White Paper: Accelerating Edge Computing with Trion® FPGAs

Edge computing has become a common buzzword. But what is it exactly, and how does it compare to cloud computing (which is a common methodology)? This white paper discusses edge computing, what it is, and how it relates to cloud computing. It also looks at some of the trends and requirements that are driving the interest in edge computing, as well as challenges in implementing these applications. The paper then discusses how FPGA technology, and 易灵思 FPGA solutions
in particular, can be advantageous to solving those challenges.

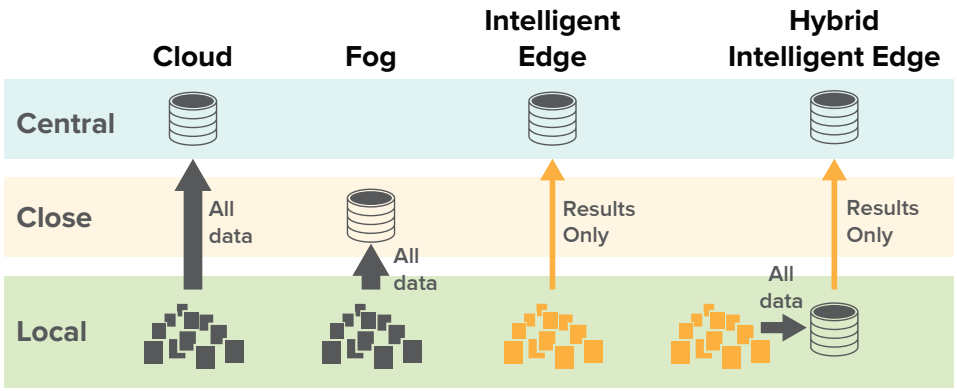
Cloud and Fog Computing

In cloud computing, data is gathered from many locations and fed into a centrally located system that stores the data and performs any analytics and/or computing using large-scale systems. The individual data streams are pushed up through connectivity networks, which vary depending on where the source devices are deployed, what those edge devices are, the data size, and speed with which they are gathering data. The connectivity medium may change node by node, and the networks may be public, private, shared, or dedicated, depending on each need.

In this model, there are costs involved with the connectivity itself. Meaning, the size of the pipe to pull all of the data from the edge and then get it to the cloud has an inherent cost. The more data there is, and the faster it is being gathered, the bigger the pipe must be, which generally drives costs upwards. For simple sensor data, connectivity may not be an issue; when gathering video data however, there can be further bandwidth challenges and concerns. Ultimately, there is a cost associated with getting the data to the cloud and storing it there.

Cloud computing also has an associated lag time, or latency. You must gather and transmit data to the cloud to perform computations, potentially pushing results back down to the edge device for further action. Each step has an associated time delay, nothing is instantaneous. That lag time can also have some variability to it, meaning, it is not always deterministic. The process depends on the connectivity type, the relative geographic relationship between the devices and the cloud server, and the actual cloud computing time required. Some of these resources are shared, which can also cause variability in result time. No matter how the latency or vari-

Figure 1 Comparing Compute Options



ability forms, the result is the same. Application performance becomes degraded or unusable.

Modern applications and architectures, including the cloud, are becoming increasingly sensitive to latency as applications require real-time communication between a client and server. Any slowdown or disruption in service is abundantly clear to end users. To address some of the latency problems, a hybrid form, sometimes called "fog," has evolved. In fog arrangements, data centers are distributed over wider geographic areas to help minimize some of the networking time to get to the servers.

Intelligent Edge Computing

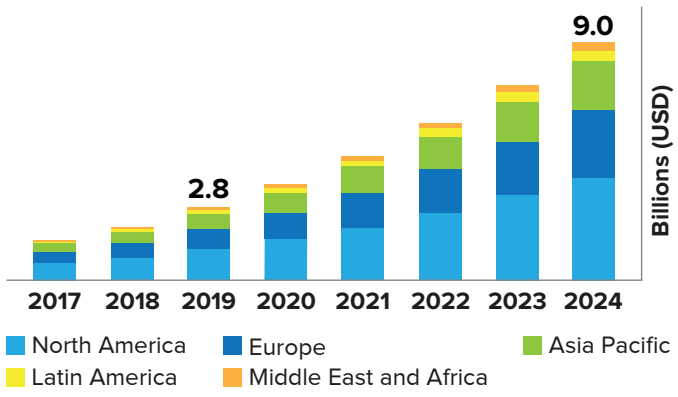
Edge computing, as the name denotes, is when computing happens closer to where the activity is. The computing is physically located in close proximity to the source of the data that is being gathered.

Edge computing is not really application or device specific. Many and any applications can take advantage of the capabilities that edge computing brings. Edge applications can occur in smart factories, smart cities, smart homes, smart transportation, smart consumer, really, any and all applications. There are some application characteristics that may play better in the edge vs. in the cloud though, and this paper discusses some of those characteristics.

To accomplish edge computing, you must include some level of processing function in the edge device that is gathering the data. Today, smartphones and many other mobile devices already do some level of edge computing or processing.

Edge computing can vary in the amount of computing done locally. For example, some edge applications may only perform a portion of the required computing. Then, those pre-processed results are pushed to the cloud or other computation nodes for further algorithms or analysis. In other cases, the analysis is done locally and only results, if any, are passed on. In these cases, the data that is gathered stays local, keeping it secure.

Figure 2 Edge Growth by Region



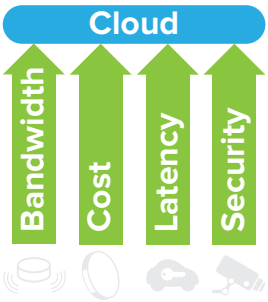
Source: https://www.marketsandmarkets.com/Market-Reports/edge-computing-market-133384090.html

Sometimes, there are many edge devices that are co-located. All of those edge devices could be directly connected to a local aggregator that does the computing. In this hybrid case, the edge analytics system, the aggregator, is still geographically local, but the computing is shared over a larger set of input data devices.

As Figure 2 shows, the need to do more local edge processing is growing, and this explosion appears to be a global market trend. Insight from marketsandmarkets.com shows that all markets are tracking growth for edge for the next 5 years in all geographic locations.

Why Do We Need Edge Computing?

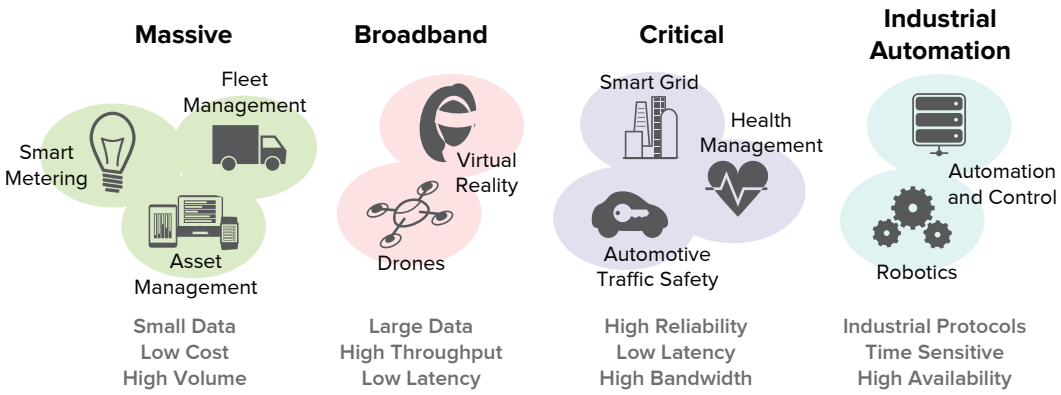
The growth towards edge computing is a worldwide phenomenon driven by the perceived challenges cloud computing faces.



- Bandwidth—The bandwidth challenge is dependent on where the device is and what amount and speed of the data you are gathering. The challenges lie in the interconnect strategy to connect the devices to the cloud and the performance needed to accomplish the connection.
- Cost—There are several points in the signal chain that incur costs for the cloud model. There is the actual connection cost to move the data to the cloud, involving fixed and variable costs. These applications are commonly always on, which drives the variable costs, continuously pushing all the data through the pipes. Lastly, once all of the data gets to the cloud, there is an associated storage cost.
- Latency—Nothing happens instantly because time is needed to get the data from the edge, over the connected network, into the cloud's data center, computed, and then potentially back down the network to do something at the edge. In some cases, the "lag" induced by data moving to the cloud and then computed may be beyond what is acceptable for your application. There also may be some level of variability to the lag because the resources can be shared.
- Security and privacy—Once the data has moved beyond the local device, there is potential for data to be insecure. The type of data may drive a desire to avoid any potential "theft or manipulation." Particularly sensitive data, such as faces, people, medical information, etc., may drive edge computing analytics for security reasons.

When you compute at the edge, you have more control of the process. The computing is done locally, under your system's complete control. It is more deterministic because nothing is shared. The computing also happens faster without the connectivity and computing lags you have in a cloud computing environment. Applications such as self-driving transportation may have requirements that make cloud computing options

Figure 3 Edge Applications Are Diverse



challenging. In some extreme applications, for example devices in places that are geographically remote, there may be very limited ability, if any, to connect, which obviously drives you to an edge computing methodology. Finally, when data is retained at the edge, it is not transferred over public networks or to the cloud, making it inherently more secure.

Diverse Edge Needs

Figure 3 illustrates a variety of edge applications. Examples to the left involve lower amounts of data and potentially large installment bases. They might be gathering simple sensor data, such as environmental, status, geographic location, or time stamps. Moving to the right, complexity, bandwidth, and criticality increase, as well as data set size and requirements for decreased latency, high availability, and time sensitivity, real-time results.

Applications for real-time automation control, smart grid, AR/VR, drones, health management, and fleet management differ greatly, but all of them need edge computing in some form and will be part of the projected future worldwide growth. With so many diverse, real-time applications, having a computing architecture equally as diverse and scalable becomes a necessity.

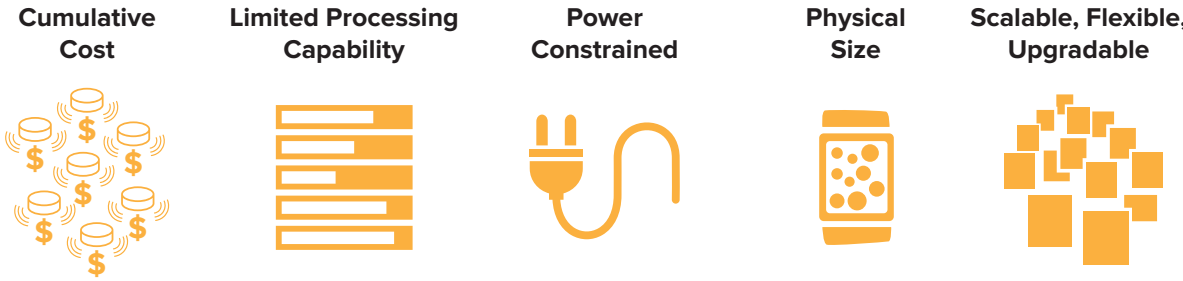
Challenges for Edge Applications

Edge devices are not without their own challenges, and many architects and designers are looking for the best solution to optimize pure edge computing and edge aggregation computing. These challenges (Figure 4) are the five-headed beast of cost, performance, power, physical size, and "change-ability" (meaning scalable, flexible, and upgradable). These are "whack-a-mole" challenges, where optimizing for one characteristic may create problems for one or more of the others. System-level decisions need to account for all these criteria to some level, although different applications and use cases obviously have their own ranking of importance.

Let us look at these challenges in some more detail.

- Cost—As a purely edge device, the cost associated with each node is not shared, it is cumulative. So whatever cost is associated with that node is multiplied by the number of nodes you need. Volumes for IoT devices can be quite large, and there is a linear total deployment cost that is dependent on the individual device cost. Decisions on the next indices can have a large effect on the overall cost of each node.
- Performance—Performance requirements depend on a number of factors. You need to consider the frequency of data gathering and computing, and whether the application is always-on application and continuously gathering data and performing analysis. Or, the application may be time stamped to gather at a specific time or based on a local or remote trigger.
- Power—Many edge devices do not have access to unlimited power. The device may be battery powered or use alternative power such as solar, or wind. Understanding what power level you have and potentially how and when you have access to that power further defines your architecture.

Figure 4 Challenges at the Edge



- Size—The size of the edge devices further defines your component choices. Many edge devices tend to be physically small, which restricts which components you can use. This limitation can affect multiple decisions such as battery, computing architecture, cost, etc.
- Scalable, flexible, upgradable—Many nodes may be similar but have different performance levels, potentially defined by a more challenging location. Having a solution that provides some scalability allows reuse in slightly different capacities. Edge computing is a young and growing market; it is changing and evolving. Having a core design that is flexible enough to interface to many sensors or data types lets you reuse that core design and easily port it to another use case with minimal effort. Many edge devices are deployed in locations with limited accessibility, so being able to remotely drive upgrades to the system is helpful.

All of these challenges affect decisions about your computing/analytic solution architecture. Different priorities can create hybrid approaches, where local aggregators may interface to more basic edge devices with the aggregator doing the computing. This is still edge computing, the data is still being processed locally, but this multi-level approach can be a better solution for your price/performance/power/size equation.

When FPGAs Meet Edge Computing

After this in depth discussion on the challenges facing edge computing, let us look at how field programmable gate arrays, or FPGAs, can help alleviate the problems and help meet system requirements. While not all FPGA families have all the characteristics required for your use cases, FPGAs do offer a great place to look for a solution, and you simply need to understand the differences in how companies have architected their products for specific use cases. Not all FPGA vendors are the same and not all families within a vendor are the same.

Advantages: Performance and Latency

CPUs are sequential processing devices. They break an algorithm up into a sequence of operations and execute them one at a time, serially. Inherently, the FPGA architecture is highly parallel, and the processing functions and memory units are accessible in parallel. With parallelism, you can accomplish a lot of processing concurrently. An FPGA might be able to process a particular function in a single stage or a handful of stages; in contrast, a typical processor takes multiple sequences to do the same function using its single arithmetic logic unit. Additionally, FPGAs have multiple multiplier blocks and large amounts of on-board block memory, enabling a lot of on-chip processing, which is faster than going off chip.

A custom hardware solution, whether in an FPGA or in some other form, presents the opportunity to unroll the algorithm so that each part of the process is done by dedicated hardware arranged in a pipeline, much like an assembly line.

Figure 5 gives a very simplified view of the concurrency advantage of FPGAs over software-oriented processor solutions. The functions of R1, R2, R3, R4 in an FPGA can be accomplished all at the same time; a typical processor solution would compute them in serial. A dedicated hardware or logic approach can actually be a more elegant solution than a processor engine and some code.

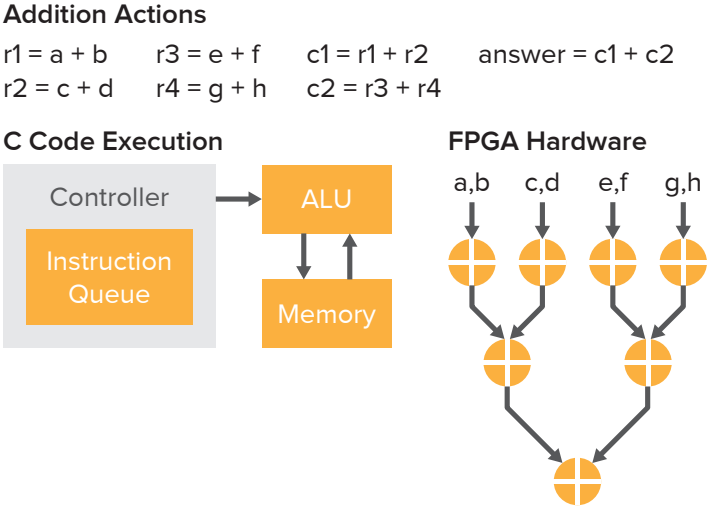
Advantages: Size and Power

Many edge applications are always on, gathering data, computing, and acting on it. Some devices that may be considered low power utilize a sleep mode. Although it can be good, sleep does not work well for always-on applications, which are common in edge computing. FPGAs, on the other hand, if designed with low power in mind, can deliver low power while always on. For example, FPGAs can use parallelism to slow internal clock frequencies. The lower clock frequency is associated with lower dynamic power.

It is important for you to look at the power characteristics of each family though, because each FPGA family has been designed with specific applications in mind. Static and dynamic power vary from vendor to vendor and family to family, and even within families, so you want to review them carefully with your design specifics in mind.

Lastly, edge applications, due to their locations, can have size constraints. Some FPGA families have been designed with small die size, to take advantage of many of the small BGA packages available. Smaller packaged devices usually have fewer IO pins to connect to the other components in the system, so you need to make sure you have ability to choose the smallest package size that has the number of I/O pins you need.

Figure 5 Comparing Sequential and Parallel Processing



Advantages: Flexible, Upgradable, Secure

As the name states FPGAs are themselves programmable. Moreover, you can change the programing at anytime, even after production. This feature gives you the ability to update and change systems in the field after you deploy them.

Most FPGA vendors group FPGAs into families of related devices that are generally footprint compatible within a common package. This flexibility means you can choose the largest device for your original design and debug, and then you can switch to the smallest device that still meets your requirements for the end of development and production. With this capability you can trade off performance, power, and cost for different systems or boards.

FPGAs are very configurable in how they interconnect with other devices on the board or in the system. One of the most popular FPGA capabilities is their ability to connect to virtually any device. Edge computing is in its early stages; therefore, the ability to interconnect with other devices is an important system level attribute for design reuse. FPGA I/O pins are configurable to many standards and can add some level of bridging. This flexibility is difficult, if not impossible, to do with other processing engines or dedicated ASSPs.

Finally, you can use FPGAs' flexibility to build functions to further secure data that is being pushed to the cloud. FPGAs are blank pages you can use to create any function you need in addition to the edge processing capability.

Accelerating Edge Computing

Much edge computing/analytics is associated with vision systems, be it pattern detection, sequences, or actions, vision systems are a big driver in the analytics world. These applications can span industrial, consumer, surveillance, and automotive markets. Anything from the doorbell that recognizes a person is at your door all the way to counting the number of devices that are moving through a production line or detecting that there are human bodies present in a building that is experiencing an emergency. Systems that analyze and make decisions based on a video stream are an important part of today's market, and there will be more unique challenges as time goes by. FPGAs are very good at accommodating these new requirements.

Additionally, with the advent of more and varied self-driving machines, be it cars, robots, luggage, or garbage cans, the need to use vision to make control decisions continue to flood the market with new, ever challenging applications. These use cases will add new requirements to power, performance, package, and price.

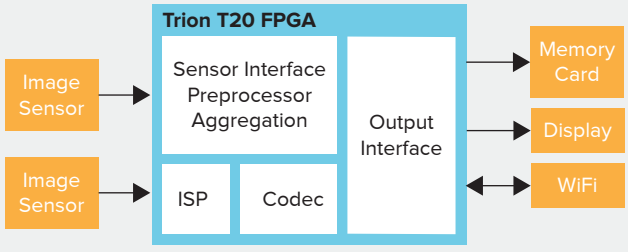
In general, FPGAs do a good job in video processing. The parallelism helps you perform many actions concurrently, which allows you to tighten up any processing loops to optimize the system latency. Also, you can create internal bus structures of various sizes for video processing optimization. The flexibility of FPGA I/O pins lets you interface to many devices natively, so you can have very integrated designs to address power and space challenges. Vision sensors use a variety of I/O standards, so flexibility is important. Finally, FPGAs have large levels of embedded block memory and many multiplier blocks for processing and/or coefficient storage, which can be valuable in AI inferencing applications.

A number of edge computing applications benefit from acceleration.

- Convolutional neural networks (CNNs), a type of AI inferencing, involve a lot of matrix multiplication operations that are very computationally intensive, and are very good candidates for compute acceleration inside FPGAs.
- For applications that capture high-resolution images or videos and transmit them over a network to a cloud server for analytic operations, the images/videos need to be compressed to reduce data bandwidth. In this case, compute acceleration comes in handy.
- For sensitive data, the information must be encrypted before sending it over a public network. Advanced data encryption algorithms can be computationally intensive, and compute acceleration is helpful.
- There are scenarios in which the application aggregates data collected from multiple sources and sends it over a single transmission medium, for example, a sensor hub in an IoT

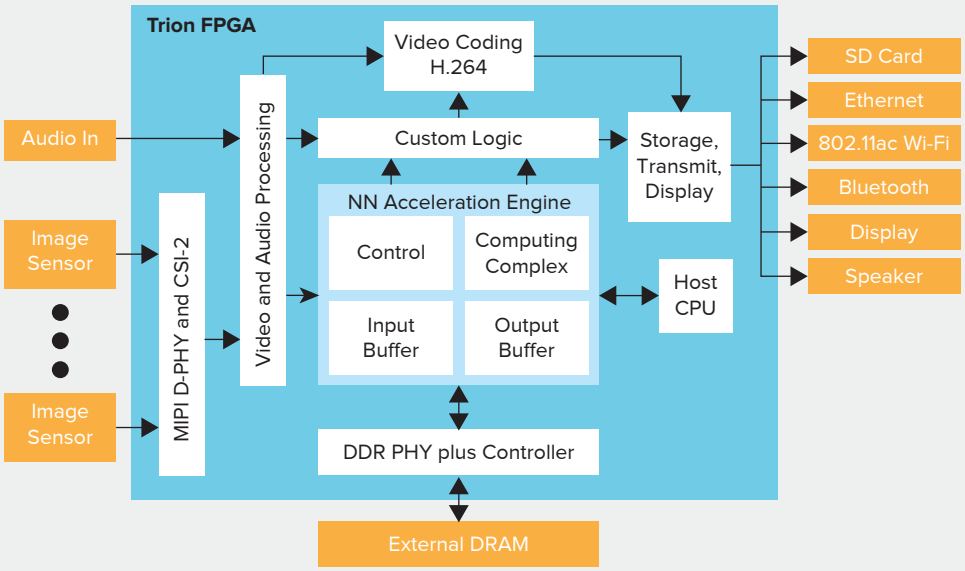
Use Case: Time Lapse Camera

In this simplified remote camera, one or more sensors collect data. The Trion FPGA aggregates and processes the image with functions such as debayering, color space conversion, etc. for display. The Trion FPGA may make decisions on what data is stored locally, perhaps defined by an environmental statistic, and continually flushes unwanted data as time goes on. Depending on the application, low power, integration, and small footprint can be important attributes, and Trion FPGAs hit these metrics.



Use Case: Smart Camera

This complicated, intelligent camera has multiple MIPI CSI-2 sensors. The FPGA provides image processing (as well as audio processing) and runs an inferencing engine based on a trained neural network defined by the implementer. Large amounts of internal block memory in the Trion FPGA allows you to keep much of the activity on chip. Additionally, the multiplier rich architecture provides additional of performance-power-area advantages.



application, data trunking in networking applications, or video trunking in big data analytic applications. Most of these applications require high speed and low latency, and they are usually accelerated by dedicated hardware.

- It is also very common for edge applications to convert data from one format to another in real time, especially when the data comes in at very high speed and requires low latency. In this case, the mission can only be completed with acceleration. Usually conversion is required to connect hardware that produce and consume data with different expected format, say converting video input data from MIPI to HDMI, from HDMI to DisplayPort, etc.

Besides vision sensors, other sensors can also be in play, such as environmental, audio, or vibration or sensors. Each sensor type drives its own need for computing or acceleration.

易灵思 Disruptive Programmable Fabric

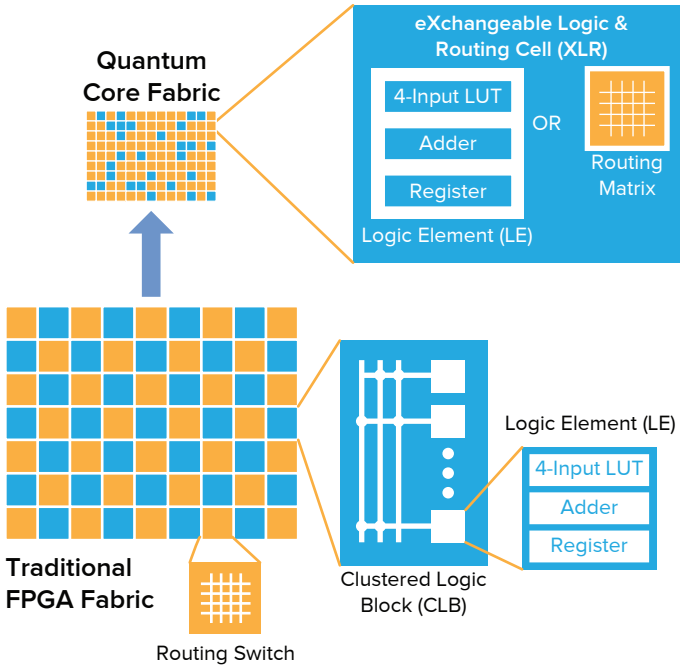
易灵思's unique programmable technology can help solve the problems facing edge computing applications. Trion is the family name for our low power, small form-factor, volume driven FPGAs. The Trion® family's disruptive technology comes from its unique architecture, called Quantum. Our Quantum™ technology gives our devices much of their unique disruptiveness and capability in the marketplace.

Quantum FPGA Technology

The basic building blocks for any FPGA are the logic elements and the routing fabric that connects it. When creating an FPGA, once the logic element design and fabric/routing is chosen, it is "spread" across the die and uses space and power whether your RTL design uses it or not. In a coarse-grained FPGA architecture, most designs utilize only a small fraction of the available functionality in the logic elements and routing switches, and much of that extra functionality simply takes up space, uses power, and is wasted.

The 易灵思 Quantum architecture has a unique fine-grained logic element called the exchangeable logic and routing or XLR cell. This unique logic element can create logical functions necessary for your design, and can also be utilized as routing. Leveraging this unique capability, our Efinity® tool delivers results that alleviate congestion issues and obtain much higher utilization than traditional FPGAs, while still producing the performance needed for mid-range, cost opti-

Figure 6 Comparing Quantum and Traditional FPGA Fabrics



mized families. Trion FPGAs have 7 to 8 metal layers and a compact logic cell, leading to a big advantage in power-per-formance-area over traditional FPGAs.

This architecture also allows flexibility, scalability, and quick time to production as 易灵思 products move to future and different silicon processes, or derivatives addressing different market needs. The Trion family offers up to 200,000 logic elements, which is larger than some of the traditional FPGA providers today. Furthermore, 易灵思 has modeled the Quan-
tum architecture to scale to a million+ logic elements, which allows us to migrate to future geometry nodes.

Architectural Advantage

When compared to competitive mid-range solutions, Trion FPGAs surpass those devices performance and utilization indices while delivering industry-leading power use. Figure 7 shows the power usage for Trion T20 FPGAs and comparable FPGAs.

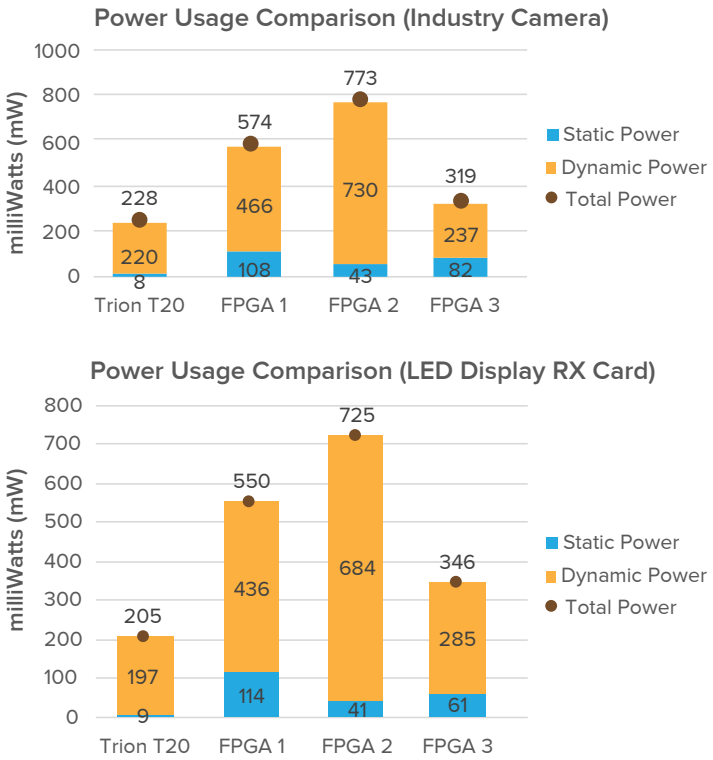
The XLR cell gives great flexibility and scalability, allowing better utilization and lowering the design complexity. Due to this flexibility, you can accomplish your requirements more easily, with better power.

Hotspots in the design, places where congestion might affect timing, can be alleviated by using XLR cells as routing instead of logic. During compilation, the Efinity software dynamically decides whether to use the XLR cell as routing or logic, and it optimizes silicon resource use specifically for the characteristics of each design.

In other architectures, you may need to move to the next bigger family member when utilization may only be 60 to 70%. With the Quantum architecture, you can utilize the amount of logic defined for that family member; you get what you paid for.

The Quantum architecture results in a small die, and drives low power, small footprint, priced optimized solutions, all items that are important for edge processing.

Figure 7 Power Comparisons



Trion FPGAs

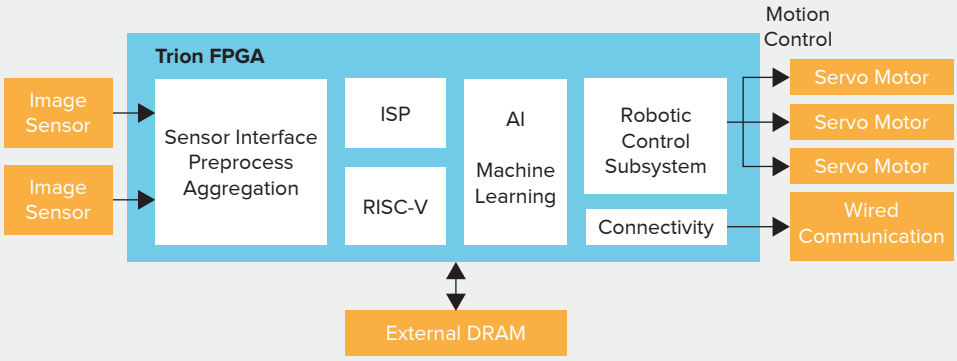
Trion FPGAs range from 4,000 logic elements to 200,000 logic elements and are based on a high-volume SRAM process. They are rich in features, including embedded block RAM and 18 x 18 multiplier blocks, which are very important in some edge processing applications, as well as many other applications.

Additionally, the smaller Trion family members have an optional mask option that allows you to eliminate the configuration device, thereby saving space and the cost of that device.

易灵思 has integrated some key hard block functions inside most Trion family members that are ideal for edge applications. These hardened blocks offer great camera sensor interfacing with multiple MIPI/CSI-2 interfaces plus enough

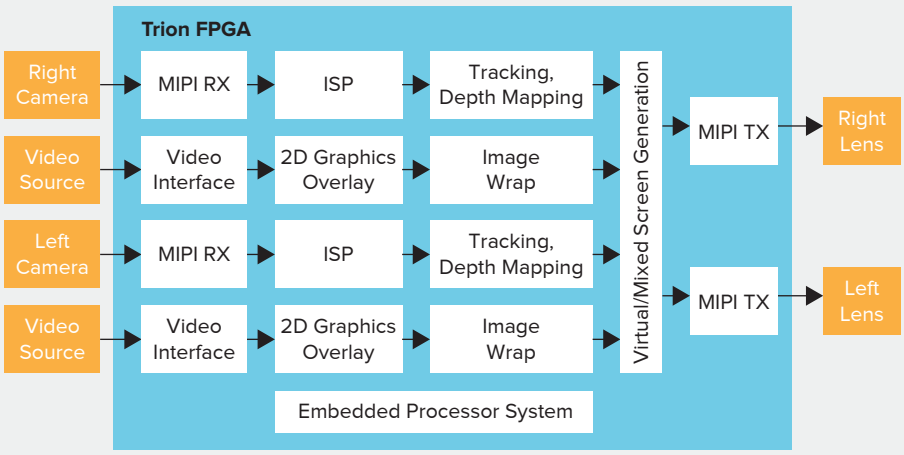
Use Case: Collaborative Robot

In the industrial market, collaborative robots combine machine vision and robotics, both of which require low latency, high compute, and low power. The FPGA can implement an AI or dedicated algorithm to process real time results, making decisions and driving control of the motors based on those inputs.



Use Case: Augmented Reality and Virtual Reality

Augmented reality and virtual reality products are smart devices that can benefit from FPGAs. Beyond entertainment, they can be useful for remote assistance and knowledge sharing. In these and similar applications, the performance advantage of Trion FPGAs is key. Additional driving factors are Trion FPGA's small size and power. Our hardened MIPI CSI-2 functionality helps reduce power, and Trion FPGAs have a smaller chip area compared to other solutions.



logic to accomplish image processing inside the device. Trion FPGAs also have a hardened DDR DRAM memory controller in some devices for interfacing to a frame buffer or some other system memory. Hardening these blocks allows Trion FPGAs to reach performance levels typically only available on much higher end FPGAs that incur more power, higher cost, and physically larger packages.

The Trion family has been very successful in many industrial power and LED controller applications. The low-power and small packages are also very applicable to IoT applications and edge applications across those markets and segments.

易灵思 has a broad market FPGA offering, meeting many needs for industrial, communications, consumer markets, etc. in addition to edge computing.

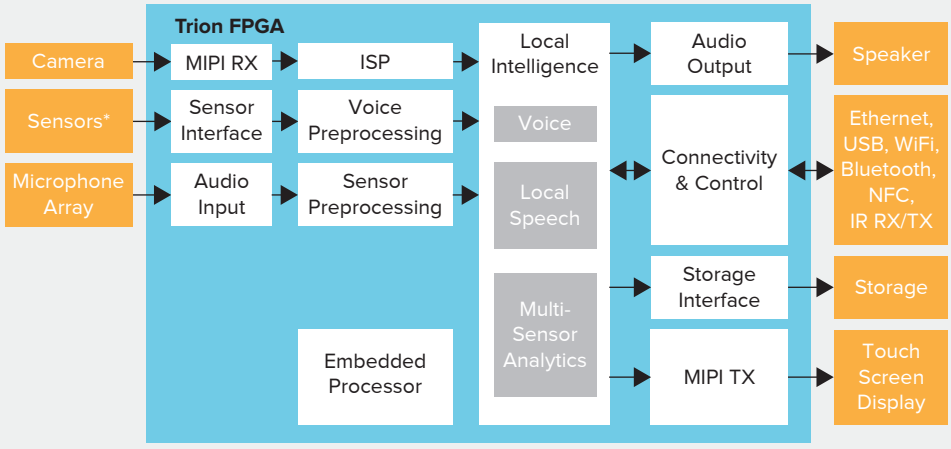
Excellent for Camera Interfaces

MIPI is an industry-standard organization that defines the interconnects needed for high-volume handset applications. D-PHY/CSI-2 is one of the most popular in the collection of MIPI defined standards. CSI stands for camera serial interface and is most likely the interface to the camera on your cell phone in your pocket. With so many of these camera sensors available today at such a low cost, many embedded customers are using those same cameras in their designs.

The simple challenge lies in the fact that most processors for embedded applications do not have a CSI port. This initiates the need for a bridge with enough logic to do some processing and manipulation to connect the camera sensor to the device. This bridging is where 易灵思 FPGAs can help.

Use Case: Smart Home Controller

This last example is a smart home application, or more generically, you could just call it smart life. These products have camera sensors, environmental sensors, body sensors, motion sensors, you name it, with localized processing and connectivity. Power and small size are important attributes in these smart home, smart life applications, especially if they are wearable.



* Sensors for pressure, relative humidity, temperature, UV, position

Trion FPGAs support the D-PHY specification up to 1.5 Gbps and up to 1066 Mbps performance for DDR3 DRAM speeds. This is leading performance for a midrange, cost focused FPGA solution. These FPGAs support x16 bit or x32 bit DDR DRAM data paths, depending on the package.

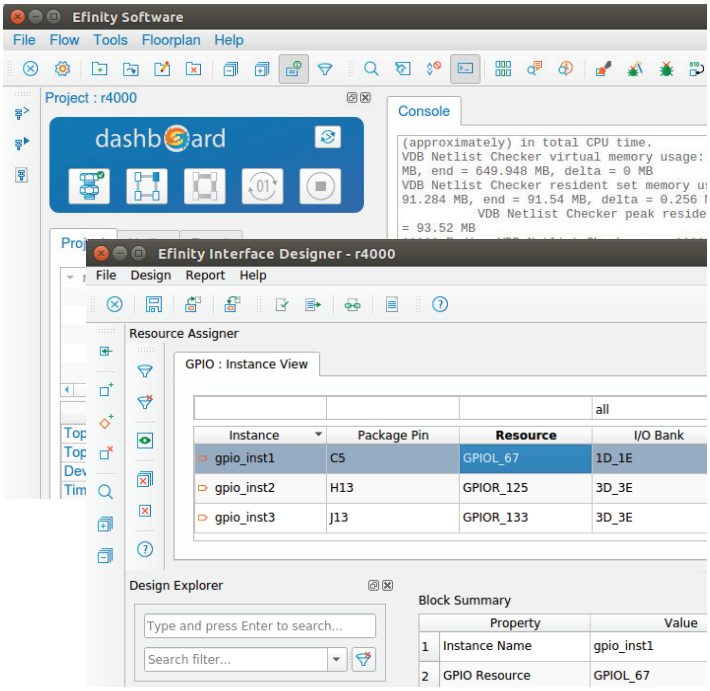
Trion FPGAs support the CSI-2 specification for 1, 2, or 4 lanes at 1.5 Gbps with 2 or 3 hardened MIPI/CSI ports, depending on the FPGA and package. In all of these CSI-enabled FPGAs, you can use both the RX and TX channels of each port simultaneously.

For edge applications that interface to CMOS imagers, having these integrated interfaces is a huge value in power, price, and package size.

Efinity Integrated Development Environment

The Efinity software is our internally created development suite. It supports standard VHDL and Verilog inputs and supports a traditional tool flow of design entry, mapping, place and route, timing analysis, Bitstream generation and programming.

The tool has two parts, the interface designer and the core designer. The interface designer is where you define the I/O pins that you need for your design. You also use it to set up configuration for any of the hard blocks, such as the memory controller or the MIPI/CSI hard blocks. The core designer is where all the logical functions happen, and where you set up timing constraints and run the rest of the tool flow such as mapping, place and route, etc.



Summary

Edge applications are diverse. There is a push to drive more processing to the edge vs, the cloud due to bandwidth, cost, security, and latency concerns. That shift is creating new applications and opportunities to use FPGAs as those edge processing entities.

Always-on edge applications in general bring further challenges, as they are deployed in very different and potentially remote locations. Getting a processing solution that meets your needs for performance, footprint, and power can be problematic, and the Trion FPGA family can help you get there.

WP-EDGE-COMPUTE-1.0

易灵思®