A Soft Control Architecture: Breakthrough in Hard Real-Time Design for Complex Systems

WHITE PAPER 2015/2016

KINGSTAR

Executive Summary

Embedded OEMs – especially those whose products have complex human-machine interfaces, manage many degrees of motion, and require hard real-time – have traditionally relied on field programmable gate arrays (FPGAs) and digital signal processors (DSPs) to meet precision and performance requirements.

Today, that hardware-centric model is undergoing intense scrutiny as OEMs face increasing market pressures to cut their costs, improve quality, and differentiate their products.

By adopting a soft control architecture, OEMs have an opportunity to do all three. They can differentiate their products and improve their competitiveness by significantly increasing yields/throughputs, and by shortening time to market. They can also reduce their bill-of-materials costs and shrink the compute footprint while simplifying and streamlining development, usability and training.

In the relentlessly changing world of technology, several important advances and trends have emerged that allow OEMs to transition to a soft control architecture that will not only move them away from dependence on FPGAs and DSPs, but also change the basis of competition in the equipment and machine tool industries.

The major trends favoring a soft control architecture include:

- Increasingly powerful x86 processor technologies
- Renewed commitment to commercial off the shelf (COTS) hardware and software
- Advances in, and availability of COTS-based field buses
- Convergence of components in system design
- The advent of touch-centered usability and motion sensing technologies

Although there are several competing approaches for capitalizing on these trends, a soft control architecture has emerged as the leader.

By adopting a Soft-Control Architecture, OEMs can differentiate their products and improve competitiveness by significantly increasing yields/throughputs, and by shortening time to market.

By utilizing hard real-time symmetric multiprocessing (SMP) support on multiprocessor architectures, and through tight integration to the Microsoft Windows environment, OEMs get a powerful, versatile soft control architecture that moves the hard real-time control logic, such as PLC or motion logic, from specialized hardware components to software components. With x86 processor advances, OEMs can take the C/C++ source-code logic that traditionally has been compiled and run on DSPs or FPGAs and port the source code to target a real-time operating system (RTOS), or a real-time extension to Microsoft Windows, such as IntervalZero's RTX64. The result is a hard real-time, SMP-enabled application that runs directly on x86, eliminating the need for an FPGA or DSP to perform the logic.

The software component runs on multicore, commodity x86 processors and uses both open standards and standard communications architectures, such as USB and real-time Ethernet. IntervalZero's RTX64 SMP real-time software extension for Windows is one of several key components of a soft control architecture. By enabling hard real-time, scalable SMP that leverages both Windows and multicore, RTX64 allows machine tool OEMs to improve yields and throughput; to shrink compute footprint; and to significantly reduce both their costs and their customers' costs.

Sources of Product Differentiation for Complex Hard Real-Time Systems

In equipment design, the real value – the source of the intellectual property and product differentiation – is the logic that is executed when a task is performed. Where it performs – on DSP, FPGA or x86 processor – is not as important as it once was. Depending upon the application, that logic can be captured in DSP or FPGA hardware – ladder logic in a PLC, for example – or captured as a software algorithm running in a C/C++ program on an x86 chip.

The platform and architecture for how the logic gets implemented is dictated by several important requirements including form factor, performance, and user experience.

For many real-time systems, the performance requirements historically have been so tightly bounded that FPGAs and DSPs were the only viable choice for implementing the logic. Therefore, the architecture was preordained. Even if programmers could be 10 times more productive by using an integrated development environment such as Microsoft Visual Studio that targeted an x86 processor, unless the resulting product could satisfy the performance requirements, it didn't make business sense to use the more productive environment.

For the last decade, x86 based hard real-time systems grew at a steady rate as companies committed to software-based hard real-time. Siemens, for example, did so with their Simatic WinAC RTX PLC. But FPGAs and DSPs have continued to rule the market when it came to hard-real time for motion control or other complex, high-precision and high-performance systems. That is no longer the case. Advancing technologies make it possible for OEMs to deploy a breakthrough soft control architecture that changes the game.

By adopting a Soft-Control Architecture, OEMs can differentiate their products and improve competitiveness by significantly increasing yields/throughputs, and by shortening time to market.

Changing the Basis of Competition for Embedded Systems

As mentioned above, there are five distinct trends in the embedded industry that are driving OEMs toward a soft control architecture. Although it is true that capitalizing on just one of these trends can give a machine designer a competitive edge, when taken together, they create clear separation from all competitors.

Once the trends are understood, it is easier to see how a soft control architecture is possible, and why it is so effective.

Advancing x86 Chip Technologies

For a long time, performance requirements trumped all other requirements for embedded real-time systems. Meeting yield and quality requirements by satisfying very tight performance guidelines was the top priority. It didn't matter how the human-machine interface was if the tool couldn't perform as designed. Often the performance and precision requirements were so tightly bounded that only application-specific integrated circuits (ASIC), DSP or FPGA chips could be considered for the hard-real time processing requirements.

Additionally, because the real-time component was isolated in order to perform the mission-critical logic on DSPs, FPGAs and an RTOS, if there was a need for a complex user interface, the machine designer would have to add an operator workstation that relied on a general purpose operating system (GPOS). In other words, the architecture demanded two computing platforms in a two-tier client (GPOS) /server (RTOS) configuration.

With advances in x86 multicore, multiprocessor computers, and 64-bit processors, that de facto two-tier architecture is no longer the most effective.

In fact, a real-time subsystem that can distribute threads across multiple cores or processors in a SMP implementation easily out-scales a DSP or FPGA solution.

Freed from architectures that isolate the real-time subsystem, machine designers can compete in new ways. They can innovate with new architectures that offer a better user experience in a unified development environment, and that also enable them to reduce product costs and increase operational efficiencies.

Because x86 processor power has increased so much, it is possible for even a single core to outperform FPGA and DSP-based applications. This has allowed machine designers to move their FPGA and DSP functionality off custom boards (e.g. motion boards) and onto one of the cores in a multicore system. We will detail the benefits later in this paper, but it is important to note that yields and quality go up dramatically and costs are reduced as well.

To add even more power to their systems, many OEMs are moving to quad-core, where multiple cores can run hard real-time processes in parallel. This positions the x86 favorably against FPGA and DSP-based applications for even the most high-end and most demanding hard real-time deployments.

Of course the relentless push to double the performance of the x86 doesn't stop with multicore. For example, 64-bit offers more flexibility and supports functionality demanded by other trends that are impacting the ideal architecture for hard real-time systems.

In summary, performance requirements alone no longer dictate the embedded system architecture. X86 processors with multicore, multiprocessing, and 64-bit enable breakthrough architectures that can outperform and out-scale a traditional embedded environment relying on DSPs, FPGAs, microcontrollers and RTOSs.

Commitment to Commercial off the Shelf (COTS)

A great deal of ink and bits has been used to describe how open standards can drive costs out of systems and increase quality. No need to belabor the point here. Suffice it to say that as the performance of the x86 increases, the move to COTS will only accelerate. All the components that previously required DSPs or FPGAs to perform the hard real-time tasks can be converted to software components and run as a process on one of the x86 cores in a multicore environment. This is COTS at the extreme and it is coming.

Siemens' soft PLCs represent a good example of how COTS can drive rapid change. About five to seven years ago, Siemens started offering industrial PLCs that ran on PCs instead of relying on proprietary hardware. Siemens innovated the industry and continues to see success. CNC manufacturers quickly embraced the soft PLC for their deployments, but when it came to their own motion logic, they still relied on motion boards built with DSPs or FPGAs because the performance of the x86 was not yet comparable.

However, as the performance and precision of the x86 improved to match the requirements in the CNC industry, thought-leading machine designers began moving toward soft motion where their motion logic runs on an x86.

In fact, it is now possible to have the soft PLCs run on one core and soft motion run on another core. The whole CNC machine can now be driven on an industrial PC without a custom board.

In the past, machine designers have complained that it can be difficult to optimize on a PC because the chipsets change frequently. Although this is true for consumer PCs, companies offer industrial PCs that circumvent the "end-of-life" issues with x86 chip sets by guaranteeing up to 10 years of availability.

Advances and Availability of COTS-Based Field Buses

The term "embedded system" gives the impression that the deployed system is stand-alone. Not so. In today's world everything is interconnected, including embedded systems.

Components that previously required DSPs or FPGAs to perform the hard real-time tasks can be converted to software components and run as a process on one of the x86 cores in a multicore environment. This is COTS at the extreme and it is coming.

Whether real-time Ethernet, USB, or IEEE1394, many complex systems demand real-time communications. The rapidly evolving standards for intercommunication are another example of the COTS trend. Increasingly, real-time standards are seeking to use the hardware available on an off-the-shelf PC such as the USB ports or NIC card or 1394 port.

Convergence of Components in System Design

Today's end customers want whole or pre-integrated solutions rather than acquiring components for assembly themselves. This trend is forcing OEMs to reconsider the scope of their product offerings.

To better meet customers' needs, responsive OEMs are looking at ways to become more vertically or horizontally integrated.

Good examples of vertical integration are the CNC and machine tool designers that are developing their own motion logic in-house rather than buying component parts, such as motion boards and soft PLCs, to be included in a machine design.

An example of horizontal integration would be the motion board vendors that are expanding their offerings to include PLCs. OEM customers want the PLC and motion logic pre-integrated so most motion vendors are now adding soft PLC.

Either way, the end user benefits because more parts are pre-integrated, which shortens the time to market, improves quality, and reduces maintenance.

The trend is gaining momentum because the scalability and performance of the multicore processors allows a deeper level of integration than could be achieved previously. Integration wasn't as possible before because motion logic demanded a dedicated processor/board and the same was true for the PLC. The architecture called for different components that all stood alone. Now that there is a scalable hard real-time SMP alternative, developers can move motion control logic and PLC into new architectures.

In fact, many stand-alone systems can become one highly integrated system, which further improves quality, speeds time to market and lowers cost.

Advances and Availability of COTS-Based Field Buses

Don't underestimate the power of the user experience.

Sizzle sells.

As we've seen, real-time systems have historically demanded an architecture that separated the user experience from the hard real-time subsystems. Focusing on achieving breakthroughs in performance rather than on user experience made perfect sense. A terrific user experience was useless if the machine couldn't deliver the required bounded latency and precision.

Now that motion logic can run in different architectures and tighter integration is possible, equipment and machine designers see an improved user experience as one means of competitive separation. This is precisely why Microsoft Windows quickly became the strategic platform for simple embedded systems. It is also why Windows is rapidly becoming the strategic platform for complex equipment and machine designs with hard real-time requirements and complex human-machine interfaces.

Many stand-alone systems can become one highly integrated system, which further improves quality, speeds time to market and lowers cost.

The Windows platform leads in market share and has the most development resources behind it. It is the de facto standard for the user experience and is the most common form factor in the world.

It's hardly surprising that as companies come to market with new multimedia experiences, boards, or technology, the first operating system they target is Windows because it represents the greatest revenue opportunity.

It is impossible for a proprietary RTOS, or even an open source real-time solution, to keep pace with the Windows user experience. And as we've seen, in a traditional two-tier hard real-time architecture, the RTOS didn't have to keep pace with the Microsoft user experience. But this two-tier architecture doubles many costs: two chips sets, two tool chains, two code bases, two development groups, two maintenance efforts and so on. By integrating the systems more tightly, costs decrease significantly.

A two-tier architecture can double many costs: two chips sets, two tool chains, two code bases, two development groups and two maintenance efforts. By integrating the systems more tightly, costs decrease significantly.

The pace of change in the user experience is accelerating, which plays to Windows' and Microsoft's tremendous strengths. In fact, two new technologies that Microsoft is innovating will dramatically change the way everyday users of embedded systems engage with PC-based solutions. Surface technologies, for example, which support touch-centered input, are starting to be reflected in current releases of Silver Light and with Windows. Using their fingers to zoom and pan dramatically changes the way users engage with a system. Imagine an ultrasound medical system that allows the technician to zoom around by pointing at the actual graphic rather than relying on a trackball or joystick. This technology – and its differentiating capabilities – will become mainstream with the release, and rapid adoption, of Windows.

Microsoft is also working on motion sensing. In grimy industrial environments where it might not make sense to have a touch screen, a motion-detection system could let an operator communicate a starting position for a CNC machine. Through motions or hand movements, the operator could walk through the set up faster.

This latter technology is more futuristic, but the point remains the same. Microsoft is absolutely committed to remaining the standard for the user experience. Now that the user experience is better able to be a differentiating factor for complex embedded systems, Microsoft Windows is the right choice. No other company is better able to maintain the pace of innovation.

Key Characteristics of a Breakthrough Soft Control Architecture

So, the technologies are shifting the competitive landscape, but before they can be evaluated for how they fit in the new world, we must first examine the ideal characteristics of a new architecture. What are the best practices that define a soft control architecture or a breakthrough architecture that capitalizes on the advances and opportunities cited above?

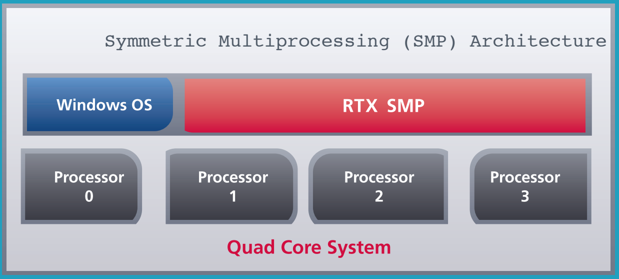
As we've seen, the playing field was level when all parties were forced to separate the hard real-time subsystem from the complex user interface. All OEMs had two sets of hardware, two tool chains, two source code models, two engineering teams and so on. Additionally, the real-time team was often hardware oriented, with a design process far different from the user interface team. This required careful coordination and made communications more difficult. Still, as all OEMs had the same challenges and costs, gaining a competitive advantage was difficult.

With the increased performance of the x86 multicore unified system, it is possible to control both the hard real-time and complex HMI on a single integrated system. The breakthrough opportunity is to have a single integrated development environment handle both the complex user interface and the hard real-time subsystem. This approach simplifies and streamlines the development process because there is only one development environment and one target environment. The engineering teams speak the same language so communications and coordination are easier. This directly translates to improved quality and time to market.

Wanting to take advantage of the increase performance of the x86, the industry has seen two basic processing models emerge that are contenders for consideration to be the underpinning of a soft control architecture – SMP and asymmetrical multiprocessing (AMP) or virtualization.

For a variety of reasons, an integrated development environment can deliver the best of two worlds when SMP is enabled. (Figure 1) Windows is known for its incredibly powerful, world-class user interface, but not for meeting the hard real-time demands. The role of the RTOS should be real-time deterministic performance, not a highly graphical interface for the user.

Figure 1

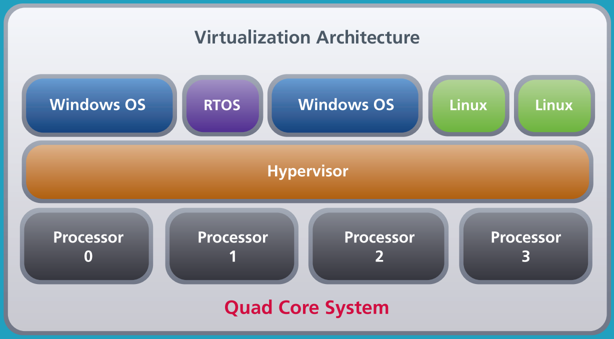


In the end, to achieve the scalability that allows an x86 multiprocessor environment to perform better than a DSP or FPGA, a true SMP environment is the most viable model because of it offers the most scalable and maintainable deployment environment.

A noteworthy characteristic of a true deterministic SMP implementation for Windows is a hard real-time extension, which serves as the RTOS in this configuration. The real-time extension adds a real-time scheduler and some other functionality to allow threads that required determinism to run in real-time container outside the constraints of Windows. In this configuration, only a single instance of Windows and a single instance of the real-time extension are required, regardless of the number of processors being used. This means system resources, like memory, are not overburdened, and it also means that the OEM does not have to maintain multiple instances of the same software. System objects and resources, such as IPC objects and shared memory, are maintained by the single instance of the subsystem. All threads on any processor have the same direct, equal access to these resources.

This approach is in direct contrast to AMP or virtualization architectures (Figure 2) that introduce great complexity and customization and cannot provide the same scalability that a true SMP model offers. There are two AMP/virtualization models.

Figure 2



In the heterogeneous virtualization model, the hardware is divided. The operating system that drives the user interface runs on one or multiple cores and the RTOS runs on separate cores. The OEM is no better off than in the traditional architecture – two tool chains, two development teams…

The first virtualization model is a heterogeneous implementation where the architecture supports two different operating systems on a single processor. The other virtualization implementation is a homogeneous implementation where the architecture replicates the same Windows OS with real-time extension in multiple virtual machines.

In the heterogeneous virtualization model, the hardware is divided. The operating system that drives the user interface runs on one or multiple cores and the RTOS runs on separate cores. The hardware is divided with a hypervisor, and even though the two systems happen to be running on the system, the user interface is separated from the real-time subsystem in much the same way it was in the inefficient, costly two-tier architecture described previously. The OEM is no better off than in the traditional architecture – back to two tool chains, two development teams, and so on.

The homogeneous virtualization approach is to have Windows and a Windows hard-real time extension replicated on as many cores as needed. A quad-core solution will require a hypervisor and some combination of four copies of Windows or copies of the hard real-time subsystem. If these instances need to communicate, or need to share resources, then programmers must develop the equivalent of interprocess communication and remote procedure calls must be created in order for the systems to be synchronized. This approach requires heavy customization on the OEMs' part because the coordination and versioning that is necessary is fraught with quality and maintenance challenges. The overhead of the interprocess communications and the hypervisor add too much latency and complexity to come close to being competitive with the SMP approach.

There is a final virtualization architecture that has yet to be released to the market and it represents the most viable virtualization platform for hard real-time embedded systems that take advantage of SMP.

This future platform is a blend of the heterogeneous and homogeneous implementations and presents an opportunity for Microsoft to win the embedded hard real-time virtualization market. This mixed virtualization environment would allow the Windows user interfaces to run in a HyperV virtualization environment and allow a Windows hard real-time extension subsystem to run in SMP mode on multiple cores that are protected from HyperV. This means the real-time extension could directly see and control multiple cores and the resources in an SMP-enabled mode and then communicate via direct memory.

This solution is contrasted to a virtual machine Windows hard real-time solution because of the fact that the Windows environment and the real times subsystem can run on different cores and the real time subsystem can run in SMP mode. Both are required to meet the scalability and performance requirements.

There are other advantages of SMP over a non-HyperV Windows virtualization model.

SMP treats the multi-processor hardware as a shared resource with a single real-time subsystem running across all configured processors. Virtualization's goal is isolation, which is completely opposite of integration. For example, having access to all resources directly enables scalability of the system. Taking advantage of a multicore chip means the OEM must be able to assign processes and threads to multiple cores and be able to set priority levels of threads within each core. Only an SMP-capable kernel can schedule threads directly to cores.

AMP/virtualization systems have an OS/scheduler per virtualized OS, so communication and synchronization between threads becomes too complex too quickly to allocate threads to any core other than the core the process it is running on. This alone limits the value of a virtualization environment because the lack of direct interprocess communication directly limits its ability to scale.

And it's not only resource access that should be considered, but also data. SMP cores must have immediate and direct access to shared data. AMP relies heavily on programmer-developed mechanisms to replicate the equivalent of interprocess communication and memory copying to allow access to shared data regions. This leads to data corruption and synchronization issues with parallel code. Again, this is a lot more complexity that the OEM must take responsibility for rather than letting the system do the work, as is the case with SMP.

SMP is superior to virtualization because it relies on a smaller memory footprint for the actual real-time subsystem. A small-memory footprint improves the overall system performance and scalability. Ideally, on four-core systems a real-time extension should only require about 250k of memory to run the actual subsystem only – not user applications. In a virtualization environment the real-time extension will require memory for each instance on each core, rather than SMP's one instance across all needed cores.

Finally, while it does require some thought, scalability of an SMP deployment is parameterizable if the code is designed from the start to do so. In other words, code can be written once and the performance will scale automatically with an increased number of processors – without code changes or even rebuilding.

In summary, the best practices that an SMP environment offers are:

- One common integrated development environment and world-class GUI – e.g. Microsoft Windows
- Single real-time subsystem instance executing directly on multiple assigned processors
- All resources visible to all real-time processes
- Ability to schedule real-time threads across multiple processors, or dedicate certain logic to specific cores
- Direct access to shared data without additional copies and heavy IPC usage
- Minimizable system memory requirements – footprint/power usage
- Ability to code once with parameters for parallelism

While virtualization is conceptually pleasing because it simplifies each isolated instance, interprocess communication is necessary for all-important scalability.

Optimizing Best Practices in a Soft Control Architecture

A soft control architecture capitalizes on all the recent technology advances – particularly x86 multicore processors – and also blends the best of a general purpose operating system and a real-time operating system to deliver a breakthrough in systems design that is changing the basis of competition across markets that typically have relied heavily on DSPs and FPGAs. Impacted industries include the Industrial Automation, Medical Systems, Test and Measurement and Digital Media markets, to name a few.

As stated at the outset, the key to a soft control architecture's value is moving the hard real-time control logic, such as PLC or motion logic, from specialized hardware components to software components. With x86 processor advances, OEMs can take the C/C++ source-code logic traditionally compiled and run on the DSP or FPGA and port the source code to target an RTOS-like real-time extension to Microsoft Windows. The extension runs as a hard real-time, SMP-enabled application directly on x86, therefore eliminating any need for the FPGA or DSP to perform the logic. The software component runs on multicore, commodity x86 processors and uses both open standards and standard communications architectures, such as USB and real-time Ethernet. As an example, a contrasting look at an SMP-enabled CNC machine design versus a traditional hardware-centered CNC machine design can be seen in Figures 3 and 4.

In the heterogeneous virtualization model, the hardware is divided. The operating system that drives the user interface runs on one or multiple cores and the RTOS runs on separate cores. The OEM is no better off than in the traditional architecture – two tool chains, two development teams…

Figure 3

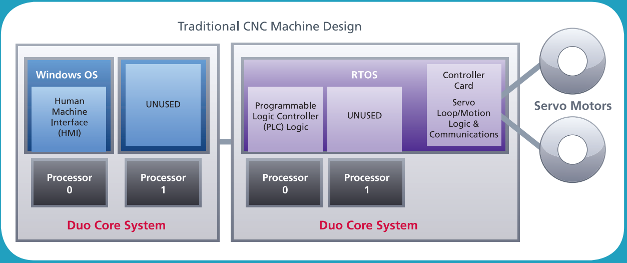
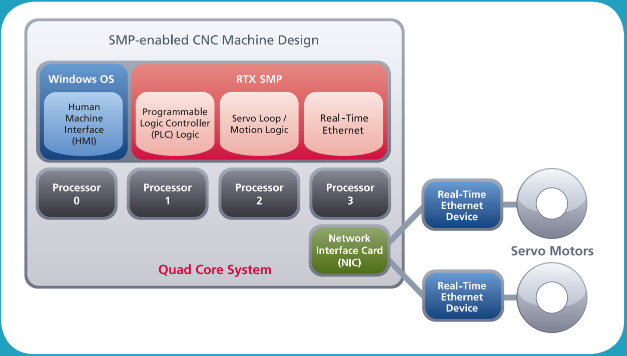

Traditional CNC Machine Design
Windows OS
Human Machine Interface (HMI)
UNUSED
Processor 0
Processor 1
Duo Core System
RTOS
Programmable Logic Controller (PLC) Logic
UNUSED
Controller Card
Servo Loop/Motion Logic & Communications
Processor 0
Processor 1
Duo Core System
Servo Motors

Figure 4


SMP-enabled CNC Machine Design
Windows OS
Human Machine Interface (HMI)
RTX SMP
Programmable Logic Controller (PLC) Logic
Servo Loop / Motion Logic
Real-Time Ethernet
Processor 0
Processor 1
Processor 2
Processor 3
Network Interface Card (NIC)
Quad Core System
Real-Time Ethernet Device
Real-Time Ethernet Device
Servo Motors

To deliver this value, a soft control architecture blends three components: Windows, x86 multicore chips and an RTOS-like, SMP-enabled hard real time extension.

Windows delivers the optimal user experience and is the foundation of this breakthrough architecture. Clearly, no RTOS can compete with its usability. It is far easier to add real-time capability to Windows than it is to get an RTOS vendor to keep pace with the user interface. To achieve a single integrated development environment, Windows must be extended with a real-time scheduler and other RTOS-like features. It is then simply a matter of selecting the best hard real-time extension for Windows.

The requirements for this Windows hard real-time extension are stringent. SMP-enabled hard real-time is a prerequisite for:

- Sufficient scalability
- The levels of performance required to support the migration of logic from DSPs and FPGAs into software that runs on an x86
- Optimizing the utilization of the x86 multicore processors.

While it is difficult to provide benchmark data directly because PCs vary, it should be noted that the minimum sleep time for Windows is 1 millisecond +/- 7.5 milliseconds. IntervalZero RTX64 offers a 1 microsecond timer, but because little can be performed in the interval, customers often use a 100 microsecond timer and experience jitter of only 2 microseconds. Other customers have opted to work with a 20 microsecond timer, which is required to sufficiently replace DSPs and FPGAs in some industries. Now, consider running multiple threads in parallel across multiple cores. The point here is that RTX is capable of delivering equivalent determinism of RTOSs or competing extensions, and can outperform DSPs and FPGAs. At the same time, x86 multicore chips are capable of supporting SMP and enabling the scalability and performance requirements that OEMs demand.

The resulting benefits are real. A soft control architecture gives OEMs clear competitive advantages and product differentiation by delivering breakthroughs in throughput and yields; in production quality; in a more compact physical footprint; and in substantial cost reductions. In addition to reducing product costs, a soft control architecture can improve operational efficiencies for an OEM. By transforming hardware components to software components, there is nothing to inventory and parts are infinitely replicable.

IntervalZero RTX64 Enables Scalable, Hard Real-Time SMP

IntervalZero's RTX64 is an essential, enabling component in this powerful new soft control architecture. The economics are too compelling to ignore, and companies that move first will have a significant advantage.

RTX is the only solution in the world that integrates seamlessly into the Microsoft Visual Studio integrated development environment; that deploys to a single integrated Windows system; that extends Windows, delivering hard real-time precision with bounded latency; and that does so on multi-core processors as a natively SMP-enabled solution.