Choose Your Embedded Virtualization Solution Wisely

Using virtualization to consolidate various types of functions on multicore platforms is becoming increasingly popular. When evaluating virtualization approaches, however, the desire to employ the newest technologies needs to be traded off with the value of preserving elements of old designs.

by Kim Hartman, TenAsys

Consolidation of various processing workloads using partitioned multicore PC CPUs has been the promise of embedded virtualization for over a decade. There is a continuum of initiatives, including the Internet of Things (IoT) and Machine to Machine (M2M), which involves merging different application types onto the same platform. The initiative’s primary value proposition is the opportunity to reduce system costs while enhancing the number and type of services supported.

The technologies are in fact promising, but the devil is in the details, as they say. The term “virtualization” gets used so loosely that the technical issues behind specific “embedded virtualization” implementations are often not fully appreciated. Many people think that the only solution is to use a hypervisor, but this type of virtualization comes at a cost. And variations within hypervisor usage models cause OEM product developers to trade off adaptability and costs.

Machine builders face some of the trickiest problems. The desire to consolidate workloads to add features and minimize costs is bringing together discrete system elements. Some of these (specifically Windows-based HMIs) have evolved to be PC-based and are being combined with real-time controllers performing special functions (e.g., motion-control subsystems) along with the need for networking capabilities (machine-to-machine and machine-to-cloud) added in. The PC-compatible portion is the most cost-driven, and engineers who design and maintain these supervisory systems are accustomed to dealing with a relentless flow of hardware and software changes. Separately, the real-time portion, closest to where the real work is getting done, comes from a world that is both resistant to change and risk-averse. Consolidating these environments requires embedded virtualization, the hosting of heterogeneous operating system environments—both real-time and non-real-time—on the same platform.

FIGURE 1

The term virtualization applies to a wide range of implementations. Embedded virtualization is applied to only those solutions that preserve determinism.
Critical Success Factors

As experienced developers of embedded systems are well aware, there are some key trade-offs among the critical factors leading to success of a new design effort. The challenge of achieving success is in delivering the existing mission-critical services without missing market window timing or overspending on the solution. Significant investments in building equity in market position and intellectual property value for existing products must be carefully protected. Though it may be tempting to start with the latest tools, experienced project leaders know that this can easily inject delays into the project and risk not meeting core market requirements altogether. A more pragmatic strategy is to reuse as much of an existing application as possible, enhancing the system with services to enable new connectivity and interoperability features.

Obviously, most embedded designs involve a merging of old and new design elements. But the problem of providing reliable support for legacy content has caused many OEMs to delay embracing the compelling use models of modern multicore processors. In this regard, different embedded virtualization approaches provide different levels of support for multicore processing and varying levels of complexity in achieving the prime objective of consolidation.

Embedded Virtualization Is About More Than Just Hypervisors

There exists a continuum of technologies that provide the means to run multiple operating systems on the same platform. For embedded virtualization we are interested in comparing and contrasting the range of approaches, which on the one hand yield the highest real-time performance and on the other hand are the most versatile.

Consider the chart in Figure 1. At the far end of the spectrum are Type 2 hosted hypervisors and full virtual machine manager (VMM) solutions. These were designed for and are generally applied to IT-type problems (e.g., VirtualBox and VMWare), where deterministic execution in the guest operating environment is a non-factor, and therefore these systems don’t qualify as supporting embedded virtualization.

Simply moving into the range of Type 1, bare metal hypervisors do not yet guarantee support for determinism (as with KVM and Hyper-V). Similar to hosted and full VMMs, they provide a complete PC environment, making them very easy to use, but the lack of real-time responsiveness disqualifies them for use in deterministic embedded applications.

Moving along the continuum brings us to deterministic Type 1 hypervisors, which can support real-time processing. There are two classes of these. The first uses the familiar Type 1 hypervisor approach, but also provides the needed guest-to-core affinity, enhanced with virtualized services only where absolutely needed. This approach ensures the guest retains its deterministic and real-time capabilities and provides the greatest versatility in supporting legacy RTOS, general or proprietary OSs, all without need for modification. TenAsys’ HaRTH, a hard real-time hypervisor technology running inside the company’s IN-time RTOS, is an example of this. When HaRTH is configured to support hosting a single guest alongside Windows, it results in a product that TenAsys calls eVM for Windows (Figure 2).

The other class of deterministic Type 1 hypervisor is supported by most other embedded hypervisor vendors. Figure 3 is a diagram commonly used to depict such an approach. While also providing for partitioning services and core affinity, members of this class are built to support a more narrow set of guests. This is primarily due to the requirement that the guest must cooperate directly with the hypervisor it is running on. Para-virtualization techniques are often used to simplify the services a host hypervisor must provide by relying upon the guest to use proprietary para-API hooks. This may be an effective way to improve some type of
guest operations, but it limits the adaptability of the hypervisor and leads to the use of modified, and most often, proprietary RTOS and GPOS guests.

Regardless of the approach of deterministic Type 1 hypervisors, they can all support time-critical applications. And when PC chipset services (e.g. Intel VT-d) are not available, developers of guest-based applications will likely have to modify the drivers of any bus mastering devices to compensate for a lack of physical address translation. This makes selection of the hardware platform a bit more complicated as there is as yet no established embedded virtualization specification for Intel-based PCs.

Explicit Hardware Partitioning

As with any deeply embedded device, the highest performance is always obtained through explicit hardware partitioning. In a Windows-based system, this process is done by modification of the base RTOS to work in cooperation with the Windows environment on the same platform. Partitioning is done explicitly with the help of standard Windows APIs. Both run natively, right on the associated CPUs (Figure 4). Neither operating system is affected by virtualization, as there is no hypervisor to take context away. As Windows runs natively, there is no violating the Microsoft licensing restrictions of running embedded versions (e.g. WES7) on a virtualized platform.

Explicit hardware partitioning, introduced in 1997, is the longest time-tested solution in the market. Developing applications for TenAsys’ INtime for Windows RTOS features the use of a native or WIN32-like API and the familiar Microsoft Visual Studio line of IDE products. This design environment in itself saves substantial costs and time in bringing consolidated solutions to market.

Hybrid approaches can lead to interesting solutions. Consider running a deterministic Type 1 hypervisor on a dedicated core next to Windows. This combination of explicit hardware partitioning and hypervisors creates a unique characteristic similar to a hosted Type 1 solution, but features isolation (partitioned CPUs) and real-time functionality.

Other non-real-time configurations with Linux on the same platform can yield some interesting solutions. Consider running a hardened Linux-based firewall /VPN appliance on its own core next to Windows (Figure 5). Assigning the system Ethernet to Linux and connecting to Windows using a shared memory based on virtual LAN produces a network-hardened platform without any additional hardware!
So Which Virtualization Solution Do You Choose?

Deciding which virtualization solution to choose depends, of course, on several factors. These include how extensible your system needs to be in terms of performance, functionality and ease of use, and how much legacy content you want to preserve. Bringing your legacy application to a consolidated platform along with new workloads could mean extensive porting costs to different OSs where a para-virtualized solution is used, or it could be done with legacy software stacks. When porting an application is an acceptable option, an explicit hardware partitioned solution supporting both Windows and real-time will have the best overall performance and can save substantial costs with reuse of familiar Visual Studio toolsets and integrated I/O stacks.

Conversely, with IoT increasingly becoming a reality, you may want to take advantage of best-in-class software from multiple, as in more than two, different OS environments. In that case, a solution involving a hypervisor may be optimal. IoT systems can drive a drastic shift in design objectives, and can intensify the challenges of balancing the needs of deeply embedded systems with the richness of the Internet environment. Major among these are the need to provide security and to simplify user interfaces, while keeping software development costs and time-to-market under control. New, creative solutions may be unfamiliar in the light of standard software architectures of the past, but may yield better results.

The key to selecting which approach is best for your application is planning. Examine your requirements and pay particular attention to valued legacy IP. Look at the needs in the kind of processing your customers will want to do, and also the technology trends in the individual elements of your solution. For example, updating embedded systems not only to be Internet aware, but also to couple with Internet-based software resources, is motivating engineers to plan for maximum flexibility and extensibility in their designs. Deciding on a particular para-virtualized solution could restrict their design’s ability to evolve. New software is continuously being developed that OEMs may want to take advantage of economically in future product versions, and designers want to be able to easily take advantage of processors with an increasing number of cores.

TenAsys
Beaverton, OR
(503) 748-4720
www.tenasyss.com

RTC PRODUCT GALLERY

Multicore Development Tools and Multicore Board

Embedded Solutions™

ADLE3800PC - Intel® E3800 Series PCIe/104 SBC
- Intel® E3800 Series SoC Processors, DC/Quad
- Up to 8 GB DDR3L-1333, 1.35V SoDIMM204 Socket
- Type 2 Downward-Stacking PCIe/104 V2.01 with 2x Gen2 PCIe x1 Lanes
- 4x USB 2.0, 1x USB 3.0, 2x Serial COM
- 2x SATA 3 Gb/s, 2x GLAN Ethernet
- PCI Express Mini Card 1.2 Socket, Compatible with Mini PCIe or mSATA Modules

Phone: (858) 490-0597  Fax: (858) 490-0599
Email: sales@adl-usa.com  Web: www.adl-usa.com

www.menmicro.com

Play it Safe!

Rugged Computer Boards and Systems for Harsh, Mobile and Mission-Critical Environments

- Safe computers for rail, road and air, up to SIL 4/DAL-A
- Modular box and panel PCs for industry & transportation
- Powerful system solutions on CompactPCI®/PlusIO/Serial
- Rugged, standard Computer-On-Modules (ESMexpress®, EMMini™)
- EN 50155- and e1-certified Ethernet switches and fieldbus interfaces

www.tenasyss.com