MULTICORE SYSTEM MANAGEMENT: HYPERVISOR OR MULTICORE FRAMEWORK?

MENTOR EMBEDDED PLATFORM SOLUTIONS

Mentor®
A Siemens Business

WHITEPAPER

EMBEDDED PLATFORM SOLUTIONS

www.mentor.com

With increasing application of multicore to embedded devices, a number of important decisions are necessary to ensure an optimal design. One such decision concerns the overall control and management of a multicore system. Is a hypervisor required or is a multicore framework a better solution?

MULTICORE SYSTEMS

There are a variety of reasons why an embedded system may be implemented using multiple cores. It may be to simply attain more processing power, but it is most likely to allow functional segmentation of a design in a number of ways. This latter motivation results in designs being configured as Asymmetric MultiProcessing (AMP) systems.

An AMP system may be constructed from any combination of core architectures; all the cores may be identical or there may be a rich mixture of core types that includes conventional processing units as well as specialized cores, like digital signal processors (DSPs) for instance. Each core executes independently in an AMP architecture, with or without an operating system. Each core's operating system may be selected on the basis of the required functionality.

There are some specific challenges with an AMP design:

- An inter-core communication facility is most likely required.
- There may be safety/security issues that require the cores to be protected from one another.
- Boot order – the sequence in which the software on each core starts – may be important to avoid synchronization and security issues.
- Debugging the disparate workloads running on the potentially heterogeneous cores can be quite challenging.

Although the cores in an AMP system are independent, these challenges indicate that some overall control facility is necessary. Broadly there are two options:

1. A hypervisor. A complex software component that runs across all the cores.
2. A multicore framework. A software component allowing enablement for AMP systems which runs on each core.

It should be noted that multicore applications can be implemented using a symmetric multiprocessing (SMP) enabled operating system, but that approach does not allow for independent workloads to be executed on different cores, and also does not support heterogeneous cores.

HYPERVISORS

A hypervisor is a fairly complex, versatile software component that provides a supervisory capability over a number of operating systems, managing CPU access, peripheral access, inter-OS communications and inter-OS security. A hypervisor may be used in a number of ways. For example, multiple operating systems may be run on a single CPU to protect an investment in legacy software, although with the growth of multicore processors this is becoming rarer.

Alternatively, hypervisors can be used in embedded applications in AMP designs, where supervision of inter-core communication and allocation of peripherals to specific cores is needed. A hypervisor can additionally take care of the boot sequence and manage shared peripheral access. One of the main advantages of using a hypervisor is that if an operating system 'crashes' it will not affect execution of workloads on other cores, and in some cases the hypervisor can even reboot that operating system without requiring a reboot of the device. It is, of course, very advisable to utilize a hypervisor that is specifically designed for use in embedded applications for better performance.

Although it is possible to develop a hypervisor that will enable all of the required separation and virtualization features in software, which is quite difficult and is unusual these days. Today hypervisors are designed to use underlying virtualization features present on most multicore processors.

PROS AND CONS OF HYPERVISORS

Hypervisors have advantages and disadvantages compared with other solutions.

PROS

- Great flexibility enables efficient resource sharing, dynamic resource usage, low latency, and high bandwidth communication between VMs
- Strong inter-core separation
- Enables device virtualization and sharing
- Ability to assign ownership of peripherals to specific cores

CONS

- Only work on a homogenous multicore device (i.e. all cores are identical)
- Significant code footprint
- Some execution overhead
- Require hardware virtualization enablement in the processor

MULTICORE FRAMEWORKS

Because of their separation, management and sharing capabilities, hypervisors have far more functionality than many embedded designs demand – they can be overkill. To address this issue, a few embedded runtime vendors developed an alternative that was specifically engineered to support an AMP multicore system: the multicore framework.

Frameworks are designed very specifically to support the multicore application, providing just the key functionality: boot order control and inter-core communications. The result is that a framework loads a system with a much lower overhead and can be run on much more basic systems. Although each core in an AMP design probably runs an operating system, one or more cores may be "bare metal" – i.e. running no OS at all. A multicore framework can accommodate this possibility.

INTER-PROCESSOR COMMUNICATION (IPC)

Once the remote processor OS and application stack are running, many use cases will require communication with other parts of the system. The Mentor Embedded Multicore Framework provides a cleanroom implementation of a remote processor messaging framework feature called ***rpmsg*** to establish a communications channel between the master operating system and the remote operating systems. In this way, data can be passed back and forth between the two in an inter-processor communication channel.

The transport layer that enables both remote processor lifecycle management and inter-processor communication is ***VirtIO***. VirtIO is a virtualization standard for high performance input/output device drivers widely adopted in virtualized Linux® environments.

REMOTE PROCESSOR LIFE CYCLE MANAGEMENT

Assuming control over a remote processor, and then starting or stopping an OS and/or application stack within that remote processor, is referred to as remote processor (***remoteproc***) lifecycle management. The Linux community has adopted a remote processor framework for managing this scenario. Remoteproc allows a master operating system to bring up other operating systems on other cores.

The remoteproc feature within the Mentor Embedded Multicore Framework allows remote processor interoperability between Mentor® Embedded Linux®, Nucleus® RTOS, and Bare Metal Environments (BME) and Linux and RTOS products from other vendors. A key benefit to remote processor lifecycle management is reduced power consumption. The remote core stays in a low power state when not in use. Only after remoteproc is used to bring up the remote core and deploy the necessary firmware does the remote core draw any notable power.

PROS AND CONS OF MULTICORE FRAMEWORKS

Multicore frameworks have advantages and disadvantages compared with other solutions.

PROS

- Provides the minimally required functionality for some applications
- Modest memory footprint
- Minimal execution time overhead
- Can work on heterogeneous multicore devices (i.e. all cores do not need to be identical)Support bare metal applications

CONS

- The core workloads are not isolated from each other
- Can be more difficult to control boot sequence, and to debug

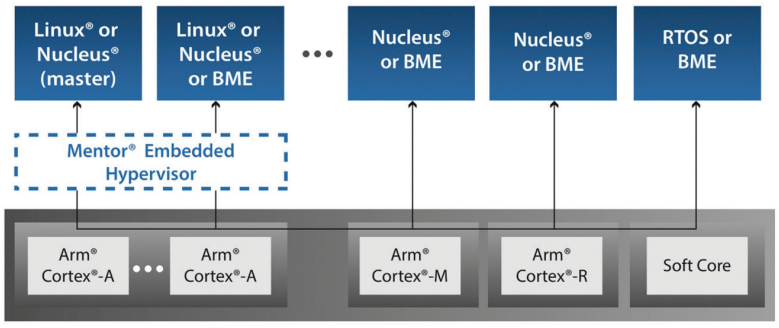
OPENAMP

Although some multicore frameworks are proprietary, some standards have been proposed that govern their functionality, interfaces etc. A popular example of such a standard is OpenAMP.

There are two key functionalities in OpenAMP:

1. Life cycle management using ***remoteproc***. This facilitates control of boot order etc.
2. Inter-core communications using ***RPMsg***.

A current reference implementation of the OpenAMP standard is available on GitHub. Mentor Embedded Multicore Framework (MEMF) and Mentor Embedded Multicore Framework Cert are proprietary implementations of the OpenAMP standard. Extensions to the standard include additional functionality to support Linux as a Remote, Large Buffer, Zero Copy, Proxy support for Ethernet and additional development tools.

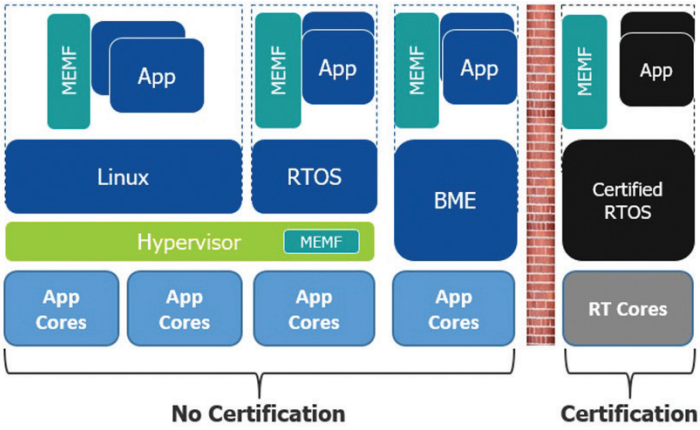


MIXED SAFETY-CRITICALITY SYSTEMS

A mixed safety-critical system is a system that requires the execution of several applications of different safety integrity levels (SIL) or different criticalities, such as safety-critical and non-safety critical, on a single SoC. Both a hypervisor and a multicore framework can support this type of configuration.

A hypervisor does this by certifying the hypervisor itself. The virtual machines can then have different criticality levels running with the certified hypervisor. The separation is provided by the certified hypervisor, which typically uses underlying hardware virtualization and separation features on the SoC.

A multicore framework leverages other hardware-assisted separation capabilities provided by some SoC architectures to obtain the required separation between the safe domain and the non-safe domain. This includes the separation of processing blocks, memory blocks, peripherals, and system functions. The multicore framework provides enhanced bound checking to ensure the integrity of shared memory data structures. It also provides interrupt throttling and polling mode to prevent interrupt flooding. It is even possible to use a non-safety certified hypervisor along with a mixed criticality enabled multicore framework as shown in the following picture.



CONCLUSIONS

Deciding to use a hypervisor or a multicore framework, or both, to control and manage a multicore system is a critical architecture decision. The final decision will depend on the specific application requirements and the use case for the device. The options should be considered as complementary solutions to unlock the power of a multicore SoC. The availability and understanding of these choices allows the designer to achieve a more optimal multicore design.

For the latest product information, call us or visit: www.mentor.com

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000
sales_info@mentor.com

Silicon Valley
Mentor Graphics Corporation
46871 Bayside Parkway
Fremont, CA 94538 USA
Phone: 510.354.7400
Fax: 510.354.7467

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics
Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
11F, No. 120, Section 2,
Gongdao 5th Road
HsinChu City 300,
Taiwan, ROC
Phone: 886.3.513.1000
Fax: 886.3.573.4734

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Trust Tower
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

Mentor® A Siemens Business

MGC 08-20