Porting CompactECC to Atmel's New SAM3S MCU

By Markus Sewing, Sales Manager, Atmel Corporation

Together with Atmel, ubisys has created a port of its CompactECC library to the new SAM3S controllers with ARM Cortex-M3 core. Despite being useful for many applications, these general-purpose microcontrollers are a perfect match for the smart metering market.

First results prove that Atmel has managed to create a controller capable of providing users more performance right out-of-the-box, while at the same time, consuming less energy. Solutions that are already based on SAM7 devices can immediately profit from this easy migration path.

Introduction

Countries in the European Union have recently created laws turning the EU's Smart Grid initiative into national legislation. In Germany, the government has decided to switch to smart metering by 2014, thereby putting energy providers and meter manufacturers under pressure. The Energy Economics Law ("Energiewirtschaftsgesetz", EnWG) mandates that meters must "reflect the actual energy consumption and the actual usage times," starting from January 2010 (§21b). Energy providers have the obligation to "create monthly, quarterly or half-year totals," and they also have to offer "load-variable or daytime-dependent tariffs" until December 2010 (§40). These legal obligations are difficult to met without some kind of data transmission facility. With this background, VDE has recently released two specifications for metering devices. Specification series EDL (EDL21 and EDL40) covers the approximate 43 million household meters in Germany, while SyM² ("Synchronous Modular Meter") aims at commercial and industrial areas.

Both specifications employ elliptic curve cryptography (ECC) for creating digital signatures on consumption measurement values. In particular, the Elliptic Curve Digital Signature Algorithm (ECDSA) guarantees the authenticity of remote meter readings. Compared to other algorithms, like RSA, ECC provides the same level of security with shorter keys, or more security when the key length is kept equal. For instance, an ECDSA signature with a key-length of 192 bits is slightly more secure than 1024-bit RSA. This is of particular importance in the context of embedded systems, since shorter keys require less memory and less computational power, resulting in a significant performance and energy-efficiency gain. This is why the ANSI-standardized curve p192r1 serves as the base curve for digitally signing measurements on SyM² and EDL meters. However, SyM² meters face stringent timing requirements, as they need to perform two consumption measurements per second, sign and transmit them. In contrast, EDL meters have to meet rather relaxed timing requirements, a signature might take nearly three minutes to complete without raising any issues.

Therefore, ubisys has extended its C++ template class library CompactECC™ with a dedicated implementation of the p192r1 curve. This particular implementation allows for an optimal blend of code size, memory footprint, and execution time, depending on individual application requirements. The library can generally be used with all 8-, 16- and 32-bit controllers for which a C++ or embedded C++ compiler with template support exists. Integration with plain C projects is also possible.

Starting Point: SAM7S/SAM7X

One of Germany's largest meter manufacturers is using this version of ubisys CompactECC since October 2009 for its latest products. Atmel's AT91SAM7X controllers are perfectly suited for SyM² meters due to their embedded Ethernet MAC. Therefore, ubisys chose SAM7 as the initial platform to support the optimized p192r1 ECC variant. In response to customer request, the next platform supported was Atmel's 8-bit AVR, which is a cost-efficient match for very high-volume, price sensitive EDL meters.

Just to provide a few implementation figures, an ECDSA signature on SAM7X running at a 48 MHz master clock completes in 140ms and requires approximately 8 KB of ROM and 2.5 KB of RAM, fulfilling all pre-requisites for SyM² meters. In this benchmark, the C++ compiler has been configured in such a way that it would mainly generate 16-bit thumb instructions (ca. 99 %). This way, the code can profit from the 32-bit wide pre-fetch buffer. This buffer acts like a minimalistic instruction cache, which Atmel has integrated into its SAM7 controllers to improve the execution speed of 16-bit thumb instructions when the controller is running in excess of the flash memory's maximum access frequency. Without this pre-fetch buffer, when running at frequencies beyond 30 MHz, access to instructions and data stored in flash memory inherently imposes penalties due to mandatory wait-states. With the help of this pre-fetch buffer, a second 16-bit thumb instruction can be loaded from the buffer and executed by the core, while the next 32-bit word makes its way from the flash ROM to the buffer. In most cases, this approach provides for a steady flow of instructions to the ARM core's instruction pipeline.

With respect to CompactECC, only a single, very computationally intensive, multiplication (192-bit x 192-bit yielding 384-bit result) is implemented as a hand-crafted assembler routine employing 32-bit ARMv4 instructions and executing directly from SRAM. Access to the synchronous RAM never takes more than one clock cycle to complete, even at maximum clock speeds.

Migration to SAM3S

Finally, how complex and time-consuming has the migration to SAM3S been? Regarding the CompactECC library itself, this was done in five minutes. In the project settings, under the C++ compiler options, the code generation target has been changed from ARM7TDMI to Cortex-M3 – that was about it… almost.

Due to the fact that the Cortex-M3 does not support ARM mode (don't worry, 32-bit ARM instructions are included in Thumb-2) a single line in the assembler source code for the multiply operation mentioned above had to be changed, namely the directive that declares the instruction set being used. The actual instructions were not touched at all. Because the SAM3 provides a flash memory bus that is four times wider than that of the SAM7, it was no longer necessary to have this code executed from precious SRAM.

Moreover, it should be taken into account that the ARM7 follows the Von-Neumann architecture, while the Cortex-M3 is based on the Harvard architecture with independent paths for data and instructions. Accordingly, one will achieve the highest throughput, when instructions are fetched from the flash memory, while data is loaded from, and stored into, SRAM. The algorithmic part of porting CompactECC was complete at this point, since there were no other hardware dependencies to care about.

Well, it was not quite that easy in the end. Besides the library, a sample application had to be ported as well. Obviously, interaction with the underlying hardware is much tighter in this case. It starts

We've got your ARM® Cortex™ Solutions in hand...

Digi-Key® CORPORATION

www.digikey.com/cortex

with Atmel having slightly changed the syntax of identifiers in the controller's header file. Prefixes like AT91C_ID_..., AT91C_BASE_..., etc. have mostly disappeared; registers and constants have got slightly different names.

Functional adaptations were also required, beginning with the start-up code, because the exception vector table has changed between processor generations, one reason being the Cortex-M3 featuring an integrated Nested Vectored Interrupt Controller (NVIC). The NVIC obsoletes Atmel's Advanced Interrupt Controller (AIC), which is used in its ARM7- and ARM9-based devices. The NVIC provides a lot of advantages, including reduced interrupt latency and the absence of spurious interrupts, which used to be tedious and cumbersome to deal with. In addition, it is now possible to change the location of this table in memory, i.e. it is no longer fixed at address 0x00000000. Instead, the table may be relocated to almost anywhere in address space. Therefore, remapping the SRAM to the beginning of the address map in order to gain write access to the exception table is no longer necessary, and consequently, the remap command has also vanished.

Processor modes have been simplified, too, which also affects the number of hardware stacks and their initialization. The ARM7TDMI supports seven modes of operation (USR, FIQ, IRQ, SVC, ABT, SYS, UND), which indicates that this core was originally intended to be used with fully-fledged operating systems featuring multitasking, paging and the like. In contrast, two modes are sufficient for the Cortex-M3, which has been tailored for controller applications from the ground-up. Normal program execution is in thread mode, while interrupts are serviced in handler mode. In practice, two or three stacks are usually set-up within ARM7TDMI programs: one for the application, which typically runs in supervisor mode (SVC), one for interrupts (IRQ), and one for the fast interrupt (FIQ), if applicable. The remaining processor modes are typically of no interest to controller applications. Again, Cortex-M3 designs get along with one ("main") or two ("main" and "process") stacks.

As the start-up sequence progresses, low-level hardware initialization will be performed soon after stack initialization. Even before the C/C++ runtime library takes over control, a SAM7 application would switch from the 32 kHz oscillator to the quartz oscillator, and program the PLL to its operating frequency. This is done as early as possible to speed-up initialization of memory sections with pre-defined contents and, as a matter of fact, to improve the over-all system start-up experience.

The SAM3S includes two independent PLL blocks, which is advantageous for applications that need to provide a 48 MHz USB clock, which might otherwise not be easily derivable from the master clock frequency. In addition to its 32 kHz RC-oscillator, the SAM3S also hosts a faster internal RC-oscillator with 4, 8 or 12 MHz selectable output frequency. This oscillator is active from the beginning and provides the initial 4 MHz master clock. It is of major importance having flash wait-states configured in accordance with the system clock. Atmel has simplified the embedded flash controller interface. With previous generations, it was necessary to notify the flash controller of the master clock frequency, at least if the application performed write operations to flash memory pages. This is no longer required with SAM3S.

A number of peripheral blocks, including the Power Management Controller, get a protection feature, which helps to prevent unintended write accesses to mission-critical registers (for example clock frequency settings). In order to write new values to registers that are protected in this way, a key must be written together with the intended setting. The new value is only accepted, if the key matches a hardcoded reference value.

As a side-effect of introducing the NVIC, the shared system interrupt combining the interrupt request lines of system peripherals like PIT, RTT, WDT and DBGU became unnecessary, too. With the previous generation, the system interrupt service routine first had to determine the interrupt source responsible for the current interrupt request. In case of SAM3S, each system peripheral has received its own interrupt vector, and the associated service routine is able to service the right interrupt with its inherent knowledge of the source. For obvious reasons, the configuration of the interrupt controller has changed as well. Interrupt vectors for peripherals are simply stored following the processor's core exceptions (IRQs 0 - 15) and initial stack setting. A unified software interface, the Cortex Microcontroller Software Interface Standard (CMSIS), offers functions for enabling/disabling single interrupts, for defining their priority, and for triggering interrupts by software. What is more, interrupt service routines do not require special prologues and epilogues any more – service routines are executed as usual functions. Integrating the interrupt controller with the core also provides the advantage that the SAM3S does not suffer from spurious interrupts, like ARM7 and ARM9 devices do. Such interrupts could occasionally occur, when an interrupt line was active long enough to trigger an interrupt, but too short to allow the interrupt controller to provide the core with the proper interrupt vector, because the interrupt had vanished before the source had been determined.

The Periodic Interval Timer (PIT) is also no longer present, due to the Cortex-M3 having its own SysTick timer embedded into the core. As long as the PIT had been used to increment a counter via its associated ISR, migration to SAM3S is a snap: It is sufficient to assign the ISR to the proper interrupt vector, ensure that the right clock source is selected, and choose the desired reload value. This will cover 99 percent of all applications.

Further adaptation is required with respect to the PIO controllers, because the package pin multiplexer now supports four instead of two peripheral signals, which can be routed to each pin. In addition, each pin can serve as edge- or level-sensitive interrupt source with programmable polarity.

The debug unit (DBGU) has also gone. This peripheral combined several features: A serial port for trace messages, a chip ID register for identifying the exact controller flavor by software or via JTAG, an

...And in Stock!

Digi-Key®
CORPORATION

www.digikey.com/cortex

interface to the ARM7TDMI's debugging data channel and access to SAM-BA, Atmel's flash programming solution. These functions are now provided by a self-contained CHIPID block and a UART.

After all these changes were made, the CompactECC sample application for SAM3S was finally complete. Now, it was possible to build executable images for AT91SAM7S64-EK and AT91SAM3S-EK evaluation boards out of the same source code base. As part of the cooperation between Atmel and ubisys, the SAM3S-EK was made available to ubisys's engineering team well in advance of the official market launch. This would allow a strong cryptographic solution to be ready for all interested customers right from the beginning. The sample application is easily controlled by any terminal program and demonstrates the generation of private and public ECC key pairs, as well as creation and verification of ECDSA signatures. Commands that show the heap and stack memory consumption and print the results of inherent profiling measurements complete the sample.

Results

This way, the ubisys research and development team managed to achieve a 30 percent increase in performance compared to the SAM7S version – without any particular optimization. At the same clock frequency of 48 MHz, the ECDSA signature on the p192r1 curve now took less than 100ms, typically about 95ms. Part of this performance gain was due to the very efficient Thumb-2 instruction set, and part was due to the four-fold data bus interconnecting flash ROM and core, which is also even better utilized, due to the architectural differences between Von-Neumann and Harvard approaches.

The maximally achievable performance gain is even higher, because the SAM3S can run at up to 64 MHz, while the SAM7S is only specified for operation up to 55 MHz. Projecting the previous results, this means that a signature on the SAM7S would take an approximate 122ms, while the same operation would take only 75ms on the SAM3S. This comparison is absolutely fair, when we take the energy balance into consideration. At 55 MHz, the total current consumption amounts to 33mA for the SAM7S, while the SAM3S draws no more than 38mA at 64 MHz – in both cases, when supplied with 3.3V. Hence, the energy balance looks like W_{SAM7} = 3.3V * 0.033A * 0.122s = 13.3mJ for the SAM7S. This good performance is still outperformed by SAM3S by 30 percent: W_{SAM3} = 3.3V * 0.038A * 0.075s = 9.4mJ. With the computations having been quickly performed, the controller might enter a power-saving state until other calculations are required, or signed measurement values are to be transmitted over power-line or ZigBee networks. In addition, static power consumption of the SAM3S is below that of SAM7S, further contributing to the higher energy-efficiency of SAM3S.

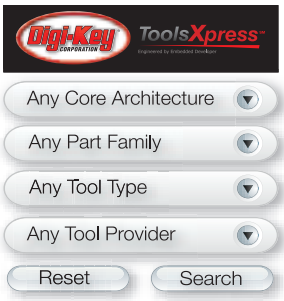
END

www.DevtoolsXpress.com

Pick a Tool. Any Tool.

Find the Right Development Tool, Compare it to Other Tools, Evaluate It, and Buy It from Digi-Key Tools Xpress -- Without Leaving Our Site.

The Digi-Key Tools Xpress intuitive research engines are used by engineers worldwide to locate, compare and evaluate hardware or software development tools.



Digi-Key Tools Xpress, engineered by Embedded Developer, is the only site in the industry where engineers can quickly find, compare and buy the leading development tools.

Compare before you buy: tools are listed side-by-side, with features and performance specs, availability, and prices, so you can make an educated decision!

Join the thousands of engineers worldwide who use Digi-Key Tools Xpress for their development tool needs.

Digi-Key CORPORATION

ToolsXpress℠

Engineered by Embedded Developer