Class 330

Configurable Embedded Systems:
Using Programmable Logic to Compress Embedded System Design Cycles

Steven K. Knapp
Vice-President, Applications
sknapp@triscendcorp.com

Arye Ziklik
Director, Product Development
arye@triscendcorp.com

Triscend Corporation (Booth 5010)
Mountain View, CA 94043
Web: www.triscendcorp.com

INTRODUCTION

Programmable logic provides configurability to embedded systems designs. Such systems exploit programmable logic's flexibility and increasing cost effectiveness, as well as the growing availability of intellectual property (IP) cores, to deliver embedded products to market faster. Programmable logic also offers advantages in higher integration, increasing performance, and improving system debugging. Deciding to create configurable embedded systems using programmable logic should be done strategically, with an understanding of the latest technologies, design trade-offs, and industry trends.

Only One Thing Remains Constant: Change

So why build a configurable system? Adaptability is essential to survival in constantly changing environments. A product must react to new markets, specification changes, customer requirements, and competitive threats. For most embedded systems, adaptability means using programmable technologies, such as FLASH memory and programmable logic.

Programmable logic itself offers new capabilities. With their flexible architectures, programmable logic devices like ***CPLD***s (Complex Programmable Logic Devices) and ***FPGA***s (Field-Programmable Gate Arrays), and the emerging class of configurable processors allow a designer to explore new design solutions. The

combination of a microcontroller and programmable logic proves to be a powerful mix. A microcontroller efficiently performs complex, general-purpose processing using a minimum amount of logic, though not always with quick response. Programmable logic, however, is extremely good at fast response though not good at efficient, general-purpose processing. The combination allows a designer to offload time-critical or fixed, computationally intensive functions from the processor and into the programmable logic.

The architectural flexibility of programmable logic also allows the designer to circumvent shortcomings of a particular processor. Need additional I/O, timers, or interrupts in your microcontroller? Supplement the MCU using programmable logic.

The configurable nature of MCUs, FLASH, and programmable logic are an advantage throughout the entire product cycle. During development, configurablility aids debugging. Buried signals can be brought out to I/O pins. Design changes can be made in a matter of minutes. During manufacturing, production boards can be tested and configured as part of the production flow, reducing manufacturing steps and inventory. If designed accordingly, configurable embedded systems can be modified, even after being deployed in the field. New features or updates can be shipped to a customer, much like software.

Key Design Considerations

Embedded system designers confront demands from four driving forces, shown in Figure 1.

Reducing time to market. Time-to-market is loosely defined as the time required to develop, test, manufacture, and deploy an application to the end user. Hardware flexibility contributes to time to market. How easy is it to use and modify the hardware? Does it accommodate change? For example, does it adapt to ever-changing I/O requirements? Software adds to the time to market figure in a different way. Are there assemblers and compilers for the hardware? Are there software engineers that have experience using the processor? For some technologies, are there cores available that implement desired peripheral functions? Lastly, is there an effective debugging capability to find and eliminate errors?

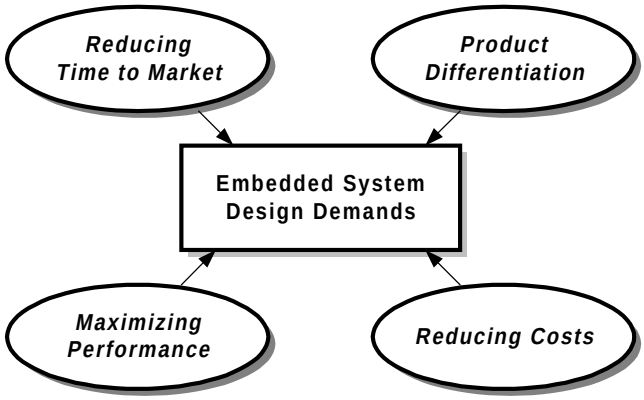


Figure 1. Four key forces drive embedded system demands.

Maximizing system performance. The overall performance of an embedded system depends on a few items. Some applications are constrained by the processor's architecture. Programmable logic offers some advantages by potentially offloading time-critical functions from the processor. The actual performance depends on how well the function integrates with the MCU and the intercommunication between the MCU and programmable logic. Other designs are limited by the processor's fabrication technology. Oftentimes, older processors are stuck on obsolete manufacturing processes and only support slower clock frequencies.

Differentiation from competitors. Like the old adage, "If it were easy, everybody would be doing it," standard, fixed-architecture devices offer little product differentiation from your competitors, except through software. Programmable logic provides significant differentiation while still providing off-the-shelf availability.

Reducing overall product costs. Many factors contribute to a product's cost. Those within the control of engineering include design and development costs and component cost. Technologies that reduce time to market usually also help reduce development cost. For example, a processor well supported by software is typically faster and easier to use in an application. This correspondingly reduces design time and its associated cost. Some technologies promising fast time to market may have higher component costs. For example, a programmable logic device is more expensive than a gate array. However, the cost to develop a gate array can quickly overwhelm component costs for all but the highest-volume applications.

CONFIGURABLE EMBEDDED SYSTEM DESIGN

System Design Trends

Like everything else in the digital world, embedded systems benefit from the constantly shrinking transistor. Improved process technologies create faster and denser devices at nearly a constant cost, as shown in Figure 2. In the early 1980's, discrete solutions dominated embedded systems design. A standard ***MCU*** (microcontroller unit) was the centerpiece and TTL devices or ***PLD***s (Programmable Logic Device) provided the glue logic between the MCU and its discrete peripherals. Even the first personal computers consisted of primarily discrete components.

In the late 1980's to the mid-1990's, MCU companies began integrating combinations of peripheral functions into a microcontroller creating a ***derivative*** device, often driven by high-volume customers or applications niches. Some derivatives even integrated additional system RAM and the boot PROM. Most of the system glue logic disappeared into programmable logic devices, either ***CPLD***s (Complex Programmable Logic Device) or ***FPGA***s (Field-Programmable Gate Array). Some discrete peripheral devices still exist. However, designers are beginning to implement some peripherals in programmable logic as CPLD and FPGA densities increase. Designers in high-volume or high-performance applications integrated most of the system components in a single ***ASIC*** (Application-Specific Integrated Circuit) or gate array.

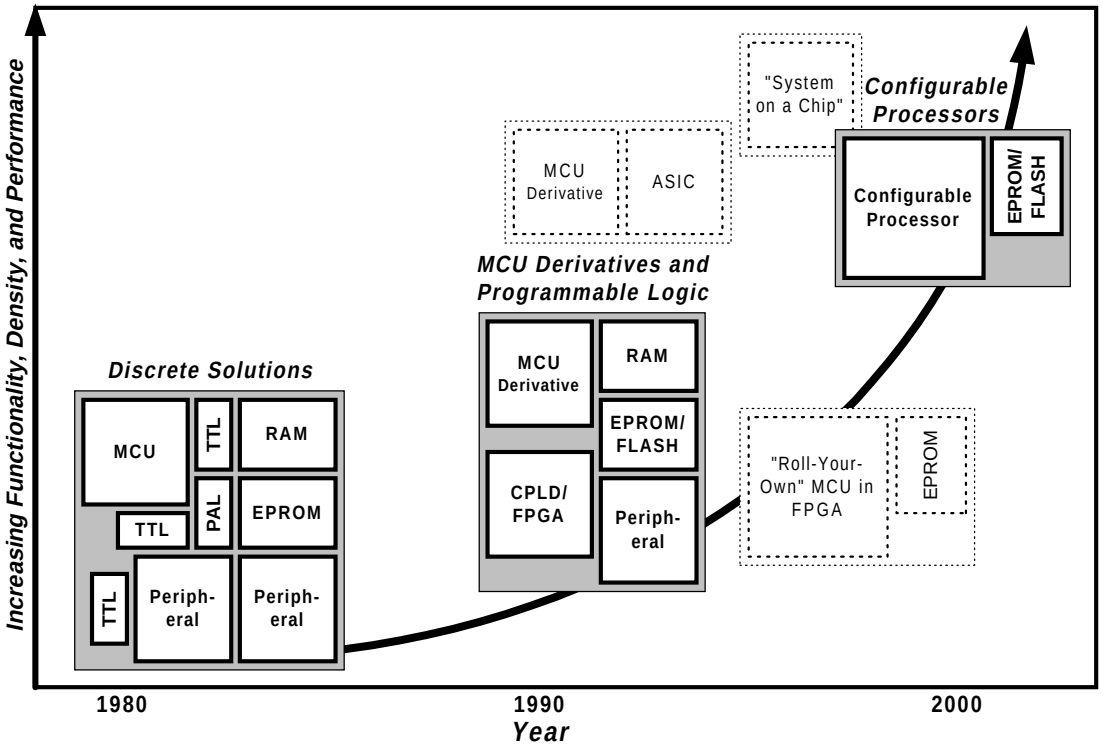


Figure 2. As embedded system design evolves, it tends to integrate functions into new programmable technologies of ever-increasing density and performance.

Others trends in the programmable logic and intellectual property (IP) arenas further shaped embedded systems development. FPGA and CPLD capabilities and performance increased, allowing them to all but implement entire systems on a single piece of silicon. The IP providers had been developing cores, such as UARTs, timer/counters, etc. for the ASIC or gate array markets. With improved programmable logic densities, IP providers began "porting" their core offerings over to FPGAs.

In the late 1990's, higher-volume designers began to integrate entire processor systems—including the MCU—into a single, large ASIC. This technique is sometimes referred to with the marketing buzzword, ***system-on-a-chip***. These designers could afford the additional time, development cost, and risk associated with semi-custom design. The disadvantage of ASICs, for most designers, is that they are inflexible. Metal-masked devices are cost-effective for high-volume applications.

Several companies have recently announced, or announced development plans for, a new class of device called a ***configurable processor***. Similar to its ASIC counterpart, a configurable processor integrates a microprocessor or MCU with its associated peripherals. Unlike the ASIC, a configurable processor is changeable, often with its configuration program stored off-chip in a standard byte-wide EPROM or FLASH device.

Comparing Today's Design Alternatives

A designer starting a new embedded systems design confronts various implementation options. Choosing the best method depends on the underlying application. No solution is universal for all design demands. The chart in Table 1 lists the advantages and disadvantages of each approach and the solution's fit in terms of time to market (TTM), improved performance (PERF), reduced cost (COST), and increased differentiation (DIFF).

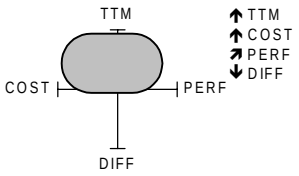
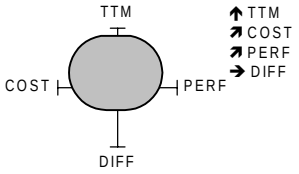
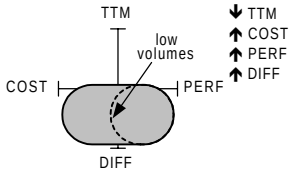
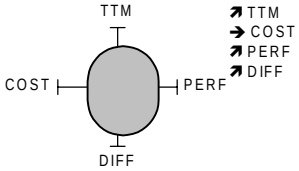
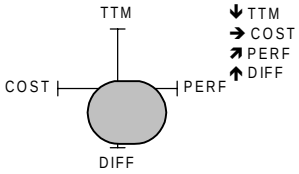
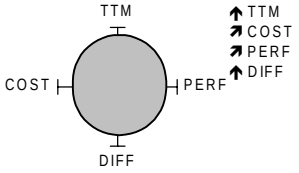
Stand-alone, standard microcontroller. A single, stand-alone MCU is often sufficient for simple designs. It offers fast time to market and significant cost advantages if the MCU is a popular, high-volume device. However, product differentiation is accomplished via software only. Often, older MCUs are built on aging process technologies, limiting overall performance.

Microcontroller derivative. Microcontroller derivatives offer improved integration, usually with extra peripherals and possibly on-chip memory (EPROM, FLASH, RAM). Derivatives have some of the same advantages and disadvantages of a stand-alone MCU. However, derivatives typically have weaker software support. Furthermore, smaller-volume derivatives suffer notoriously bad availability problems resulting in long lead-times.

Microcontroller with ASIC support logic. Some higher-volume applications integrate the microcontroller's support logic in a supporting ASIC or gate array. This solution offers extremely high product differentiation with excellent potential performance and low costs in high-volume manufacturing. However, ASICs require extensive design simulation and test vector creation before manufacturing. There are also non-recurring engineering (NRE) charges for mask making and production plus minimum volume requirements by the manufacturer. In addition, there is little software support for custom functions implemented in the ASIC. Debugging an ASIC-based design is challenging unless extra debug logic is included in the design. Also, many of the peripheral functions must be designed from scratch or purchased as an IP core, adding to the design cost.

Microcontroller with CPLD/FPGA support logic. This option provides some of the same benefits as the ASIC solution. Time to market concerns are dramatically reduced because CPLDs and FPGAs are standard, programmable, off-the shelf devices. Currently, there are few IP cores available for CPLDs and FPGAs so custom design may add to development time and expense. Component costs are usually good for all but higher-volume applications.

Table 1. Comparing Today's Embedded System Design Alternatives.

Solution/Fit	Pros	Cons
Standard, stand-alone MCU 	Good for simple designs Fast time to market Off-the-shelf availability Strong software support System-wide debugging Low cost for popular MCUs	Limited number of I/O Little product differentiation Performance limited on older MCUs
MCU Derivatives 	Fast time to market Integrated peripherals, memory Off-the-shelf availability	Limited number of I/O Little product differentiation Weaker software support Long lead-times, poor availability for some derivatives
MCU + ASIC 	Increased I/O Superior performance Excellent differentiation Architectural flexibility Low cost in high volumes	Complex design/verification Manufacturing lead-times No modifications possible Little software support High cost in low volumes Poor debugging support
MCU + CPLD/FPGA 	Good time to market Increased I/O Off-the-shelf availability Good performance High differentiation Architectural flexibility Good cost in moderate volumes	Moderately complex design Limited software support Performance limited by intercommunication delay Fair debugging support
Roll-Your-Own MCU in FPGA 	Increased I/O Excellent differentiation Higher performance for some applications Architectural flexibility	Complex design/verification No software support Performance limited by FPGA architecture Poor cost benefit for standard processors Poor debugging support
Configurable Processors 	Fast time to market Increased I/O Integrated peripherals, memory Off-the-shelf availability Strong software support System-wide debugging Good performance High differentiation Architectural flexibility Good cost in moderate volumes	Processor centric solution Moderately complex design

TTM=improved time to market, PERF=higher performance, COST=lower cost, DIFF=increased differentiation, ↑=excellent, ↓=poor

System performance sometimes suffers due to poor data communication between the MCU and the logic implemented within the CPLD/FPGA. Functions within the CPLD/FPGA do not always integrate well with the MCU. Some require additional decoding, bussing, and control logic. Nearly all CPLDs and FPGAs lack a dedicated internal bus for efficient signal distribution. In addition, decoding address signals taxes the narrow fan-in blocks found in most FPGAs. Some hybrid devices, like WSI's PSD devices, contain CPLD or FPGA logic with an integrated, dedicated MCU interface. The ability to integrate a new function into MCU's software flow also becomes an issue. How will the MCU read and write to functions implemented in programmable logic?

Roll-Your-Own MCU with FPGA. With the ever-improving performance, density, and cost of FPGAs, adventurous designers have opted to create their own MCU and support logic in a single FPGA. While offering the ultimate in differentiation plus performance benefits for some applications, a custom MCU in FPGA presents some significant challenges. The designer must create the design and bus structure from scratch or purchase an MCU core from an IP provider. Even pre-canned solutions have weak assembler or compiler support. Some designers choose to implement standard processors, such as a subset of an 8051, but at extreme cost compared to an off-the-shelf solution. On the same process technology, a processor implemented in FPGA consumes an estimated ten to fifteen times more silicon than a dedicated implementation. This approach requires substantial verification for complex processors.

Configurable processors. Configurable processors are an emerging and important new solution for embedded system designers. Configurable processors are standard, off-the-shelf devices that provide a silicon-efficient dedicated embedded processor, usually well supported by software tools. An internal, high-performance bus offers high-bandwidth communication between the processor and dedicated or programmable peripherals available on chip. The integrated programmable logic allows a designer to specify and efficiently use a variety of application-specific peripherals. The pre-designed bus reduces development time and dramatically simplifies verification. On-chip breakpoint hardware provides extensive debugging support.

CONFIGURABLE EMBEDDED SYSTEMS TECHNOLOGIES

There major configurable embedded systems technologies available today include

- Complex Programmable Logic Devices (CPLDs),
- Field Programmable Gate Arrays (FPGAs),
- Hybrid Devices (CPLDs or FPGAs with an integrated microcontroller interface),
- Configurable Processors

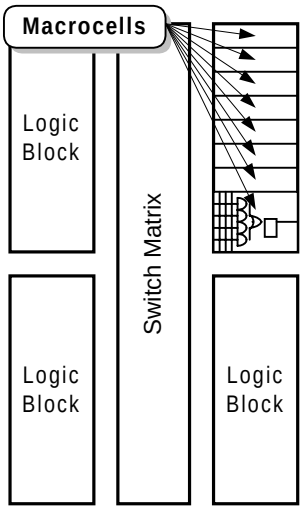


Figure 3. A typical CPLD device contains numerous macrocells, combined into larger logic or function blocks, interconnected with a programmable switch matrix.

CPLD - Complex Programmable Logic Device

A CPLD is much like integrating multiple PALs or PLDs all on a single chip, as shown in Figure 3. A typical CPLD is the equivalent of four to 32 simple PALs, typically containing from 18 to 256 macrocells, which typically equates to a board-full of 7400-series TTL devices. Each CPLD macrocell contains a flip-flop and a wide combinatorial logic function that efficiently implements sum-or-products functions. Usually, each macrocell has an associated device pin.

A group of eight to 16 macrocells is grouped together into a larger logic block. The macrocells within a logic block are usually fully connected. The logic blocks are further interconnected to one another through some form of switch matrix. Not all CPLD switch matrices are fully connected, depending on the vendor and product family.

CPLDs are generally best for control-oriented designs, due in part to their fast pin-to-pin performance. The wide fan-in of their macrocells makes them well suited to complex decoding and high performance state machines.

Generally, modern CPLDs are CMOS and use non-volatile memory cells such as EPROM, EEPROM, or FLASH to define their functionality. Many of the most-recently-introduced CPLD devices use re-programmable EEPROM or FLASH technologies, and are designed to be programmed in-circuit (also called ISP for in-system programmable).

EEPROM and FLASH processes are erasable technologies. However, not all EEPROM- and FLASH-based can programmed while soldered to the circuit board. In-system programmability (ISP) requires special on-chip circuitry to boost internal programming voltages. Devices lacking the on-chip programming circuitry require a device programmer to be programmed and erased. Those with the programming circuitry

can be programmed in the system, either via a proprietary interface or a standard JTAG (IEEE 1149.1) interface.

Table 2. Vendors of in-system programmable (ISP) CPLDs.

Altera
Atmel
Cypress
Lattice
Philips
Vantis
Xilinx

In-system programmable CPLDs are recommended for configurable embedded systems because they can be modified, even when soldered to a printed circuit board. Vendors supplying such CPLDs are listed in Table 2.

FPGA - Field Programmable Gate Array

FPGAs are a distinct from CPLDs and typically offer the highest logic capacity, plus the greatest number of flip-flops and I/O. An FPGA consists of an array of logic blocks, surrounded by programmable I/O blocks, and linked with programmable interconnect, as shown in Figure 4. A typical FPGA contains from tens to thousands of logic blocks and an even greater number of flip-flops. Sophisticated software places the

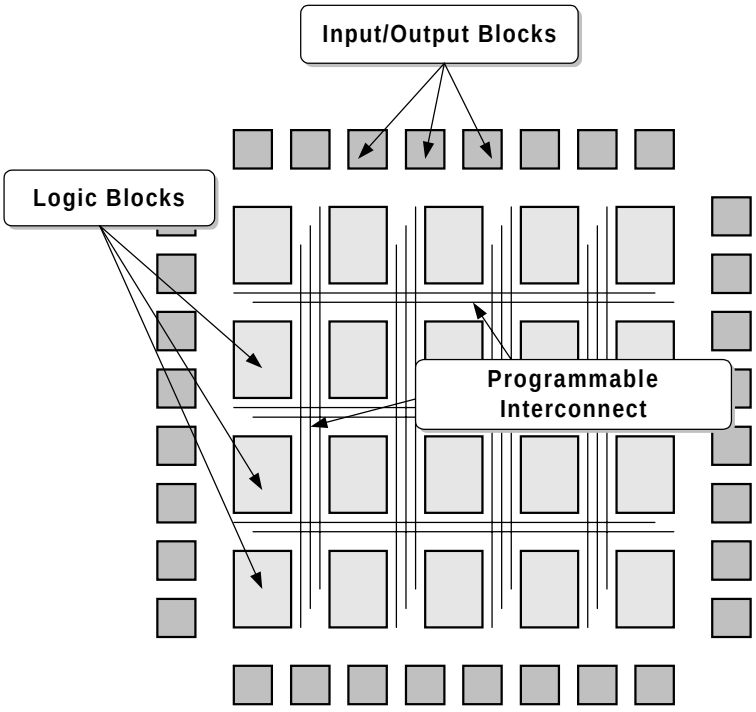


Figure 4. An FPGA consists of a core of programmable logic blocks, surrounded by programmable I/O blocks, interconnected with programmable wiring.

Table 3. FPGA vendors supply different architectures and process technologies.

	Process Technology		
Architecture	**SRAM**	**Anti-Fuse**	**FLASH**
Coarse-Grained	Altera Atmel DynaChip Lucent Vantis Xilinx	QuickLogic	
Fine-Grained	Actel Atmel	Actel	Gatefield

user's logic and routes the interconnections on the device much like a PCB autorouter would place and route components on a board.

There are two primary classes of FPGA architectures—coarse-grained and fine-grained, as listed in Table 3. Coarse-grained architectures are the most popular and consist of larger logic blocks, often containing two or more look-up tables capable of building complex logic functions and two or more flip-flops. The larger logic block usually results in improved performance, especially as interconnect delay becomes more significant with shrinking process geometries.

Most modern coarse-grained architectures also offer advanced features like internal bi-directional bussing, fast arithmetic logic to boost the performance of adders and counters, and on-chip RAM for building small register files or FIFOs.

The other architecture type is called fine-grained. In these devices, there are a large number of relatively simple logic blocks. The logic block usually contains a flip-flop and either a two-input logic function or a 4-to-1 multiplexer. These devices are good at systolic and shifting functions and have some benefits for designs created by logic synthesis.

Another difference in architectures is the underlying memory technology used to manufacture the device. Currently, the highest-density FPGAs are built using static memory technology, similar to most microcontrollers and microprocessors. The other common process technology is called anti-fuse, which has benefits for more plentiful programmable interconnect because of its small cell size.

SRAM-based devices are inherently re-programmable, even in-system. However, they require some form of external, configuration memory source. The configuration memory holds the program that defines how each of the logic blocks functions, which I/O blocks are inputs and outputs, and how the blocks are interconnected together. The FPGA either self-loads its configuration data or an external processor downloads the data into the FPGA. When self-loading, the FPGA addresses a standard byte-wide PROM—much like a microcontroller addresses its boot PROM—or it uses a special sequential-

access serial PROM. When downloaded by a processor, the FPGA appears much like a standard microcontroller peripheral. The configuration time is typically less than 200 ms, depending on the device size and configuration method.

In contrast, anti-fuse devices are one-time programmable (OTP). Once programmed, they cannot be modified, but they also retain their program when the power is off. Anti-fuse devices are programmed in a device programmer either by the end user or by the factory or distributor. Again, in-system programmable technologies are an advantage in configurable embedded systems.

Hybrid Devices - FPGA or CPLDs with microcontroller interface

A few vendors offer FPGA or CPLD devices with an integrated microcontroller interface. Most popular among these is the WSI PSD (Programmable System Device) family, which integrate a small amount of CPLD logic, up to 2Kbytes of RAM, and up to 128Kbytes of EPROM. Some variants include additional peripherals such as timer/counters and power supervisory functions.

Only a few FPGA devices offer dedicated interfaces. Lucent Technology's ORCA 3 FPGA family offers a rudimentary, glueless microcontroller interface to Power PC and i960 processors. Other devices, such as the Xilinx XC6200 family provide a memory-mapped interface to the FPGA's internal configuration storage.

Configurable Processors

A configurable processor is a highly integrated device, combining a dedicated processor and programmable logic in a single, configurable device. A configurable processor incorporates most—if not all—of an embedded system’s logic. While each vendor's product family offers some unique features, most contain

- A silicon-efficient, dedicated processor
- Dedicated peripherals attached to the processor or to the internal bus, including such functions as additional timer/counters, UARTs, or DMAs
- Programmable logic using CPLD- or FPGA-style technology designed to implement programmable peripherals or other user-defined functions
- A dedicated, high-performance bus providing high-bandwidth communication between the processor and its peripherals, allowing easy MCU integration of new functions built from programmable logic
- Abundant programmable I/O pins, many more than even the most I/O-rich microcontroller

Configurable processors have the good assembler, compiler, and debugging support of a stand-alone MCU. Compared to the 'MCU plus CPLD/FPGA' or 'Roll-Your-Own' approaches, the predefined and tested bus architecture dramatically reduces system design and verification time. Likewise, the configurable processor’s dedicated bus improves system performance by reducing intercommunication delays between the

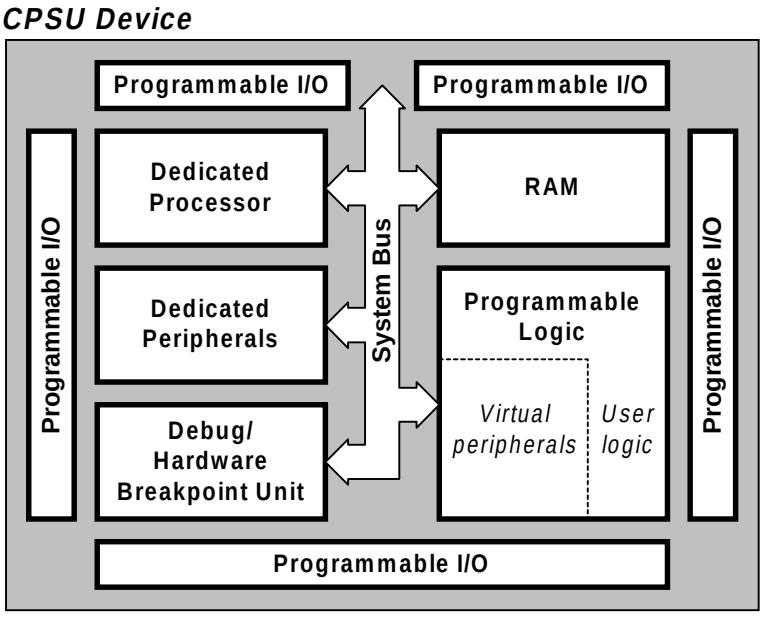


Figure 5. A configurable processor integrates a dedicated processor, its peripherals, an on-chip bus, and programmable logic all onto a single pre-tested device.

processor and any peripherals implemented in programmable logic. The bus also provides easier integration of programmable peripherals with the MCU and its development software.

Compared to a fixed-architecture stand-alone processor, a configurable processor offer incredible architectural flexibility and significantly more programmable I/Os. A configurable processor's integration reduces board size, power consumption, and EMI emissions.

Configurable processors are a new, emerging category of devices with several companies announcing devices or development plans. Siemens plans to integrate their 32-bit TriCore processor with Gatefield's FLASH-based, fine-grained FPGA family. No further details are yet available.

Motorola planned to integrate the 32-bit ColdFire processor, peripherals, and their MPA fine-grained FPGA technology in a new product family called CORE+. Unfortunately, Motorola's CORE+ efforts died when Motorola decided to exit the FPGA business due to their long-suffering FPGA market share.

National Semiconductor, with funding from DARPA, is developing a device with a 32-bit RISC processor and a fine-grained FPGA technology acquired through the now-defunct Concurrent Logic. National's technology, called NAPA1000, is targeted toward reconfigurable computing and National has not yet revealed their commercial plans.

Triscend Corporation is developing various families of configurable processors based on popular, industry-standard microcontrollers and processors. Their devices feature a high-performance, integrated bus and SRAM-based, coarse-grained programmable logic. Additional interface logic simplifies the design and improves the performance of ***virtual peripherals*** implemented in programmable logic.

SUMMARY

Configurable embedded systems can be built from off-the-shelf programmable technologies such as microcontrollers, FLASH, programmable logic, and configurable processors. Configurable systems provide time-to-market, performance, and integration benefits. Configurable processors are an emerging class of devices that promise to accelerate this trend.

USEFUL SITES & MORE INFORMATION

FPGAs/CPLDs/Configurable Processors

"The Programmable Logic Jump Station":
`http://www.fpga-site.com`

Programmable ASICs (from **Application-Specific Integrated Circuits** by Michael John Sebastian Smith):
`http://spectra.eng.hawaii.edu/~msmith/ASICs/HTML/Book/CH04/CH04.htm`

"Roll-Your-Own" MCUs

"Homebrewing RISCs in FPGAs", presentation by Jan Gray:
`http://www3.sympatico.ca/jsgray/sld001.htm`

Embedded System Conference, Fall 1998, Related Classes

Class 509 & 529: "An Introduction to FPGA Design", Bob Zeidman

Class 470: "Prototyping Embedded Microcontrollers in FPGAs", Eric Ryherd

Class 402: "Roll Your Own RISC", Tom Cantrell