AMD

Adaptive Computing

Technology Overview

Contents

Executive Summary

Adaptive computing builds on existing FPGA technology yet makes it more accessible to a wider range of developers and applications than ever before. The fundamental functionality of this technology is built on adaptive hardware, which has the unique ability to be modified after manufacture. Arrays of hardware blocks, each configurable, can be connected as needed, allowing highly efficient, domain-specific architectures to be built for any application.

This adaptability is a unique differentiator from CPUs, GPUs and ASSPs, which all have a fixed hardware architecture. The benefit of adaptability varies by application, but it's not uncommon to see a 20X performance improvement vs. a CPU implementation of the same function.

Adaptive computing can be used by a wide variety of hardware, software and AI developers. Adaptive platforms, including comprehensive development tools, provide a strong base from which efficient yet differentiated end products can be built. The benefits of deploying adaptive platforms in production systems include reduced time-to-market, reduced operating costs, and an innate futureproofing of the system.

Adaptive platforms are available for a wide variety of end applications, from the data center, to the network, to the edge and even the endpoint. From self-driving automobiles to the Mars rover, adaptive computing is empowering a new generation of intelligent, efficient applications

Introducing Adaptive Computing

The principles of adaptive computing were established in 1984 when Ross Freeman brought his brilliant idea for Field Programmable Gate Arrays (FPGAs) to life and founded Xilinx. This technology, and its ability to serve the needs of numerous applications, has come a long way since then. Although adaptive computing is built on top of FPGA technology, it has grown to encompass a great deal more.

At its core, adaptive computing comprises silicon hardware that can be highly optimized for specific applications. This optimization occurs after the hardware is manufactured and can be performed and reperformed an almost infinite number of times. This unique flexibility allows hardware changes to be made after the device has been fully deployed into a production setting. Just like a production CPU can be given a new program to run, an adaptive platform can be given a new hardware configuration to adapt to, even in a live, production setting.

The ability to have the hardware function changed after deployment is the "FP," or "field-programmable," in FPGA. It means hardware can be programmed in the field, after deployment into its production environment. The "GA" in FPGA refers to "gate array." Adaptive computing platforms have come a long way since the days of gate arrays, but the concept is still a valid way to explain how the underlying technology works. We will further explore this topic in the next section.

Adaptive Hardware

Adaptive hardware allows the underlying function of the hardware to be configured after manufacturing. It can do this because it has two unique capabilities.

The first unique capability is a regular structure of configurable hardware blocks. In an FPGA, these are called “configurable logic blocks” and each are capable of being configured to perform one of many possible arithmetic functions that operate on multiple inputs to produce an output. In newer adaptive computing platforms, these blocks can be significantly more complex, comprising highly complex configurable functions, including vector processors. We will focus on the former type for now, but the latter is covered in a subsequent section.

The second unique capability is the configurable connectivity between the blocks. This configurable interconnect gives adaptive hardware the ability to make connections between blocks as needed, thus enabling more complex functions to be built by connecting specific blocks together. The configurable interconnect also allows connection between non-configurable blocks the end-system may need, such as memory, embedded CPUs, digital signal processors (DSPs) and the inputs and outputs (I/O or ‘pins’) of the hardware device.

The figure below shows a representation of the unconfigured blocks (left side), and the same blocks after being configured for an application (right side). In this example, two independent functions, A and B, have been implemented. Each has numerous stages, some of which can be parallel (AF5a and AF5b). A and B themselves are also completely parallel. The arrows show the connectivity that has been configured into the adaptive hardware, while the white text represents the blocks that have been uniquely configured to implement a specific function. Four blocks remain unconfigured in this example (still grey in color) but remain available to use should future changes be desired.

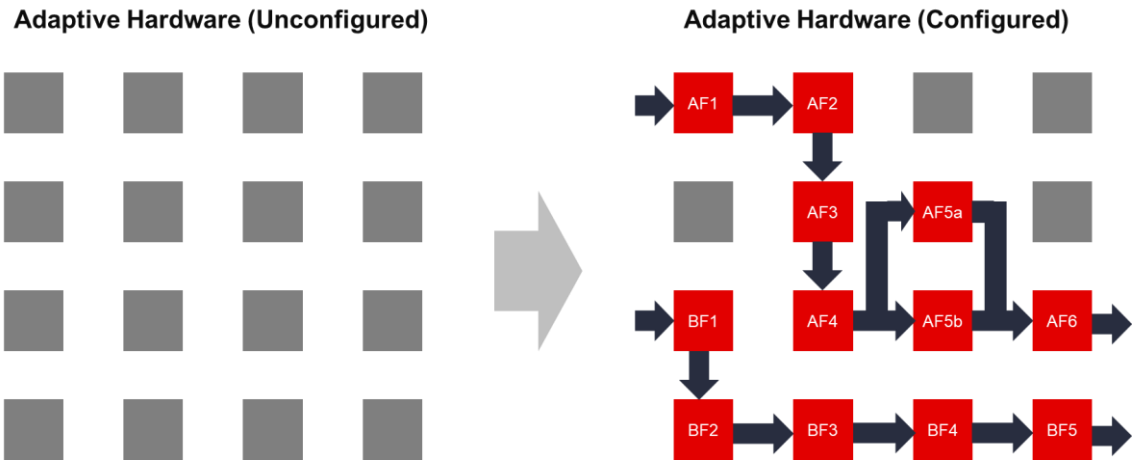

Figure 1: A representation of unconfigured and configured adaptive hardware

Adaptive Hardware vs. CPUs

Although CPUs are highly flexible, their underlying hardware is fixed. Once a CPU is manufactured the hardware cannot be changed. It relies on software to tell it which specific operation (arithmetic function) to perform, on which data in memory. It selects from a fixed number of pre-determined operations each time. The hardware must be capable of performing all possible operations, which are called using software instructions, though the atomic processing components of a CPU (known as an algorithmic logic unit, or ALU) can generally only execute one instruction at a time. Modern CPUs can have 32 or more ALUs, but fundamentally, each one operates in a sequential fashion – one instruction at a time.

As discussed, software is written to tell (instruct) the CPU which exact operation to execute, where to get the data to operate on, and where to store the resultant data. CPUs still operate on the basic Von-Neumann architecture (or more accurately, stored-program computer), where data is brought to the processor from memory, operated on, and then written back out to memory. Fundamentally, the architecture is centered around the ALU and requires data to be moved in and out of the ALU for every operation.

CPUs offer immense flexibility as they are software programmable. Some applications, however, do not fit efficiently into the Von-Neumann architecture, regardless of how advanced the CPU architecture is.

For example, consider a smart-vision application that monitors vehicle license plates for road tolls. It must take in a stream of video data, decode the data (preprocessing), and pass it to an AI Inference model that recognizes the license plate characters. It feeds the character information into a post-processing block that combines them and reports the license plate number via a network to a centralized data center to enact the billing process. In streaming data applications such as this, a Von-Neumann architecture is far from optimal.

Rather than moving the data to the processor, it is significantly more efficient to build a processing pipeline that operates on the data as it flows through the processing stages described above. Such an architecture would first receive the stream of video data, then perform the preprocessing to prepare it for the AI model. Then the data flows directly to the AI model for license plate recognition. While the AI processing is occurring on the previous frame, the next frame is streaming through the preprocessor.

The same application implemented with a CPU will need to preprocess the data first - reading data in from memory, processing, then writing back out to memory. It must then read in the new data to perform the AI function. This is less efficient than the streaming architecture as it must continually move the data to the processor, rather than building a processing pipeline to operate directly on the streaming data.

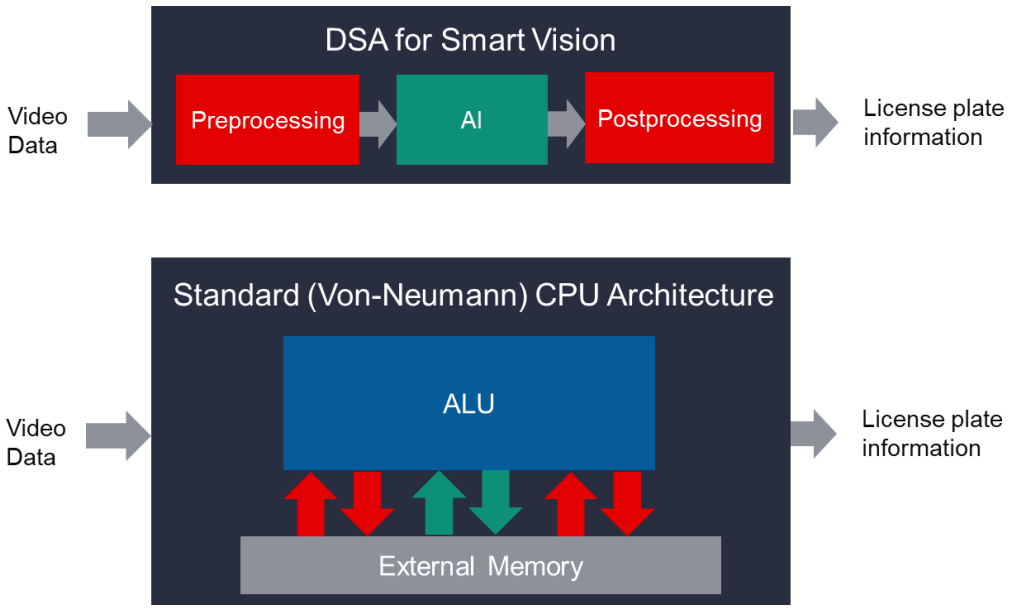

Figure 2: Streaming dataflow architecture vs. Von-Neumann architecture

The architecture on the left is an example of a domain-specific architecture (DSA)--in this case, a specific architecture built for the smart-vision domain. DSAs closely match compute, memory bandwidth, data paths, and I/O to the specific requirements of the domain. For some domains, this delivers a much higher level of processing efficiency compared with general-purpose CPU architectures.

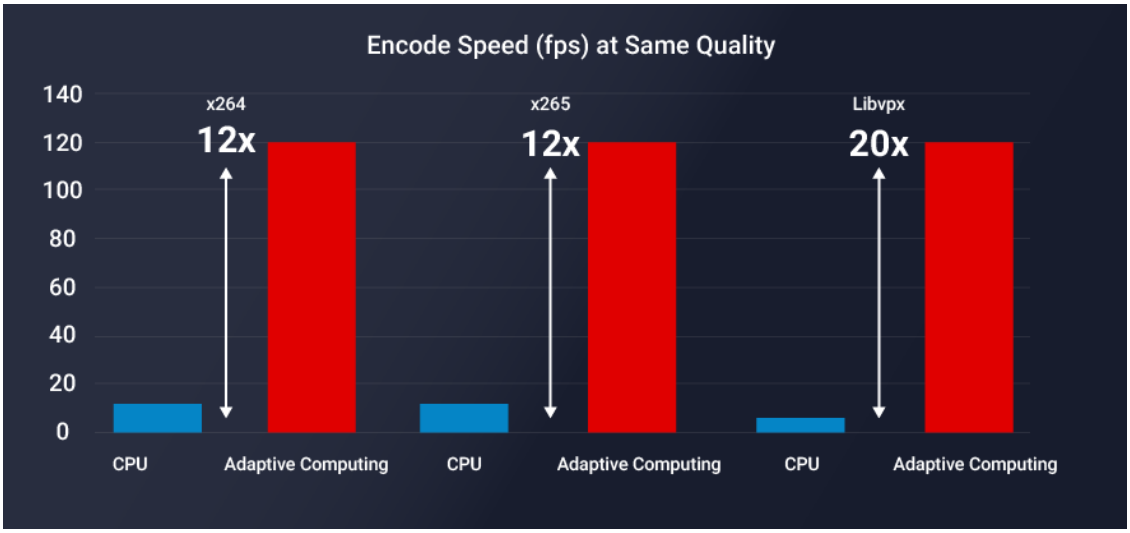

Figure 3: Video encoding/decoding efficiency - Adaptive computing vs. general- purpose CPU (Source: Xilinx)

Despite these limitations, many applications still benefit significantly from being partially implemented with CPUs. Most complex applications will benefit from a heterogeneous array of hardware, with each underlying hardware type providing an optimal implementation for different parts of the application.

Adaptive Hardware vs. GPUs

General-Purpose GPUs (GPGPUs), which we will abbreviate to GPUs for brevity, address a major drawback of CPUs – the ability to process a large amount of data in parallel.

Fundamentally GPUs are CPU-like because they have fixed hardware and operate using software instructions. They differ from CPUs because they can operate on very wide data sets. A single instruction could process a thousand pieces of data or more, though typically the same operation must be performed on every piece of data being simultaneously processed. Modern GPUs have significant architectural complexity, but parallel data operation is the fundamental differentiation from CPUs.

Despite this difference, the core of a GPU still contains a type of Von-Neumann processor. The atomic processing elements operate on a data vector, but still perform one fixed instruction per ALU. Data must also still be brought to these processing units from a memory via a fixed data path.

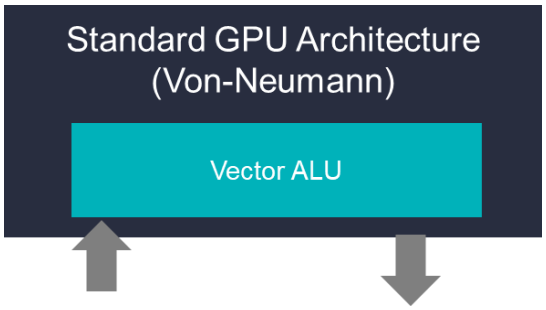

Figure 4: GPUs still fundamentally use the Von-Neumann architecture

Like CPUs, GPUs can be optimal for certain types of applications. However, GPUs, like CPUs, are still built with fixed hardware--the basic architecture and data flow is fixed prior to manufacturing. It cannot be adapted to a specific application, and certainly can't be changed once built. GPUs do not have the ability to enable domain-specific architectures to optimally implement specific applications within specific domains.

Adaptive Hardware vs. ASSPs

DSAs can be built using a dedicated (fixed) silicon device, which is typically called an application-specific standard product, or ASSP. This is an oversimplification, but generally ASSPs implement optimized hardware for a given application or domain, so they can typically be considered a DSA.

There are advantages of implementing a DSA in a fixed ASSP, but there are also disadvantages. Let's consider two of the main disadvantages.

First is the pace of innovation. To keep up with the pace of innovation, manufacturers are expected to create and deliver new services in shorter timeframes than ever before. More specifically, the timeframes are shorter than the time it takes to design and build a new fixed-silicon DSA. This creates a fundamental market misalignment between the market demands on innovation and the time it takes for companies to design and manufacture ASSPs. Changes to industry standards or other fluctuating requirements can quickly render such devices obsolete.

The second consideration is the cost of custom silicon. The one-time cost to design and manufacture a unique silicon design, such as a complex 7nm ASIC, can cost several hundred million dollars in non-recurring engineering (NRE) costs. Costs are projected to rise further as device geometries shrink to 5nm and below. When you buy an ASSP, that upfront NRE is amortized across each unit sold. ASSPs are only cost-effective when a large number can be sold. The increase in cost is slowing adoption of advanced nodes for ASSPs, which can leave their users with outdated and less efficient technology.

Adaptive Platforms

The term *adaptive platform* refers to any type of product or solution that has adaptive hardware at its core. From an application development point of view, a platform is something that provides a set of functionalities from which a product can be built. It provides a solid base on which developers can build their applications. By providing the base of the application, a platform allows developers to focus on their application's specific differentiation.

Adaptive platforms are all based upon the same fundamental adaptive hardware foundation; however, they include much more than just the silicon hardware or device. Adaptive platforms encompass a comprehensive set of design and runtime software. In combination, the hardware and software deliver a unique capability from which highly flexible, yet efficient applications can be built.

Advantages of Adaptive Platforms

Adaptive platforms make adaptive computing accessible to a broad range of software and system developers. These platforms can be used as the basis for many products. The benefits of using an adaptive platform include:

Reduced Time-to-Market. An application built using a platform such as the Alveo™ data center accelerator card can leverage accelerated hardware for a specific application yet require no hardware customization. A PCIe card is added to the server and accelerated libraries are called directly from an existing software application.

Reduced Operating Costs. Optimized applications based on an adaptive platform can provide significantly higher efficiency per node than CPU-based solutions, due to increases in compute density.

Flexible and Dynamic Workloads. Adaptive platforms can be reconfigured depending upon current needs. Developers can easily switch the applications deployed within an adaptive platform, using the same equipment to meet changing workload needs.

Future-Proofed. Adaptive platforms can be continually adapted. If new features are needed in an existing application, the hardware can be reprogrammed to optimally implement these features, reducing the need for hardware upgrades and therefore expanding the system's lifespan.

Accelerating the Whole Application. Rarely does AI inference exist in isolation. It is part of a larger chain of data analysis and processing, often with multiple pre- and post- stages that use a traditional (non-AI) implementation. The embedded AI parts of these systems benefit from AI acceleration. The non-AI parts also benefit from acceleration. The flexible nature of adaptive computing is suited to accelerating both the AI and non-AI processing tasks. This is called "whole-application acceleration" and it has become increasingly important as compute-intensive AI inference permeates through more applications.

Adaptive Platform Accessibility

In the past, benefiting from FPGA technology required developers to build their own hardware boards and use a hardware description language (HDL) to configure the FPGA. In contrast, adaptive platforms allow developers to benefit from adaptive computing directly from their familiar software frameworks and languages such as C++, Python, TensorFlow, etc. Software and AI developers can now utilize adaptive computing without having to build a board or be hardware experts.

For example, video application developers can use adaptive computing directly from industry-standard frameworks such as FFmpeg. They do not need to be experts in FPGAs or hardware-implemented video codecs, yet can deploy state-of-the-art, efficient applications. Additionally, if they have existing software code, it can be ported to use adaptive computing with straightforward API call updates.

Adaptive platforms have enabled adaptive hardware to be used at multiple design abstraction levels. Many accelerated APIs are already available from an independent software vendor (ISV) ecosystem and from vendor-provided, open-source libraries. Larger design teams can also have their own hardware engineers create custom accelerated APIs, which their software team utilizes in an end application.

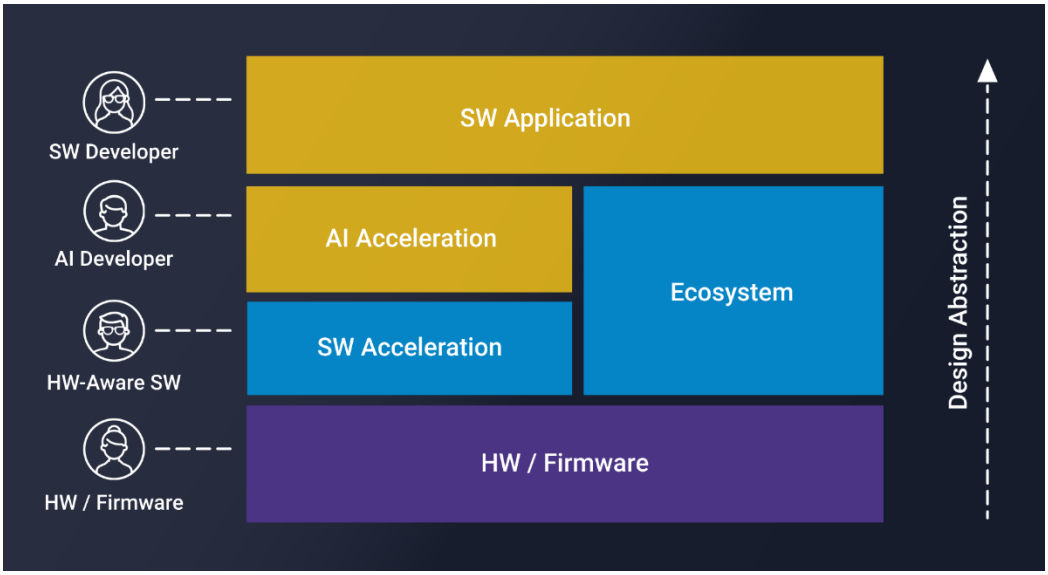

Figure 5: Example adaptive development stack (Source: Xilinx)

Types of Adaptive Platforms

There are many types of adaptive platforms based on the application and need, including data center acceleration cards and standardized edge modules. Multiple platforms exist to give the best possible starting point for the desired application. Applications vary widely, from latency-sensitive applications, such as autonomous driving and real-time streaming video, to the high-complexity of 5G signal processing and the data processing of unstructured databases.

Adaptive computing can be deployed in the cloud, the network, the edge and even at the endpoint, bringing the latest architectural innovations to discrete and end-to-end applications. The range of deployment locations is possible thanks to a variety of adaptive platforms – from large-capacity devices on PCIe accelerator cards in the data center, to small, low power devices suitable for endpoint processing needed by IoT devices.

Adaptive platforms at the edge include Kria™ adaptive system-on-modules (SOMs) from Xilinx. Kria adaptive SOMs are built around Xilinx's Zynq® UltraScale+™ MPSoC architecture and give developers access to a turnkey adaptive platform for edge applications. By standardizing the core parts of the system, developers have more time to focus on building in features that differentiate their technology from the competition.

Figure 6: Kria adaptive SOM

Adaptive platforms in the data center include the Alveo accelerator card. This uses industry-standard PCI-express to provide hardware offload capability for any data-center application. Adaptive computing is not limited to just compute offload in the data center. Adaptive platforms also exist for SmartSSD storage, where acceleration occurs at the storage access point, and SmartNIC, where acceleration occurs directly in the network traffic flow.

Figure 7: Alveo PCIe acceleration card

Recent Advancements in Adaptive Computing

Earlier we discussed adaptive hardware as having two unique characteristics – configurable blocks and configurable interconnect. These form the fundamental differentiation between adaptive and non-adaptive (or fixed) hardware.

One of the biggest innovations in adaptive computing was the introduction of the AI engine by Xilinx. The AI engine is a revolutionary new approach that provides unprecedented compute density for mathematically intense applications.

The AI engine is still fundamentally a configurable block, but it is also programmable like a CPU. Instead of being formed from standard FPGA processing hardware, an AI engine contains high-performance scalar and single-instruction multiple-data (SIMD) vector processors. These processors are optimized to efficiently implement math-rich functions typically found in AI inference and wireless communications.

Arrays of AI engines are still connected with FPGA-like, adaptable data interconnects which enable efficient, optimized data paths to be built for the target application. This combination of computationally dense (math-rich), CPU-like processing elements connected with FPGA-like interconnect is ushering in a new generation of AI and communications products.

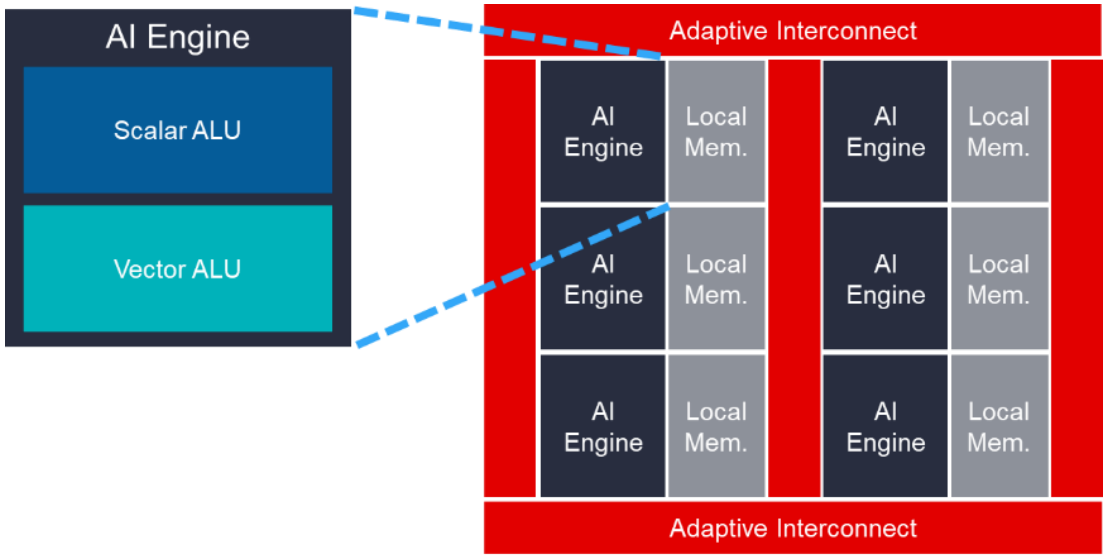

Figure 8: State-of-the-art adaptive hardware – an AI Engine Array

Adaptive Computing in Action

As we have seen, adaptive computing allows applications to be dynamically updated. It enables over-the-air (OTA) updates not just for software, but also for hardware. This is especially important as processing becomes more distributed and when the application is deployed somewhere hard to reach. There are fewer places harder to reach than Mars. The NASA Perseverance rover, now exploring the surface of Mars, contains adaptive computing technology.

Perseverance uses adaptive computing for its comprehensive vision processor. Built using an FPGA-based platform, it accelerates AI and non-AI visual tasks, including image rectification, filtering, detection and matching. The images that Perseverance sends back to NASA are being processed using adaptive computing.

If a new algorithm was invented during the eight months it took Perseverance to reach Mars, or a hardware bug is discovered, adaptive computing allows hardware updates to be sent remotely, over the air - or over space, in this case. These updates can be done as quickly and easily as a software update. When the deployment is remote, such remote hardware updates are more than just a convenience, they are a necessity.

Figure 9: Adaptive computing touching down on Mars for the third time (Image: NASA)

Conclusion

Adaptive computing builds on existing FPGA technology yet makes it more accessible to a wider range of developers and applications than ever before. Software and AI developers can now build optimized applications using adaptive computing, a technology that previously was unavailable to them.

The ability to adapt hardware to a specific application is a unique differentiator from CPUs, GPUs, and ASSPs, which have fixed hardware architectures. Adaptive computing allows the hardware to be tailored to an application, yielding high efficiency, yet still enables future adaptation if workloads or standards evolve.

Adaptive platforms are ideal for a wide variety of end applications – in the data center, the network, the edge, and the endpoint. They accelerate time-to-market and deliver highly efficient, yet upgradable solutions.

Adaptive computing is an advanced technology that enables optimized, hardware-accelerated applications to be built without needing hardware expertise. It has been widely deployed across many industries and has proven itself in numerous applications - around the globe and well beyond it.

As the world becomes more connected and intelligent, adaptive computing will continue to be at the forefront of optimized, accelerated applications, empowering all developers to build a better tomorrow.

Learn more about adaptive computing at: www.xilinx.com/adaptivecomputing

Revision 1.0, June 2021.