New Technologies Bridge the Gap Between Custom ASICs and ARM-Based Microcontrollers

By Mark Saunders, Product Marketing Manager Sr, Cypress Semiconductor Corp.

One perspective for thinking of the embedded world is as a spectrum of complexity ranging from ASIC to MCU. ASIC designs are enormously expensive and take years to complete but the potential for innovation is almost unbounded. MCU projects are almost free in comparison and take months or even just weeks to finish, but they are limited by the third-party chips they run on. However, as different as these approaches are technically, there are many similarities in both their end-products; they both predominantly use ARM CPU cores, include standard communications interfaces, integrate significant amounts of analog functionality on-chip, and support low power operation and fast wakeup times.

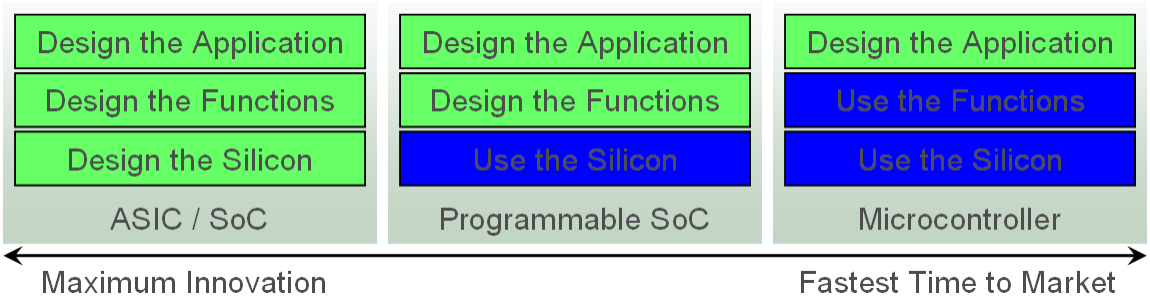

Figure 1: The spectrum of embedded design from SoC to MCU

Sitting in the middle of this spectrum are programmable platforms where design does not take place at the silicon level but still offers enormous flexibility in terms of the functions that can be integrated into a device. The obvious examples of this genre are FPGAs and CPLDs, which are fabulously powerful and have enormous capacities. These devices, however, do not epitomize the definition of a programmable platform because they are so focused on the digital domain.

To truly bridge the gap between ASIC and MCU requires a programmable platform that provides flexible in terms of both analog and digital functions without a designer having to become an expert in either discipline. Designers will want to complement that programmability with standard CPUs and their off-the-shelf tools and ecosystems. Finally, an ideal system will be perfectly suited to an application, offering all of the required peripherals and interfaces on-chip, the right level of performance, and no unneeded functions that have to still be paid for even if they are never used.

To achieve such flexibility requires a platform that enables custom configuration of high-performance analog and programmable digital logic in the SoC but without developers needing to be experts in HDL or analog design. Microcontrollers have the tools and analog functions but ultimately lack the configurability. FPGAs have configurable logic and are beginning to offer better software, but their weakness still lies in analog and true low power operation.

ARM is the De-Facto Embedded Standard

For many years now the ARM architecture has been the de-facto standard for ASIC design, as dominant in most deeply embedded markets as Intel have been in Personal Computers (and without the problem of reverse-engineered clones). After the release of the Cortex M-series processor cores a few years ago, one would be quite hard pressed to find a modern microcontroller without an ARM at its heart.

The truth of this statement can be seen from the support ARM receives throughout the embedded industry. Talk to SoC IP vendors about the first bus architecture they target for their new products: it's always ARM's AMBA. Talk to real-time operating system (RTOS) companies (those which haven't been bought by semiconductor companies) and they'll tell you ARM support is the top priority for their new products too.

Published in EE Times Design (http://www.eetimes.com/design) November 2010

Realistically, any platform that seeks success in the programmable sector simply has to use an ARM CPU. The reason for this is continuity. Engineers do not fear change. Rather, they fear aggravation and wasted time. In terms of SoC design, it is extremely difficult to lure engineers away from their traditional platforms if you don't offer the same CPU architectures, the same compilers, the same IDEs and debuggers, the same RTOS and the same middleware packages. Put simply, software rules because no one wants to port it. With the exception of really low-end products that can perform dedicated embedded functions on trusty 8-bit devices like the 8051, any programmable platform that does not offer an ARM CPU is quickly pigeon-holed into the few remaining market segments that aren't dominated by that architecture.

Add Analog Functions to Programmable Devices

The perennial downfall of programmable devices is, of course, analog functionality. While there are many platforms available that integrate significant analog in the form of high-speed communications interfaces, the true pain point that needs to be addressed is the low-level circuit functions that integrate traditionally off-chip components. After all, the physical layer (PHY) implementations found in today's FPGAs completely isolate the designer from the analog portion of their problem, presenting a standard digital interface just like any other IP block.

The real analog challenge is in implementing general-purpose functions like analog-to-digital and digital-to-analog converters, amplifiers, and voltage comparators. This is not just because analog circuits represent a tricky design problem but rather that some of the difficulties get passed onto the end user. With a digital function, for example, it is possible to place the implementation into a design, route to the appropriate I/Os, run static timing checks, and everything works as planned. Certainly timing plays a key role in design but timing issues are often less a property of the IP being integrated and more a function of the speed at which a device is being run, the overall complexity of the entire design, and the level of utilization of the part, all of which impact the available routing resources. With any analog design, however, the behavior of even simple circuits is directly impacted by configuration options, on-chip routing, and the external board design, and these are tricky to get right.

Switched capacitor blocks, for example, are powerful analog components because they can be configured in many ways, such as programmable gain amplifiers (PGA), trans-impedance amplifiers (TIA), analog filters, and even frequency mixers. However, they also create a problem for designers because their behavior is dependent upon the configuration of the block and the frequency at which the capacitors are switched. Getting that functionality into a single chip is compelling but figuring out how to make it work from a datasheet and a bunch of configuration registers is quite the reverse.

The solution to this problem is, of course, software. Getting well-behaved, high-performance analog functions into a device is one thing. But without a development tool that takes the mystery out of the configuration process, the quick route to market often entails more off-chip components than were planned for when the programmable device was first selected.

The cure for these analog woes is a software tool that offers abstraction and parameterization by supporting devices with configurable analog blocks and abstracting implementation details away from how the function is presented to the designer. Put another way, just as a designer does not need to understand how an old-fashioned ADC chip works, nor should they have to learn an integrated device's register names and bit-fields just to set up an on-chip ADC. Rather than getting bogged down calculating clock rates and messing with R and C values, developers need to be able to configure an ADC based on its features and characteristics such as available resolution, maximum sample rate, voltage range, and so on.

Once the ADC is selected for a design, the next step is to tailor it to the exact needs of an application. Ideally, this process is achieved using parameters. If an ADC supports a range or fixed set of values for, say the input range (the band of voltages that will not cause the ADC to saturate), then these choices should be presented in an intuitive manner and the decision achieved through the single action of selecting the desired parameter value.

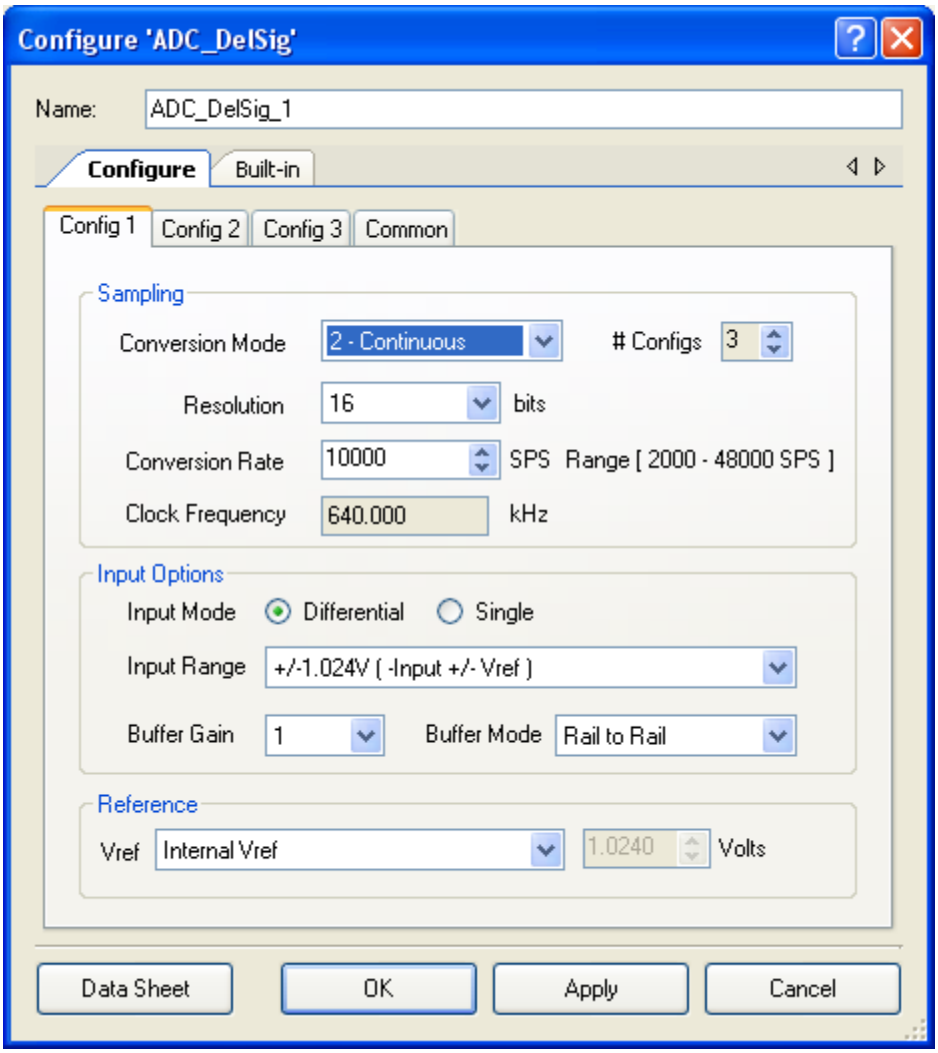

Figure 2: Tools like PSoC Creator from Cypress Semiconductor enable the design of programmable analog circuitry through the selection of the desired functionality and avoid implementation details

An example of a tool that does this for a programmable platform is PSoC Creator from Cypress Semiconductor. PSoC Creator supports the PSoC 3 and PSoC 5 devices with a schematic capture interface that lets the user draw the design they need and configure the selected parts through user-centric parameter editors. Pre-built analog (and digital) components are presented to the user in a catalog, where they have access to example projects and data sheets, just like an off-chip ADC. When a component is instantiated in a design, the tool generates APIs for the application to interface to it without the need to decode register sets or worry about the order and timing of the ADC setup code.

Integrate analog and digital design with software development

While solving the programmable analog problem with schematic capture of analog components is effective, it is not the whole solution. Developers also need a tool that supports digital design too and, possibly more importantly, the software application.

Schematic capture is by no means new in digital design, and platforms that support the integration of digital and analog design into a single device are becoming more popular. Using multiple tools to complete one design, however, is not appealing to developers. Developers would rather be able to draw a combination of digital and analog design elements in a single editor and build, debug, and test such designs from the same environment.

Vendors of programmable platforms need to remember that, in the MCU world, designers live and breathe design in software-centric Integrated Development Environments (IDEs) that bring together source editing, project management, compilation tools, and debugging together in a single framework. The same is basically true – for application development – in the ASIC world where the chip designers and software developers are rarely a single team. Neither of these groups are going to get excited about losing the convenience of their IDEs and adding new tools to solve problems they never had to worry about before. Both teams need a modern IDE that looks and behaves like the ones they are used to and they need even more compelling features to justify the change.

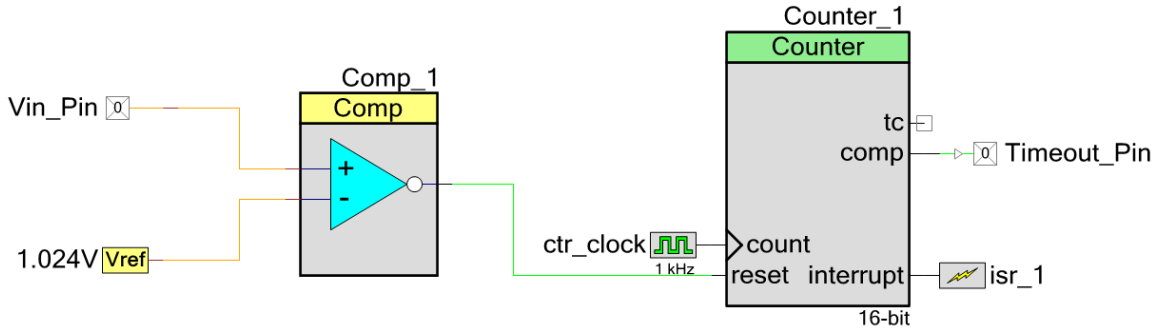

Figure 3: An over-voltage timer circuit integrates an analog comparator seamlessly with a digital counter.

Some ideas can be illustrated through an example of an over-voltage timer which uses an analog comparator and a digital counter to monitor an input voltage. If the voltage on the pin exceeds the reference, then the comparator starts the counter which, after a pre-determined time (set by a parameter, naturally), signals an error state on a pin and triggers an ISR. This simple example integrates digital and analog design effortlessly while demonstrating other functions that are difficult to complete if a developer is designing a device from scratch or working within the constraints of an MCU; namely, the ability to just “drop in” pins, clocks, voltage reference, and interrupts.

When this design is built, the environment generates APIs for the components. This allows the developer to use these components without having to expose the implementation. With APIs, for example, a developer can change the timeout period of a counter or poll its value, disable the interrupt, turn off the clock, and so on, all without consulting the reference manual for the device or hacking away at sample code. By enabling hardware setup from the schematic, a programmable device offers a real advantage over fixed-function silicon because the tool provides all the setup code automatically and make it a straightforward process to interface to all the on-chip functions.

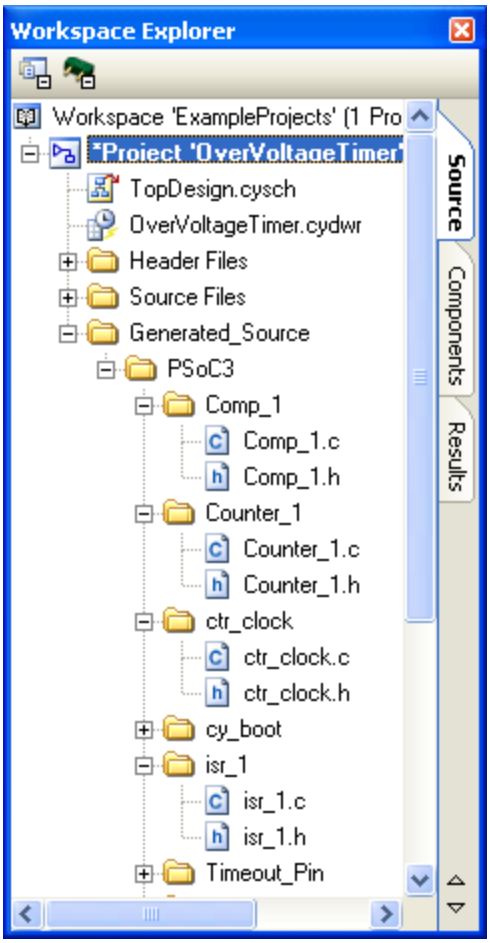

Figure 4: The PSoC Creator Workspace Explorer window showing the generated source files for the comparator, counter, clock and interrupt service routine

API generation is really just an extension of the abstraction idea from hardware-setup parameters to software. To set up a clock, for example, a developer just has to drop it into the design and ask for the desired frequency. The tool figures out how to derive the appropriate frequency, within tolerance, from the available on- and off-chip clock sources. To start and stop the clock requires a simple call to a cunningly named API like ctr_clock_Start() or ctr_clock_Stop(). There is never a need to modify registers to select the clock source, set a divider, choose it as the input to the on-chip function, or to enable/disable it through cryptic bit manipulation.

A tool that generates APIs for system resources like clocks, interrupts, DMA, and pins is a major time-saver and, likewise, APIs for on-chip analog, digital and communications peripherals make developing for a programmable device as easy, or easier, than for MCUs or ASICs. When combined with the most popular embedded ARM cores, an integrated schematic capture tool represents an attractive alternative to both ends of the design spectrum. As the programmable silicon gets better, with greater capacity enabling ever more innovation, more powerful (ARM) cores to support it, and powerful analog functions reducing overall chip count, it seems clear that the software supporting the platform will determine whether it will succeed in the market or not. The everyday development tool is the key to unlocking the silicon potential and expanding the range of designs that can be supported but, most importantly, it will win fans among embedded engineers who are just doing what they always do: looking for a better way to solve difficult problems.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com