Building solutions with the new Xilinx All Programmable Zynq Ultrascale+ MPSoC and the new Aicas Jamaica 8 Toolchain and Runtime

Xilinx is the world's leading provider of All Programmable FPGAs, SoCs, MPSoCs and 3D ICs, enabling the next generation of smarter, connected, and differentiated systems and networks. Aicas is the world's leading provider of high performance, hard realtime, deterministic Java bytecode virtual machine runtime environments, and component and application platforms enabling the next generation of secure, safe and reliable software for embedded, realtime and cyber-physical systems. Combining these technologies offers systems designers and developers the tools to solve their current and future problems.

The Xilinx Zynq Ultrascale+ MPSoC is a combination of several technologies of which two main technologies are multicore CPU System-on-Chip and FPGA fabric. The MPSoC is the only product that combines multicore CPUs targeting general purpose application processing, multicore CPUs targeting realtime software, and FPGA logic for soft IP. The Aicas Jamaica Toolchain and Runtime transparently combines and executes mixed interpreted bytecodes with compiled machine instruction code for an optimized mix of code size and throughput performance. Jamiaca is the only product that combines hard realtime execution, write-once/run-anywhere code and standards-based I/O API libraries for intelligent device software. This paper provides brief overviews of how the Xilinx Zynq Ultrascale+ MPSoC and the Aicas Jamaica Toolchain and Runtime combine to provide systems designers and developers a set of flexible, useful, and scalable tools to apply to their new designs.

The MPSoC ARMr5 Realtime Processing Unit (RPU) and The JamaicaVM Runtime on Flat Memory Model Architectures

The MPSoC RPU is intended for applications requiring the lowest jitter, highest deterministic realtime software execution possible on the MPSoC. The RPU has two distinct operating modes: dual symmetric multicore operation; and dual execution fault detection lockstep operation. In both modes, the MPSoC can be configured such that the RPU always has a higher priority access to the memory controller and peripherals than the application processors (APUs). This eliminates the possibility of hidden dependencies between the RPU and APUs such as bus contention starvation. The MPSoC can be configured such that the APUs do not have any access to the RPU system memory completely isolating the RPU instruction and data from the APUs. This eliminates the possibility of instruction and data corruption caused by defects in APU hosted software.

The RPU does not have a memory management unit (MMU). This eliminates virtual memory paging as a source of non-determinism. The absence of an MMU means that the entire address map of the RPU is available to all software running on the RPU. MMU and page table management is the only mechanism operating systems use for memory isolation and protection between multiple processes. Thus application processes written to run directly on operating systems on the RPU without an MMU can be subject to non-local software defects from other processes co-hosted on the RPU. That is, any component of any application can accidentally write over any part of memory corrupting any other application. Because it is impossible to anticipate and test for all operating conditions, it is impossible to verify in complex multiprocessed non-virtualized applications, the absence of all non-local software defects on the RPU.

The JamaicaVM runtime uses a virtual machine memory model that does not make any assumptions of the presence of an MMU and virtual memory. Instead, JamaicaVM's memory model virtualizes all memory references for both interpreted and compiled bytecodes. Aicas' Light Weight Process Model (LWPM) layers on top of JamaicaVM's memory virtualization reintroducing process memory isolation to flat memory model CPU architectures. As JamaicaVM runtime is deterministic, LWPM does not introduce non-deterministic artifacts.

Hosting JamaicaVM with LWPM on the RPU introduces process isolation without compromising the realtime software execution of the RPU. As process isolation eliminates most of the non-local multiprocessing code defects related to memory accesses, the combination of LWPM on the RPU can greatly improve the reliability of realtime software executing on the MPSoC.

The MPSoC ARM53 Application Processing Unit (APU) and the JamaicaVM Runtime on Virtual Memory Operating Systems

The MPSoC APU consists of a quad of ARM53s. In contrast to the RPU, the ARM53s have MMUs for virtual memory page table management and support process memory isolation. Each ARM53 can be separately dynamically enabled and disabled for power management. The ARM53s may run a single symmetric multiprocessed (SMP) operating system such as QNX Neutrino, SysGO PikeOS or a Linux distribution. Alternatively, the ARM53s may run a hypervisor such as Xen, SysGO PikeOS, Lynx Systems LynxSecure, or QNX Hypervisor. An hypervisor can in turn run multiple guest OS partitions, as well as offer a direct API, which is called a minimal runtime environment (MRE). Both the SMP OS's and the Hypervisors provide CPU affinity APIs to control the distribution of threads of execution across the ARM53 CPUs.

One would expect that an MPSoC software design that utilizes both the APU and RPU must take into consideration the differences between the CPU memory models as well as operating system scheduling differences. As noted above, the RPU memory model does not offer OS process memory isolation. Modules that run on the APUs may be dynamic memory intensive which virtual memory management effectively handles, while the same module run on the RPU may experience unpredictable memory exhaustion due to memory fragmentation. Further the OS for the RPU is likely to be an RTOS supporting only fixed priority scheduling, while the OS for the APU may support both fair scheduling and fixed priority scheduling. Multithreaded applications written for an APU OS may have implicit OS fair scheduler dependencies such that the same application moved to the RPU may have unexpectedly poor performance where only fixed priority scheduling is available. These factors and others can contribute to difficulties for systems designs that incorporate both the APU CPUs and RPU CPUs and attempt to maintain a common code base to facilitate code sharing and avoiding code duplication.

As stated above, the JamaicaVM Runtime uses a virtual machine memory model that does not make any assumptions of the presence of an MMU and virtual memory. The JamaicaVM virtual machine memory model is equally compatible with the APU MMU as with the RPU. JamaicaVM virtualizes thread affinity and scheduler disciplines. The standards-based APIs implemented by JamaicaVM abstract event handling, thread priorities, and peripheral I/O. The JamaicaVM runtime offers a form of fair scheduling and fixed priority scheduling whether the underlying OS provides both disciplines or only fixed priority scheduling. In the special case of an OS that only provides fair scheduling, the JamaicaVM runtime scheduler emulates fixed priority scheduling to the extent possible.

At the application source code level no changes are required when moving modules from a MMU enabled CPU to a non-MMU enabled CPU. Similarly no source code level changes are required when moving modules from a fair and fixed scheduler capable OS to a fixed scheduler only OS.

The JamaicaVM runtime can not completely abstract and isolate the variations between the APU and RPU environments. It is likely that latency and jitter will vary between the APU and RPU for the same module. For example, the APU CPUs will have higher jitter than the RPU CPUs. However, the elimination of many of the sources of OS and CPU dependencies encourages code sharing, avoidance of code duplication and creation of a common code base by developers.

The MPSoC FPGA fabric and the JamaicaFPGA Highlevel Language Direct Synthesis

The FPGA fabric is one of the main features of the MPSoC that sets it apart from other multicore SoCs. Xilinx provides tools, Vivado HLx and SDSoC, to synthesize FPGA logic from the VHDL and Verilog hardware languages as well as from the C and C++ software languages. The MPSoC includes a feature to dynamically reconfigure the FPGA fabric under control of either the RPU or APU. Dynamically reconfiguring the FPGA fabric and synthesis of highlevel languages enables the concept of software-defined hardware.

A new Jamaica toolchain, JamaicaFPGA, integrates with the Xilinx tools to expand the number of high-level languages available for FPGA synthesis to all languages that compile to Java bytecodes. As the primary language compiled to Java bytecodes, JamaicaFPGA supports Java bytecodes compiled from Java programming language source code. Just as with targeting the APU and RPU described above, no source code changes are required when moving from either the APU or RPU to the FPGA fabric. Similarly, the JamaicaFPGA toolchain can not completely abstract the variations between the CPU and FPGA targeted implementations. In particular, FPGA synthesized code jitter can be an order of magnitude lower than the same code hosted on APU or RPU CPUs. Extending the common code base from the APU to RPU to FPGA synthesized code offers the developers not only the opportunity for code sharing, but a new dynamic, flexible choice in hardware and software implementation decision.

Summary

The Xilinx Zynq Ultrascale+ MPSoC and the Aicas Jamaica 8 Runtime and Toolchain both offer world leading computer hardware and software technology, respectively. Both products are evolutionary culminations of 32 and 16 years of successful technology development, respectively. Each independently delivers optimal solutions for embedded, realtime, intelligent devices. Combining the MPSoC's APU, RPU and FPGA technologies with the Jamaica MMU OS, MMU-less OS, and FPGA deterministic technologies, repositions the application software, realtime software and hardware implementation decision as a dynamic, flexible, runtime changeable decision instead of a static early stage design cycle decision.

For further information

David Beberman
aicas incorporated
6 Landmark Square, Suite 400
Stamford CT 06901-2711 USA
Tel: +1 (508) 210 4083
email: dbeberman@aicas.com
www.aicas.com

aicas GmbH
Haid-und-Neu-Str. 18
Karlsruhe, Germany
Tel: +49 721 663 968-0
email: info@aicas.com