Configurable Processors for Embedded Computing

Nikil Dutt, University of California, Irvine
Kiyoung Choi, Seoul National University

Ever-increasing chip capacities have given rise to configurable processors that offer virtually unlimited choices in core architectures.

We have all heard about the increasing software content of embedded systems. To those who think of embedded software as autonomous programs hidden deep within the system, plugging away transparently and reliably on dedicated tasks, this increase might suggest that these programs are somehow becoming larger. In reality, the ongoing increases in processor performance let system designers implement in software what previously required dedicated or custom hardware blocks and accelerators.

Indeed, given a choice, system designers might actually prefer the flexibility of implementing all embedded applications in software on programmable processors. However, parts of the applications must often run under critical time, performance, power, and cost constraints. Thus, designers have traditionally mapped these segments into custom hardware, such as application-specific integrated circuits (ASICs), or into reprogrammable fabrics, such as field-programmable gate arrays (FPGAs).

EMBEDDED PROCESSOR LANDSCAPE

With the continuing integration of yesterday's board into today's chip, the role of embedded processors is also changing. An embedded processor used to mean a 4- or 8-bit low-end microcontroller, designed primarily to perform simple control applications. The contemporary embedded system landscape cuts a broad swath—from low-end microcontrollers to high-performance processing engines.

Further, contemporary systems typically employ one or more processor cores, integrated into a system-on-chip design. Traditionally, such cores are fixed processors drawn from a library of well-known processor architecture families, such as those from ARM and MIPS. Designers can integrate the processors into SoCs as either hard cores with fixed layouts or custom-designed soft cores synthesized from Hardware Description Language specifications. But the processor's instruction set and the architectural features and parameters are typically fixed and not configurable for application-specific optimization. Coprocessors can accompany fixed-processor cores to improve overall system performance. Synthesizing the coprocessors for specific applications can boost the processor core's performance. Integrating the core with reconfigurable logic is another way to boost software performance while retaining hardware-acceleration benefits.

However, the need for differentiation in the marketplace, coupled with ever-increasing chip capacities, opens the door for significant customization of the processor cores themselves.

CONFIGURABLE PROCESSORS

Companies such as ARC, Improv, and Tensilica now offer configurable processors. This gives embedded system designers virtually unlimited choices in processor core architectures, allowing them to customize several features to suit the application and design constraints at hand.

For instance, designers can tune the processor's instruction set architecture to the application's characteristics. Similarly, they can optimize the processor pipeline or data path components for critical code segments in specific application domains. The resulting processors remain programmable and, in principle, can run any application, but they are optimized for a targeted application domain.

While the idea of customizing processors is appealing, it opens up new challenges: customization of the entire software tool chain, including compilers, simulators, debuggers, and so on; synthesis of processor models; and validation and verification of generated processor designs.

Flexibility and efficiency tradeoffs

Figure 1 shows a view of the efficiency and flexibility tradeoffs for different embedded SoC implementations in the hardware-to-software continuum. A traditional hardware-software codesign flow maps applications into the far ends of the continuum: For performance or power reasons, critical code segments are mapped to dedicated hardware, such as ASICs; the rest of the application software is mapped to general-purpose programmable processor cores.

The hardware implementation gains some flexibility with reconfigurable hardware, such as FPGAs, while sacrificing some performance or power. Similarly, domain-specific processors—for example, application-specific instruction-set processors, or ASIPs—trade flexibility for improved performance and power savings.

As the performance of fixed-processor cores improves, they assume more and more embedded system tasks. However, their still-limited performance and power savings provide little competitive advantage over other processor cores. System designers can use fixed cores together with dedicated hardware to boost performance with less power consumption, but this approach increases cost and decreases flexibility.

Configurable processors fill the gap in the center of a design continuum that starts with a pure hardware implementation and ends with a full software realization on general-purpose processors.

But can configurable processors actually achieve this balance? It may not be efficient to reuse tightly optimized configurations in new application domains. More importantly, unless the supporting software tool chain is available, the design time and effort can increase rapidly and nullify the potential efficiencies in reuse. Further, since configurable processors typically use an automated design-synthesis process at the back end, they are unlikely to outperform handcrafted fixed processors. Despite these problems, experiments with commercial configurable processors have reported speeds up to six times faster than fixed processors for specific application domains.

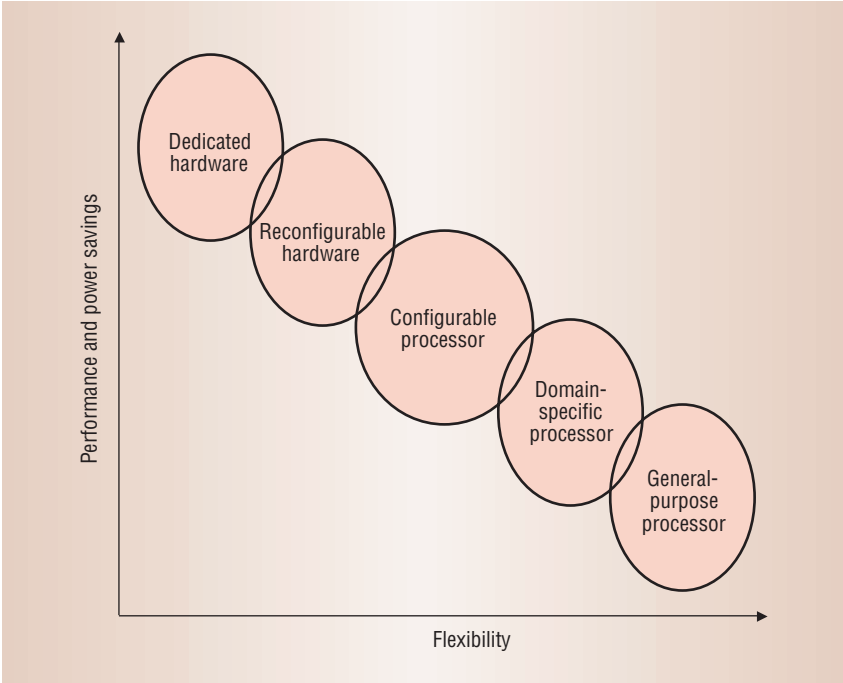

Figure 1. Embedded SoC design-space continuum trades the performance and power savings of dedicated hardware for the flexibility of software-based solutions.

Tool-chain customizations

The viability of the configurable processor approaches rests on the ability to quickly generate a customized software tool chain covering the entire development process, including compilers, simulators, debuggers, and validation and verification tools. For a given application, the customized processor configuration must not only match or outperform the design based on fixed processor cores and handcrafted development environments, it must do so without delaying the short time-to-market requirements for embedded systems.

Widespread adoption of configurable processor technology depends on rapid generation of an entire software tool chain and a customized, validated development environment.

ANALYZE THIS... CONFIGURE THAT

Since configurability increases the design space, taking maximum advantage of this potential requires an efficient system for design space exploration. A DSE system will support aggressive optimization in, for example, the instruction set architecture, memory organization, and overall system architecture in conjunction with coprocessors, hardware logic, multiprocessor and bus architectures, parallelism, and so on.

Each new configuration needs a new compiler and a customized simulator. We cannot explore a huge design space successfully unless we can generate such tools quickly and automatically. Arguably, the tools for design exploration need not generate optimal code or simulate behaviors at a detailed level. Rather, they need only support consistent comparative results.

Thus, a separate exploration phase for design flow efficiency would let system designers evaluate different base processor candidates and memory organizations. A simulator and a com-

Design Space Exploration

The following research projects use architecture description languages and machine description models to support intelligent tradeoffs among alternative processor architectures.

Expression

Center for Embedded Computer Systems, University of California, Irvine

An ADL supporting design space exploration for embedded SoCs and automatic generation of a retargetable compiler-simulator toolkit.

http://www.cecs.uci.edu/~aces/Projects/Expression/EXPRESSION_distribution.htm

Mescal

Gigascale Silicon Research Center, University of California, Berkeley

A project, "Modern Embedded Systems: Compilers, Architecture, and Languages," to develop methodologies, tools, and algorithms for fully programmable platform-based designs in specific application domains.

http://www.gigascale.org/mescal/

Lisa

Aachen University of Technology

A machine description language and generic machine model that allows bit-true and cycle-accurate modeling of pipelined processor architectures.

http://www.ert.rwth-aachen.de/lisa/lisa.html

ASIP Meister

Osaka University

A design environment that generates both data path and control units for application-specific instruction-set processors from behavioral descriptions of pipeline-stage operations.

http://vlsilab.ics.es.osaka-u.ac.jp/ed-meister/

piler generated in this phase could support rapid DSE in an estimation mode. A subsequent refinement phase could generate a cycle-accurate simulator and an optimizing compiler that allow the designer to fine-tune the processor characteristics and memory subsystem.

To fully evaluate the effects of different processor configurations, the DSE must consider not only the processor but also the interactions with the entire SoC, including the memory subsystem, buses, peripherals, and any other processing or coprocessing engines. An architecture description language could facilitate this functionality. The "Design Space Exploration" sidebar lists some university research projects that study ADLs and models for exploring tradeoffs among programmable SoC solutions.

SOFTWARE FOR SOFTWARE'S SAKE?

Finally, configurable processors execute software. So how does a designer generate production-quality software automatically? The process requires a customized development environment that will, in turn, rapidly generate several critically interdependent tools:

- a compiler that compiles the application software efficiently on the customized processor architecture;
- a simulator that allows for rapid, yet accurate performance evaluation;
- debuggers that permit probing and bug-fixing;
- synthesizable model generators for downstream processor implementation; and
- test suites that support efficient validation of the customized processor.

Production-quality software requires production-quality tools. The compiler is perhaps most important for achieving good system performance. ADLs provide a possible approach to generating a production-quality compiler capable of exploiting the processor architecture customizations to yield performance superior or equivalent to fixed processors with handcrafted compilers.

If the processor core does not have complex dynamic execution semantics, the hardware-synthesis flow for the configured processor seems to be relatively simple. Most existing approaches generate HDL specifications at the register-transfer level from the ADL view of the configurable processor, relying on other commercial synthesis tools for the rest of the design. However, the generated hardware again must compete with handcrafted fixed processors, regardless of whether they are soft, firm, or hard cores. The trick for efficient generation of hardware is to use predefined modules, user-customized modules, user-designed modules, or a judicious combination of the three.

Generating validation and verification suites for each instance of a configurable processor presents a major challenge.

AND THEN THERE WERE MORE

Many current SoCs already contain multiple processors that work collaboratively on application tasks. Local optimization of a configurable processor is insufficient for such systems. A critical need exists for tools that let system designers explore a heterogeneous multiprocessor system, selecting and customizing each processor core and performing system-level optimization across multiple configurable processors.

Further, developers typically base existing configurable processors on one of three architectural styles: reduced-instruction-set computing, very large instruction word, or digital signal processing. As designers begin to investi-

gate alternative architectures, such as simultaneous multiple threading, they will need tools that permit the exploitation of thread-level and instruction-level parallelism in an application.

CONFIGURE OR RECONFIGURE?

So far, we have talked about configurable processors. Configurable, in this context, implies a one-time customization of the processing engine prior to manufacturing.

By comparison, reconfigurable processors support both postmanufacturing and dynamic runtime configurability. Reconfigurability promises effective design reuse across multiple applications, without incurring new fabrication costs. An application could reuse the reconfigurable block on silicon across multiple tasks by dynamically changing the configuration. System designers could also combine reconfigurability with configurable processor technology—for instance, as a reconfigurable coprocessor or as part of the data path, control, or interconnect of the configurable processor itself.

Work on reconfigurable processors is primarily in the research phase, with many interesting problems still unsolved.

Proponents of configurable processor technology claim that it presents a paradigm shift in the design of next-generation embedded SoCs. They see it empowering application designers to create customized processors that can run their application codes efficiently while eliminating the entire complex chain of the processor design flow.

While this is true in principle, significant challenges remain in making this technology robust across a variety of architectural models and platforms. Although the technology has demonstrated its viability for niche domains, whether its applications will actually extend broadly across embedded system design remains to be seen. Nevertheless, ADL-driven research is moving the design process in this direction, and recent commercial offerings have already demonstrated the power of configurable processors for some application domains. ■

Nikil Dutt is a professor in the Information and Computer Science and the Electrical and Computer Engineering departments at the University of California, Irvine, as well as a faculty member in the Center for Embedded Computer Systems. Contact him at dutt@uci.edu.

Kiyoung Choi is a professor of electrical engineering and computer science at Seoul National University. Contact him at kchoi@azalea.snu.ac.kr.

Editor: Wayne Wolf, Dept. of Electrical Engineering, Princeton University, Princeton, NJ, 08544-5263; wolf@princeton.ecu.