

Cost-reduce your embedded system using multicore processors

By Paul Fischer

Embedded systems developers can reduce multiple embedded systems into a single hardware platform by allocating CPU cores in a multicore processor to dedicated real-time tasks.

By using operating system partitioning on a multicore processor platform, embedded system developers can shrink multiple hardware platforms into a single hardware platform, with a concomitant reduction in cost, size, and hardware complexity. Embedded systems that incorporate a mix of dedicated subsystems, performing functions such as real-time control, data acquisition, and a Human Machine Interface (HMI), can be built to run on a single computing platform where each operating environment has a dedicated processor core. The key to supporting a mix of real-time and general-purpose operating systems on individual cores within a multicore processor is to use a real-time operating environment that supports virtualization.

Case study: Precision motion operations

For a case in point, consider a high-speed electronic assembly machine that performs precision motion operations (Figure 1). The system includes a DSP subsystem for calculating coordinated multi-axis motion profiles in real time that drive a positioning platform, while another subsystem performs real-time data acquisition from a vision subsystem. A third subsystem serves as an HMI to monitor and control overall machine operation and to set up and initiate assembly tasks from a menu of options.

The motion subsystem needs to complete a complex sequence of calculations in a predetermined amount of time. In one such high positioning system, the cycle time between position updates is 100 microseconds (10 kHz). Within each cycle, the DSP performs a set of floating point calculations, position set points, that represent the coordinated profiles of several axes of motion. In addition to calculating motion set points, the DSP also monitors and controls a collection of digital and analog I/O points connected to system safety switches

and status and control points within the equipment.

In the vision subsystem, a real-time processor captures and analyzes images from cameras to determine placement and orientation of the components to be placed by the motion system. The decisions made by the real-time processor guide the motion system, in the form of multi-axis

positioning targets. Again, a complex set of operations must be completed within a time-limited window (typically within a few milliseconds) in order to ensure that the motion system is able to hit its targets precisely, repeatedly, and on time.

Many problems plague machine builders that require complex real-time processing systems (Table 1). First is the cost penalty

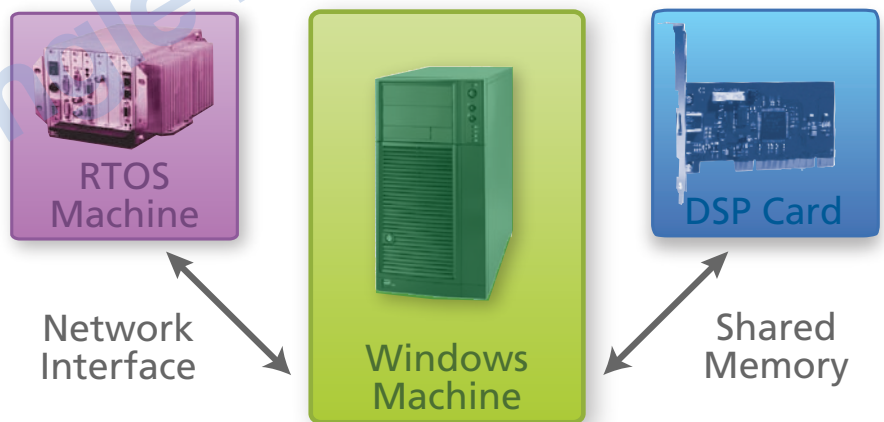


Figure 1

Costs associated with multiplatform systems

Issues	Solutions
Cost of specialized processing subsystems (for example, DSP cards)	Dedicate a CPU core and utilize libraries that exploit Single Instruction, Multiple Data (SIMD) instructions such as SSE on x86 processors
Hardware duplication (for example, memory, power supplies, enclosures) increases costs	Merge multiple hardware platforms into one single hardware platform
Increased complexity and decreased system reliability	A single hardware platform simplifies the system, increasing reliability
Redesign costs and delays due to obsolescence of hardware platforms	Building on operating systems designed for forward-moving platforms ensures the ability to move forward
Development and maintenance expenses associated with specialized subsystems	Availability of engineers for standard software/hardware platforms is much greater, resulting in less development and maintenance risk

Table 1

“The key to supporting a diverse mix of real-time and general-purpose operating systems is to use virtualization.”

due to using specialized processing subsystems like DSP cards. Second is the duplication of memory components, power supplies, circuit cards, and enclosures within the machine. Third is an increase in complexity of assembly with a decrease in system reliability, due to more parts and interconnects between subsystems. Fourth are the cost and availability issues that arise due to the eventual redesign resulting from the obsolescence of nonmainstream processors by their manufacturers. Fifth are the development and maintenance expenses associated with writing and maintaining specialized DSP code that is too frequently debugged using expensive and less than state-of-the-art development systems.

The proliferation of multicore implementations on mainstream processors, such as Intel Architecture (IA) CPUs, provides a cost-effective solution to these problems.

By adapting DSP and real-time functions to run on dedicated cores of a multicore IA CPU, where each core is running its own RTOS, OEMs can achieve the following results:

- Significant cost reduction in the control hardware
- Upgrades to the control hardware and software more easily adapt to new processing platforms
- Substantially improved development and debugging of control software using up-to-date development tools
- Additional processor cores provide expanded resources for real-time processing of more complex algorithms
- Newer and cheaper standardized I/O subsystems can be constructed, a benefit of PC market economies of

scale (for example, USB, PCI Express, and Ethernet)

The availability of multicore processors on the PC architecture enables system designers to maintain the integrity of the real-time environments by dedicating one core of a multicore processor to an RTOS. With a dedicated core, real-time applications are ensured 100 percent of the CPU instruction cycles; there is no performance penalty for sharing the platform with a General-Purpose OS (GPOS) such as Windows.

Virtualization-supporting RTOS is key

The key to making multi-OS embedded systems work on a multicore CPU is an RTOS that supports virtualization. Virtualization provides the isolation needed between multiple operating environments and also enables legacy real-time systems to be integrated with new functionality and minimal impact on legacy software.

The latest Intel multicore processors include a feature called *Intel Virtualization Technology* or *Intel VT* that enables hardware-enforced isolation of

the processor’s I/O and memory (Table 2). With virtualization, multiple control loops can run simultaneously. A distinct boundary is established between real-time processes and threads and nondeterministic tasks that execute on different processor core(s). VT hardware includes a collection of new processor instructions, traps, and a privileged “root” operating mode that enables Virtual Machine Manager (VMM) software to host multiple virtual machines on a single hardware platform.

Exploitation of VT hardware for embedded systems requires a VMM that understands the needs and requirements of real-time operating systems and applications. An example of such an operating environment is the TenAsys’ INtime RTOS for Windows (Figure 2).

Removing contention for resources in a multi-OS platform has a dramatic impact on real-time performance metrics such as interrupt latency, when compared to multi-OS platforms that must share a single CPU. TenAsys has measured a 10 to 1 improvement in interrupt latency when running the INtime RTOS on dual-

Intel Architecture virtualization attributes

Software-only virtualization	Limited to non time-critical applications such as consolidating servers. Available today for Pentium grade and above.
VT-x: CPU virtualization	Special CPU root mode to trap system-level instructions. Available today in most multicore processors.
VT-d: Chipset virtualization	Hardware-supported DMA and interrupt remapping. Available in upcoming chipsets.
PCI-SIG: I/O device virtualization	Address Translation Services (ATS) to accommodate sharing PCI Express I/O cards between multiple operating systems. ATS 1.0 spec available now.

Table 2

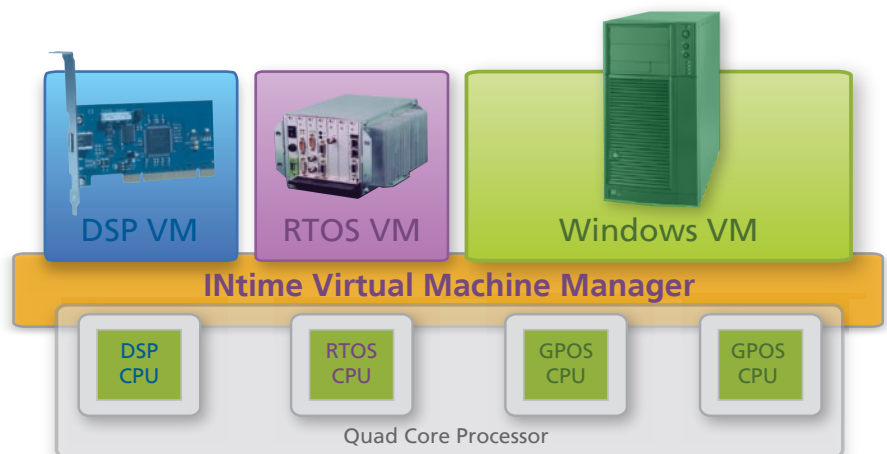


Figure 2

core multi-OS platforms compared to equivalent clock speed single-core platforms. Interrupt latencies measured in the 10-30 microseconds range on single-core systems are reduced by an order of magnitude to 1-3 microseconds on equivalent speed dual-core systems. With such low latencies, real-time control loops can execute in the 50-200 microsecond range with very high precision, while simultaneously supporting a general-purpose OS such as Windows on the same hardware platform.

Virtual devices interface with inter-OS protocols

To facilitate existing application reuse, virtual devices can be used as the interface for inter-OS protocols. For example, an inter-OS protocol can be implemented entirely within a virtual PCI hardware interface. The guest operating systems are configured to share an area of shared memory in which common data is posted (Figure 3). After a guest updates its data structure in the shared memory region, it signals the other guests of the update via a register in the virtual PCI interface.

In this example, each virtual PCI device presents two memory ranges to each guest. The first memory range, pointed to by PCI configuration register BAR0, maps the shared memory buffer. The second range, pointed to by BAR1, presents an I/O address to each guest OS. When an application within the guest OS accesses the BAR1 I/O address, a trap is made into the virtual device driver hosted by the VMM. The virtual device driver then injects a virtual IRQ into the target guest OS, which responds by accessing the shared memory area for updated data.

RTOSs and virtualization: Smoothing the way

The net gains from the application of real-time virtual machine technology on multicore processor platforms are the elimination of redundant computer and communication hardware, faster communication and coordination between RTOS and GPOS subsystems, improved reliability and robustness, reuse of proven legacy applications, and simplified development and debugging. **CS**



Paul Fischer is a senior technical marketing engineer at TenAsys Corporation. He has more than 25 years of

experience building and writing about real-time and embedded systems in a variety of engineering and marketing roles. Paul has an MSE from UC Berkeley and a BSME from the University of Minnesota.

TenAsys Corporation

1400 NW Compton Drive, #301
Beaverton, OR 97006

503-748-4720

info@tenasys.com

www.tenasys.com

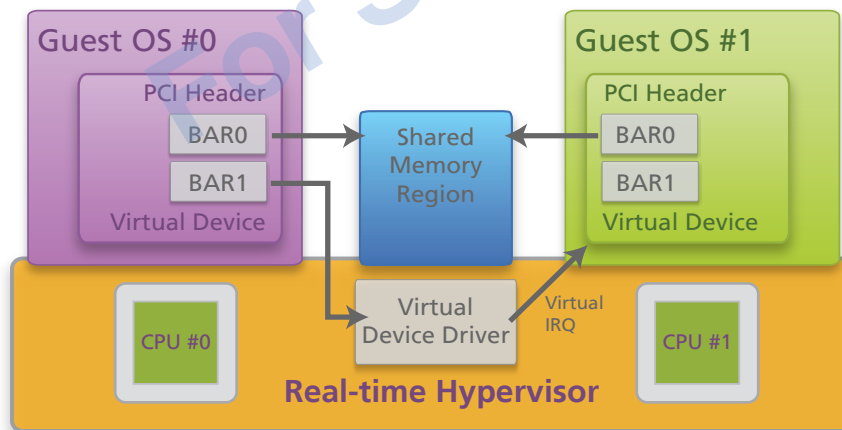


Figure 3