SmartFusion Customizable System-on-Chip:

Intelligent, Innovative Integration

November 2011

Table of Contents

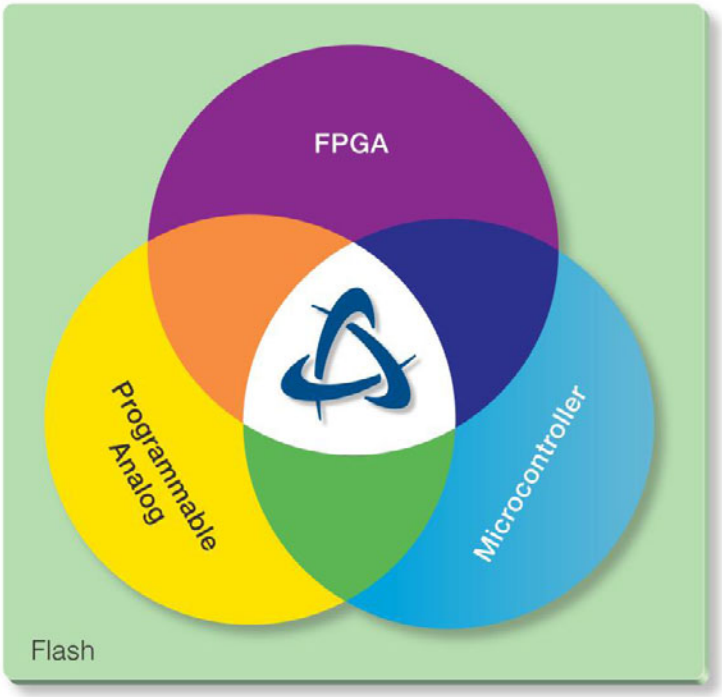
The whole point of an FPGA is flexibility. We could also mention integration and say instead that the whole point of an FPGA is flexibility and integration. But then there is cost savings. So the whole point of an FPGA is flexibility, integration and cost savings. Yet there is also power reduction. And then there's security.

All these advantages (and others besides) have made FPGAs very popular over the years. Engineers like the flexibility, space and power reduction and integration that FPGAs provide. So it stands to reason that adding more flexibility, more integration and more cost and space savings would be a good thing.

Microsemi SoC Products Group's new family of SmartFusion® customizable system-on-chip (cSoC) devices takes all the traditional advantages of FPGAs and combines them with equally flexible analog circuitry and the world's most popular embedded processor. It's all rolled up in one package, and all under your control. It's a single "super chip" that could very probably be the only chip in your system. How's that for space, time, power and cost savings?

Overview

SmartFusion cSoC devices have three basic features: a processor, an FPGA and an analog circuit block. All three are in the same chip and all three are programmable. The processor is programmable (through software); the FPGA is programmable (through normal FPGA design flow); and the analog circuitry is programmable (through some clever configuration software). Almost everything about SmartFusion cSoC devices is under the engineer's control, so even though the chips themselves are standard when they leave the factory, the products they enable are anything but.



You can think of SmartFusion cSoCs as a three-part chip or a three-legged stool that includes the processor, the programmable logic on an FPGA, and the programmable analog circuitry. In a typical engineering team, you'll have people who specialize in each domain: software, logic design and analog. All three will be comfortable with SmartFusion cSoC devices, but what's even better is that a single engineer working alone can also use a SmartFusion cSoC as a universal, all-in-one component to realize almost any design. All three design tools and domains are integrated, and all three talk to one another. It's the first real synthesis of the three major branches of embedded development. And it may be the only chip your system needs.

Some designers see the SmartFusion cSoC device as a processor; some think of it as an FPGA; some consider it an all-purpose analog chip with a processor and FPGA attached. All of that's true, because no one part is more or less important than the others.

But what's really interesting is that it brings all these things under one roof. That's not just good for space and cost savings——it also makes designs more secure and more flexible. With most or all of your product's electronics and interconnects inside the chip instead of outside it, you don't have to worry about PC board interconnect, unexpected component changes, interface problems, supply-chain headaches, reverse-engineering, power-on surge current, radio frequency (RF) interference. SmartFusion cSoC devices take care of most of these things in one fell swoop.

The Security Angle

Security can mean a lot of different things, from safety to reliability, to hacker-proofing and more. Saying your product is secure sounds nice but it's not specific enough. SmartFusion cSoC devices bring different kinds of security to your new product; security that's worth considering.

A SmartFusion cSoC uses flash memory, not SRAM, for its entire internal configuration. This has been a Microsemi SoC Products Group hallmark for years. It may sound like a trivial distinction, but it's an important one. Here's why.

When an SRAM-based FPGA starts up, it's a blank slate, so it has to be retrained on every power-cycle by reading from an external ROM. That means that (a) your FPGA won't work for several milliseconds until it's been retrained; and (b) all that training is out in the open. In other words, your precious FPGA configuration stream, which you spent several months or years developing, is there for anyone with an oscilloscope to see. It's as simple as reading a ROM, and there's no way to prevent it. Duplicating your design is as easy as duplicating the ROM. Is that how you want to protect your design?

In contrast, SmartFusion cSoC devices use internal flash memory that's never exposed to the outside. There is no ROM for disreputable reverse-engineers to read. Your configuration data—your design, really—never leaves the chip. There is no known way at all to tease it out, not even by "decapitating" the part and using an electron microscope to examine the silicon layers.

A nice side effect of this is that your chip actually works when it's powered up. In other words, it behaves like most other chips do; the way most engineers assume chips are supposed to work. That's more than just a mere detail. If your main (and perhaps only) component doesn't work on power-up, what happens to all of its floating inputs and outputs? How does that disrupt power-on initialization? What will the other chips do, the ones it's connected to, when their inputs are connected to this brain-dead FPGA that hasn't been programmed? And what's to prevent someone from yanking various configuration pins during that vulnerable pre-configured state? What happens then? There's something to be said for having your FPGA work all the time, every time, just the way you designed it.

Security Part II

There's another whole aspect to security that SmartFusion cSoC devices improve. With your FPGA, your processor and your analog circuitry all on the same device, nothing ever leaves the chip. That is, all the data transferred from the processor to the FPGA fabric, from the analog circuits to the processor or between the FPGA fabric and the analog area, is all secure. It never leaves your chip. It's never passed across an external bus that anyone can snoop. It's never exposed. That's a terrific boon when you're transferring a large amount of sensitive data throughout your system. (Oh, and it's also faster and more power-efficient, by the way. Who knew going green could have so many benefits?)

You might never have to encrypt data again. If you're encrypting data transactions merely to prevent them from being exposed to prying eyes, that whole problem goes away once the bus transactions are confined on-chip. If nothing leaves the secure area of your own chip, you can dispense with the cloak-and-dagger obfuscation and encryption and get on with the real work.

Your design is more secure, too. With none of its internal workings or interfaces exposed, all hackers see is a literal black box. Nothing about your architecture, structure or partitioning is visible or discernable in any way. Just the way you want it. They may not even know you're using SmartFusion cSoC devices at all: Microsemi SoC Products Group can private-label your chip with your own logo and part number.

Reuse

A good rule of engineering is never to design anything twice. Management consultants call this design reuse. We never seem to practice it as often as we should, often reinventing particular wheels from one project to the next. With so much programmable logic on the SmartFusion cSoC devices, this could present a real problem. How are you going to fill all that logic without spending months on the task? And isn't it likely that somebody, somewhere has already developed similar bits and pieces to what you're designing? Common features like pulse-width modulation (PWM) and Reed-Solomon encoder shouldn't have to be designed from scratch.

That's where Microsemi SoC Products Group's DirectCore library of IP comes to the rescue. DirectCore contains more than 50 different IP blocks in its library—and they're all free! If that doesn't suit your needs there are also independent third-party suppliers of IP blocks ready to plug into your SmartFusion design. Just plug and chug.

The same goes for software, which can often be more time-consuming to develop than hardware. Here again, Microsemi SoC Products Group provides a big library of commonly used middleware, such as TCP/IP stacks, file systems, HTTP, SMTP and plenty of other acronym-heavy bundles of code. And again, it's all free for the taking.

Dependence

The skeptical engineers out there are saying, "I like all this, but it means I'd be getting my main (and possibly only) component from a single source. That'll expose me to nefarious extortion from an unscrupulous silicon supplier." To which the more reasonable engineers reply, "How is that different from what you're doing now?" Aren't most of your components coming from a single source already? In fact, aren't all of your chips single-sourced? Does anything you're using have a drop-in replacement from another supplier? And wouldn't a disruption in the supply of any one of those components throw a wrench into your production?

Statistically speaking, you increase risk by using multiple components and decrease risk by reducing component count. Ideally, you'd have just one component from one supplier. Even more ideally, that one component would be programmable and customizable so that you're not simply buying the same generic component that your competitors are using, which would leave you with no differentiation and no "value added." But if you're using a single component that is programmable and customizable, you get all the advantages of a low-component-count bill of materials plus the freedom and flexibility to design whatever you want around it. The best, as they say, of both worlds.

Size and Cost

When we say "big component" we don't mean massive in the physical sense. SmartFusion cSoC devices aren't large. Nor are they expensive. So this is perfectly suitable for something like a battery charger, small motor controller, PC card gadget, display driver, or what have you. Big capabilities and big possibilities don't necessarily mean big price tag. Good things come in small packages.

Processor Subsystem

For a lot of engineers, the processor on the SmartFusion cSoC is the best part. Or at least, the most familiar part. That's because SmartFusion cSoCs are all based on the popular ARM® Cortex™-M3 processor. The ARM Cortex-M3 processor is among the world's most popular 32-bit processors, and an ideal basis for a mid-range embedded system.

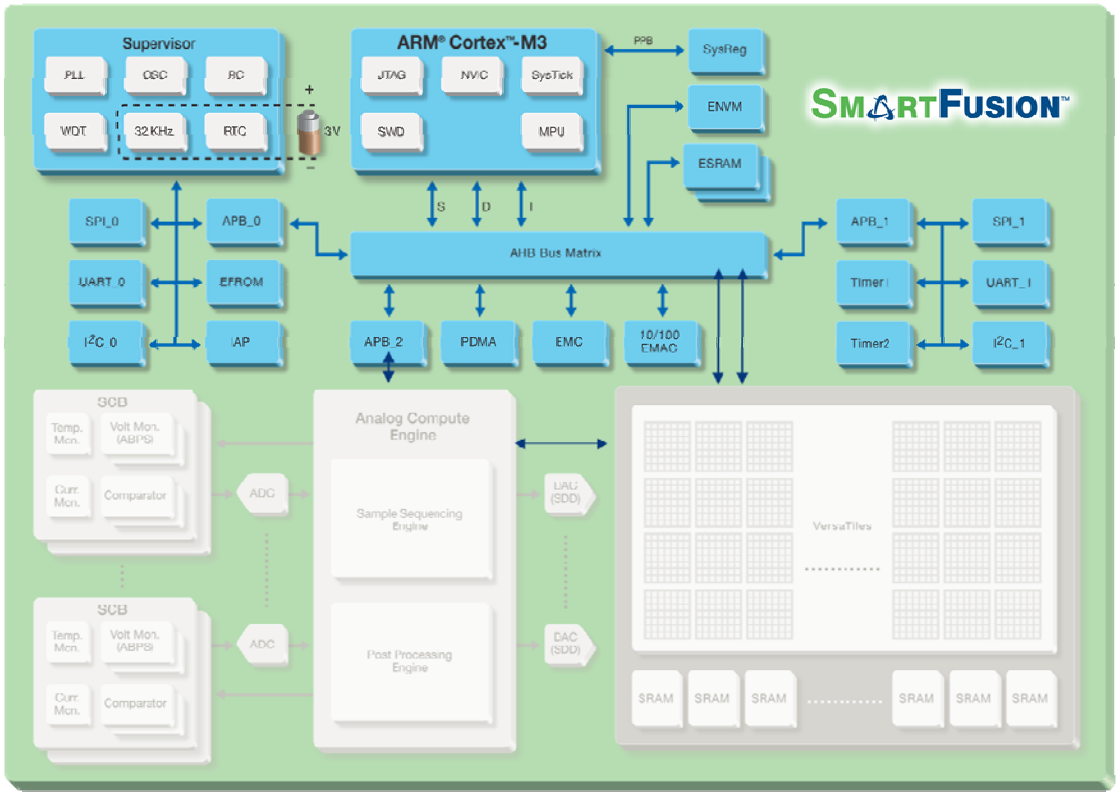


Figure 1: SmartFusion Microcontroller Subsystem

Some engineers hear the words "processor" and "FPGA" in the same sentence and think, "Oh no, not another slow, balky, configurable processor!" That's not the case here. This is no build-it-yourself construction set with a user-defined instruction set and no software support. Far from it; this is a factory-stock ARM Cortex-M3 processor core, just like millions (really) of other Cortex-M3-based chips in use all around the world. ARM offers the world's most popular 32-bit processor family. Contrary to popular belief, Intel® doesn't make 90% of the world's processors. In reality, it's closer to 2%. In contrast, about five times that many ARM-based chips go out the door every week. The Intel x86 chips may dominate the PC market, but ARM-based processors dominate just about everything else.

The advantage of all this popularity is, well, popularity. The more popular your processor, the easier it is to find software, tools and talent for it. Studies show that most engineers choose their operating system and code-development tools first, and then choose a processor to run them on. With more software, middleware, operating systems, debuggers and hardware and software development tools than you ARM processors are the de facto choice for smart developers around the world. It's easy to hire programmers with experience with ARM processors and easy to get good support, software and service for ARM-based chips. You feel like you're swimming with the stream instead of against it.

The processor hums along at 100 MHz, so you won't have to make any excuses for its performance. If you're not familiar with the Cortex-M3 processor, it's ARM's mid-range embedded processor, much newer (and faster) than the popular ARM7 but less exotic (or power-hungry) than ARM's Cortex-A8 or Cortex-A9 processors. It's a 32-bit RISC processor that's extremely well-supported with software development tools. If you're a believer in the Dhrystone benchmark, it'll deliver about 125 Dhrystone MIPS.

Unlike a lot of RISC processors, the Cortex-M3 processor has a hardware multiplier and divider, so you won't have to do these basic math functions in software. The processor also has a memory-protection unit (MPU), which is like a streamlined memory management unit (MMU). It's enough to segment and rearrange memory maps to suit your whims, and to prevent rogue code from damaging memory, but not so complex that you have to deal with page faults and translation lookaside buffers (TLB.) Naturally, the processor comes with its own memory and other resources. In this case, it's got 16 KB to 64 KB (depending on the specific SmartFusion cSoC device) of high-speed SRAM, which you'll want to use for executing code and storing local variables. It's also got anywhere from 64 KB to 512 KB (again, depending on the chip) of nonvolatile flash memory.

Initial accesses to the flash take from 5 to 7 cycles, depending on the speed of the processor. Obviously, as the clock frequency gets higher the cycle time gets shorter, so even though the number of cycles may increase, the amount of time stays the same.

SRAM accesses always take just one cycle, regardless of processor speed. This no-wait-state memory is fast, and it's divided into two banks so that two masters (the processor and the DMA, for example) can both access it simultaneously. Another clever feature is the SRAM's "bit banding" region that allows your C compiler to store single-bit Boolean flags in just a single bit of RAM, without using up a whole byte (or word). There's an external memory controller if you want to expand beyond the chip's boundaries.

Rounding out the processor area of the chip is a multitude of peripheral controllers. The set includes a 10/100 Ethernet MAC, two I^2C ports, two UARTs, two SPI interfaces, a pair of timers, and an 8-channel DMA controller. In short, the processor subsystem alone would be a $5 microcontroller all by itself.

FPGA-Based Processors

FPGA-based processors have a checkered history. Some are terrific; some less so. First of all, there are two kinds of FPGA-based processors: soft and hard. Soft processors are those that use the FPGA's own programmable logic to build the processor core. That uses up some (perhaps most) of the FPGA's logic, leaving little left over for other circuitry—which defeats the purpose of using an FPGA in the first place. More important, however, these early soft processors just didn't work very well. It turned out that microprocessors just didn't map well onto the type of logic and interconnect that FPGAs provide. They don't have the right mixture of registers, MUXes, storage, and wiring. The result was that soft PowerPC®, MIPS®, or ARM processors implemented in FPGA logic were slow and an expensive waste of valuable FPGA resources.

Following that initial experiment, the big FPGA companies started over with their own FPGA-compatible processors. These worked much better. Microsemi SoC Products Group created CoreABC, Altera designed Nios®, and Xilinx developed MicroBlaze™. In each case, the processor was designed to suit the FPGA's available resources, rather than forcing an existing processor into an unwilling FPGA. This second generation of soft processors all use FPGA resources much more efficiently, and deliver modest performance at minimal extra cost.

The other kind of FPGA-based processors are hard processors. These don't use an FPGA's programmable logic at all, but are really standard processors permanently attached to an FPGA. It's the best of both worlds: the processor gets to run at full speed and the FPGA fabric gets left alone to do what it does best. Neither the processor nor the FPGA fabric is compromised, but instead work together. Everybody wins.

The SmartFusion cSoC device family follows this latter philosophy. The FPGA fabric on the chips is entirely free for the user, while the Cortex-M3 processor is well-equipped to run at full speed with caches, buses and interfaces suited to its demands.

Analog Subsystem

“Programmable analog” sounds like an oxymoron, but the concept is elegantly simple and one that will delight both hardened analog engineers and analog-phobic digital designers.

Think of SmartFusion’s programmable analog area as an FPGA without the lookup tables (LUTs). Instead, it combines the familiar programmable interconnect with an array of analog components and building blocks. By connecting and combining different analog components, you can create just about any analog front-end, function or feature you want. There are prescalers, converters, sensors, analog/digital converters, comparators, current monitors, temperature monitors and more at your disposal.

Every SmartFusion cSoC device has at least one successive-approximation register analog-to-digital converter (SAR ADC). The converter converts voltage into an 8-, 10-, or 12-bit number (your choice) and can do this at 500 to 600 KHz (depending on the resolution). That means you can take half a million samples per second, which ought to be enough for most designs. The ADC is fed by a 16-to-1 multiplexer, the idea being that you can sample 16 different analog inputs simultaneously but you’re only going to read the digital value from one at a time.

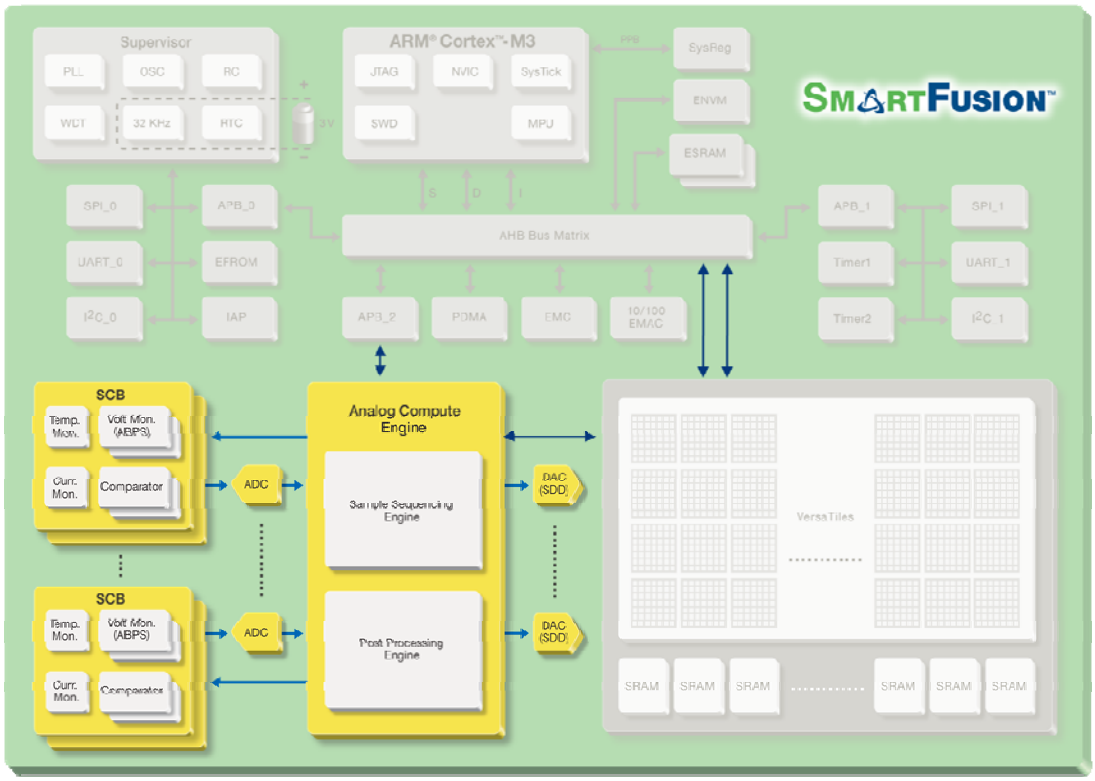


Figure 2: SmartFusion Programmable Analog

If you’re going the other way (that is, converting digital to analog), SmartFusion cSoC devices have a first-order sigma-delta DAC with effectively 12 bits of resolution. The qualification “effectively” is given because the internal architecture is based on a one-bit DAC, but the noise and resolution performance is comparable to that of a conventional 12-bit DAC. You get from one to three of these DACs, depending on the specific SmartFusion cSoC device.

The analog portion connects directly to the other two main portions of the chip: the FPGA fabric and the processor's peripheral bus (APB #2 in the block diagram). This makes good sense, but deserves a little deeper explanation.

The path between the analog circuitry and the FPGA fabric is 32 bits wide, so there's a big pipe between analog and digital logic. This makes the two areas real partners, not just acquaintances in an arm's-length relationship. Your programmable logic can be deeply integrated with your programmable analog circuitry, and vice versa. This is much better than having only an 8-bit "keyhole" peek into the analog area, squeezing signals and data through a tiny interface.

The second interface, between the analog and processor realms, also makes good sense. The processor sees the analog block as a peripheral (or several peripherals, if you wish) so it has no trouble reading and writing values to and from it. From the analog perspective, it sees the processor as a digital source or sink, much like the digital side of a DAC or ADC. In short, it works like you'd expect it to.

Where the analog portion of the chip really gets interesting, though, is in its analog computing engine (ACE). It sounds like another oxymoron, but it's actually quite impressive and a big time-saver for programmers and analog engineers alike. In brief, the analog computing engine is like a second processor that you can program, allowing the analog circuitry to follow sequences and look after itself without so much processor intervention.

For example, the ACE can automatically adjust the resolution of the ADCs sample by sample, or after a preset amount of time. You could have it automatically adjust resolution up or down over time, for instance, or delay the samples from one ADC channel to another. The ACE can also launch an analog conversion (either direction: ADC or DAC) based upon a signal coming from the FPGA portion of the chip. Conversely, the ACE can interrupt the Cortex-M3 processor when some event occurs, such as when a particular sampling level is detected or reached.

It gets better. The ACE can do its own linear transforms (of the form yy = mx + b, including calibration) or do its own digital low-pass filtering, threshold comparison or state filtering. All of this without intervention by the processor. The CPU can either go to sleep or it can do something else while the analog circuitry looks after itself.

FPGA Subsystem

FPGAs are how Microsemi SoC Products Group made its name, and for good reason. All SmartFusion cSoC devices include a generous amount of FPGA fabric, starting with 1,536 tiles of logic on the smallest A2F060 part. The biggest A2F500 device comes with 11,520 tiles. Each tile is roughly equivalent to a D-flip-flop or three-input LUT.

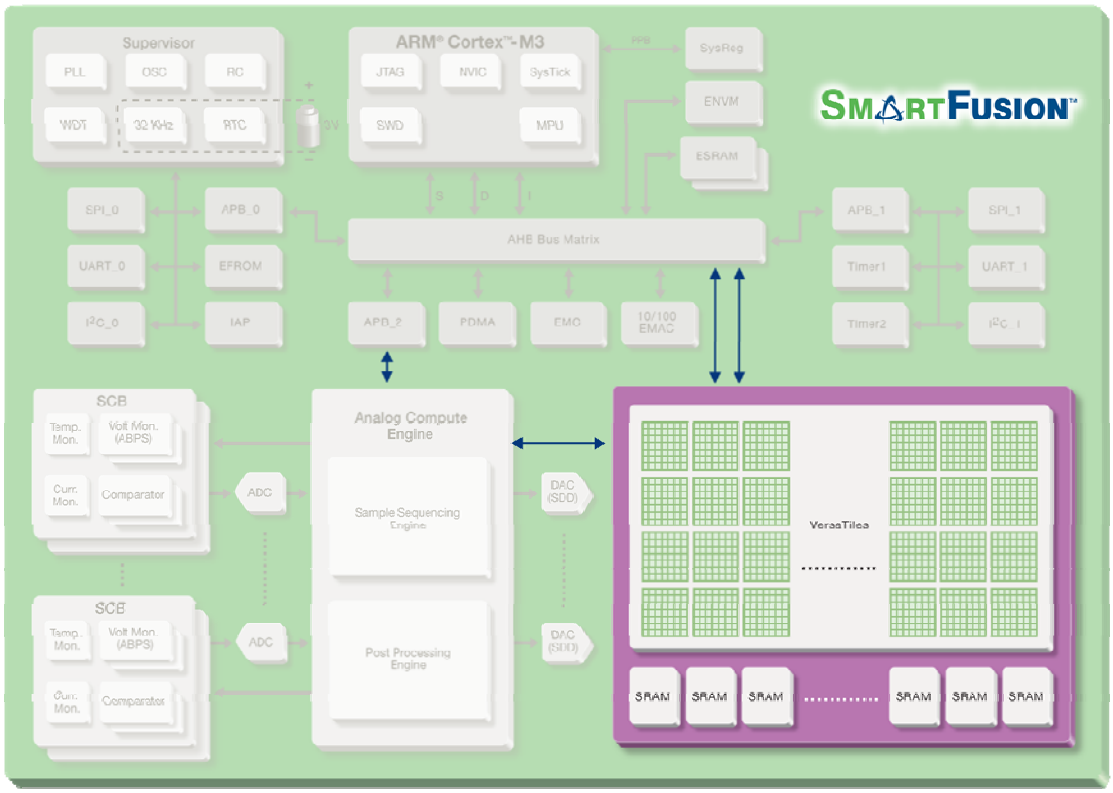


Figure 3: SmartFusion FPGA Fabric

Summary

Engineering is all about finding the most efficient solution to a given problem. Sometimes a software approach works best; sometimes it's a hardware problem; sometimes the solution lies in the analog domain, and so on. Usually it's a combination of all these things. There's no such thing as a "typical" embedded system because there are so many different engineering approaches to different product requirements.

In the US, there's an old saying that when the only tool you have is a hammer, every problem looks like a nail. We've all met inexperienced engineers who try to hammer every problem with the one tool they know, whether it's hardware, software or analog circuitry. But integrating some combination of hardware, software and analog is usually the best and most efficient solution—if only we could actually do that.

Harmonious integration makes for the best engineering solutions, but it also makes good commercial sense, too. Management cuts and reductions in engineering budgets and headcount mean fewer engineers are doing more work. Time and budget pressures certainly haven't loosened, and product life cycles tend to get shorter, not longer. Nimble flexibility is becoming a corporate requirement, and engineering flexibility can be a career-saver. Switch-hitters make great team players.

Rub the genie's lamp and, *voila!*—a perfect three-way combination of hardware, software and analog appears. Also a three-way combination of flexibility, security and cost-effectiveness. Or a three-way combination of space savings, power savings and time savings. It's hard to tell what's best about SmartFusion cSoC devices.

Could this be the only chip your system needs?

By Jim Turley

Jim Turley is the founder of Silicon Insider, a technology-business consulting firm advising chip makers, investors, legal professionals, and software companies' manufacturers in the computer industry. As a technology expert, Turley is a sought-after thought leader, consulting with investment firms, semiconductor companies, and legal professionals to provide informed advice and analysis on technology trends and market requirements, or to review strategy and technology developments around the world.

Jim Turley is an acknowledged authority on microprocessor chips, embedded systems, semiconductors, and intellectual property licensing. He is the editor of Embedded Technology Journal. He was previously the President/CEO of a publicly traded microprocessor firm and Senior vice president of another public microprocessor IP company, was editor of the prestigious industry journal Microprocessor Report (a three-time winner of the Computer Press Award), editor-in-chief of Embedded Systems Design magazine, and conference chairman of many annual industry events. Jim is a regular speaker at industry gatherings and is frequently quoted in The Wall Street Journal, New York Times, San Jose Mercury News, and other media and appears regularly on television, radio, and internet broadcasts.

He can be reached at info@jimturley.com or www.JimTurley.com, or by calling +1.831.375.8086.

Microsemi Corporate Headquarters
One Enterprise Drive, Aliso Viejo CA 92656
Within the USA: (800) 713-4113
Outside the USA: (949) 221-7100
Fax: (949) 756-0308 · www.microsemi.com

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

55900116-1/11.11