TABLE OF CONTENTS

Embedded Virtualization .......................................................... 2
Multi-core Software Configurations ............................................. 3
Applications for Embedded Virtualization ..................................... 5
  Cost Reduction and Increased Capacity Through OS Consolidation .......... 5
  Creating Separation and Safety Critical Applications ....................... 9
Features ................................................................................. 10
  Virtual Board ..................................................................... 10
  Guest Operating System .......................................................... 11
  Flexible Configuration ............................................................. 13
  Device Model ..................................................................... 13
  Inter-guest Communication ....................................................... 15
Architectural Design ................................................................. 16
  Deterministic ..................................................................... 17
  Event-Driven ..................................................................... 17
  Minimal Footprint ................................................................. 17
  Scalable ............................................................................. 17
  Flexible Device Model ............................................................. 17
  OS Agnostic ..................................................................... 18
Development Tools ................................................................. 18
  Graphical Configuration ......................................................... 18
  Automatic Configuration (Intel Platforms) .................................... 19
  Building and Debugging ........................................................... 20
  Flexible Terminal Console Options ......................................... 20
  JTAG On-Chip Debugging ....................................................... 22
Integration ............................................................................. 22
Professional Services ................................................................ 22
  Installation and Orientation Services ........................................... 23
Embedded Development Kits ................................................... 23
Education Services ................................................................ 24
  Public Training ................................................................... 24
  Private Training ................................................................... 25
Customer Support ................................................................ 25
  North America, South America, Asia/Pacific .................................... 26
  Japan .................................................................................. 26
  Europe, Middle East, Africa ..................................................... 26
Wind River® Hypervisor, an essential building block in the Wind River portfolio of embedded software development solutions, is an embedded hypervisor providing a Type 1 embedded virtualization layer that partitions software running on single or multi-core processors into multiple virtual boards with varying levels of protection and capabilities. It is optimized to host general purpose operating systems, operating systems with real-time requirements, and operating systems used in safe and secure systems simultaneously.

Embedded virtualization is one of the key technologies that enable teams to build competitive next-generation embedded devices using multi-core processors, and Wind River Hypervisor was designed to address the future of such devices. A low latency, event driven, multi-core distributed hypervisor with a minimal footprint, it provides direct access to devices, and is processor, architecture, and OS agnostic. Optimized for hardware virtualization, it can also operate on processor families running paravirtualized guests in user space.

**EMBEDDED VIRTUALIZATION**

The embedded systems market faces a set of challenges for next-generation embedded devices:

- How can device performance be increased and the bill of materials decreased?
- How can existing features across various devices be migrated, consolidated, and colocated with new innovative, higher value, and disruptive technologies?
- How can noncertified applications coexist with applications with differing levels of certification and performance?

A new class of virtualization technology has emerged to address the design of next-generation embedded systems: embedded virtualization. Embedded virtualization allows applications, guest operating systems or bare metal applications, devices, and resources of an embedded device to be integrated and tuned, holistically, as a system.

For example, a human-machine interface (HMI) for a device may require high performance I/O for graphics, but data acquisition and control software may require synchronous real-time processing. With embedded virtualization, each of these workloads can run in parallel on a single device in its own guest operating system. A general purpose operating system such as Linux or Windows could be chosen for the HMI because of available third-party applications, and a real-time operating system such as VxWorks® could be chosen for the data acquisition and control software because of its real-time performance and determinism. Device, memory, and core utilization is configured and tuned during development so that the embedded device is optimized for performance.

Embedded virtualization enables device manufactures to increase the performance of devices and lower the bill of materials by migrating and consolidating the software from multiple devices onto a single device; software running on the CPU of multiple blades, for example, can execute as multiple virtual boards on a single-core or multi-core processor existing on a single blade. Similarly, devices with multiple CPUs can use a single multi-core processor. This technique facilitates product scalability and enables reuse of software and capabilities that can be positioned from entry-level, low margin, high volume offerings to high cost, high margin offerings, using the same software.
These next-generation embedded devices can leverage the software from new, innovative, and high-value technologies coupled with the required “table stake” legacy software assets (with potentially thousands of person-years invested) by collocating virtual boards onto a single microprocessor. These software elements can be a mixture of general purpose operating systems and real-time operating systems; a heterogeneous configuration of operating systems can be engineered without sacrificing the deterministic response time required by real-time operating systems and their applications. If required, instances of these operating systems can span multiple cores in symmetric multiprocessor (SMP) mode. These devices can leverage virtualization to allow separation of safety critical applications and to reduce certification costs.

MULTI-CORE SOFTWARE CONFIGURATIONS

Until recently, configuring embedded systems was relatively simple: The processor had a single core that hosted a single operating system. Depending on the product requirements, a general purpose or real-time operating system was chosen. If more than one operating system was needed, the design would have to accommodate more than one processor.

Today’s powerful single and multi-core processors can be used in many different configurations. A multi-core processor can be managed by a single SMP operating system that manages all the cores. Alternatively, each core can be given to a separate operating system in an asymmetric multiprocessing configuration (AMP) configuration. Both SMP and AMP have their challenges and advantages: SMP does not always scale well, depending on workload; and AMP can be difficult to configure in regard to which operating system gets access to which shared system device.

The use of virtualization in the form of an embedded hypervisor enables a software architect to optimize the embedded device software using a wide variety of configurations, including mixes of AMP, SMP, and core virtualization. The hypervisor manages the hardware and creates partitions in which operating systems execute. Each partition is given access to resources (processing cores, memory, devices) as specified by the design of the embedded system. Each partition can hold an operating system (also known as the guest operating system) and is protected from the other partitions. The hypervisor can execute a single partition on a single core, a single partition across multiple cores, or multiple partitions on a single core.
Figure 1: Primary multi-core configurations

Figure 2: Consolidation of multiple single cores to a single multi-core

Figure 3: Scalability to high core counts
APPLICATIONS FOR EMBEDDED VIRTUALIZATION

Embedded virtualization provides new capabilities to developers and equipment manufacturers. The following are some use cases that are enabled when employing Wind River Hypervisor.

Cost Reduction and Increased Capacity Through OS Consolidation

Many current systems use multiple processors, either on the same compute board or on multiple blades in a rack. The rationale for using multiple processors includes performance demands and the need for separation between different types of functionality (real-time, general purpose, safety, security, etc.).

Embedded virtualization can be used to maintain necessary separation, whether on a single or multi-core processor, allowing the previously separate and disparate functions to be consolidated onto a single compute platform. Multi-core processors offer increased compute performance with lowered power consumption by enabling this consolidation while maintaining—or increasing—the compute performance available for the individual functions.

The benefits of such consolidation include reducing the bill of materials of the device or system, increasing the performance and capacity of the device or system, and reducing the amount of power used. Each of the operating systems executing on a separate compute platform on the original system can be migrated onto a separate partition on the multi-core processor with embedded virtualization. The embedded hypervisor enforces the partitioning that provides separation and fault containment. Figure 4 shows an example of a migration from a multiprocessor system to a multi-core processor on a single board.

![Diagram](image)

Figure 4: Reduce component costs by consolidating multiple single-processor boards onto a single multi-core CPU

Use Case: Industrial Control Systems

Multi-core processors offer industrial equipment vendors the opportunity to increase product performance and functionality. When migrating existing designs to multi-core CPUs, often there remain available CPU cores on which new functionality can be added.
Such new functionality could be the addition of a graphics-rich interface for an industrial controller, for example, or the addition of a standards-based protocol stack. Each of these new functions can be added to the existing product while maintaining the necessary spatial and temporal separation to ensure that the real-time application is not impacted by the non-real-time applications. This is shown in Figure 5.

Adding new applications and code to the product’s software image and hosting that new content on separate CPU cores provides the ability to add general purpose applications without impacting the existing application. This can greatly reduce retesting efforts when adopting new hardware platforms.

**Migrating Existing Software**

Traditionally, embedded systems were designed as “fixed purpose” devices, often working with other devices to provide some service. Many of these devices have required thousands of person-years to develop, tens of millions of dollars invested in software development, and deployment life cycles tied to service contracts that can span decades. Often these devices provide services that directly generate revenue.

In order to stay competitive, the market demands increased functionality and performance and lower operating costs without affecting the current capabilities of a product. Often, rewriting or porting the needed legacy software functionality for new operating environments is cost prohibitive and high risk (in terms of budget, time, and regression testing) or next to impossible—the software may be complex and contain many undocumented modifications that have been added throughout the life cycle of the product, directly affecting the operations and timing inherent to the correct functionality of the software. Fortunately, modern multiprocessors and embedded virtualization of the complete operating environments of the legacy software (operating system and applications) have carved a migration path for existing applications to be collocated with new, innovative features.
Embedded virtualization allows vendors to migrate their existing software assets from older single core CPUs to multi-core CPUs to benefit from performance and power improvements. By leveraging embedded virtualization to isolate legacy applications into a single virtualized partition running on a single core, developers can port the existing code base with minimal effort to the new hardware.

Wind River Hypervisor supports unmodified guest operating systems, allowing the existing code to execute in the environment in which it was designed. The underlying hardware is abstracted by the hypervisor. A single-core virtualized partition can be created for existing applications and operating systems that do not support multi-core CPUs. The specific hardware devices required by the existing application can be presented to the partition in such a way that the migrated application detects the devices that it requires and behaves as it does on the older, single core hardware.

By porting unmodified legacy applications in virtualized partitions, device manufacturers can retain their software investment and leverage general purpose operating systems to provide enhanced HMIs, or offer a scaling range of product features, for example. New software with innovative features can be added in other virtualized partitions executing on unused CPU cores alongside the legacy unmodified guest operating system, allowing developers and vendors to add to the functionality of the device with minimal effort. The real-time behavior of Wind River Hypervisor allows the new product to maintain the real-time determinism while providing new capabilities, allowing device manufacturers the opportunity to deliver innovation and differentiation.

Figure 5 depicts a situation where an existing operating system and application can be augmented with a new operating system and application to develop a new, differentiated next-generation product.

**Unmodified Guest Operating System**

Having the ability to migrate an existing application along with its operating system from older, lower performing hardware to newer multi-core CPUs is an attractive proposition for device manufacturers; however, not all operating systems support multiple cores. Migrating an application to a new version of the operating system can be prohibitive. Embedded virtualization can be used to migrate the existing application and operating system to new hardware, provided that the hypervisor supports unmodified guests and the hardware provides the necessary support.

Examples of CPUs that offer this hardware support for privilege levels that allow for unmodified guest operating systems include Intel® CPUs with vPro technology and Freescale CPUs with e500mc CPU cores, such as the P3 and P4 families of CPUs.

Matching Wind River Hypervisor with a CPU that provides hardware support for virtualization allows seamless migration from older CPU technology to newer technology, offering the ability to leverage newer CPU devices with increased core counts, computational power, and lower power consumption.
Modified Guest Operating System

While the ability to migrate an existing application to new CPU hardware without making modifications to the OS kernel is attractive, it is not always possible. Developers may have access to the OS kernel source code, allowing them to recompile the kernel and operating system for a new CPU target. To maintain the privilege levels necessary for the OS kernel to execute under hypervisor control, the system calls within the kernel can be instrumented or augmented with API calls to the hypervisor rather than to the BIOS or hardware directly.

When selecting a CPU, this approach can be an attractive alternative to unmodified guests because the set of CPUs that offers hardware support for unmodified guests is relatively small; the option of instrumenting and recompiling the kernel allows a wider range of CPUs to which migration can be targeted.

Use Case: Consolidation of Hardware and Software in Networking Equipment

Embedded virtualization offers network equipment vendors the opportunity to meet scalable performance requirements with significant savings in hardware costs, size, weight, and power consumption, as well as significant decreases in time-to-market for deployment of the services.

By migrating existing designs to multi-core CPUs and consolidating the separate compute functions into partitions enforced by an embedded hypervisor, the total number of physical boards, or physical CPUs, can be significantly reduced.

Also, disparate compute functions such as control plane management processing can be consolidated on the same multi-core hardware in isolated partitions, executing on dissimilar operating systems to meet the differing needs of the applications. Control and management capabilities may require a general purpose operating system or a real-time operating system, while data processing engines may only require lightweight processing executives or a real-time operating system.

These different operating requirements can be met with the isolation provided by Wind River Hypervisor and the increased compute capacity of multi-core processors. Figure 6 demonstrates how the control and data plane functionality that spans heterogeneous operating systems, multiple hardware modules, and multiple blades can be colocated on the same blade with a high performance multi-core processor with new functionality added.
In this example a functional block of a carrier card running Wind River Linux on a single processor, and two four-module carrier cards, with each module running VxWorks on a single processor, are migrated to a new blade running Wind River Linux SMP on two cores of a quad-core processor and VxWorks on each of two remaining cores. The performance of the multi-core processor allows optimizations such as increased functionality and a four-to-one aggregation of applications running on VxWorks. This configuration quadruples capacity, lowers the bill of materials dramatically, and allows disruptive and innovative applications to extend the current offering.

Creating Separation and Safety Critical Applications

Safety critical systems typically comprise multiple individual components of varying levels of safety certification, with the safety-certified components physically separate from the noncertified components.

Modern safety-certified systems are doing more with less by leveraging separation with virtualization. Wind River Hypervisor, a certifiable and secure real-time embedded hypervisor, provides this level of separation, ensuring safety, certification, and subsystem separation on a single, complete system, and eliminating the need for multiple, and expensive, components.
An embedded virtualization solution that is safety certifiable to levels conforming to IEC 61508 or its derivatives can segregate mission-critical subsystems into certified partitions, while non-mission-critical subsystems can be separated and isolated in noncertified partitions.

The following are some advantages of certifiable embedded virtualization partitioning:

- Reduced certification costs
- Separate safety-related code
- Partial certification of an application
- Addition of new functionality without recertification costs and efforts

These embedded virtualization benefits can be realized while maintaining the guaranteed real-time requirements that are demanded by embedded systems.

**FEATURES**

Virtual Board

Wind River Hypervisor separates a system into multiple virtual boards (partitions). The virtual board can contain a guest operating system with applications or an application without an operating system (virtual board application). The virtual board is managed by the hypervisor. Wind River Hypervisor controls which cores the virtual board executes on and which memory and devices it can access. A virtual board can share a single core with another virtual board, run dedicated on a single core, or span multiple cores.

The virtual board is a fault container. The hypervisor prevents faults that occur within a single virtual board from affecting other virtual boards and the rest of the system. Wind River Hypervisor can be designated as the fault handler, or the fault can be propagated to the virtual board for handling within the guest operating system.
Virtual boards can be individually created, paused, resumed, restarted, and migrated between cores, providing an easy way to recover from critical failures inside a virtual board. This provides the ability to dynamically scale applications to meet changes in demand. For example, as throughput or demand increases, new instances of virtual boards can be created on available CPU cores to meet the demand. Similarly, as demand wanes, instances can be paused or collocated on a single CPU core, allowing a subset of CPU cores to be freed for other processing tasks or powered down completely, resulting in overall power savings.

**Guest Operating System**

Wind River Hypervisor executes guest operating systems, which host applications. A guest operating system executes with near-native performance. Wind River Hypervisor uses a mix of hardware assist and paravirtualization of the guest operating system to deliver optimized performance and determinism.

Wind River Hypervisor supports guest operating systems such as Wind River Linux and VxWorks, and its open interface allows other operating systems and executives to run, including those that are open source, proprietary, or internally developed.

**Unmodified Operating Systems**

Wind River Hypervisor supports the execution of unmodified guest operating systems—operating systems and kernels that do not need to be modified to run on top of a hypervisor. This is in contrast to paravirtualization, discussed in a later section.

Typically an OS kernel running without a hypervisor has the highest level of privilege and complete write access to all hardware. With virtualization and hypervisors, the hypervisor has the highest level of privilege. This level must be higher than even the OS kernel.

To continue to behave as intended, the guest operating system must be able to continue to execute with these necessary changes in privilege levels, without recompiling the OS kernel. This is the intent of support for unmodified guest operating systems.

Latest-generation CPUs offer hardware support that allows the hypervisor to retain the highest level of privilege and the guest operating system to retain complete write access to the hardware devices within its partition that it detects at boot time.

Wind River Hypervisor supports the use of unmodified guests such as Red Hat Enterprise Linux or Microsoft Windows inside a virtual board when running on top of Intel processors. This enables general purpose operating systems such as RHEL or Windows to run alongside applications on Wind River Linux and/or real-time content hosted by VxWorks.

Wind River Hypervisor supports 32- and 64-bit single core and multi-core processors and can host 32- and 64-bit single core and multi-core guest operating systems, including 32-bit guest operating systems on 64-bit processors.
Bare Metal Executives

Wind River Hypervisor can host the execution of a virtual board without an operating system. This is useful for certain tasks that don’t need the capabilities and code footprint of a complete operating system, such as data processing engines or a fast polling loop, but require a subset of services typically provided by operating systems, such as interrupt control and scheduling.

Wind River Hypervisor provides programmable APIs that present these bare metal executives with access to system resources such as the memory management unit (MMU), interrupts and exceptions, CPU cores, and other hardware devices necessary to operate as if the executives were hosted by a complete operating system. These APIs allow the data processing engines to benefit from virtualization while retaining the necessary real-time responsiveness. Completely event driven and lock free, Wind River Hypervisor is only as involved as needed when providing scheduling and hardware access to all operating systems on all CPUs.

Hardware Assist and Paravirtualization

Some processor architectures provide features in hardware that facilitate virtualization (e.g., Intel VT-x, PowerPC e500mc). Wind River Hypervisor can use these features to provide virtualization services. On CPUs where these features do not exist, or if these features impact performance and determinism, the hypervisor implements them in software, and paravirtualization of the operating system is required.

Paravirtualization is the modification of an operating system’s privileged system calls to collaborate with a hypervisor. The hypervisor must execute the privileged instructions on behalf of the guest operating system. This collaboration works through programmatic interfaces executed from within the guest operating system. The amount of modification depends highly on the processor architecture. It typically concerns the following:

- **Privilege levels**: A virtualized system typically requires three privilege levels: the hypervisor; the guest operating system; and the application. Many CPU architectures that do not support hardware-assist have only two privilege levels. The task of paravirtualization is to emulate the missing privilege level. Privileged instructions in the guest operating system need to be replaced by hypervisor APIs.

- **Device access**: A driver in a native operating system is able to access any device. In a virtualized environment, the hypervisor arbitrates whether a virtual board has access to a device. This includes device interrupts, device I/O, device register access, and direct memory access (DMA). Device access includes access to timers, network cards, graphic cards, and so forth. Environments with hardware assist often allow device drivers to operate without modification.

- **MMU**: The hypervisor controls the MMU in a virtualized environment. Many processors with hardware assist features allow a guest to modify the virtual board MMU. If hardware assist is unavailable, the guest may have to collaborate with the hypervisor to modify the MMU.
Flexible Configuration
Wind River Hypervisor provides flexible tooling to configure virtual boards on single core and multi-core processors. The configuration defines how the virtual boards are distributed over the available processing cores; how they are scheduled; and how the physical hardware is partitioned between the virtual boards. There are a number of basic building blocks that the developer can use to assemble a system:

- **One virtual board allocated to a single dedicated core**: This allows the guest operating system and its hypervisor services exclusivity to the core, providing performance comparable to the operating system running natively on a single core processor.

- **Multiple virtual boards allocated to a single core**: The hypervisor provides scheduling services. The hypervisor’s scheduler decides how processor cycles are provided to each virtual board. These decisions are either made on a priority basis or a (time slice) partition basis. The scheduler in Wind River Hypervisor is pluggable; so, if needed, teams can design their own schedulers for integration with the hypervisor.

- **Multiple cores allocated to a single virtual board**: The guest operating system is executing in SMP mode across a number of cores.

System designers can decide how to use these building blocks in their designs. They can, for example, partition an eight-core processor in a virtual board spanning multiple cores and running an SMP guest operating system. They can run several virtual boards, each using a single core; and they can run multiple virtual boards on a single core.

The ability to explicitly configure which hardware devices are presented to each virtual board via the hypervisor, and in which manner the devices are accessed, delivers the benefits of virtualization while maintaining the real-time determinism of a virtualized platform.

Device Model
Wind River Hypervisor offers a unique device driver model that allows developers to assign devices to virtual boards in one of several access configurations. Device access can be direct, shared, virtualized, or emulated. Each device access method has characteristics that allow flexibility and optimization of device utilization for a particular system configuration.

**Direct**
With direct access to hardware devices, the physical device is mapped into the guest’s memory space, the guest has full control of the device, and there is no virtualization layer between the guest and the physical device. The guest and application have bare metal access to the device, and other virtual boards do not detect the presence of the device in their hardware scans. This provides the highest level of performance in guest behavior.

![Figure 9: Direct guest access to devices](image)
**Shared**

Shared device access is desirable when one guest “owns” access to the device but there is a need for other guests to have access to the data presented through the device. This might be the case when one guest has access to a hard drive partition, for example, but other guests need to access the file system of the hard drive. The native device driver resides in the “owning” guest while a stub device driver resides in the guest that shares the device.

![Figure 10: Shared device access](image)

**Virtualized**

Virtualized access to devices is beneficial when more than one guest needs individual access to a device. In this manner, the physical native device driver is placed within the hypervisor and each guest that needs individual access to the device loads a “stub” device driver that communicates with the driver in the hypervisor.

![Figure 11: Virtualized access to devices](image)

**Emulated**

Emulated device access is necessary when the device driver in the guest cannot be replaced with a stub driver. Wind River Hypervisor provides hardware emulation to the native driver in the guest operating system but uses its own native driver to access the real hardware device. The hypervisor can emulate devices for multiple virtual boards and handle the physical device for all of them. Emulated devices provide the highest level of flexibility for the virtual boards, but there is a performance penalty compared to virtualized or shared devices.

![Figure 12: Emulated access to devices](image)
Inter-guest Communication

Wind River Hypervisor facilitates fast communication between virtual boards and their guest operating systems by providing point-to-point messaging, multi-OS inter-process communication (MIPC), and IP communications using a virtual network interface controller (VNIC). There are two available transport methods for messaging: Shared Memory Transport (Standard Profile) for MIPC connections, and a secure transport that can handle MIPC connections as well as VNIC connections. VNICs have a gateway feature for VNIC-to-MIPC communications. Fast IP connections among virtual boards are achieved by using the guest operating system’s VNIC driver and creating one or more virtual Layer 2 Ethernet switches inside Wind River Hypervisor. This capability is available in the standard profile only.

![Diagram of choices for virtual board-to-virtual board communications](image)

**Figure 13: Choices for virtual board-to-virtual board communications**

Multi-OS Inter-process Communication

Communication between virtual boards is an important capability when building embedded systems. MIPC provides a mechanism for sending messages and data between virtual boards. MIPC provides the MIPC API—which includes socket-like routines—for VxWorks and Wind River Linux kernel applications, and for bare-wire VBAs. MIPC also provides the AF_MIPC socket API—which includes standard socket routines—for Wind River Linux user-mode applications. Both APIs support connection-based and connectionless datagrams and connection-based byte streams. MIPC can be used with either the Shared Memory Transport or SafeIPC transport layer.

**Virtual Network Interface Controller**

Wind River Hypervisor supports VNIC drivers in the guest operating system for TCP/IP connections between virtual boards. The connections can be through a Layer 2 switch (standard profile) or a point-to-point connection over the SafeIPC transport. VNIC drivers have a gateway capability of converting between UDP/IP packets and MIPC packets, which allows communication between a network stack on one virtual board and MIPC on the other.
Shared Memory Transport Layer (Standard Profile)
The Shared Memory Transport layer is designed for high-performance MIPC connections, which can be optimized by adjusting socket and channel options.

SafeIPC Transport Layer (Safety Profile)
The SafeIPC transport layer is designed for security, but it can also be optimized by adjusting the queue length of SafeIPC channels in relationship to scheduling. The SafeIPC transport layer supports both MIPC channels and VNIC interfaces. MIPC-to-VNIC connections are also supported.

Layer 2 Ethernet Switch (Standard Profile)
Wind River Hypervisor provides an internal virtual Layer 2 Ethernet switch that provides standards-based IP connectivity among the guests hosted by the hypervisor.

Based on needs, guests can be configured to have one or more VNICs, each directly connected to a virtual port on the switch. Multiple internal subnets can be configured and connected to multiple isolated internal switches. Guests can belong to one or more virtual internal networks. MAC addresses are assigned by the virtual NIC drivers or can be configured along with IP addresses through the board configuration.

ARCHITECTURAL DESIGN
Wind River has decades of experience in developing real-time embedded operating environments and partitioned systems, and this knowledge has been leveraged in the architecture of Wind River Hypervisor.
Deterministic
An embedded hypervisor can only support operating systems and applications with deterministic capabilities if it too is deterministic. It needs to respond within known time constraints without fail. Embedded developers rely on a deterministic operating environment that needs to include the hypervisor as well as guest operating systems.

In addition to the requirement for deterministic response times, Wind River Hypervisor is optimized to provide low overhead, even on multi-core processors with large numbers of cores.

Event-Driven
Wind River Hypervisor works as passively as possible. It provides separation, configures resources, and allows the guest operating systems to run at full speed. It has no active threads and is completely event driven; there is no overhead or activity unless requested by the guest operating systems. Examples of events are system calls from the guest operating system or a driver, interrupt delivery, and so forth.

Minimal Footprint
Wind River Hypervisor’s footprint is kept to a minimum, ensuring that unnecessary code is not executed as part of a hypervisor-based system run-time, and that the hypervisor has minimal impact on system operations. With minimal code size, applications’ determinism and safety are retained without impact to expected behavior. The virtualization layer is scaled down to the level containing only the minimal functionality required to provide its services. Any additional code, such as for device drivers, is not contained inside the hypervisor but resides directly inside the guests that require it.

Scalable
It is not uncommon for embedded systems (especially in the networking space) to use up to 32 microprocessor cores or more, and this number of cores will continue to increase in the coming years. Unlike many hypervisors and SMP operating systems, Wind River Hypervisor was designed from the ground up to be scalable, ensuring that performance is not compromised as the core count increases.

Flexible Device Model
Embedded device developers must pay specific attention to local hardware device access. In some devices, I/O throughput is critical; in others, latency or timing constraints are critical.

With Wind River Hypervisor, memory, PCI attributes, and interrupts can be directly mapped into a specific guest. The hypervisor is not involved in the data path to or from the device in this case. This capability provides native performance while delivering the development and operational benefits of embedded virtualization.
OS Agnostic
Fragmentation of processor types and operating systems is common in the embedded industry, particularly across a product line. Many device developers use low-cost processors for small devices and higher-performing processors for their high-end products. Wind River Hypervisor, together with Wind River operating systems, provides a single API and environment that can be used across many different devices. The hypervisor itself is completely OS agnostic, meaning that other operating systems, either third-party or in-house, can be added as guests.

DEVELOPMENT TOOLS
An integrated development environment is typically the cockpit from which the development team builds embedded systems. Wind River Workbench provides an Eclipse-based development environment that improves developer productivity for tasks such as configuring, diagnosing, and analyzing hypervisor-based systems.

Graphical Configuration
The configuration choices made while architecting an embedded device can be graphically selected, viewed, and verified using Wind River Workbench development tools that are tightly integrated with Wind River Hypervisor. Graphical configuration allows developers to select interrupts and device offsets (for hardware devices, for example) and graphically view the physical board configuration to see the relationship and mapping among the virtual boards, the processor cores, the device drivers, and the physical devices. Through this capability, developers are also able to graphically view the complete memory map of the generated system image to verify virtual board placement in system memory.

Figure 15: Wind River Workbench Architectural Overview tab showing a configuration of Windows SMP, VxWorks, and Linux on a quad-core processor
Automatic Configuration (Intel Platforms)

The Wind River Hypervisor development and deployment workflow includes an extension to static configuration of device ownership on Intel targets. The boot sequence allows PCI devices to be discovered and allocated to guest operating systems based on an auto-configuration policy, enabling a critical subset of the device allocation to be statically configured and the remainder of the configuration to be deferred to boot time. Workbench provides a facility to define a static configuration of a subset of the devices, as shown in Figure 16, and to download a new configuration based on the auto-configuration of a device after it has booted, as shown in Figure 17.

This approach allows automation during the development cycle and flexibility during boot. For example, during the development cycle, you may not have the correct configurations or definitions for the target. The auto-discovery/configuration retrieval cycle can automate and help tune the configuration of the Wind River Hypervisor integration project.

![Figure 16: Architecture overview of Wind River Hypervisor integration project before auto-configuration](image1)

![Figure 17: Architecture overview of Wind River Hypervisor integration project using downloaded auto-configuration](image2)
Building and Debugging

Workbench provides single-click system build capabilities that include the virtual boards and the complete system file. Once the build process completes, the system image can be automatically downloaded for debugging and analysis. Debugging in a multi-core and virtualized environment often requires the developer to keep track of many things at the same time. Workbench enables this through the multi-context debugger.

Workbench provides a single integrated environment to configure, build, debug, and analyze systems based on Wind River Linux, VxWorks, and Wind River Hypervisor, which simplifies a developer’s workflow and increases developer productivity in comparison to switching back and forth between tools from different vendors.

Flexible Terminal Console Options

A terminal interface can be a powerful mechanism for interacting with Wind River Hypervisor or guest operating systems, especially during development. For example, a user may need access to the hypervisor shell as well as the bash shell of a Wind River Linux guest OS and the target shell of a VxWorks guest OS. When configuring Wind River Hypervisor, there are a number of options to make the terminal interfaces available to the user.

However, often an embedded device will have just one serial port for this purpose. Wind River Hypervisor and Wind River Workbench provide options that address this issue.

Direct Ownership of Serial Ports

Wind River Hypervisor or a guest operating system can be given exclusive access to a serial device. Although this is the simplest method, the number of guest operating systems requiring a serial port may exceed the available devices. In this case, there are other available options.

Console Port Sharing with Application Multiplexed I/O

Wind River Hypervisor provides a mechanism for debugging guests over a shared serial connection called application multiplexed I/O (AMIO). The serial device driver exists in the hypervisor, which communicates directly to the serial device. Both unmodified and paravirtualized guest operating systems have the appearance of ownership of the serial device, but Wind River Hypervisor multiplexes and demultiplexes data sent to the driver and received by the driver through a special device called a multiplexed virtual serial device. The architectural overview diagram in Figure 18 shows a configuration of AMIO for two guest operating systems.
Figure 19 shows the Workbench AMIO Console view for a serial connection to com1 of a booted target with the configuration shown in Figure 18. The VxWorks guest owns channel 2 and displays the VxWorks target shell in the Channel 2 tab. The Core OS and Channel 1 tabs control the I/O for the Wind River Hypervisor console and Wind River Linux console, respectively.

AMIO requires configuration in the Wind River Hypervisor only, since the guest operating systems function as though they are communicating to a serial port. No guest resources are needed to host connections to other guests.
JTAG On-Chip Debugging
Having fine-grained control over an embedded target is paramount for success in debugging a virtualized embedded system. Wind River Workbench On-Chip Debugging allows JTAG debugging of drivers, interrupt routines, board bring-up, and other low-level scenarios. Often when debugging a single thread, it is necessary to stop other threads at the same time to avoid buffer overruns or underruns that would change the debug scenario. Workbench On-Chip Debugging allows the developer to stop all cores of a multi-core target with short lag time, providing an overview of the entire context.

INTEGRATION
Wind River Hypervisor is part of the Wind River multi-core software solution and integrates with the other components: VxWorks, Wind River Linux, Wind River Workbench, and Wind River Simics.

Wind River Hypervisor supports an open and well-defined interface to virtual boards. Any operating system can be executed inside a virtual board with the appropriate paravirtualization modifications. Wind River Workbench provides the interface to configure, build, and debug small and large hypervisor-based systems through graphical editors, elaborate build configurations, and multi-context debuggers.

PROFESSIONAL SERVICES
The increasing use of multi-core and virtualization provides many more choices to design teams delivering next-generation devices, but many programs do not yet have the experience or expertise required to leverage these new technologies. Wind River Professional Services is on the leading edge of the multi-core and virtualization revolution and can help accelerate the introduction of these technologies while reducing risk.

A CMMI Level 3–rated organization, Wind River Professional Services enables you to focus on development activities that add value and differentiate your design. Wind River offers industry-specific services practices, with focused offerings that help you meet strict market deadlines while keeping development costs down. Our experienced team delivers device software expertise globally to solve key development challenges, and directly contributes to our clients’ success.

Backed by our commercial-grade project methodology, Wind River Professional Services includes the following:

- Multi-core/virtualization architectural review
- Requirements discovery and definition
- Multi-core and virtualized board support package (BSP) and driver optimization
- Software system and middleware integration
- Application and infrastructure development

Typical projects range from two to four person-weeks for driver and BSP implementation; one person-month to one person-year for hardware design or extensions to an existing software solution; and multi-person-year programs that bring customer concepts to reality.
through design, creation, and system test and verification.

Wind River Professional Services has extensive experience with platform design, including safety critical systems and navigation/infotainment systems. Professional Services has implemented both hardware and software solutions for the embedded device market, and continues to work with standards organizations to establish the next generation of platforms.

Installation and Orientation Services

The Wind River multi-core software solution provides a host of technologies that may seem overwhelming at first. Wind River offers Installation and Orientation Services to ensure your project starts on time and without hassle by delivering the following:

• **Onsite installation:** Guided install on your hardware and host platform, along with a sample multi-core software configuration, demonstrations, and examples of customizations

• **Hands-on orientation:** Introduction to architecture, development file system, open source packages, porting drivers, and design issues

• **Advice:** Introduction to Wind River support channels and processes, additional services, project reviews, and consultation

Wind River Installation and Orientation Services will expedite your path to productivity, eliminating common sources of user error and helping you maximize the benefit of multi-core and virtualization.

**EMBEDDED DEVELOPMENT KITS**

Wind River has partnered with leading embedded board vendors to provide embedded development kits (EDKs) that enable developers to begin application development within minutes of opening the kit. EDKs provide a technically comprehensive platform to address complex development requirements.

Each EDK includes a bootable USB flash drive that immediately turns any host computer into a fully integrated development environment, with absolutely no installation required. Each board comes with a pre-flashed, run-time trial version of the Wind River VxWorks real-time operating system, Wind River Linux, or Wind River Hypervisor.
EDUCATION SERVICES

Education is fundamentally connected not only to individual performance but also to the success of a project or an entire company. Lack of product knowledge can translate into longer development schedules, poor quality, and higher costs. The ability to learn—and to convert that learning into improved performance—creates extraordinary value for individuals, teams, and organizations. To help your team achieve that value, Wind River Education Services offers flexible approaches to delivering product education that best fits your time, budget, and skills development requirements.

Public Training

Wind River public courses are scheduled for your geographical convenience. They are conducted over one to five days, using a mixed lecture and interactive lab classroom format that leverages the experience of Wind River instructors and other course participants. Courses provide a fast, cost-effective way for students to become more productive with Wind River technology.

Benefits of public courses include the following:

- Conceptual introduction that orients students to the subject matter
- Selective examination of the details, focusing on the most commonly used areas, or on areas with which users tend to be least familiar
- Personal guidance and hands-on application of individual tools and course concepts
- Chance to grasp device software concepts, as well as the fundamental issues involved in real-time design
- Knowledge needed to develop device drivers, perform hardware porting, or develop applications
- Answers to specific questions about topics addressed in the course

Consult your local Wind River sales representative for course schedules and fees.
Private Training
If you have a large project team or a number of new users, you may benefit from private training. Instructors will consult with you and, based on the workshop series curriculum, determine which topics should be included and emphasized. This type of education offers an opportunity for one-on-one discussions with our instructors about your specific project needs, technical requirements, and challenges.

Advantages of private training include the following:

- Your entire team gains a common knowledge base.
- Knowledge and skills are transferred directly from the classroom to your workplace.
- Training is scheduled for your convenience.

Consult your local Wind River sales representative for further information about private training.

CUSTOMER SUPPORT
Wind River Customer Support, a Service Capability and Performance (SCP)–certified organization, provides support for Wind River Hypervisor.

Your subscription includes full maintenance and support delivered through the Wind River Support Network website and our worldwide technical support team. While under subscription or support agreement, customers receive both maintenance updates and major updates.

Visit the Wind River Support Network at www.windriver.com/support for fast access to product manuals, downloadable software, and other problem-solving resources. The Support Network offers a comprehensive knowledge base with robust search features for locating information quickly.

Additional support features, including proactive email alerts covering particular technologies, platforms, or product patches and technical tips for common problems, are available for all customers by subscription. Support Network visitors can also access a community of developers to discuss their issues and experiences.

If you cannot find the information you need through the Wind River Support Network, contact our global support team for access to the industry’s most knowledgeable and experienced support staff.


Wind River Customer Support contact details are at www.windriver.com/support/.
North America, South America, Asia/Pacific

support@windriver.com
Toll-free: 800-872-4977 (800-USA-4WRS)
Tel.: 510-748-4100
Fax: 510-749-2164
Hours: 6:00 a.m.–5:00 p.m. (Pacific time)

Japan

support-jp@windriver.com
Tel.: +81 3 5778 6001
Fax: +81 3 5778 6003
Hours: 9:00 a.m.–5:30 p.m. (local time)

Europe, Middle East, Africa

support-ec@windriver.com
Toll-free: +800 4977 4977
France tel.: +33 1 64 86 66 66
France fax: +33 1 64 86 66 10
Germany tel.: +49 899 624 45 444
Germany fax: +49 899 624 45 999
Israel tel.: +972 9741 9561
Israel fax: +972 9746 0867
UK tel.: +44 1793 831 393
UK fax: +44 1793 831 808
Hours: 9:00 a.m.–5:30 p.m. (local time)