tesla V100	72 GFLOP/s/W(3) (FP16)	288 GFLOP/s/W (FP16)
Intel Stratix 10	136 GOP/s/W (INT8)	
Xilinx Virtex® UltraScale+™	277 GOP/s/W (INT8) [Ref 27]	

Notes:
1. The numbers quoted are for comparison purposes only. Realizable device efficiency depends on the end application and the user.
2. Giga operations per second per watt of power consumed.
3. Giga floating point operations per second per watt of power consumed.

Because of the limitations highlighted earlier in this white paper, GPUs struggle to achieve anywhere near the numbers shown in Table 1 for real workloads and systems.

The flexibility and other benefits of Xilinx devices, coupled with Xilinx's new software development stack, ensure that Xilinx based solutions realize much higher efficiency for a massive array of end applications and workloads.

Flexibility in All Programmable Devices

Xilinx devices are carefully crafted to deliver the compute, efficiency, costs, and flexibility needs of a large array of high-performance end systems. Xilinx achieves this balance through a mix of hardware programmable resources (e.g., logic, routing, and I/O) and flexible, independent, integrated core blocks (e.g., DSP slices and UltraRAM), all built on leading edge process technology, such as TSMC's 16nm FinFET process technology.

The hardware programmability and flexibility of Xilinx devices mean that the underlying hardware can be configured to address the needs of a given workload. Later, even during run time, the datapath can easily be reconfigured using partial reconfiguration[Ref 35]. Figure 4 attempts to capture some of the flexibility offered by Xilinx All Programmable devices, but the true flexibility offered by Xilinx devices cannot be captured in a single figure. Kernels (or user design elements) can interface directly to programmable I/O, any other kernels, LUTRAM, block RAM, and UltraRAM, external memory etc.

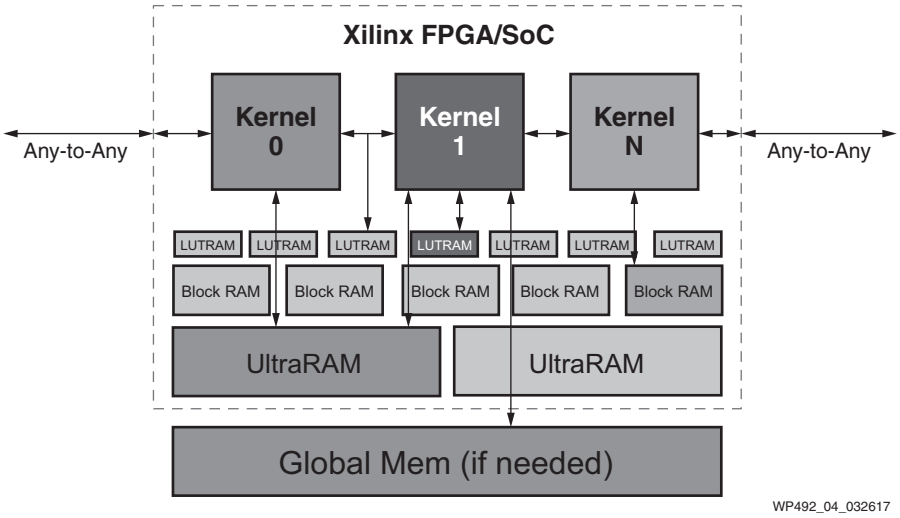

Figure 4: **All Programmable Datapath and Any-to-Any I/O**

The unique hardware programmability of Xilinx devices means they are free from certain restrictions, such as SIMT or a fixed datapath. Whether massively parallel, moderately parallel, pipelined sequential, or a mix, the compute and efficiency of Xilinx devices is accessible. In addition, if the underlying algorithms change (e.g., developments in machine learning networks), the platform can adapt.

The benefit of the flexibility of Xilinx devices can be seen in many systems and workloads. One such workload is machine learning inference. One of the trends in machine learning inference is a move towards sparse networks. Users of Xilinx devices are already taking advantage of these trends. NVidia itself has been one of these users. In a recent paper on speech recognition co-authored by NVidia, the use of Xilinx FPGAs resulted in a 43X speed-up and a 40X efficiency improvement relative to a CPU, and a 3X speed-up and an 11.5X efficiency improvement relative to an NVidia GPU[Ref 36]. The programmable datapath also reduces the need to batch with Xilinx FPGAs. Batching is a significant determiner of latency vs. real-time performance of the system.

From a big data perspective, the flexibility of Xilinx devices has also shown clear benefits. Xilinx FPGAs have been shown to be highly efficient and fast for SQL workloads, including those with complex data, e.g., variable length strings. Baidu has shown speed-ups of over 25X using a Xilinx Kintex® UltraScale™ KU115 device-based card. The card consumes only 50W, making Baidu's solution 4X more efficient relative to a GPU implementation[Ref 37]. Similarly, for text and pattern-matching workloads, studies have shown that Xilinx-based RegEx implementation is 14.5–18X faster than CPU implementations, and close to 3X faster than GPU implementations[Ref 38][Ref 39].

Genomic analysis is yet another tangible example. GPUs have been leveraged by some to help speed up genomic analysis, demonstrating a 6X–10X speed-up vs. an Intel Xeon CPU implementation[Ref 40]. However, Xilinx FPGAs have demonstrated much higher speed-ups, over 80X versus equivalent CPUs[Ref 41].

The flexibility of Xilinx devices also makes them an ideal element for Cloud service providers as part of their compute platform as a service. Large varieties of software as a service can reap the benefits of Xilinx devices.

Finally, for automotive systems designers striving for autonomous driving, the flexibility of Xilinx devices provides a scalable platform that can be used to address the various SAE levels, in the journey to full autonomy. For more information on SAE levels, go to SAE's website. Xilinx devices can efficiently process the sensor data coming from a variety of sources, e.g., radar, camera, and ultrasound, while maintaining the real-time/latency targets of the system.

Any-to-Any I/O Flexibility

In addition to the flexibility of the device's compute resources, Xilinx's any-to-any I/O flexibility ensures that its devices can integrate seamlessly into existing infrastructure, e.g., to connect directly to network or storage devices without the need for a host CPU[Ref 42]. The I/O flexibility also allows the platform to adapt to any changes or updates to the infrastructure.

Further details on the Xilinx UltraScale architecture-based devices are available in Xilinx's large and growing library of white papers.

On-Chip Memory

As shown in Table 2, Xilinx devices offer an industry-leading 500Mb of flexible high-bandwidth, low-latency on-chip memory[Ref 44]. This massive on-chip memory cache means that much of a workload memory requirement is delivered with on-chip memory, reducing memory bottlenecks associated with external memory access, as well as the power and cost of a high memory bandwidth solution, e.g., HBM2. For example, the coefficients/feature maps for most deep learning network topologies, e.g., GoogLeNet, can be stored in on-chip memory, increasing compute efficiency and decreasing cost.

Table 2: **Device On-chip Memory Size**

Device	On-Chip Memory (Mb)
NVidia Tesla P4	80
NVidia Tesla P40	101
NVidia Tesla P100	144
NVidia Tesla V100	300
Intel Stratix 10	244
Xilinx Virtex® UltraScale+	500

Staying on-chip eliminates the massive latency associated with off-chip memory access, maximizing the real-time performance of the system.

In-package HBM

Where additional high-bandwidth memory is required, Xilinx offers HBM in some Virtex UltraScale+ devices. In addition to the 460GB/s of memory bandwidth associated with the in-package HBM stacks, the Xilinx HBM memory controllers add additional flexibility to help efficiently map workloads to the device and available memory bandwidth, maximizing efficiency and compute efficiency.

Functional Safety

Xilinx has a long history of addressing the needs of safety-critical applications, such as industrial automation and, more recently, ADAS. Xilinx tools and devices have been designed from the ground up to support functional safety applications, achieving the appropriate certification level[Ref 45].

As a result, Zynq®-7000 All Programmable SoCs are in production for safety-critical application ADAS applications at a number of automotive manufacturers. Zynq UltraScale+ MPSoCs further extend support for functional safety-critical applications.

Conclusion

System designers are faced with a difficult choice in this new era of computing. Xilinx FPGAs and SoCs offer the lowest risk for system designers to address the core requirements and challenges of tomorrow's systems, while offering the flexibility to ensure the platform remains relevant into the future.

In the context of deep learning, the parallelism inherent in the DSP architecture in UltraScale architecture enhances convolution and matrix multiplication throughput for neural networks with a scalable performance of INT8 vector dot products. This delivers lower latency for deep learning inference. Fast DSP array coupled with the most efficient memory hierarchy of block RAM, and UltraRAM memory arrays deliver best in class power efficiency.

The many benefits of Xilinx devices can be explored today using development kits available on https://www.xilinx.com/products/boards-and-kits.html and the many design entry tools, such as HLS, SDSoC, and SDAccel tools.

References

1. ZDNET. "Vision and neural nets drive demand for more powerful chips." Accessed April 6, 2017. http://www.zdnet.com/article/vision-and-neural-nets-drive-demand-for-better-chips/.
2. NVidia Corporation. http://www.nvidia.com/object/IO_20020111_5424.html.
3. MJ Misic, DM Durdevic, MV Tomasevic. "Evolution and trends in GPU computing" MIPRO, 2012 Proceedings of the 35th International Convention, 289-294. http://ieeexplore.ieee.org/abstract/document/6240658/.
4. Nicole Hemsoth. "FPGAs Focal Point for Efficient Neural Network Inference." Last accessed on April 6, 2017. https://www.nextplatform.com/2017/01/26/fpgas-focal-point-efficient-neural-network-inference/.
5. http://www.cse.psu.edu/hpcl/docs/2016_PACT_Onur.pdf.
6. Babak Falsafi, Bill Dally, Desh Singh, Derek Chiou, Joshua J. Yi, Resit Sendag, "FPGAs versus GPUs in Datacenters," IEEE Micro, vol. 37, no. 1, pp. 60-72, Jan 2017. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7866802.
7. Richard Vuduc, Aparna Chandramowlishwaran, Jee Choi, Murat Guney, Aashay Shringarpure. "On the limits of GPU acceleration." HotPar'10 Proceedings of the 2nd USENIX conference on Hot topics in parallelism. Pages 13-13. Berkeley, CA. June 14–15, 2010. http://hpcgarage.org/wp/vuduc2010-hotpar-cpu-v-gpu.pdf
8. Fowers, J., Ovtcharov, K., Strauss, K., Chung, E.S., Stitt, G.: A High Memory Bandwidth FPGA Accelerator for Sparse Matrix-Vector Multiplication. In: IEEE Int. Symp. on Field-Programmable Custom Computing Machines (2014). http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6861585
9. B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. "Sparse Convolutional Neural Networks." Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference. http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Liu_Sparse_Convolutional_Neural_2015_CVPR_paper.pdf.
10. Yaman Umuroglu, et al. "Random Access Schemes for Efficient FPGA SpMV Acceleration." Microprocessors & Microsystems Volume 47 Issue PB, November 2016 Pages 321-332. http://www.sciencedirect.com/science/article/pii/S0141933116300023 (figure 4 - shows utilization)
11. Sparsh Mittal, Jeffrey S. Vetter. "A Survey of CPU-GPU Heterogeneous Computing Techniques." ACM Computing Surveys (CSUR) Volume 47 Issue 4, July 2015 Article No. 69.
12. Xilinx white paper, "Reduce Power and Cost by Converting from Floating Point to Fixed Point." Last accessed April 6, 2017. https://www.xilinx.com/support/documentation/white_papers/wp491-floating-to-fixed-point.pdf.
13. Marc Baboulin et al. "Accelerating Scientific Computations with Mixed Precision Algorithms." Computer Physics Communications 180 (2009) 2526-2533. http://www.netlib.org/utk/people/JackDongarra/PAPERS/202_2009_Accelerating-Scientific-Computations-with-Mixed-Precision-Algorithms.pdf.
14. Suyog Gupta et al. "Deep Learning with Limited Numerical Precision." ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 Pages 1737-1746. https://arxiv.org/pdf/1502.02551.pdf.
15. William Dally. "High-Performance Hardware for Machine Learning." Last accessed April 6, 2017. https://media.nips.cc/Conferences/2015/tutorialslides/Dally-NIPS-Tutorial-2015.pdf.
16. NVidia Corporation. "NVIDIA TensorRT." Last accessed April 6, 2017. https://developer.nvidia.com/tensorrt.

17. Xinxin Mei, Xiaowen Chu. "Dissecting GPU Memory Hierarchy through Microbenchmarking." IEEE Transactions on Parallel and Distributed Systems Volume: 28, Issue: 1, Page 72-86, Jan. 1 2017. https://arxiv.org/pdf/1509.02308.pdf.
18. NVidia Corporation. "Cuda C Programming Guide" Last accessed on April 6, 2017. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#memory-hierarchy.
19. Mark Gebhart et al. "Unifying Primary Cache, Scratch, and Register File Memories in a Throughput Processor." MICRO-45 Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, Pages 96-106, Vancouver, B.C., CANADA - December 01–05, 2012. https://research.nvidia.com/sites/default/files/publications/Gebhart_MICRO_2012.pdf.
20. Chris Edwards. "Minimize Memory Moves for Greener Data Centers" Last accessed April 6, 2017. http://www.techdesignforums.com/blog/2016/06/08/dac-data-center-acclerators/.
21. Vincent Vanhouckehttps et al. "Improving the Speed of Neural Networks on CPUs" Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop 2011 [online] Available: http://research.google.com/pubs/archive/37631.pdf.
22. Christian Szegedy et al. "Going Deeper with Convolutions" Proceedings Computer Vision and Pattern Recognition (CVPR). Pages 1–9, 2015. http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf.
23. Funding Universe. "Xilinx, Inc. History" Last accessed April 6, 2017. http://www.fundinguniverse.com/company-histories/xilinx-inc-history/.
24. Xilinx Inc. "Software Zone." Last accessed April 6, 2017. https://www.xilinx.com/products/design-tools/software-zone.html.
25. Xilinx Inc. "Acceleration Zone." Last accessed April 6, 2017. https://www.xilinx.com/products/design-tools/acceleration-zone.html#libraries.
26. ANANDTECH. "NVIDIA Announces Tesla P40 & Tesla P4 - Neural Network Inference." Last accessed April 6, 2017.
27. Xilinx white paper, ''Deep Learning with INT8 Optimization on Xilinx Devices." Last accessed April 6, 2017. https://www.xilinx.com/support/documentation/white_papers/wp486-deep-learning-int8.pdf.
28. Philipp Gysel et al. "Hardware-Oriented Approximation of Convolutional Neural networks." ICLR 2016. https://arxiv.org/pdf/1604.03168v3.pdf.
29. Chenzhuo Zhu et al. "Trained ternary quantization." ICLR 2017. https://arxiv.org/pdf/1612.01064.pdf.
30. Yaman Umuroglu et al. "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference." 25th International Symposium on FPGAs, February 2017. https://arxiv.org/pdf/1612.07119.pdf.
31. Wonyong Sunget et al. "Resiliency of Deep Neural Networks under Quantization." ICLR 2016. https://arxiv.org/pdf/1511.06488v3.pdf.
32. Nicholas J. Fraser et al. "Scaling Binarized Neural Networks on Reconfigurable Logic." HiPEAC 2017. https://arxiv.org/abs/1701.03400.
33. Cl´ement Farabet, Yann LeCun, Koray Kavukcuoglu, Eugenio Culurciello, Berin Martini, Polina Akselrod, Selcuk Talay. Large-Scale FPGA-based Convolutional Networks. http://yann.lecun.com/exdb/publis/pdf/farabet-suml-11.pdf
34. C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. ACM/SIGDA ISFPGA, pages 161–170. ACM, 2015. https://pdfs.semanticscholar.org/2ffc/74bec88d8762a613256589891ff323123e99.pdf

35. Xilinx, Inc. "Partial Reconfiguration in the Vivado Design Suite." Last accessed April 6, 2017. https://www.xilinx.com/products/design-tools/vivado/implementation/partial-reconfiguration.html.
36. Song Han et al. "ESE: Efficient Speech Recognition Engine with Sparse LSTM on FPGA." International Symposium on FPGA 2017. https://arxiv.org/pdf/1612.00694.pdf.
37. Jian Ouyang et al. "SDA: Software-Defined Accelerator for General-Purpose Big Data Analysis System." Hotchip 2014. http://www.hotchips.org/wp-content/uploads/hc_archives/hc28/HC28.23-Tuesday-Epub/HC28.23.80-Big-Data-Epub/HC28.23.832-SDA-BD-Analysis-JianOuyang-Baidu-v06.pdf.
38. Shreyas G Singapura et al. "FPGA Based Accelerator for Pattern Matching in YARA Framework." CENG 2015. http://ceng.usc.edu/techreports/2015/Prasanna%20CENG-2015-05.pdf.
39. Yuichiro Utan et al. "A GPGPU Implementation of Approximate String Matching with Regular Expression Operators and Comparison with Its FPGA Implementation." PDPTA 2012. https://pdfs.semanticscholar.org/2667/ac95d36ab63ae6eeb4b352f4c20dc46344c3.pdf.
40. BarraCUDA. "The BarraCUDA Project." Last accessed April 6, 2017. http://seqbarracuda.sourceforge.net/index.html.
41. Edico Genome. "DRAGEN Genome Pipeline." Last accessed April 6, 2017. http://www.edicogenome.com/wp-content/uploads/2015/02/DRAGEN-Genome-Pipeline.pdf.
42. Microsoft paper, "A Cloud-Scale Acceleration Architecture." Last accessed April 6, 2017. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/10/Cloud-Scale-Acceleration-Architecture.pdf.
43. Xilinx, Inc. "UltraScale Architecture." Last accessed April 6, 2017. https://www.xilinx.com/support/documentation/white_papers/wp470-ultrascale-plus-power-flexibility.pdf.
44. Xilinx white paper, "UltraRAM: Breakthrough Embedded Memory Integration on UltraScale+ Devices." Last accessed April 6, 2017. https://www.xilinx.com/support/documentation/white_papers/wp477-ultraram.pdf.
45. Xilinx white paper, "Xilinx Reduces Risk and Increases Efficiency for IEC61508 and ISO26262 Certified Safety Applications." Last accessed April 6, 2017. https://www.xilinx.com/support/documentation/white_papers/wp461-functional-safety.pdf.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
06/13/2017	1.0.1	Typographical edits.
06/08/2017	1.0	Initial Xilinx release.

Disclaimer

Automotive Applications Disclaimer