Ensuring real-time predictability

Leveraging TI's Sitara™ processors programmable real-time unit

TEXAS INSTRUMENTS

Melissa Watkins,
Application Engineer
Texas Instruments

Carlos Betancourt,
Product Marketing Manager
Texas Instruments

System developers have discovered that many, if not most, contemporary applications require a finely tuned combination of high-speed software processing and real-time hardware performance. In other words, high-performance is not the equivalent of real-time performance. In fact, they both have their own unique and often mutually exclusive set of requirements which prevents either one from performing the role of the other very well.

For example, high-performance processors like Arm® Cortex®-A cores have an entirely different set of resources and processing capabilities than those of real-time processing cores, like the Programmable Real-Time Unit (PRU) co-processor in TI's Sitara processors. The capabilities that make an Arm core so powerful at processing software could also impede its real-time determinism and predictability. And, in many of today's most sophisticated applications, real-time capabilities are just as critical as high-performance, if not more so.

Real-time requirements

Outside of the data center, many systems need the low-latency predictability of real-time processing in one way or another. In fact, even many general-purpose systems that require a high-level operating system (HLOS) often have a real-time component or subsystem, such as communication protocol processing, audio processing, lighting control, sensor monitoring, factory or home automation, motor control and others.

In these types of systems, a general-purpose processor (GPP), no matter how high its performance, cannot deliver the guaranteed response time within strict time constraints that typify real-time applications and subsystems. Many of the features of a GPP like instruction pipelines and memory and interconnect architectures which make it so effective running a HLOS (**Figure 1**)

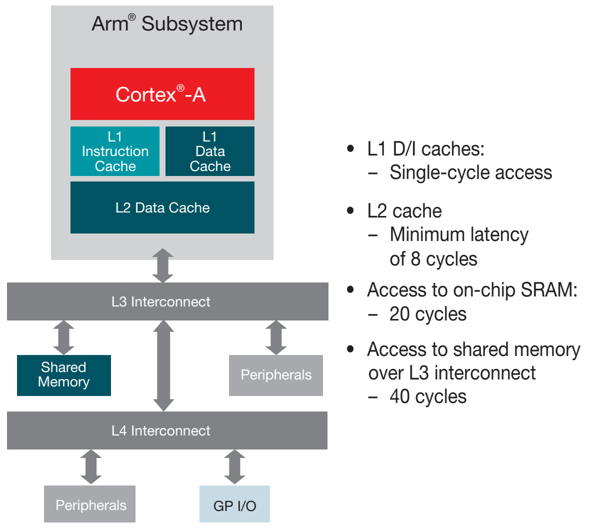

Figure 1*. A typical Cortex-A-based SoC general-purpose processing architecture.*

will often become counter-productive in real-time applications. The typical architecture surrounding a GPP core on a system-on-a-chip (SoC) will include several layers of memory, which may be internal to the processing core or external, as well as shared memory or dedicated to one core. Moreover, the usual SoC architecture will also include several layers of interconnects which link the various on-chip modules, peripherals and eventually lead off the chip via specialized or general-purpose input/output (GPIO) pins. All of these facilities and structure can get in the way of real-time processing.

The architecture surrounding a GPP cannot provide the predictable low-latency response times that must be guaranteed in a real-time application. Accessing data on any of the various levels of memory and communicating over any or all of the several layers of interconnects will add to the core's response time and ensure the unpredictability of the response. The processor's response time to an incoming time-sensitive interrupt from a sensor, for example, will vary according to a number of factors, such as where the needed data is stored, how many layers of interconnects must be traversed to access or store data, the processing load currently executing on the GPP core and others. One experiment on a certain GPP architecture showed that a simple toggle on a GPIO pin could take as much as 200 nanoseconds (ns) whereas the equivalent toggle on a real-time processor with direct GPIO pin access would have a response time of five ns, or forty times faster. In addition to the often slower response typical of GPP processors, the actual response time is usually unpredictable. In other words, the response time for a GPP to handle a real-time event will likely vary each time the event occurs.

To overcome the real-time limitations of GPP cores, certain capabilities such as real-time co-processors are often integrated into an SoC architecture. In the example below (**Figure 2**), the Texas Instruments (TI) Programmable Real-Time Units (PRU) form the basis for a real-time subsystem on processors. Such an architecture gives the SoC direct and fast access to the outside world since each PRU has its own single-cycle I/O. Additionally, local memory and peripherals dedicated to each real-time engine means that each unit is able to guarantee low-latency responsiveness. Plus an incoming interrupt has direct access to a real-time processing engine without encountering the delays caused by crossing several layers of interconnects and memory.

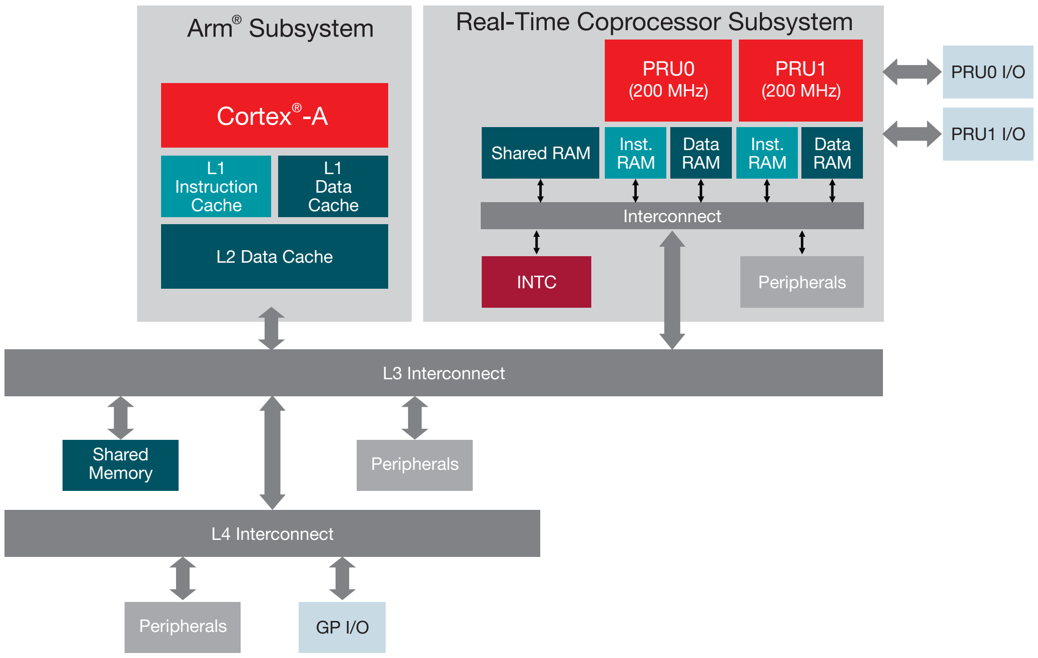

***Figure 2**. A general-purpose core supplemented with real-time co-processors.*

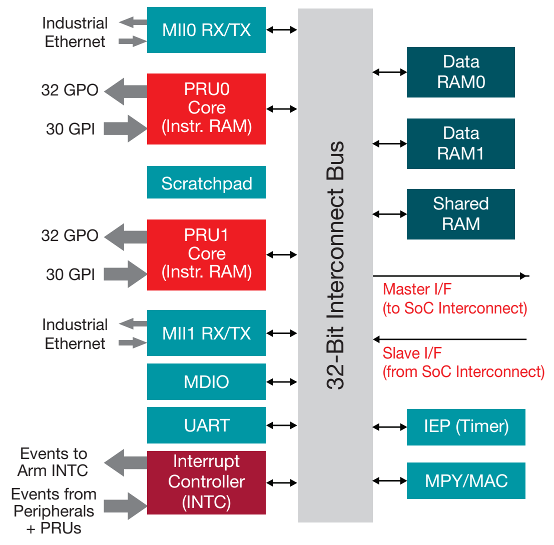

***Figure 3**. The TI Programmable Real-Time Unit (PRU) subsystem.*

Programmable Real-Time Unit (PRU)

The PRU (**Figure 3**) which is deployed along with Arm cores in the Sitara AM335x, AM437x, AM57x, 66AK2Gx, and AM6x(1) processors fulfills the role of a low-latency, deterministic real-time subsystem. Each PRU subsystem is made up of two 200-MHz real-time cores (or PRUs), each with a five nanosecond (ns) cycle time per instruction (the PRUs on some devices can run up to 225 or 250-MHz). Since the real-time cores are not equipped with an instruction pipeline, single-cycle instruction execution is ensured. The PRU's small, deterministic instruction set with multiple bit-manipulation instructions is easy to learn and use.

Shared memory as well as instruction and data memory dedicated to each real-time core allows for flexible program execution among all of the real-time and GPP Arm cores that might make up the SoC. One program or task might be better performed on one core while another could be executed faster when the processing load is shared among several PRUs and Arm cores. Direct access from the PRU to the Arm cores' layers of interconnects enables either tightly coupled execution or independent core operations.

(1) The PRU in the AM6x processors includes 2 additional PRU cores (called RTU_PRU), as well as additional features including support for gigabit Ethernet.

Access to all of the system's interconnects allows a PRU to call on any resource in the system when needed for a particular program implementation. In addition, each PRU subsystem comes with its own set of dedicated peripherals to ensure the unit's responsiveness. These peripherals avoid the data traffic on secondary and tertiary interconnects in the system by accessing directly the PRU's real-time cores. Several peripherals, such as Management Data Input/Output (MDIO) and Media Independent Interface (MII), enable real-time Ethernet capabilities. Other PRU subsystem peripherals include a UART interface.

Because of the interrupt controller and fast I/O pins dedicated to each dual-core PRU, the unit is able to closely monitor external events and respond in a predictable period of time. Each PRU has its own set of up to 30 inputs and 32 outputs that directly access external pins on the device package.

The best of both worlds: Combining Arm and PRU cores

The Sitara line of multi-core processors features escalating combinations of Arm Cortex-A cores and PRUs. With this architecture (**Figure 4**), the system designer can select the device with the right combination of high, general-purpose processing and real-time performance to meet the specific needs of the application. Arm cores have all of the resources and instruction support expected for high-performance operating system execution, either for a HLOS or a real-time operating system (RTOS), or both at once.

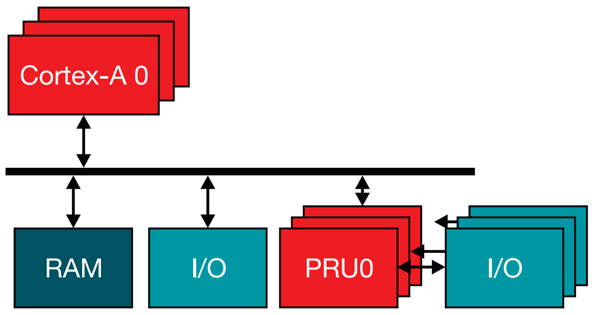

***Figure 4**. Base Sitara architecture with Arm cores and PRUs.*

At the same time, the five nanosecond cycle time of PRU cores as well as their low-latency data transfers and high-speed I/O accesses assure designers that transfers and data modifications will be performed in a predictable period of time. The responsive simplicity of a PRU core make it a natural fit for bit-banged communication interfaces like the Serial Peripheral Interface (SPI), the Inter-IC (I^2C) bus interface and others, including many industrial automation protocols.

Some end equipment architectures incorporate a field programmable gate array (FPGA) for real-time processing. Unlike the integrated PRU co-processor (**Figure 4**), an FPGA that is external to an Arm-based system processor would necessarily increase the bill of materials (BOM) costs, require additional board space, add complexity to the design and increase the power consumption of the system. With a PRU solution, system developers also benefit from a common software code base which simplifies feature upgrades, and multi-protocol processing across systems or on the same system. Development of successive generations of systems is accelerated by simply migrating the code base to the next model in the product line.

The Sitara hardware structure in **Figure 4** supports a software architecture (**Figure 5**) that tightly couples Arm cores and PRUs to ensure seamless and high-speed application processing. Typically, a Linux™ kernel running on an Arm core would include RemoteProc and rpmsg kernel drivers for the PRU subsystem. The RemoteProc is the basic control mechanism, allowing the Arm core to load PRU firmware, enabling PRU processing and other functions. Through the rpmsg driver user space applications and the PRU can pass messages (buffers) back and forth.

Applications

The simple implementation of a PRU combined with its full complement of resources make it a versatile processing engine for a wide range of real-time tasks, subsystems and application modules. Designers have also taken advantage of the PRU to deploy straight forward subsystems like stepper motor control units, bit-banging communications processors and sensor interfaces to the more complex tasks such as camera and LCD display interfaces, smart card processing and even very complex applications like Ethernet industrial automation protocol processing (**Figure 6**).

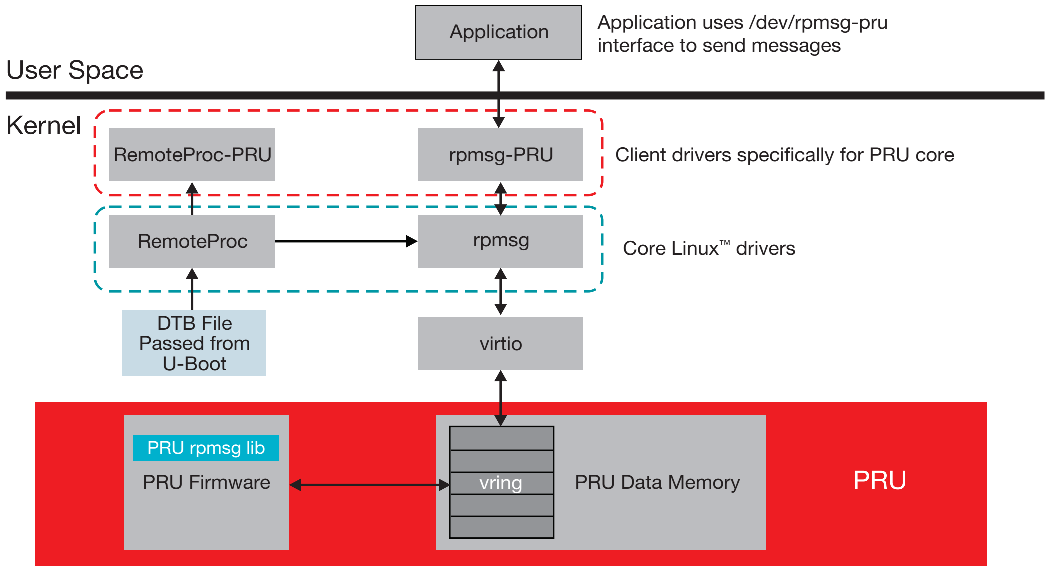

Figure 5*. Sitara Arm/PRU software architecture.*

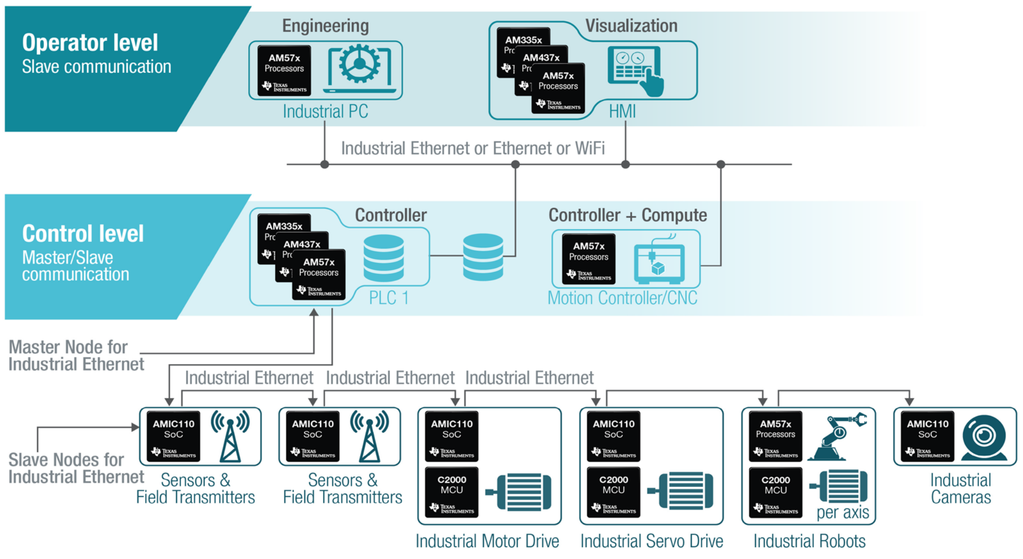

***Figure 6**. Factory automated systems with AMIC110.*

***Figure 7**. Sitara AM335x with Arm Cortex-A8 core and PRU running a 3D printer.*

The PRU is optimized for low latency and no jitter in the real-time execution and is able to process the MAC layer functionality of Industrial Ethernet standards. For example, since PRU-ICSS can support multiple protocols, it enables devices like the AMIC110 SOC to seamlessly bridge legacy motor drive designs by adding support for Industrial Ethernet such as EtherCAT, PROFINET, Sercos III, Ethernet/IP and PowerLink.

One of the more interesting applications of the PRU involves a 3D printer (**Figure 7**). In this case, designers took advantage of the BeagleBone development board with a Sitara AM335x processor featuring an Arm Cortex-A8 core running Linux, the user interface and model processing while the PRU performed the real-time control of five stepper motors. A shared region of memory was reserved for communications between the Arm core and the PRU.

Conclusions

In today's complex application world even the most straight forward general-purpose systems will frequently need the predictability, determinism and low-latency responsiveness that only real-time processing can deliver. Teaming the PRU with high-performance Arm Cortex-A cores in the Sitara line of processors provides the best of both worlds. By offering the best of general-purpose and real-time processor elements in a single package, end equipment is optimized for cost and power consumption, yet flexible for feature upgrades through efficient reprogramming.