Class 413

Rapidly Developing Embedded Systems Using Configurable Processors

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

Triscend Corporation (Booth 160)
Mountain View, CA 94043
Web: **`www.triscend.com`**

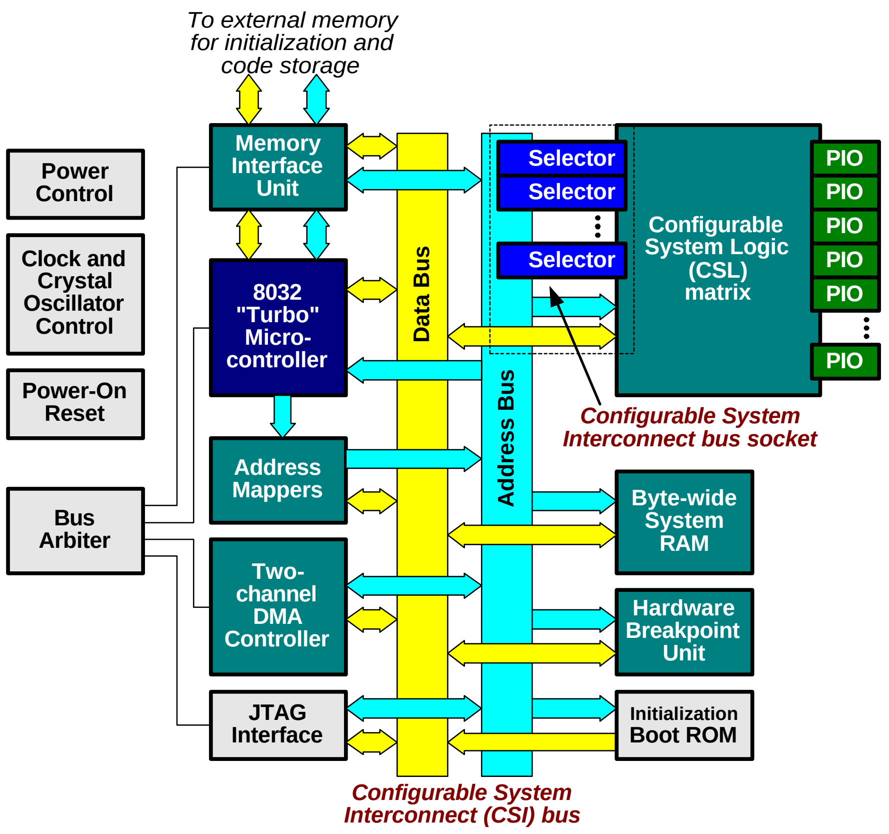
CONFIGURABLE PROCESSORS

A Configurable Processor is a single-chip combination of a microprocessor or microcontroller, programmable logic, memory, and a dedicated system bus. The programmable logic quickly implements user-defined peripheral functions, creating a customized processor derivative. The user-programmable nature of Configurable Processors minimizes time-to-market while offering a high degree of product differentiation.

In many ways, Configurable Processors are the next evolutionary step in embedded system component technology. Microcontrollers combine processor and peripheral functions on a single chip, but allow little product differentiation. A single gate array or large programmable logic device might implement multiple custom peripherals, but must still interface to a separate processor. Embedding the gate array or programmable logic on a single-chip with the processor represents the next integration step, allowing product differentiation in a single-chip solution. The programmable-logic-based Configurable Processor is more effective than the gate-array-based embedded array in applications where fast time-to-market is imperative or production volumes do not justify a gate array solution.

Configurable Processors are highly-integrated devices capable of incorporating most, if not all, of an embedded system's logic. The basic functions included within most Configurable Processor are described as follows.

- A silicon-efficient, dedicated processor—often using an "industry-standard" instruction set to leverage readily-available third-party development tools and legacy code.
- Dedicated peripherals attached to the processor or to the internal bus, including functions such as timer/counters, UARTs, or DMAs.



To external memory for initialization and code storage
Power Control
Clock and Crystal Oscillator Control
Power-On Reset
Bus Arbiter
Memory Interface Unit
8032 "Turbo" Micro-controller
Address Mappers
Two-channel DMA Controller
JTAG Interface
Data Bus
Address Bus
Selector
Selector
Selector
Configurable System Logic (CSL) matrix
PIO
PIO
PIO
PIO
PIO
PIO
Configurable System Interconnect bus socket
Byte-wide System RAM
Hardware Breakpoint Unit
Initialization Boot ROM
Configurable System Interconnect (CSI) bus

Figure 1. A block diagram showing the functions available within the Triscend E5 configurable processor. In the E5, the embedded 8032 "Turbo" microcontroller communicates to its programmable logic resource (CSL) via a dedicated internal bus structure [1].

- On-chip RAM for storing local data or code.
- Programmable logic designed to support user-defined peripherals or other functions.
- A method to communicate between the processor and the programmable logic. Some devices have a dedicated, high-performance internal bus providing high-bandwidth communication between the processor and its peripherals, allowing easy MCU integration of new functions built from programmable logic. Other architectures memory-map configuration bits within the programmable logic allowing the embedded processor to "peek and poke" to hardware functions.
- Abundant programmable I/O pins, many more than even the most I/O-rich microcontroller
- Dedicated debugging hardware such as a breakpoint unit and JTAG interface.

Table 1. Available and Planned Configurable Processors (December 1998)

Vendor/ Family	Processor	Status	Dedicated Resources	Programmable Resources	Embedded Bus Structure
Triscend/ E5	8032 "Turbo"	Sampling	2-channel DMA 8K-64K bytes RAM Hardware debug JTAG	Triscend Coarse-grained, bus oriented	8-bit Data 32-bit Address
Triscend	ARM 7TDMI	In Development		Triscend Coarse-grained, bus oriented	32-bit Data 32-bit Address
Motorola/ CORE+	ColdFire	Cancelled	2-channel DMA 3K bytes RAM DMA controller Hardware debug	Motorola MPA Fine-grained	Multiple busses Unknown format
National/ NAPA 1000	Compact-RISC	In Development	16K RAM 8x256 RAM Timer JTAG debugger	Concurrent Logic Fine-grained	
Siemens	TriCore	Plans Announced		Gatefield Fine-grained	
Atmel	?	Plans Announced		Atmel AT40K Coarse-grained	
SIDSA/ FIPSOC	8031	Sampling?	Programmable analog	SIDSA Coarse-grained	None, Memory-mapped

CONFIGURABLE PROCESSORS DEVELOPMENTS

Configurable processors are a new, emerging category of devices. As of December 1998, various companies have announced configurable processor devices or development plans, as shown in Table 1.

Triscend Corporation is sampling a Configurable Processor based around a high-performance 8032 8-bit microcontroller architecture, shown in Figure 1 [1]. Triscend is also developing a family of devices based on the ARM 7TDMI 32-bit RISC processor [2]. Triscend's devices feature a high-performance, integrated bus and SRAM-based, coarse-grained programmable logic.

Motorola announced a configurable processor family called CORE+, shown in Figure 2, which was to integrate a 32-bit ColdFire processor, peripherals, and Motorola's fine-grained FPGA technology [3, 4]. Unfortunately, Motorola's CORE+ efforts died when Motorola exited the FPGA business due to their long-suffering FPGA market share [5].

National Semiconductor, with funding from DARPA, is developing a device with a 32-bit RISC processor and a fine-grained FPGA [6]. National's technology, called NAPA1000 and shown in Figure 3, is targeted toward reconfigurable computing. National has not yet revealed their commercial plans.

Other companies are reportedly developing configurable processors, including Siemens with the TriCore media processor and Atmel [7, 8].

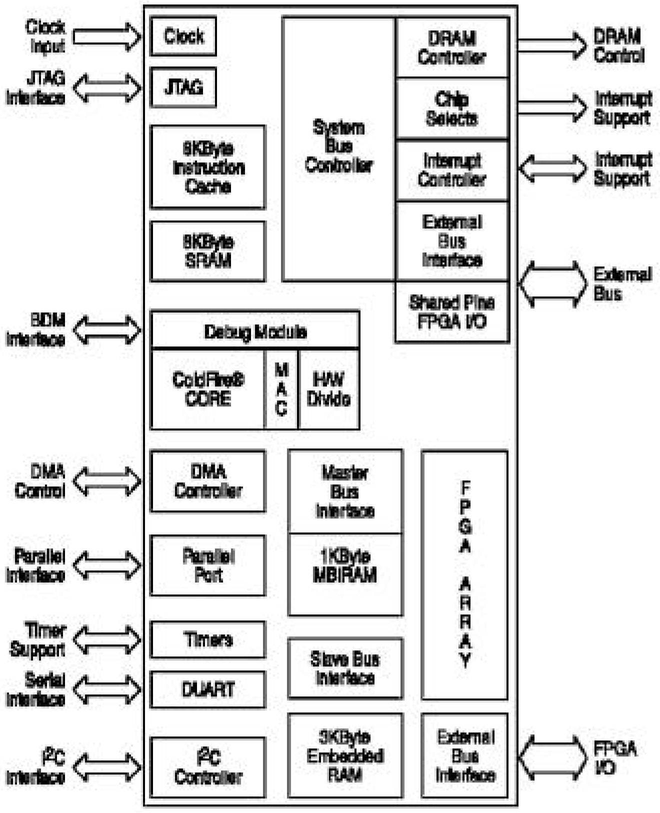


Figure 2. The CORE+ Configurable Processor based on Motorola's ColdFire processor. The CORE+ project died along with Motorola's MPA FPGA family [4].

CONFIGURABLE PROCESSOR BENEFITS

The Configurable Processor combines many of the best aspects of microprocessors with those of programmable logic. Basing a Configurable Processor on a popular, widely-used processor architecture takes advantage of the breadth of widely-available third-party tools, as well as the wealth of knowledge and experience with that processor within the embedded design community. The user-programmable nature of Configurable Processors adds the flexibility, adaptability, and fast time-to-market advantages that have fueled the rapid growth of the CPLD/FPGA market.

Design changes can be made easily, downloaded to the target device, and tested within the target system. The extensive simulation and verification tasks associated with gate array technology are avoided. Moreover, a Configurable Processor can have both its peripheral logic and processor code upgraded in the field.

Configurable processors generally have the extensive assembler, compiler, and debugging support of a stand-alone MCU. The predefined and tested bus architecture dramatically reduces system design and verification time, especially compared to a system-on-a-chip ASIC approach. Likewise, the configurable processor’s dedicated bus improves

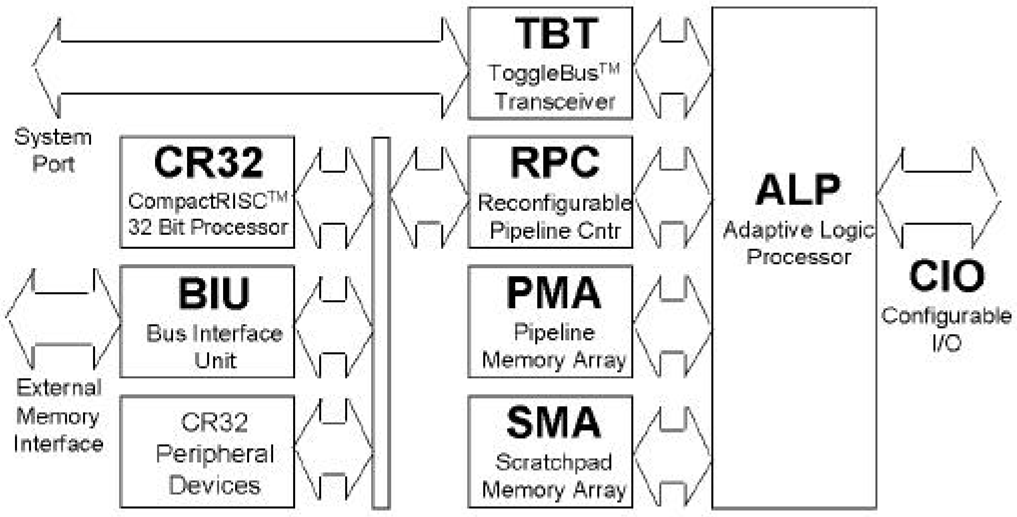


Figure 3. National Semiconductor's NAPA1000 configurable processor is a product of DARPA research funding [6].

system performance by reducing intercommunication delays between the processor and any peripherals implemented in programmable logic. The bus provides easier integration of programmable peripherals with the MCU and its development software.

APPLICATIONS OF THE PROGRAMMABLE LOGIC

The programmable logic portion of a Configurable Processor has many potential uses including

- Implementing common peripheral functions—*e.g.*, additional timers, I/O ports, etc.—to create a custom processor derivative.
- Implementing custom digital functions not typically available in any processor.
- Hardware/software trade-offs, implementing algorithms in programmable hardware instead of processor instructions.
- Offloading time-critical functions from the processor.

Custom Functions or Peripheral Set

The programmable logic integrated within a configurable processor provides ample flexibility to create custom peripherals. If an application requires additional timer/counters, pulse-width-modulations, UARTs, or other functions, these can be implemented using programmable resources and tailored for the specific application. Consequently, a designer can rapidly create a customized processor derivative.

Should the application require unique digital functions not available in any processor or microcontroller, the designer could create the function using standard logic design methods and integrate the function within the configurable processor.

Hardware/Software Trade-Offs

The very nature of a Configurable Processor allows hardware/software trade-offs with functions implemented using code in the processor or as hardware within programmable logic. There are often significant performance advantages in offloading a function into programmable logic.

Hardware Acceleration: Mapping an Algorithm to Silicon

Hardware Acceleration: Mapping an Algorithm to Silicon. The following example, shown in Equation 1 and Figure 4, compares two implementations of an algorithm—one implemented in software using the processor, the other implemented through hardware in programmable logic. The example function computes the average of four inputs originating from device input ports.

$$Z = \frac{(A + B + C + D)}{4} \qquad \text{(Equation 1)}$$

The processor implementation samples each port sequentially, adding the input to the accumulator. After computing the final sum, shifting the accumulator to the right twice accomplishes the divide-by-four operation. The overall performance of the algorithm is determined by the sum of the timing for all the individual instructions. This inherently parallel function must be processed sequentially when using a processor.

Compare this to the programmable logic solution. Offloading the function into programmable logic allows the function to be computed in parallel. The programmable logic implementation computes the average and provides it to the processor in less than 50 ns. The processor could read the final result, Z, from a register in the programmable logic. Multiple functions can operate concurrently, independent of the processor.

Programmable logic is also quite useful if the width of incoming data values exceeds

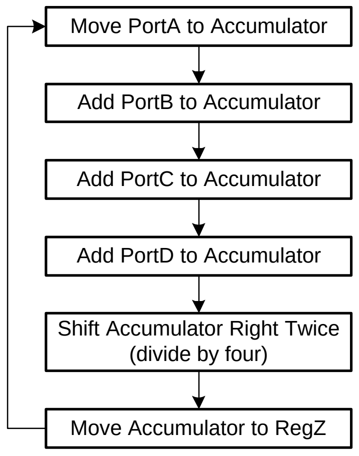
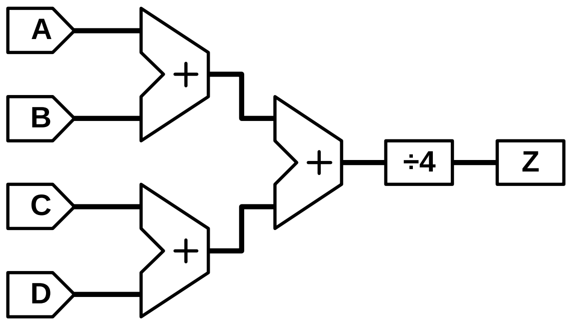
Processor Implementation	**Programmable Logic Implementation**
 Move PortA to Accumulator Add PortB to Accumulator Add PortC to Accumulator Add PortD to Accumulator Shift Accumulator Right Twice (divide by four) Move Accumulator to RegZ	

Figure 4. Two methods to compute the average of four incoming data values—one using a processor, the other using programmable logic. A Configurable Processor allows various hardware/software trade-offs.

the processor's natural word width—*i.e.* processing 13-bit data on an 8-bit processor. When incoming data is wider than the accumulator, the processor must spend additional instruction cycles handling overflow conditions. By contrast, the width of the programmable function uniquely matches the requirements of the function.

Rapid Response

Time-critical tasks can also be mapped into programmable logic, offloading the processor and saving processor bandwidth. Programmable logic allows very fast response to real-time events—down to tens of nanoseconds, much faster than the fastest interrupt response time of any processor. Furthermore, programmable logic can respond to multiple real-time events simultaneously.

Also, the programmable logic in conjunction with an internal DMA controller offloads mundane data transfers from the processor. If the Configurable Processor has a DMA controller, the DMA can transfer data directly from I/O pins or peripherals into RAM. Once the transfer is complete, the DMA could then interrupt the processor, indicating that data is available.

COMMUNICATION BETWEEN THE PROCESSOR AND PROGRAMMABLE LOGIC

One important aspect of any Configurable Processor a high-bandwidth link between the processor and the programmable logic resource. The benefits of an embedded bus also include lower power consumption and reduced electromagnetic interference (EMI) because bus signals rarely need to leave the device.

There are two basic approaches to providing communication between the processor and programmable logic—memory-mapped architectures and bus-oriented architectures.

Memory Mapped

In memory-mapped devices, programmable functions appear at fixed addresses within memory. Data and control values are written to memory locations within the programmable logic resource. Using this approach, multiple read or write operations may be required to send or retrieve the desired data, resulting in lower communication bandwidth than a bus-oriented approach.

Bus-Oriented

In a bus-oriented approach, the system address and data busses connect to the programmable logic resource. In some architectures, these busses are available only at the edge of the programmable logic resource while in others, the busses are distributed throughout the resource. The Triscend E5 is one example of a bus-oriented Configurable Processor where the bus is distributed throughout the programmable logic resource.

In the Triscend E5, the Configurable System Interconnect (CSI) socket specifies physical signaling for address and data paths, clock and control signals, and address decoding. Built-in programmable address decoders provide "chip select" signals for the various peripherals implemented in programmable logic. Additionally, the CSI socket steers

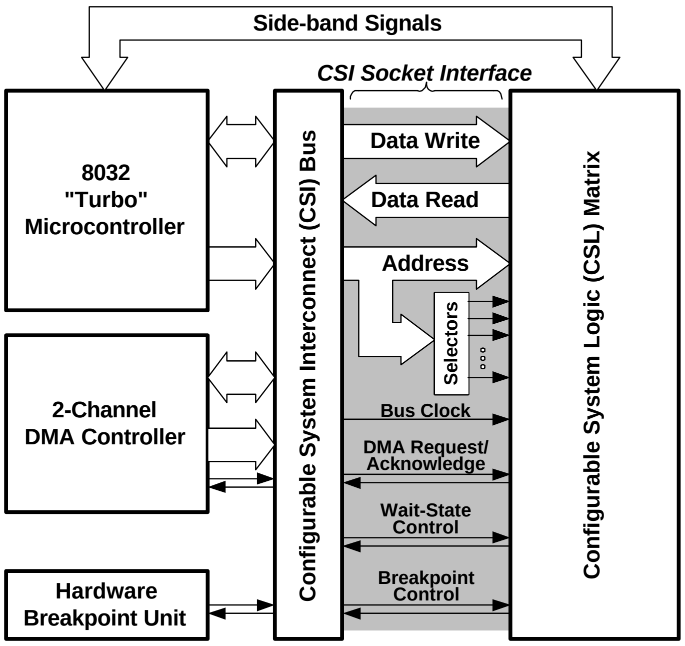


Figure 5. The Configurable System Interconnect (CSI) distributes address, data, control signals, and address decoding to the programmable logic resource (called the Configurable System Logic) in the Triscend E5 Configurable Processor.

DMA request and acknowledge signals to the appropriate DMA channel, provides wait-state control, and interfaces to the on-chip hardware breakpoint unit.

Processor signals unassociated with the CSI bus are called "side-band" signals and connect directly to the programmable logic function. The CSI socket is designed to be forward-compatible with future Configurable Processor families.

DEVELOPMENT TOOLS AND TECHNIQUES

Development flow for a Configurable Processor design follows the steps shown in Figure 6. The steps shown are for the Triscend FastChip Development Software but the flow is similar for other Configurable Processor families.

Some applications require minor customization where the user wishes to create a processor derivative on demand. In this case, the design can draw from libraries of common peripheral functions and the overall flow is similar to that of other catalog processor derivatives. Other applications require more substantial customization. In these cases, the design flow is more similar to a gate array or system-on-a-chip design.

Defining the Hardware

The vendor's Configurable Processor development system allows the designer to define the hardware functionality. The hardware definition consists of both the dedicated

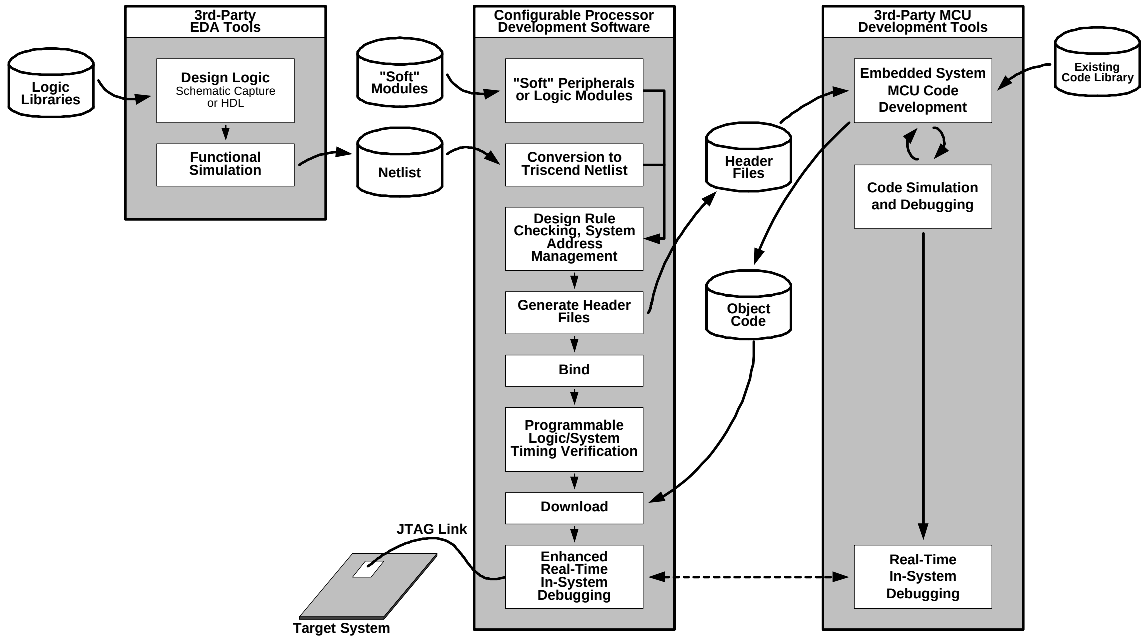

3rd-Party EDA Tools
Logic Libraries
Design Logic
Schematic Capture or HDL
Functional Simulation
"Soft" Modules
Netlist
Configurable Processor Development Software
"Soft" Peripherals or Logic Modules
Conversion to Triscend Netlist
Design Rule Checking, System Address Management
Generate Header Files
Bind
Programmable Logic/System Timing Verification
Download
Enhanced Real-Time In-System Debugging
JTAG Link
Target System
Header Files
Object Code
3rd-Party MCU Development Tools
Existing Code Library
Embedded System MCU Code Development
Code Simulation and Debugging
Real-Time In-System Debugging

Figure 6. Detailed configurable processor design flow. Designers create a custom processor derivative using pre-defined “soft” modules. User can also create custom peripherals using schematic capture or logic synthesis design tools.

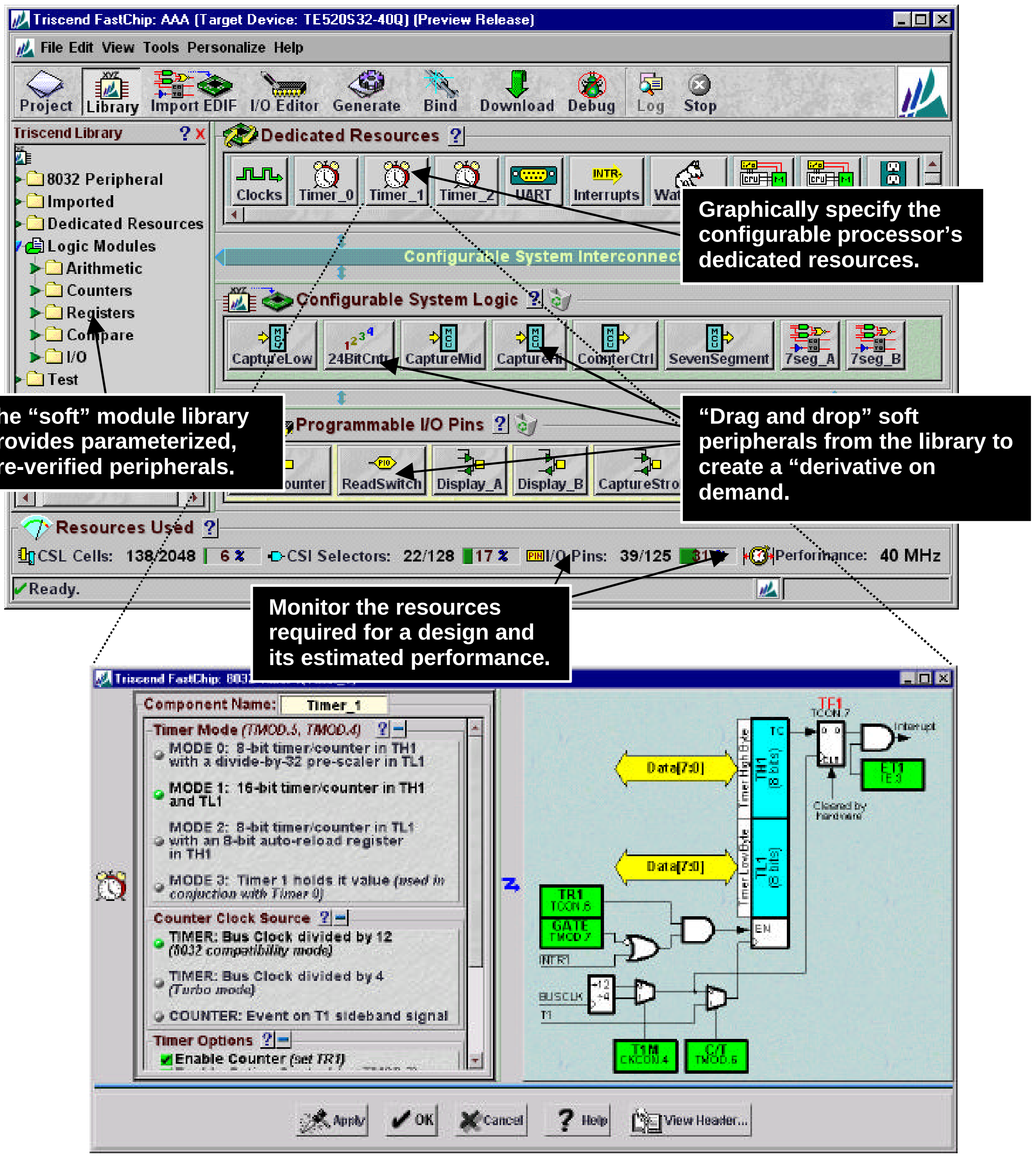
resources such as the processor and its "hard" peripherals plus any "soft" functions implemented using programmable logic resources.

To define the functionality of the programmable logic resources, a designer either leverages a library of pre-defined "soft" functions or creates custom logic using standard design entry methods such as schematic capture or hardware description languages.

Figure 7 shows how a user defines the hardware, using the Triscend FastChip Development System as a reference. The designer first decides which peripherals are necessary for the embedded system. From the library of pre-existing soft peripherals, the designer then "drags and drops" the desired peripherals into the programmable logic area. As each soft peripheral is added, graphical gauges provide the user with dynamic feedback on the amount of programmable logic resources that remain.

The Triscend FastChip development system includes many soft peripheral modules. However, the system also allows the user to create custom functions using standard schematic editors or logic synthesis tools and import them via an EDIF 2.0.0 netlist.

After the modules are "dragged and dropped", graphical interfaces are then used to select the parameterized options for each function (for example, timer/counter options would include operating mode, clock source, and initial value). This graphical parameterization is also used for the dedicated functions, such as the processor's native timer/counters.



Graphically specify the configurable processor's dedicated resources.

"Drag and drop" soft peripherals from the library to create a "derivative on demand.

The "soft" module library provides parameterized, pre-verified peripherals.

Monitor the resources required for a design and its estimated performance.

CSL Cells: 138/2048 6 % CSI Selectors: 22/128 17 % I/O Pins: 39/125 31 % Performance: 40 MHz

Component Name: Timer_1

Timer Mode (TMOD.5, TMOD.4)

- MODE 0: 8-bit timer/counter in TH1 with a divide-by-32 pre-scaler in TL1
- ● MODE 1: 16-bit timer/counter in TH1 and TL1
- MODE 2: 8-bit timer/counter in TL1 with an 8-bit auto-reload register in TH1
- MODE 3: Timer 1 holds it value *(used in conjuction with Timer 0)*

Counter Clock Source

- ● TIMER: Bus Clock divided by 12 *(8032 compatibility mode)*
- TIMER: Bus Clock divided by 4 *(Turbo mode)*
- COUNTER: Event on T1 sideband signal

Timer Options

- ✓ Enable Counter *(set TR1)*

Figure 7. Defining the configurable processor hardware using the Triscend FastChip Development System. Standard peripheral functions can be loaded from the "soft" module library. Dedicated resources are also specified graphically.

Creating the Hardware

After defining the hardware functionality, the Configurable Processor software then maps the user’s specification to hardware resources. In the Triscend development flow, this process is called “Bind” and is similar to the place and route step in printed circuit board design. The “soft” modules are mapped to specific programmable logic resources and connected to internal bus resources, if applicable. The result of this process is a binary file that, when combined with the object file from the processor’s compiler, initializes the processor and programmable resources upon power-up.

Interfacing to third-party development tools

After all the soft peripherals are designated and the parameterization is complete, the Configurable Processor software automatically produces a “header file” that assigns logical addresses and symbolic names for the registers within the “soft” peripherals. The user references this header file from within the application code to assist with the application code development. The assembler/compiler uses this information to generate the appropriate object code.

Program Development

After defining the hardware function, the bulk of the development tasks are those similar to any embedded system design. The processor’s application code is developed and debugged using standard development tools.

A programmer uses standard processor development tools to begin developing application code—typically in "C" or "assembly". Many programs also re-use code written for previous applications. The header file, created earlier, provides all symbolic naming and address locations for any soft peripheral. The programmer can then treat these functions as dedicated functions, removing the need to know how they were physically implemented. After the code development is complete, the finished code is then compiled (or assembled) into object code.

After creating code, the designer verifies its functionality. The first phase is typically logical verification, to test code functionality. In applications with complex real-world signal conditions, it may be difficult or impossible to model comparable conditions. Consequently, thorough verification is often left to in-system testing.

Creating and Downloading the Initialization File

An initialization file contains the information to completely specify the configurable processor design. It combines the binary pattern that defines all of the “soft” functionality plus the user’s object code.

The initialization file can be loaded in the device in a variety of ways. Typically, the initialization data is stored in a standard external byte-wide PROM, usually EPROM or FLASH. Upon power-up, the configurable processor loads bootstrap code from the external PROM, loading data for both the embedded processor and the programmable logic. Some devices even support serial PROMs to conserve I/O pins on the configurable processor.

For fast debugging, the initialization file might be downloaded directly into the part using a serial interface such as JTAG.

In-System Debugging

The stage is now set for exhaustive system verification. At this point, the designer has the Configurable Processor in its target system operating at full-speed, possibly in an environmental chamber to provide worst-case operating situations. The Configurable Processor typically connects to a PC for debug monitoring and runtime control.

Some development systems, such as Triscend's FastChip software, allow the designer to use familiar third-party debugging tools such as a source-level debugger. Commands from the debugger are translated by the FastChip software into JTAG commands that control the Triscend Configurable Processor's hardware breakpoint unit. This provides the designer with a familiar real-time debugger environment to control and monitor the Configurable Processor, including managing breakpoint events and examining internal register contents.

In some situations, a designer may wish to perform more advanced debugging that interacts with functions in the programmable logic, including those unassociated with the processor or the internal bus. The Triscend FastChip software provides a debugging utility for designers who desire to delve into this level of detail. This utility provides observability and control of each logical and sequential node within the programmable logic allowing the designer to mix processor debugging techniques with logic debugging.

COMPARISONS WITH OTHER TECHNOLOGIES

Configurable Processors offer significant time-to-market advantages compared to ASIC "system-on-a-chip" solutions. Programmable technologies eliminate the non-recurring engineering (NRE) charges associated with semi-custom solutions, though at a higher per-unit cost. In general, Configurable Processors share many of the cost/benefit advantages of other programmable logic technologies such as FPGA and CPLD.

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

SUMMARY

Configurable Processors are an emerging category of devices targeted at embedded system designs with stringent time-to-market demands. Configurable Processors allow a designer to quickly create a cost-effective custom processor derivative.

REFERENCES

[1] "Triscend E5 Configurable Processor Family," product description, Triscend Corporation, November 1998 (www.triscend.com/products/dse5cpsu.pdf).

[2] "Triscend Announces Industry's First 32-bit Configurable Processor Will Be ARM-based," press release, Triscend Corporation, November 9, 1998 (www.triscend.com/whatsnew/IndexSharpArm.html).

[3] "Motorola chip to combine ColdFire, FPGA cores," **EDTN**, February 3, 1998 (http://www.eetimes.com/news/98/992news/motorola.html).

[4] "MPACF250 - MPA's CORE+ ™ Reconfigurable System," product brief, Motorola, March 1998.

[5] "Motorola throws in towel on FPGAs," **EE Times**, June 29, 1998 (http://www.techweb.com/se/directlink.cgi?EET19980629S0034).

[6] "NAPA 1000 Adaptive Processor," National Semiconductor (http://www.national.com/appinfo/milaero/napa1000/ppframe.htm).

[7] "Oxford lab work rekindles codesign interest -- STM looks anew at integrating cores with FPGA," **EE Times**, April 20, 1998 (http://www.techweb.com/se/directlink.cgi?EET19980420S0045).

[8] "Atmel, startup prep configurable microcontrollers," **EE Times**, August 3, 1998 (http://www.techweb.com/se/directlink.cgi?EET19980803S0007).