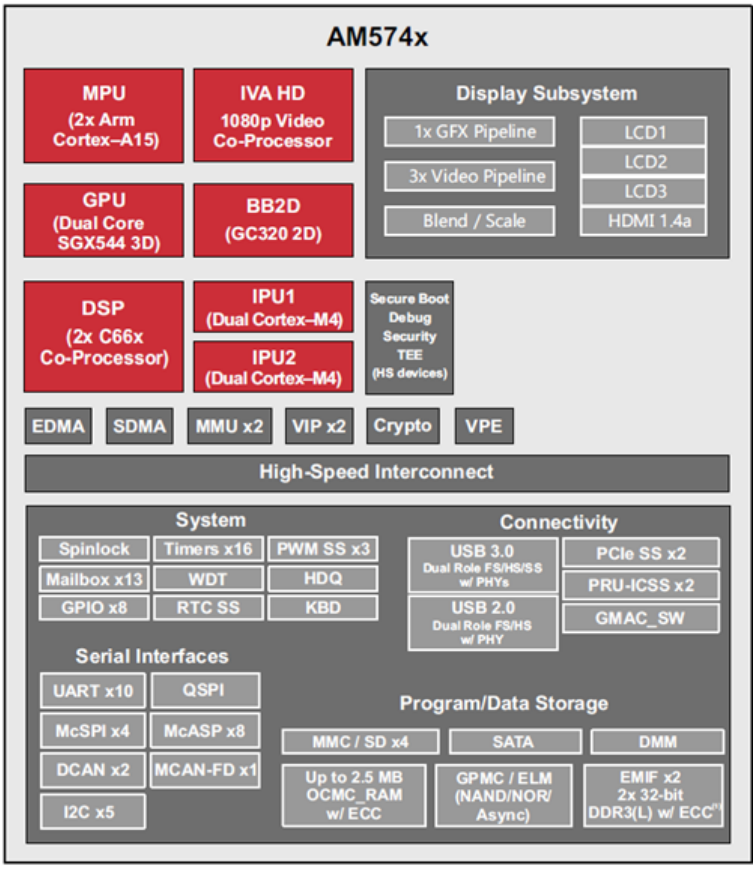
Technical Article

Hypervisors in Embedded Systems

Pekka Varis

**updated 4/18/2018*

Virtualization, hypervisors and containers have proven to be valuable solutions in the general IT and enterprise space. With virtualization you can run operating systems called guests, inside a sandbox contained under the control of another piece of software (the hypervisor) that manages all interaction of the guest toward hardware. The hypervisor runs at a more privileged level of hardware access than the operating system (OS) kernel (supervisor) and user privilege of the guest OS. Linux containers are a simplification to this without the need for a hypervisor when the guests are Linux as well. Virtualization solutions have matured to the point that with a couple clicks I can install and experience the benefits of virtualization on my laptop. Windows, Mac or Linux, no problem, I can compile code in Linux and use the required Windows based corporate IT tools on my Mac if that is my preference. Why can't we enjoy the same freedom in embedded systems?

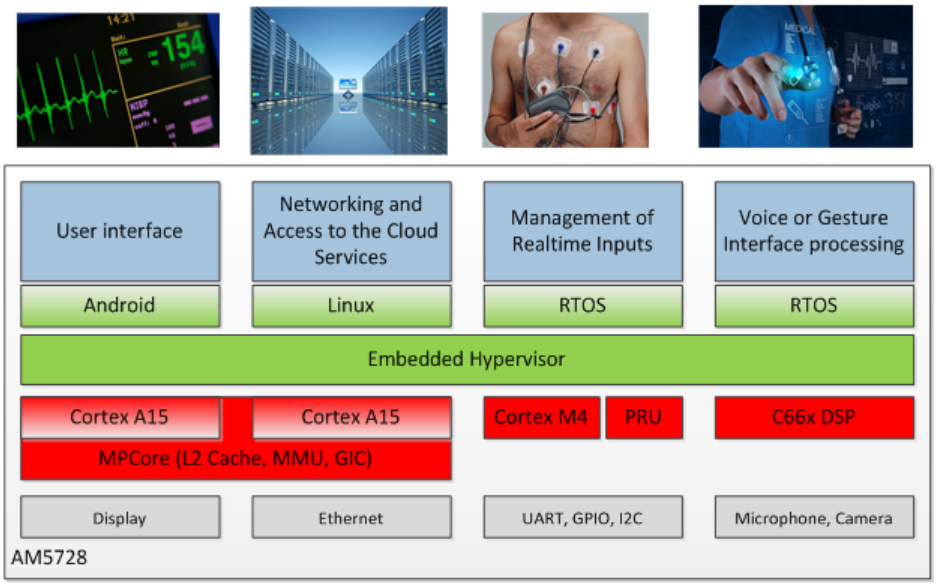


Embedded systems have different concerns and priorities from enterprise or data center server infrastructure or laptops. Typically the main difference is the priority of meeting the real-time patch requirement of the use case, as opposed to meeting a metric like average throughput or transactions per second. Embedded systems are different. Often a good embedded system is invisible and meets the use case requirement under the hood. There is often no value in going beyond the required performance, and vice versa the system becomes useless if it does not meet the required performance. For example, an elevator rate of acceleration will be right at the level which the engineer specified to make the passengers feel comfortable, irrespective of the number of passengers in it. A well-engineered elevator will not accelerate faster with less people or slow down with more.

The enabler for virtualization in embedded systems is the availability of hardware support in the system on chips (SoCs) in the right power consumption and price range. Sitara™ AM5748 processor SoC brings ARM® Cortex®-A15 cores, ARM Cortex-M4 cores, C66x digital signal processor (DSP) cores, video acceleration, graphics, a display subsystem and many peripherals and connectivity options into a single chip.

Patient monitoring systems are great examples of integrated multiple discrete processors or microcontrollers (MCUs) in an SoC. A patient monitor might utilize an MCU to manage the input from sensors for heart rate, pulse, blood oxygen level, etc. but may require another processor to present this information in a graphical format on a display. Mentor Graphics provides a great example to these patient monitoring systems.

The Internet of Things (IoT) drives the need for a full featured IP connectivity solution. This has led many embedded systems to run Linux for the convenience of having a full featured networking stack with latest constantly updated security features and application level protocols written in high level languages like Java. An embedded engineer used to bit banging— a UART with GPIO pins— might wonder at the magnitude of inefficiency, but, for example GE's Predix cloud platform gives a great overview of why the world has moved beyond hand coded TCP/IP stacks. However, for all the goodness of Linux, even with real time control patches, it might not satisfy the real-time constraints. In these cases, being able to run a deterministic real-time OS alongside Linux is needed.



A hypervisor, such as the Mentor Embedded Hypervisor, enables the integration of multiple proven software components in a single heterogeneous SoC. The application might have been implemented on two or three discrete processors or microcontrollers that can now be consolidated to a single SoC. The proven and possibly even certified RTOS system managing the input from the sensors and calculating the values to present the graphical user interface (GUI) might be Windows or Android based, and the secure networking stack can be written in Java and run on Linux.

Another typical difference from an enterprise use case for a hypervisor is in the relationship of physical cores to guest OSes. In a typical enterprise use case, a hypervisor runs a number of guest OSes larger than the number of physical cores, sharing a single networking interface. In most embedded applications, the real-time requirement is fulfilled by strict core affinity. Some or even the entire guest OSes are pinned to be the only OSes that ever run on a given core. The goal is to guarantee performance of the real-time application because no other software will run on that core and isolate the use of peripherals. This constraint can allow simpler implementations; Jail House has an interesting example for embedded systems. It will not solve all the problems of a commercial hypervisor or Linux Kernel-based Virtual Machine (KVM), but in the niche of running Linux and one or two RTOSes side by side it seems like an elegant approach. Another interesting development in the open source domain is an OpenAMP project where the commercial hypervisor and RTOS vendors are

also participating; this will lower the complexity of providing heterogeneous software and OS configurations to heterogeneous SoC's like Sitara AM5748 processors.

Learn more about our Sitara processors platform:

- Read more about Processor SDK
- Order Sitara AM5748 processor now
- Learn more about Sitara processors
- Learn more about AM57x processors

Commercial hypervisors are available from Mentor Graphics, QNX, Sysgo, and Green Hills . Of the open source approaches Linux containers (LXC) can be used today and we are looking at Jailhouse, KVM and Xen.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265