

When number-crunching alone is not enough

High performance cores and even optimised instruction sets may not always be the solution to high throughput applications in telecoms and multimedia. Frequently on-chip data bandwidth becomes the deciding issue.

INTRODUCTION

In many network, telecommunications and data-com applications, the system has to move massive amounts of data. Such systems typically use a high-end general-purpose processor to handle the housekeeping problems, then rely on hard-wired logic or specialist processing technologies like DSP or manipulate the packets. Partitioning the choices in this way may have architectural drawbacks as well as creating software engineering issues.

Challenges like this can be resolved by using a single CPU/DSP processor core and adding flexible bus interfaces. This is the approach adopted by ARC Cores.

The ARC reconfigurable processor is a completely new breed of microprocessor/controller. In contrast with conventional processor IP, which often simply rehashes existing fixed instruction RISC or DSP

processors as ASIC cores, the ARC has total versatility built in from the start. ARC does not adhere to the traditional load/store model; neither does it tie the developer down to fixed bus timing or a rigid instruction set.

ARC Cores' approach began some nine years ago, and has had a number of imitators in the ensuing years. For example, several vendors including ARC are offering extended instruction sets, allowing developers to readily create custom single instructions from HDL, then build them into the silicon.

This can have enormous benefits in eliminating software bottlenecks, but it only addresses part of the problem. MIPS aren't everything. The truth is that bandwidth can be as important as processor efficiency in pushing large amounts of data through the processor.

In part, providing a compact processor core eases the situation. The ARC can be implemented in a triple-level metal process, with core size just 16k gates, equivalent



Figure 1. Architecture

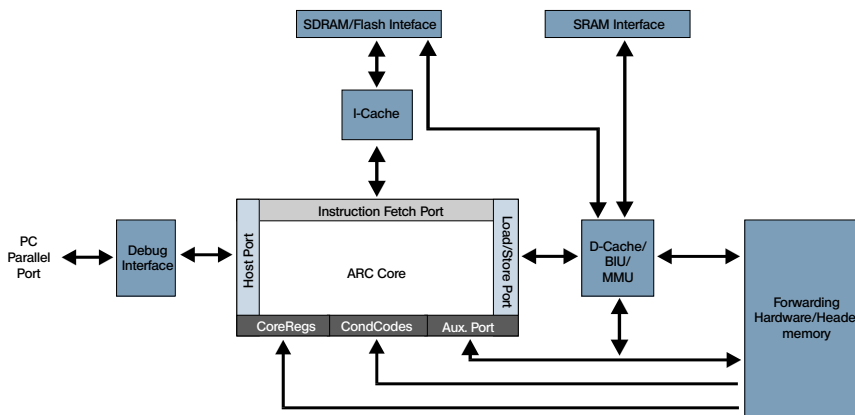


Figure 2. Toplayer.

to a footprint less than 2mm². Implemented as a 0.35 micron CMOS standard cell, it runs comfortably at 100MHz.



Figure 3. Interface.

But the real issue is architectural. It's "How much data can be shifted through the CPU?". While traditional microprocessors follow the standard Harvard architecture, with separate address and data buses, the ARC provides no less than four buses into and out of the core, leading to dramatically higher throughput. For example, the ARC exposes all internal signals, allowing the designer to build tighter coupling between peripheral hardware and the RISC engine. Also, the ARC provides a host interface port permitting external devices to completely control the internals of the processor. External hardware can start, stop, and single-step the processor-even reading and writing internal processor registers.

Provided as standard are: a host interface to provide easy bus integration; user-definable local memory interface; coprocessor interface to access specific logic blocks; and a memory interface for conventional data movement.

Using these multiple buses, the ARC can simultaneously fetch instructions, load or store data, process DMA packets and handle peripheral I/O. It makes for an unusually powerful data and bandwidth engine, as

the following examples illustrate.

APPSWITCH™ , HIGH BANDWIDTH IN PRACTICE

One high-bandwidth design to benefit from this approach is a switch that recognizes applications, all the way up to the Application Layer of the OSI model. This networking device forwards network traffic from one port to another based not only on the Layer2/MAC or Layer3/Network address, but also applies a policy to each flow based on the application and user of the application. The policies include prioritization, bandwidth guarantees, rate limits, rate shaping and graduated priorities. The applications are recognized based on information contained in higher levels of each data packet, and by stateful analysis of traffic flows to enable recognition of applications that use dynamically assigned Layer 4 ports.

To provide application switching at wire speed, analysis of network traffic needs to occur at an extremely high rate. The solution developed by Top Layer Networks - and incorporated in its TopFire™ chipset for the AppSwitch™ family - comprises three chips: an MII Octal MAC, which segments packets into cells, formats them, then transfers them over a fast internal bus to the

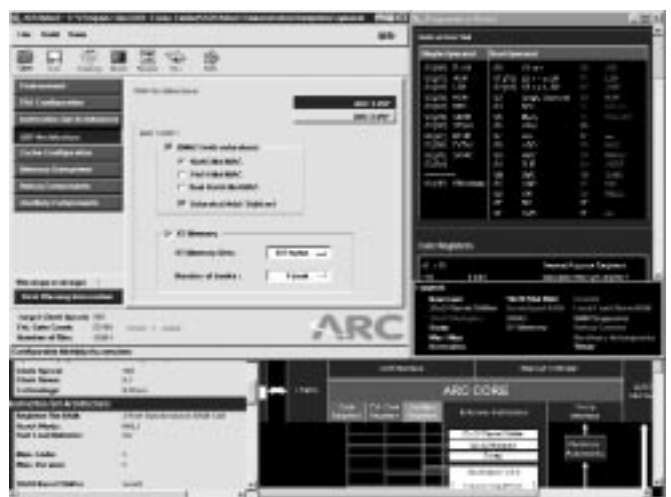


Figure 4. Interface.

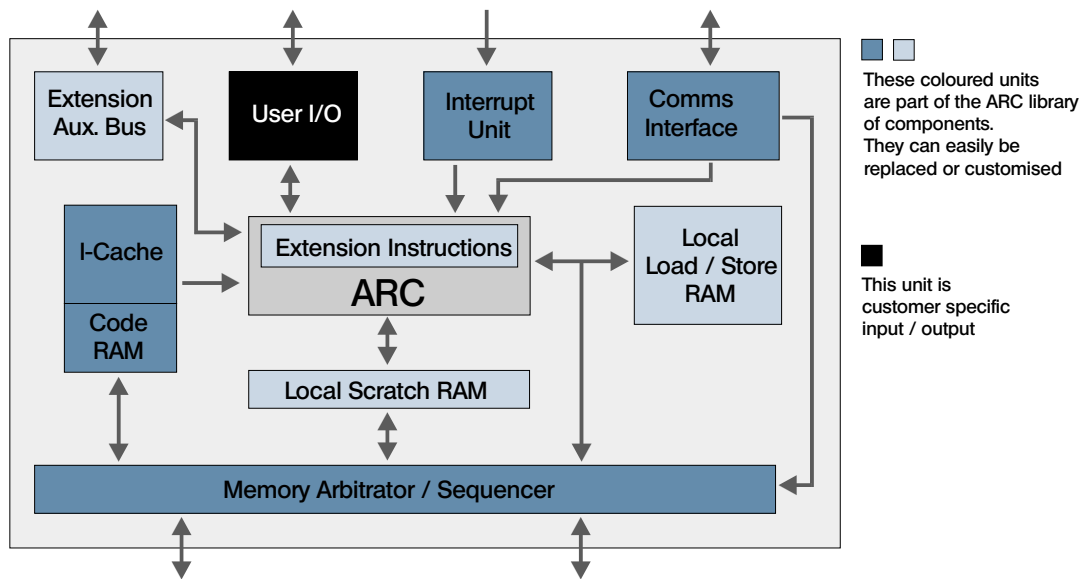


Figure 3. Interface.

second chip; the Queue Manager, which shuffles data to and from memory and manages all low-level queuing operations; and the Relay Engine, which performs packet classification using complex pattern recognition and analysis.

Top Layer Networks' Relay Engine has the ARC processor at its heart with a unique mixture of hardware and firmware that combines cost and performance benefits with flexibility and manageable complexity. While the optimised hardware performs the most frequently-executed instructions directly, the firmware comes into play with every data packet, so

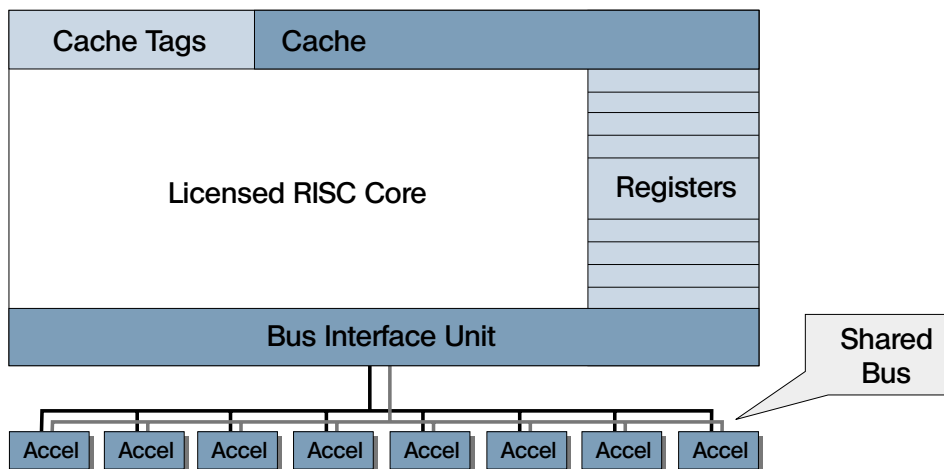
processor bandwidth is crucial.

ARC IMPLEMENTATION

The Relay Engine core takes packet data from the queue manager (QM), then uses hardware-based coprocessors to classify the packet. Once classified, information about the packet is presented to the processor for further handling. Based on the packet classification, the processor performs a series of computations and memory accesses before forwarding information back to the QM.

The processor is the logical center of the chip and has

Hyperchip typical compact RISC



Four RISC cores needed per ASIC
66-100 Mhz average speed

Figure 3. Interface.

Hyperchip with full ARC

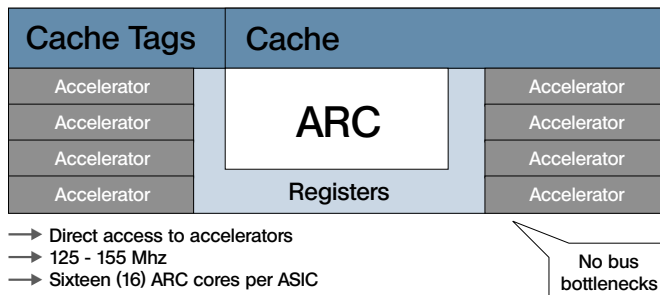


Figure 3. Interface.

visibility into and control over all operations. Top Layer designed a custom Bus Interface Unit (BIU) and Data Cache to interface to the ARC's load/store port. The BIU provides the ARC with access to external memory (SDRAM, Flash, SRAM, etc.) as well as internal registers and memory. The ARC core supports both big and little endian conventions, and the BIU was designed for both, ensuring compatibility with network byte ordering. Other features in the BIU include the arbitrary alignment of data accesses, a four-deep post buffer, and address decode.

Top Layer Networks also designed the data cache. It's a 1Kx32, direct-mapped cache with a 32-byte line size. Memory references can be marked cacheable or not using a built-in feature of the ARC LD (load) and ST (store) instructions. When the "DI" suffix is used with a LD or ST instruction, a signal will be set on the load/store interface to the ARC when the access is issued to the BIU. This signal is used to tell the data cache controller that the address is not cacheable.

The ARC Auxiliary Port - accessed using an entirely different address space from the load/store port - is used to interface to a variety of custom-built silicon functions, including two timers and a simple memory management unit developed by Top Layer. Using the special LR and SR instructions, the firmware can perform single cycle accesses to Aux-mapped registers.

Another key ARC extension used on the RE is custom

condition codes. Conditional branches are performed based on the state of up to 32 condition codes. 16 of these codes are reserved by the base instruction set. The other 16 are free to use as extensions. We were able to speed up performance significantly by mapping coprocessor conditions directly to the ARC condition codes.

Additional speedup was achieved by extending ARC's Core Register space. Certain coprocessor registers are mapped directly into the Core Register space so that the firmware has a zero-cycle penalty when accessing them.

ARC provided the Instruction Cache used on the RE as part of the ARC Development Tool kit. The design is configurable in many dimensions including size, address width, and line size. We use a 4Kx32 direct-mapped cache with 32- byte line size. The cache implements run-refill and can be disabled for debug purposes. Cache lines can be locked on a line-by-line basis.

The ARC instruction set was also extended for our application. ARC provides a library of instruction extensions. We added several instructions including multiply and barrel-shifter instructions. These instructions were added to the core automatically by the configuration program provided with the ARC Development Kit.

There are further advantages, according to Barry Spinney, Top Layer Network's Chief Technical Officer: "Perhaps one of the most important speed benefits of the ARC in our system is the ability to have multiple scheduled loads outstanding before the pipeline is stalled. The deep pipelining of our design imposes a large latency penalty for doing loads from memory. With other processor designs, the pipeline will stall if the data does not return within its scheduled slot in the pipeline. The standard ARC uses scoreboarded loads and can post up to four loads before stalling. As long as the target of a pending load is not used, the pipeline may continue with other operations and allow the data to return to the registers independently. At Top Layer Networks, we required posting up to eight loads

Hyperchip with customized ARC

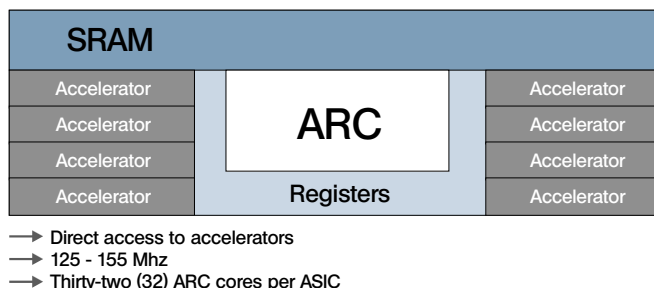


Figure 3. Interface.

AD NAT. INSTRUMENTS

before stalling.”

“The ARC’s powerful instruction set, and its unique ability to be customised, make it well suited to Top Layer’s architecture. Other RISC architectures that were examined would not have achieved the level of performance that was gained with the ARC. The ARC is technology-independent, so improvements in ASIC processes can be leveraged over time, without being tied down to one vendor’s technology rollout schedule. The ARC Development Kit is a well designed package that makes building customised versions a snap”, concluded Barry Spinney

EXTREME PERFORMANCE ROUTER

A more extreme example of ARC implementation in data communications is a recent design by Canadian parallel processing specialists Hyperchip. Their high-performance router, aimed at massively parallel processing systems, has a single-chip architecture based on 16 cells per chip, with the possibility of using multiple chips per system arranged in a variety of topologies. There are two ARC cores in each cell - a total of 32 ARC cores per chip. More impressively, each chip offers up to 160 gigabit/s of I/O bandwidth and the architecture can scale to provide petabit (1015bits) packet routing capability in small air-cooled units.

Proof, if it were needed, that overall data bandwidth is about more than just MIPS.

MORE ON ARC CORES

The ARC is a 32-bit RISC processor designed and licensed by ARC Cores, Inc. It is supplied as a set of technology-independent VHDL source files, which are ultimately synthesized using the target ASIC vendor’s library. A key differentiator of the architecture is its extensibility. The ARC can be customized by extending the design in a variety of ways. The VHDL source code has been designed with extensibility and synthesis in mind and the ARC development package comes complete with the tools necessary to build custom configurations. The ARC is well suited to embedded applications. Multiple interfaces allow a true Harvard architecture to be implemented. A special “host” interface port provides powerful debug capabilities. The innovative instruction set allows performance improvements over traditional RISC architectures. Below is a list of key architectural features:

- 32, 32-bit general purpose registers, extensible to 60.
- 4 stage pipeline
- 32-bit address space for data
- 26-bit address space for instructions
- 2- and 1-operand instructions
- 9-bit or 32-bit signed immediate data
- delayed branches with selectable delay slot modes
- conditional instruction execution
- optional flag setting
- zero overhead loops
- scoreboarded delayed loads allow operations to proceed

while high-latency memory accesses complete

- loads/stores of 1-,2-,4-byte data at any address
- address write-back feature (auto-increment)
- 2-level priority interrupts ■

Jim Turley is ARC Cores vice president of marketing. Jim Turley joined ARC from MicroDesign Resources in 1994. He was a Senior Analyst specializing in high-performance embedded microprocessors, Senior Editor of Microprocessor Report, and Editor-in-Chief of Embedded Processor Watch. He is author of the MDR Technical Library report, "Selecting a High-Performance Embedded Microprocessor," which provides technical analysis of more than 120 different microprocessors for embedded applications. Jim is the author of several technology books including "Advanced 80386 Programming Techniques," "PCs Made Easy," "Compact Guide to Windows 95," "Upgrading PCs Made Easy," and "Windows NT Training Guide."

Prior to MicroDesign Resources, he was engineering manager for Adept Technology, responsible for hardware design for robotic products that are widely used in welding, food packaging, automotive manufacturing, and electronics assembly. Earlier, Turley spent nine years at Force Computers as a design engineer of computer boards for telecommunications, industrial control, simulators, and transportation applications. He designed systems with many embedded microprocessors including the 68030, 68040, and 80386. He also wrote firmware such as debuggers, assemblers, disassemblers, and floating-point handlers. He spent two years in Munich, working with European companies developing custom and standard products. Before Force Computers, Turley was the Western region field application engineer for TeleVideo, providing technical support for the company's line of private label terminals and computers



EDITOR'S NOTE

Inclusion in this Company Directory about Development Tools for Dedicated Systems should not be taken as a Dedicated Systems Magazine endorsement or recommendation. Likewise, omission from the company directory should not be taken negatively. The information here was believed to be accurate at the time of writing, but Dedicated Systems Magazine cannot be responsible for omissions, errors or changes that occur after compilation.