Executive Summary

The proliferation of cameras and sensors into autonomous devices, calls for solutions that improve computational power while consuming less energy. Cloud computing has revolutionized the way we store and process data, but several handicaps, such as performance and bandwidth, limit applications as decisions on the Edge must be made with minimal latency. As autonomy and robotics work their way into critical functions in society—such as driverless cars, medical technology, and logistics—the high latency, limited bandwidth, fragile security, and lack of offline access inherent in cloud-computing presents serious concerns. Machines are required to recognize and process a complex and growing class of stimuli and algorithms, and real-time, direct communication between sensors and decisions is needed. These new demands are leading the drive towards Edge Supercomputing, whereby applications demand that data acquisition and processing occur at the edge of the access network and closer to users.

While advancements in edge supercomputing have increased in recent years, developers still lack a unified product architecture that offers reconfigurability, generality, and scalability. To bring the power and performance of server-class hardware to the resource-constrained edge, Quadric's team built the world's first and only purpose-built platform for low latency edge computing. Our Supercomputer enables the deployment of tomorrow's algorithms today, whereby developers can customize their code, then deploy them with popular libraries and frameworks such as OpenCV, Tensorflow, and C++. Reconfigurability and broad support for artificial intelligence, as well as high-performance computing, gives the developer freedom to push the algorithmic envelope without compromising performance.

In this white paper, we introduce Quadric's technical platform and product vision, showcasing the unique software stack, a rundown of product offerings and practical applications, and the technology blueprint that differentiates the Quadric Supercomputer. The reader will learn what to expect from the developer experience, how high-performance kernels can be incorporated with artificial intelligence models, and how kernels can be deployed on Quadric's hardware products. We present Quadric’s product offering in terms of both software and hardware and show how the developer can minimize complexities inherent in both while simultaneously improving overall system performance.

Introduction

Recent advancements in software algorithms, compute performance and deep learning are revolutionizing human-machine interaction. When applying these developments to transportation, an autonomous vehicle can deliver people and goods safely and efficiently. In drone applications, safety inspections of remote pipelines and infrastructure assets can be undertaken without risks to humans. In industrial applications, developers can achieve greater levels of efficiency, precision, and scalability of manufacturing processes. When applied to consumer products, we can unlock long-promised automation, freeing up time to do more of the things we enjoy.

Because machine intelligence on the Edge relies on various sensors embedded in devices making real-time decisions, the computational power and low-latency required are greater than that which current data processing infrastructure (i.e., the cloud) is equipped to handle on a massive scale. These requirements create a shift in how and where data is processed. Data centers are moving portions of their computing closer to the devices receiving and sending data, and more users of AI enabled devices are preferring to process data on-site rather than in the cloud. Because data is stored locally rather than sent off, it enhances some aspects of security, particularly concerning privacy. The edge-computing space is opening up new avenues for innovation in modern computing, where the demand for high performance, low latency, energy efficient products has never been greater.

Despite progress in many areas, developers deploying cutting edge algorithms on the Edge remain resource constrained. Limitations define available edge-tailored product architecture for
both hardware and software, and thus the true potential of machine intelligence to improve tasks and processes has not been achieved. AI and High-Performance workloads are tailored, by developers, for target hardware and not the other way around. Hardware should be purpose-built for these workloads. Developers seeking to orchestrate algorithms for new and novel challenges require room for experimentation and innovation. Available edge computing products tailored for innovator may allow for design flexibility, but lack the processing power to turn ideas into market-viable applications that can be put to use on a large scale.

On the other hand, products offering high-performance computing lack programming flexibility. So, as developers try to match more focused algorithms to a narrow product-defined roadmap, they find their efforts bottlenecked. Being developers ourselves, we met all of these challenges. So, we founded Quadric to build a product that brings server-class performance to the edge.

The Problem: Heterogeneous Compute

Developers deploy today’s leading autonomy algorithms on custom-built heterogenous hardware, where discrete components such as Digital Signal Processors, Field Programmable Gate Arrays, Graphics Processing Units, and General Purpose Processors share access to memory or DMA provisions and take turns processing stack components to meet software demands. To accommodate evolving software sophistication, the complexity of hardware must increase accordingly. In turn, because each hardware component in the heterogeneous compute stack has its unique software framework and programming models, software systems become even more complex, and hardware-specific constraints hinder the developer.

![Figure 1: Heterogeneous computing systems contain various components that share access to memory or DMA provisions and take turns processing parts of the software stack.](image)

Each of these processors occupies their own space in the market, and the shifting demand of applications determines the suitability of one over the other. For example, while various workloads in image processing have advanced to new and novel neural network techniques, many parts of the sense, prediction, decision loop still rely heavily on rule-based components. Because neural networks and high performance compute need to co-exist, the result is exponential system-level complexity. Path planning workloads may run on the Host CPU itself while forward-inferencing of artificial intelligence workloads run on a purpose-built AI accelerator. FPGA’s, reprogrammable and designed for generality, offer flexibility for developers but come with an increase in hardware complexity. They are not software programmable, and hardware reconfigurability requires a team of specialized FPGA experts. GPU’s are a standard choice for many AI applications because the need for parallelizable compute with a good reprogrammable software model is so pronounced. They offer high per-pin bandwidth but are memory constrained and good throughput. CPUs offer high single-threaded performance but lack the
parallel architecture to accelerate modern workloads. The programming models and frameworks for each of the discrete processors can be vastly different. When optimizing each piece of a full-stack sense and control loop with its optimized hardware, the system-level orchestration software makes integration even more complicated. Scheduling which task is run when and on which hardware while maintaining throughput at each component becomes as complex as solving the actual product problem at hand. When characterizing the entire stack holistically, system level scheduling and memory domain marshaling consume a significant amount of total system resources and power. Such full-stack sensor-based edge machine intelligence applications demand a new architecture to enable the entire workload to be accelerated in a single latency, power, and performance-based architecture.

In all cases, the more programming flexibility and general purpose oriented the processor means compromises on performance, cost, and energy use. The need for performance leads to ASIC's, which are specialized processors designed for more specific functions. These can be client-specific or general but still oriented towards a single sector, or algorithm. The specificity of the application allows for a more targeted design, that cannot change to take into account innovation in algorithms, offers high performance. However, when taking into account system-level performance, simply replacing a single piece of a heterogeneous stack with a more performant one, does not always yield a large improvement for overall workloads.

The solution demands a new processor architecture be built from the ground up. By considering all the products and applications that will be run from the top-down, Quadric has built an Edge optimized processor that provides the developer uncompromising performance for power constrained applications.

**The Solution: Unified Compute**

With this in mind, Quadric’s team set out to develop a single, unified processor architecture where developers have the power to write and unify all parallelizable algorithms onto a latency-optimized processor. Unified Compute gives the developer power to accelerate works onto a single cohesive software approach, without the need for complex hardware integration, varied software languages, and frameworks. To address the power and latency challenges at the edge, we designed a novel processor architecture that is flexible enough to handle all workloads of complex heterogeneous systems, without sacrificing performance. Quadric's Processor blends the best of current processing methods, offering ASIC level performance, the flexibility of an FPGA, the graphics processing power of a GPU, with the ease of use of a standard x86 processor. It’s the perfect balance between programmability and performance.

**Figure 2: The Quadric Processor replaces FPGA, GPU, and AI Accelerator hardware components. All software workloads previously running on those hardware components now run on a single**
latency, performance and power optimized processor architecture.

Figure 3: Ultimately, the Quadric Supercomputer execute all workloads, including those performed on the Host CPU itself.

In the following sections, we will describe the software ecosystem that is supported by Quadric Hardware products. We will also describe all forms of Quadric’s hardware products. First, the Quadric Supercomputer, a purpose-built latency-optimized system with full out-of-the-box support for eight cameras. Customers can integrate the Quadric Processor, the processor at the heart of Quadric’s Supercomputer into their hardware platforms. The Quadric IP, the purpose-built processor architecture at the heart of it all that can be integrated into customer’s SoC products. Quadric’s software ecosystem supports all of these hardware products. Write your code once, compile to any hardware target, and deploy algorithms at the edge.

Software Ecosystem

Unified Software

Algorithmic logic, rule-based approaches, and trained AI models will all co-exist in next-generation algorithms. Add to that the proliferation of cameras, LiDAR, and inertial sensors, and system complexity and bandwidth of incoming data only grows. As a result, we need more compute to understand the surrounding scene, localize within it, and plan a path through it. More sensors equal more compute. While the Quadric Processor addresses speed and power challenges, it does not simplify the software experience for the developer. To address the latter, Quadric developed a Software Framework that fuses these high-performance and graph-based development environments and their difficult-to-integrate components into a unified software experience.

A single architecture within an open software development kit (SDK) environment the unifies hardware and software integration efforts. Developers can combine and accelerate the development of applications requiring high performance for sense -> predict -> plan -> decide loops. Leveraging open source standards such as OpenCV, OpenVX, Tensorflow, Caffe, and standard C++ along with Halide and LLVM, developers can homogenize and simplify development efforts, more readily compiling code that would otherwise work on disparate heterogeneous components. Rule-based approaches lower elegantly onto Quadric’s platform where neural network approaches intertwine with standard algorithmic automata. Quadric’s hardware products accelerate all of these algorithms in a single latency-optimized compute fabric. The resulting product is ideal for deployments in power-constrained applications edge applications.

Most importantly, our SDK allows for programming flexibility. Software algorithms and best practices are in constant flux. The best-in-class algorithms of today may not look the same tomorrow. Unfortunately, current edge-tailored processors do not generally support easy customization of algorithms. Developers with increasingly complex algorithms, ones often designed for applications of societal importance and urgency, are finding that existing software platforms only provided support for a narrow set of specialized deployments. Running only the algorithms that fit within the pre-programmed platform means less than optimal accuracy.
Quadric’s Software Framework opens up new avenues for developers seeking to put their algorithms to work today.

![Quadric Supercomputer’s software ecosystem](image)

Figure 4: The Quadric Supercomputer’s software ecosystem allows for the use of popular programming languages and frameworks.

To illustrate various workloads, three example workloads of increasing complexity will be presented. The first kernel demonstrates a basic computer vision computation: the image histogram. Something as simple as an image histogram is typically a standard library call. However, we use it as an example to display the ease of use of our Intermediate Language, expressed in C++. The second kernel describes support for a common deep neural network graph operation: RESNET50. As we import RESNET directly from its description in common graph frameworks, like Tensorflow, we compare the network’s performance with other common hardware. The third workload is not a single kernel, but an entire sense and decision loop. The workload showcases the end-to-end versatility of Quadric’s architecture. The workload contains high performance compute kernels such as A* search and graph-based neural network kernels such as RESNET50. Once compiled, the entire stack runs on Quadric’s edge computing architecture.

Quadric’s software approach can improve the computing of a key component of computer vision: the image histogram. A graphical representation of the tonal content within a captured image, an image histogram is used to discern whether an image is overexposed or underexposed, as well as for thresholding. Thresholding is a technique for generating segmentation masks, which allows images to be represented in simpler components that make them easier to interpret and analyze. These qualities make the image histograms useful in many fields. While in camera pipelines, histograms can be utilized for automated camera exposure control. Doctors use them to enhance medical images, and driverless cars to better recognize objects.
Figure 5: In this image histogram of Quadric’s founders, the x-axis represents varying levels of tonal content, and the y-axis represents the total number of pixels within that tonal bucket. “Dark” content is represented close to the 0 on the x-axis. While the “light” content is represented closer to 255 on the x-axis. Since this is a portrait-style picture, most of the pixels are bright and reside in bins 150-200.

```c
#define HISTO_BINS 256
int result[HISTO_BINS];

cv::mat img;

// step through the image row-wise then col-wise
for (int i = 0; i < img.rows; i++) {
    for (int j = 0; j < img.cols; j++) {
        // You can now access the pixel value with cv::Vec3b
        uint intensity;
        uint bin;

        // calculate the intensity of the pixel
        // calculateIntensity will take the total number of bins and the pixel data
        intensity = calculateIntensity(HISTO_BINS, img.at<cv::Vec3b>(i, j));

        // lookup which bin this intensity will fall into
        // lookupBin will take the total number of bins and the intensity itself
        bin = lookupBin(HISTO_BINS, intensity);

        // increment the bin itself
        result[bin]++;
    }
}
```

Figure 6: An image histogram can be computed simply by iterating through each pixel and comparing its tonal value against the number of discrete bins. Once a match is found, the bin is incremented, and we can move on to the next pixel. Above, pseudocode for computing the histogram of an image on a single-threaded machine.
Classically, the parallelization of the image histogram computation is achieved by:

1. Subdividing the input image between execution threads
2. Processing each subdivided array and computing an image histogram for each
3. Merging the subdivided image histograms into the final image histogram result

When utilizing parallelized compute elements such as GPGPUs to compute image histograms, the developer must pay careful attention to how data is accessed and stored and possess deep knowledge of the nuances of hardware such as the arrangement of processing elements into groups and how the groups access physical memory.

Quadric’s Supercomputer offers a simplified approach, whereby the developer can define their algorithm in a single-threaded way (as the pseudocode in Figure 6 indicates), and the parallelism will be inferred and executed on Quadric’s processing fabric. While the image histogram is a very basic example that most platforms have a highly optimized library call for, it demonstrates how the importance of data locality in computer vision is better supported by Quadric’s software approach.

**Deep Neural Networks: RESNET 50**

The enhancements our software approach lend to computer vision applied to deep neural networks, which have been a subject of intense interest for researchers and AI companies in recent decades. Deep neural networks (DNN) are algorithms characterized by multiple layers between input and output, whereby subsequent layers learn input from the previous layer. Convolutional Neural Networks (CNN), a category of DNN, convert complex patterns into many, many small, simple patterns. They have been used extensively in classification tasks in image and video recognition, classification, and natural language processing.

The basic premise of a CNN is to convert data complexity into many-deep classifiers, or filters, that are simpler in dimensional complexity than the previous layer’s data. This approach leads to much less computationally complex networks than fully connected neural network approaches.

![Figure 7: converting input tensors into several-deep filters of less width and height than the original layer.](image)

By assembling CNN layers in clever ways, one can construct network architectures that can perform point-tasks as well or, in some cases, better than a human being. For example, in 2012 a neural network called Alexnet developed a novel neural network architecture that proved CNN-based architectures could someday exceed human classification in the now-famous ImageNet challenge. By 2015, researchers were proving classification results on the Imagenet challenge that exceeded human error rates. One such advanced CNN network architecture, RESNET50 strikes a balance between total computational network complexity and error rate.
Figure 8: an example of a CNN-based neural network architecture

RESNET uses the concept of residual layers, or shortcuts, that are preserved for future computation to improve network accuracy while minimizing overall network complexity.

Figure 9: a RESNET residual building block. Notice the layer at the input (TOP) is preserved and utilized in computing the final sum (BOTTOM).

By exploiting data locality within a receptive field and by placing memory next to processing elements, Quadric’s products can perform feed-forward neural network inference at high speed, low latency, and low power. For RESNET50, batch one latency is an important compute metric. It quantifies the total amount of time to compute a single RESNET50 computation end to end. Quadric processor is latency and power optimized to meet the demands of the edge. Figure 8 shows the batch one latency versus an NVIDIA Xavier. Figure 9 shows the compute efficiency against the same product. With these performance and power metrics, Quadric’s processors are well suited for all low-power high-performance edge applications.
While RESNET50 performance is an important benchmark, the application of Artificial Intelligence is an ever-evolving field. High-performance neural network acceleration at low latency is only one aspect of what software developers needs. More importantly, the developer requires the freedom to change and experiment with network architecture. The algorithms of tomorrow will not look like the algorithms of today, and in the domain of Artificial Intelligence, that change is happening faster than ever. Quadric has developed a scalable general-purpose parallel processing architecture that gives developers the freedom they need to develop and deploy the networks of tomorrow.
While artificial intelligence applications solve or improve many algorithm domains, general-purpose high-performance computing algorithms are required to be deployed alongside neural networks. In the next section, we will describe, at a high level, a full stack Sense -> Decision loop.

**Full Robot Application: Sense and Control**

Before a robot can make decisions and effectively navigate and operate in real-world scenarios, it must be able to read and act upon the environment in a way similar to humans. It must sense and perceive terrain and geography while recognizing stationary and in-motion objects. This process begins with multiple sensors designed to detect objects, measure the terrain, and provide the robot with a sense of its location. Software and algorithms then make decisions based on the data provided by the sensors and send directions back to the robot. The Quadric Supercomputer is capable of running all algorithms required to implement an entire sense and control loop within a robot. To sense the environment, we employ two cameras, an inertial measurement unit, and one LiDAR.

![Full Robot Application: Sense and Control](image)

*Figure 12: On-board sensors such as cameras and LiDAR capture data about the environment, the data from the various sensors is fused, and software and algorithms interpret this data to decide and interact safely within the environment.*

Most datasets for self-driving are proprietary, the example provided by [AVS.auto](https://av.auto) is used to explain, at a high level, various aspects of the Self Driving Car control loop and how they map to the Quadric Supercomputer. AVS is an open source visualization framework open-sourced by Uber. LiDAR, two cameras, and an inertial measurement unit are used to transduce the analog information surround the vehicle’s environment to a raw digital representation. The control outputs are steering wheel angle and throttle position (e.g., longitudinal and lateral control).

A robot utilizes cameras to capture images, and high definition video streams of objects in the surrounding environment then make decisions based on its resulting interpretation of the perceived environment. It employs [LiDAR](https://lidar.com) to determine the distance between objects and the robot by sending out approximately 100 light pulses/three-tenths of a second, then measuring the differences in return times and wave-lengths to assemble a 3-D digital representation of the environment. It uses an Inertial Measurement Unit, or IMU, which consists of accelerometers and gyroscopes that measure orientation, speed, and gravitational forces. These metrics are reported back to the robot to key it into where it resides within its environment. It can then form productive paths and ultimately issue braking, acceleration, and steering angle control instructions.

Algorithms close to sensors are typically used to condition and remove noise from input sensors. For example, the developer may run a histogram equalization to feedback exposure parameters to the camera. Color space conversion can be utilized to trivialize downstream computations, such as traffic lanes extraction or traffic sign detection. Speeds and positions of other objects can be pre-computed to inform downstream vector space annotation during localization. We may want to run a connected components kernel on the LiDAR data to infer which points belong to the same object. Again to make localization and prediction easier downstream.
A robot's cognitive function rests in its ability to predict and infer based on the data gathered from the sensors. Software that relies on high-performance interpret images, terrain, and self-location. In the case of driverless cars, the vehicle needs to know the difference between a stop sign and a pedestrian, between a minivan up ahead and a semi on the right, then make split-second decisions based on this knowledge. Neural networks have proven to be the defacto standard for classification of objects within the perceptive field of the robot. Classical algorithms such as optical flow and segmentation are still very useful alongside neural networks to augment their use to understand the robot's environment with higher accuracy. Further, the way the neural networks behave can be reinforced with rule-based expert knowledge within a particular domain.

The CNN matches objects in the real-time image captures and video streams with those it is trained to understand. To isolate different objects in an environment and bring them into better focus relative to the background. Similarly, segmentation takes a digital image and divides into simpler components so that it is easier to analyze. Objects in an environment are a mix of stable and in motion, and because a robot is also in motion, optic flow is used to represent the motion of objects relative to the robot. Since their position and the view will change concerning the movement of the robot, both static and in-motion objects will be represented as vector fields. Because a photo is a 2-D representation of a 3-D landscape, the camera will capture an image of an object from different angles. As the convolutional network is seeking correlation to match the captured image with stored images in the database, it needs to identify a key point whereby the image, even at an angle, can be matched with its appropriate counterpart in the database. All of these approaches help to prepare and unify the robot's environment so that they may plan and make a decision on which actions to take.

<table>
<thead>
<tr>
<th>Category</th>
<th>Algorithm</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Processing</td>
<td>Histogram Equalization</td>
<td>0.30 ms</td>
</tr>
<tr>
<td></td>
<td>Integral Image</td>
<td>0.35 ms</td>
</tr>
<tr>
<td></td>
<td>Histogram</td>
<td>0.30 ms</td>
</tr>
<tr>
<td></td>
<td>Channel Combine</td>
<td>0.35 ms</td>
</tr>
<tr>
<td></td>
<td>Channel Extract</td>
<td>0.30 ms</td>
</tr>
<tr>
<td></td>
<td>Color Convert</td>
<td>0.10 ms</td>
</tr>
<tr>
<td>Vision</td>
<td>Canny (5x5 Filter)</td>
<td>2.75 ms</td>
</tr>
<tr>
<td></td>
<td>Harris Corners</td>
<td>1.50 ms</td>
</tr>
<tr>
<td></td>
<td>Hough Lines</td>
<td>17.0 ms</td>
</tr>
<tr>
<td></td>
<td>Dilation/Erosion Filter</td>
<td>0.10 ms</td>
</tr>
<tr>
<td></td>
<td>Box, Gaussian Filter</td>
<td>0.10 ms</td>
</tr>
<tr>
<td></td>
<td>Gaussian Pyramid</td>
<td>0.50 ms</td>
</tr>
<tr>
<td></td>
<td>Median Filter</td>
<td>1.50 ms</td>
</tr>
<tr>
<td>3D Reconstruction</td>
<td>Connected Componer</td>
<td>5.00 ms</td>
</tr>
<tr>
<td></td>
<td>Dense Optical Flow</td>
<td>14.0 ms</td>
</tr>
<tr>
<td></td>
<td>Stereo Depth</td>
<td>15.0 ms</td>
</tr>
<tr>
<td></td>
<td>LiDAR Scan Matching</td>
<td>0.70 ms</td>
</tr>
</tbody>
</table>

Table 1: summary of algorithm benchmarks running on a single q1-64 processor.
To arrive at their objective—both literally and figuratively—robots must know how to formulate and follow a path within an environment. Thus, they must have an understanding of their position within the frame of reference. A stereo camera, equipped with two or more lenses, can assemble captured images from a scene and translate them into a 3-D representation, essentially constructing a map of the environment. As the robot moves within the environment, and as the objects within the environment are also fluid, there is a need to constantly update the map as well as the location of the robot within it. This process is called Simultaneous Location and Mapping, or SLAM.

A robot navigating an urban environment, such as the one in the AVS demo, must predict and plan the best path based on all information garnered from previous steps. It has utilized all of the above approaches to sense the environment, classify objects within that environment and place itself within the surrounding environment accurately. Now it must decide what plan of action it will take and finally make a decision: speed up, or slow down; swerve around an obstacle or come to a complete stop. These choices are not easy ones to make. And ideally, they must be made instantaneous. Because as time is taken to make a decision, the environment truth that the robot has built up for itself is getting stale. Because of this, computer algorithms utilized to make decisions must generate as many possible futures predictions and decide which one is the best course of action with minimum latency. Dijkstra's shortest path first algorithm, developed in 1959, which calculates the shortest possible distance between two nodes, and A*- an extension of Dijkstra's algorithm- that assists in finding a path between two points, or nodes. These algorithms run natively on the Quadric Supercomputer and Quadric Processor.
Path planning is hard and often single-thread limited and data-access intensive. Data patterns are random as opposed to the predictable data patterns that are present in graph-based algorithms such as feed-forward neural networks. These types of applications are where Quadric’s technology stands out. The Quadric Processor has an efficient mix of both dataflow paradigms and random access paradigms at the hardware level. This unique characteristic makes the Quadric Processor the first purpose-built architecture capable of executing a full stack sense and control loop, including path planning, for complex robots such as self-driving cars.

**Hardware Ecosystem**

Quadric’s hardware offering is threefold offering that will be staggered in time. The first product that we will offer is Quadric Supercomputer. The Quadric Supercomputer integrates four Quadric q1-32 processors. Secondly, we will be releasing the q1-64 processor on its own for system integrators to design into their own purpose-built edge computers. Lastly, Quadric will make the Quadric Processing Array available for SoC integration.

**Quadric Supercomputer**

The Quadric Supercomputer is Quadric’s first single board computer product and the world’s first purpose-built supercomputer for autonomy and robotics applications. It supports standard input/output for 8 HD cameras, LiDAR, radar, and IMU, along with the computational horsepower to back it up. With low-latency being the primary design principle, a developer can minimize the total photon-to-decision loop time. This newly enabled control will enable safer, smarter, more reactive robots and autonomous vehicles.

![Quadric Supercomputer](image)

**Quadric Processor, Array, and Core**

Quadric’s underlying compute element structure, the Quadric Array, is a scalable compute fabric containing the q1-32 Quadric Array fabric. Depending on the market and application, the Quadric Array can scale to meet the demands of various uses on the Edge.
Figure 16: At the heart of the Quadric Array is the Quadric Core (rendered above), a proprietary and purpose-built compute element that helps accelerate algorithms with heavy parallelism and data locality. A q8 contains 64 Quadric Cores; a q64 contains 4096, etc.

All embodiments of the Quadric Compute Array are supported by Quadric’s Compiler and Software Ecosystem, as discussed in the previous section. Maintaining software support across all embodiments of hardware allows for the development of algorithms once and the deployment of the algorithms to various endpoints.

For example, a RESNET50 neural network can be implemented by the end user. Depending on the target Quadric Array product, the neural network will have varying levels of performance. The following table offers the reader examples of single batch latency and power for various Quadric Array sizes.

<table>
<thead>
<tr>
<th>product</th>
<th>Array Size</th>
<th>RESNET50 Latency (ms)</th>
<th>Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1-128*</td>
<td>128x128 (16384)</td>
<td>0.266</td>
<td>100</td>
</tr>
<tr>
<td>q1-64</td>
<td>64 x 64 (4096)</td>
<td>0.65</td>
<td>12</td>
</tr>
<tr>
<td>q1-32</td>
<td>32 x 32 (1024)</td>
<td>1.2</td>
<td>4</td>
</tr>
<tr>
<td>q1-16</td>
<td>16 x 16 (256)</td>
<td>3.4</td>
<td>0.8</td>
</tr>
<tr>
<td>q1-8</td>
<td>8 x 8 (64)</td>
<td>11.6</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 2: INT8 performance The quadric array contains 4096, 1024, 256 or 64 Quadric Cores, which can be scaled up to meet the demands of various uses on the edge, depending on the application. *q1-128 with HBM2 memory.

A chip utilizing the q1-64 Quadric Array is well suited for high-performance low-latency applications where responsiveness and total edge computing are the primary design parameters. The q1-32 Quadric Array is incorporated in Quadric’s first processor offering: the Quadric Processor. This IP will see a total of four placements within the Quadric Supercomputer. The q1-16 and q1-8 are well suited for incorporation into product-specific SOCs.
Applications

The applications for Quadric’s edge supercomputer are those that require ultra-low latency, high bandwidth, and energy efficiency. Similar to self-driving cars described above, several well-known technologies are seen as having the potential to transform society, but have yet to overcome computational barriers within the power and latency constrained environments inherent in the current platforms.

Transportation

Current computing solutions available to self-driving auto manufacturers and users are primarily application-specific standard products (ASSPs), which are integrated circuit products designed for limited applications and sold to various users within that market segment. Production of ASSPs is an expensive and time-intensive process, but they provide high performance, low energy compute solution. The algorithms of ASSP’s are written for the end-user and built-in to the semiconductor device, meaning that though the product can be sold to multiple users, there is not a possibility for the user to program their tailor-made algorithms. As the world of AI and driverless cars is in rapid development, and best-practice algorithms in constant flux, the ability to create and modify algorithms with open source software programs gives the user flexibility to push the creative envelope. Quadric’s first product offering allows for powerful user experience, without increasing energy consumption.

Mobile SoCs

As mobile phone users increasingly utilize AI-backed applications such as voice assistance, intelligent imaging, and facial recognition, phone manufacturers will be looking to include AI processors in their SoC platforms. Recent research suggests that by 2022, three-quarters of smartphones produced will have onboard AI, or about 1.25 billion smartphones (compared with the 190 million that utilize AI today). Currently, smartphone AI processing relies on the cloud and various processing components within the phone itself. However, the latency inherent in these processing methods limits the applications. With the race on among smartphone manufacturers to develop the best mobile SoC’s for AI applications beyond the trivial, the demand for low latency, high compute solutions only possible via edge supercomputing will also increase.

Camera Sensors

While certain AI capable devices may have only one or two sensors, others- like self-driving cars as discussed above- have several. When one considers not just one car, but entire intersections and cities filled with commuters, it's evident that the total amount of sensors added to the network will be immense. One estimate predicts that by 2020, the number of devices in IoT will be 50 billion. Cameras, LiDAR and other sensors that need instant feedback will be less reliable as the number of sensors- and thus raw data- is sent to the cloud for processing. Some concerns, besides latency, include limited bandwidth and potential privacy violations. Edge supercomputing moves the processing onto the device itself, or nearer, reducing latency and increasing security. It solves the problem of unreliable internet connection so that that camera sensors can work continuously regardless of distance from cloud infrastructure.

Smart Sensors

As companies, consumers, and developers find further applications for AI, the types of sensors on the market will increase. These sensors will be designed for more specific end-users, and require precision and accuracy that can lead them to be considered “smart sensors.” With CNN’s and deep learning networks, sensors will be more than just pre-programmed, but rather learners within their domains, and further, need to communicate with one another. Sensors in networks in communication with one another will only serve to improve accuracy and precision in the sending back of decisions. The increased use of multiple sensors all gathering different data simultaneously requires high compute, low latency. In military and surveillance applications, the
demand for tight security is critical. In heterogeneous processing, each hardware component in the heterogeneous compute stack has its unique software framework and programming models, which creates software complexity and hardware constraints. Edge supercomputing provides lower latency, higher computer, lower energy consumption, and more security. Quadric’s SoC and software flexibility resolves the problems caused by the complex hardware/software interplay in typical heterogeneous computing systems. With “smart sensors” relying heavily on deep learning, the ability of programmers to write their software, and improve upon previous models, is a huge benefit.

Robots in the Industry

Industries in which robotics are being looked into to improve processes, increase efficiency and ease the work burden on humans include agriculture, healthcare, manufacturing, construction, quality control, military, and banking, among others. Automation in manufacturing processes already relies on robots to perform menial or dangerous tasks such as welding, handling of raw material, and packaging. The further integration of machine vision and deep learning is allowing for manufacturing robots to adapt to problems as they occur and respond quickly with a solution. The trend in the manufacturing industry is moving from product automation to “smart” automation, and this evolution is based on the collection and processing of data from the manufacturing process as it is occurring and being captured by sensors. For instance, an industrial arm may be augmented with a camera to improve assembly and testing. Robots and sensors in geographically dispersed points will be able to share real-time information and improve processes for a company not just in a single factory, but at every step of the supply chain. The low latency and security necessary for these robotic applications will require on-site IT infrastructure. Edge computing brings data processing on-site, allowing for a host of robotics applications that can improve efficiency in several industries and processes. Quadric’s flexible software approach will allow developers to match the algorithms to idiosyncratic industrial processing blueprints.

Augmented and Virtual Reality

Immersive systems that provide users with computer modified and computer generated experiences are one such arena. Augmented Reality (AR), in which digital renderings are imposed over a real-world environment, and Virtual Reality (VR), where the environment in which the user interacts is simulated in 3-D- both require real-time computing of graphics, sensors, and user inputs for the experience to be fluid and believable. If the user senses a delay in input-feedback, the human brain detects the artifice and the experience fails. While the general public is most familiar with AR/VR headsets for gaming experiences, uptake by industries such as military, healthcare, retail, mobile phones, and entertainment could create a multi-billion dollar industry in the coming decade. For that to happen, there needs to exist a computing architecture to support visual displays of 100 - 120 frames per second without intensive battery drain and the need to be tethered to a powerful PC station. With the low latency promise of edge super-computing, the potential for untethered mobile AR/VR gets closer to reality, and with Quadric’s open software feature, a developer or company with a very specific AR/VR application can experiment with and deploy algorithms tailored to their unique objective.

To demonstrate this further, consider the possibilities for AR in healthcare. A nurse taking blood could superimpose a digital map of the veins over the patient, improving accuracy, or a surgeon could view radiology images overtop of the patient he is about to remove a brain tumor from, identifying in real time the best point of incision. Consider also the split-second accuracy that would be required as the surgeon moves about the patient and the superimposed radiology image needs to be continuously fitted to the patient in the real world. Computing latency must be ultra-low, meaning the processors need to be on-site or within the nearby vicinity, and because of the hyper-specific application, the software solution must allow for tailor-made flexibility.

Conclusion
The immensity of data collected by sensors, and the immediate need to process that data to send back decisions, requires data processing speed and security that cannot be achieved in remote data processing centers or the cloud. This need for immense processing capability, as well as having the processing unit in closer physical vicinity to the sensors, provides the impetus for edge supercomputing. The dependence on algorithms and deep learning via CNN's and other high-performance algorithms to not only process visual stimuli, but to learn from it, requires an adaptable and malleable software solution conjoined with the processor technology that enables it. Quadric's team has designed a purpose-built edge supercomputer specifically to fill this void. With Quadric’s technology we are enabling the future of algorithms to be deployed today.