Choosing between an ARM7 and a Cortex-M3 processor

Anders Lundgren - July 28, 2009

With the introduction of microcontrollers based on the ARM Cortex-M3 core, a developer wanting a low-cost 32-bit device can choose either a Cortex-M3 base or an ARM7TDMI. What are the criteria to consider when making that choice?

The ARM Cortex-M3 is an implementation of the ARM7v architecture, the latest evolution of ARM’s embedded cores. It is a Harvard architecture, using separate buses for instructions and data (rather than a von Neumann architecture, where data and instructions share a bus). The Harvard architecture is intrinsically significantly faster but physically more complex. With Moore’s Law, the complexity is not such a significant issue and the increase in throughput is valuable. The Cortex-M3 is aimed by ARM at the deeply embedded market. It is designed to be low cost and low power and to provide good performance for the cost and power. ARM sees it as particularly suited to automotive and wireless communications applications. As with all ARM designs, the company licenses the design to manufacturers that produce their own implementation, and already a number of manufacturers have committed to producing microcontrollers based on the Cortex-M3. The first of these companies was Luminary Micro (acquired by Texas Instruments), which is shipping devices with volume pricing at less than $1 (Figure 1).
There is a developing ecosystem of development tools and system software evolving from the ARM7TDMI ecosystem and anticipating further manufacturers entering the area. The other ARM core aimed at the same market is the ARM7TDMI (and ARM7TDMIS). This core has been around for more than 10 years. Many manufacturers (ARM claims more than 16) are selling families of microcontrollers based on ARM7 cores and the ecosystem of software and development and debugging tools is impressive. The ARM7TDMI is, in many ways, the workhorse of the embedded world.

As well as using the Harvard Architecture, the Cortex-M3 has other significant differences. It is a smaller basic core, reducing price and increasing speed. Integrated with the core are system peripherals, such as the interrupt controller, bus matrix, and debug functionality, which would normally be added by the microcontroller implementer. It has an integrated sleep mode and the option of an integral eight-region Memory Protection Unit. It is designed for the THUMB-2 Instruction set and reduces assembler usage to a minimum instruction set.

While the ARM7 implements both the ARM and Thumb instruction sets, the Cortex-M3 supports only the Thumb-2 instruction set. The Thumb-2 instruction set does not require the system to switch state between Thumb and ARM code, which reduces performance with earlier processors. Thumb-2 is designed specifically to implement C and includes an If/Then construct (predicating conditional execution of the next four instructions), hardware division, and native bitfield manipulation. Thumb-2 allows designers to maintained and modified applications at the level of the C code; it includes functionality that would normally require calling assembler level code (Luminary Micro claims that there is no need ever to drop into assembly language). These advantages add up to easier implementation and possibly faster time to market for new products.

Another innovation on the Cortex-M3 is the Nested Vector Interrupt Controller (NVIC). Unlike external interrupt controllers, as are used in ARM7TDMI implementations, this is integrated into the Cortex M3 core and can be configured by the silicon implementer to provide from a basic 32 physical interrupts with eight levels of pre-emption priority up to 240 physical interrupts and 256 levels of priority. The design is deterministic with low latency, making it particularly applicable to
automotive applications.

The NVIC uses a stack-based exception model. Program counter, program status register, link register, and general-purpose registers are all pushed on to the stack to handle the interrupt, and once interrupt processing is completed, the registers are restored. Stack handling is in hardware, so there is no longer a need to create assembler wrappers for stack manipulation for interrupt service routines.

Interrupts can be nested. An interrupt can exert a higher priority for earlier servicing, and priority levels can be changed during run time. Using a technique of tail-chaining, successive interrupts takes only three cycles, compared with 32 needed for a successive stack pop and push, which reduces latency and increases performance. If the NVIC is stacking (pushing) when an interrupt of higher priority arrives, fetching a new vector address is all that is needed to service the higher-priority one. Similarly, the NVIC will abandon a pop to service a new interrupt. This also achieves lower latency and is completely deterministic.

To generate regular time intervals for interrupts, the NVIC has an integrated System Tick timer, which can also be used as a heartbeat for scheduled tasks or for an RTOS. This means that there is no need for an external clock, unlike previous ARM architectures. The Cortex-M3 power-management scheme supports Sleep Now, Sleep on Exit (from exiting the lowest-priority ISR), and SLEEPDEEP modes, through the NVIC.

The memory protection unit is an implementation option. When implemented it allows areas of memory to be associated with specific processes in the application with rules governing access by other processes. For example, some memory can be totally blocked for all other processes, while other areas can be read-only for specific other processes. Another rule could halt execution if a process attempts to enter a memory area. This provides a significant improvement in reliability, particularly in real-time code.

The integrated debug and trace Debug Access Port can be implemented as either a two-pin Serial Wire Debug Port or a Serial Wire JTAG Debug Port. In association with the Flash Patch and Breakpoint unit, the Data Watchpoint and Trace unit, the implementation option of the Embedded Trace Macrocell and the Instrumentation Trace Macrocell, it is possible to carry out debug and monitoring functions within the core. It is possible to set breakpoints, watchpoints, define fault conditions, or carry out debug requests, and either halt operations or continue while monitoring. All of these facilities have been available in ARM architectures, but the Cortex-M3 pulls them together in a standard package for the developer.

While the ARM7 cores do not have such deeply integrated peripherals as the Cortex family, there is range of ARM7-based devices with an even larger range of peripherals, shading from general-purpose microcontrollers, through to application-oriented microcontrollers, SoCs, and even an ARM7 core in an FPGA from Actel. There are around 150 different microcontrollers based on the ARM7 (and that number can be even higher, depending on how you count versions).

For almost every application in the embedded space it is possible to find an ARM7 implementation that has been customized, to some degree or other, to meet the requirements. To the standard core the implementer adds different memory types and sizes and other peripherals, such as serial interfaces, bus controllers, memory controllers, and graphics units. They use a range of packaging types and for industrial, automotive, and other demanding applications, provide extended temperature range versions. They may also bundle software, such as TCP/IP stacks or even application-specific software.
For example, the STR7 product line (from STMicroelectronics) has three major families, with a total of 45 members with variants in packaging and memory. Each family has a different peripheral set aimed at specific applications—for example, the STR730 family is designed for industrial and automotive applications so is available in an extended temperature range and includes multiple I/O and 3 CAN interfaces, while the STR710 is aimed at consumer, point-of-sale, and high-end industrial applications and has multiple communication interfaces, including USB, CAN, ISO7816 and four UART, large memory and an external memory interface.

Implementers also choose to add help for developers, for example, by implementing ARM’s Embedded Trace Macrocells (ETM) and by supplying development and debug tools (Figure 2). By contrast, the number of companies shipping Cortex-M3-based products is limited, although other companies have announced their intention to produce products.

Figure 2: The Embedded Trace Macrocell makes it possible to carry out a wide range of debug and monitoring functions within the core.

Tools

The ubiquity of the ARM7 has led to an explosion in third-party products for developing and debugging applications. The ARM Web site lists more than 130 companies, and other companies are also serving this market.

Most manufacturers provide a basic development board, housing the device and providing interfaces to load programs, attach debugging tools and drive peripherals. It will include a status display of LEDs or a single line screen. Normally the board is packaged as part of a kit with a compiler and some debugging software. More advanced kits, including those from third parties, will include a full integrated development environment (IDE) with a compiler, linker, debugger, editor, and other tools. They may also include hardware, such as JTAG probes to connect the JTAG port on the device to the PC. In-Circuit Emulators (ICEs), one of the earliest and most useful forms of debugging tools, are available with ARM7 interfacing, from a range of manufacturers.

Software development tools range from modeling and visual design front-ends through to compilers. Application modules and middleware sit on top of real-time operating systems (RTOS) for rapid
development, and more of these are coming to the market. And, perhaps more important, there is a huge base of experienced ARM7 developers.

There is an emerging tools base for the Cortex-M3, but obviously this has still some distance to go. However, the integrated debug in the Cortex-M3 will make system bring-up and debug much easier and more efficient and removes the need for In-Circuit Emulators (ICEs).

So, which do you choose today? If cost is an absolute driver for you, consider choosing the Cortex-M3. If you are looking for better performance and improved power at low cost, you might again consider the Cortex-M3, particularly if the application is in the areas that ARM sees as primary targets—automotive and wireless. The integrated elements within the core and the Thumb-2 instruction set should make developing and debugging on the Cortex-M3 easier and faster than the ARM7TDMI.

The drawback at the moment is that there are a limited number of suppliers. However, many of the companies with ARM7-based devices are working on Cortex-M3 designs, and it would be worthwhile talking to suppliers to get a feeling for timescales and implementation details.

But as retargeting an application from the ARM7TDMI is not difficult, particularly when using an RTOS, the conservative route may be to use a device with an ARM7TDMI core for now but ensure that design and implementation do not use features that will make retargeting more complex.

Further reading

Two ARM white papers that provide further reading are “Introduction to the ARM Cortex-M3 Processor” and “Running ARM7TDMI Processor Software on the Cortex-M3 Processor.” Both can be downloaded from [www.arm.com/documentation/](http://www.arm.com/documentation/).

Author information

Anders Lundgren has been with IAR Systems since 1997. He currently works as product manager for the IAR Embedded Workbench for ARM. During the first years with IAR Systems he worked with compiler development and as project manager for compiler and debugger projects. Prior to joining IAR Systems, Lundgren worked with space science instruments at the European Space Agency and spent one year at the space science laboratory at the University of California, Berkeley. He received a master’s in computer science from the University of Uppsala, Sweden, in 1986.