Structural Object Programming Model: Enabling Efficient Development on Massively Parallel Architectures

Laurent Bonetto, Brad Budlong, Michael Butts, Paul Wasson
Ambric, Inc., Beaverton, Oregon
laurent@ambric.com

Introduction

High-performance embedded computing (HPEC) systems run compute-intensive applications such as video compression, image processing, networking and software-defined radio. Familiar CPU, DSP, ASIC and FPGA architectures, which once delivered increasing performance at a Moore's Law rate, have reached several fundamental scaling limits. A new platform for HPEC is required.

A number of parallel processing devices have appeared, seeking to overcome these limits. Those adapted from general-purpose or scientific computing may not be long-term scalable, reliable, or appropriate to HPEC applications. A new architecture, the Massively Parallel Processor Array (MPPA) has been expressly developed as a powerful, scalable and easy to use HPEC platform.

Ambric chose a parallel programming model first, for development, performance and scalability. This model is realized in the Am2045, which contains 168 32-bit DSPs, 168 32-bit RISC CPUs and 336 memory banks. It executes over one trillion operations per second, fifty 16-bit GMACS and over 125 thousand 32-bit MIPS [1,2]. This paper discusses development of a representative application (JPEG image compression) on the Am2045 to illustrate the methodology and effort involved in programming MPPAs.

SMP and SIMD Architectures

General-purpose microprocessors, microcontrollers and DSPs use a single CPU with a single memory space. Performance increases from processor architecture reached their limits by about 2003, so speedup fell from 52% to about 20% per year, roughly the silicon technology alone. If CPUs and DSPs had maintained the earlier rate, today they would routinely have 15 GHz performance. ASICs and FPGAs have their own scaling limits. ASIC fabrication expense is going up exponentially with the cost to build a foundry. ASIC and FPGA hardware design and verification productivity grows much slower than Moore's Law, so development cost has become the major project expense, also rising exponentially.

A number of single-chip parallel processing architectures are entering mainstream use in HPEC systems. Symmetric MultiProcessors (SMP), now common in general-purpose multicore CPUs, are appearing in embedded systems. Multiple processors are connected to cached shared memory, also used for inter-processor communication.

SMP's compelling property is its ability to run large, existing single-processor applications without modification. This is required for general-purpose computing, but it brings a heavy cost. The relative ease of adoption of multicore SMPs in general-purpose computing is misleading in the long run.

SMP interprocessor communication is a side-effect of cache coherency, which is slow and inefficient. Massively parallel SMPs require many-to-many cache coherency, which scales poorly. Achieving correct interprocessor synchronization is left up to the programmer. Debugging massively parallel applications promises to be difficult at best. SMPs are seriously non-deterministic, and get more so as they get larger [3], not a good property for time-critical and mission-critical embedded systems. Single-chip massively parallel SMP platforms are unlikely to be well-suited to the development, reliability, cost and power constraints of HPECs in the long run.

SIMD (single-instruction, multiple data) architectures have dominated the scientific high-performance computing (HPC) world since the time of the Cray-1. Workloads such as computational geoscience, chemistry and biology, structural analysis, and medical image processing are massively data-parallel, feed-forward, regular and floating-point intensive. SIMD parallel processors harness this regularity with tens to hundreds or more deeply pipelined datapaths, all under the control of one instruction stream.

Serial dependencies, feedback loops or short vectors put SIMD's long pipelines at a disadvantage. Data-dependent branching is expensive and deep pipelining makes branches cost many cycles. Some embedded application kernels in image and video processing resemble HPC workloads enough to run well on SIMDs. But as video, radio and networking algorithms get more sophisticated and complex, shorter vectors, irregular structure and data-dependency are common, putting SIMD at a severe disadvantage [4].

SIMD scalability is also limited by vector length. As graphics processing units evolve into HPC processors [5], they are adopting SMP/SIMD hybrid architectures, with the disadvantages of each in HPEC applications.

MPPA Architecture

The Massively Parallel Processor Array is a purpose-built platform for HPEC. Its objective is to optimize performance and performance per watt for HPEC applications, with reasonable and reliable application development, and long-term hardware and development scalability.

By starting with a good programming model, the Structural Object Programming Model, and developing silicon and tools to realize that model, Ambric's MPPA architecture is very well suited to HPEC applications, reliable, and has long-term hardware and software development scalability.

MPPA Programming Model and Tools

In Ambric's Structural Object Programming Model, objects consist of one or more software programs running concurrently on an asynchronous array of Ambric

processors and memories. A leaf object runs on a single processor or memory, while a composite object is a hierarchical composition of objects. Objects run independently at their own rates. They are strictly encapsulated, execute with no side effects on one other, and have no implicitly shared memory.

Objects exchange data and control through a structure of self-synchronizing Ambric channels. Each channel is word-wide, unidirectional, point-to-point, strictly ordered, and acts like a FIFO-buffer. Objects are mixed and matched hierarchically to create new objects, snapped together through channel interfaces. These composite objects can be as simple as a scaler or as complex as an entire application. Many pre-tested objects kept in libraries are easily reused, with excellent reliability thanks to strict encapsulation.

The developer expresses an application's object-level parallel structure in a coordination language called aStruct, which defines how objects are connected to each other. This language is essentially a simple textual representation of an implementation block diagram.

Having hundreds of processors available to implement an application provides rich flexibility to the developer, who can break down the target application into individual function objects that map naturally onto separate processors. Then the task of implementing, testing and tuning each of these objects becomes simple and self-contained.

This Structural Object Programming Model makes it efficient for a team of software developers to work in parallel and quickly implement any given application, each programmer being assigned a number of simple, well-defined functions to run on individual processors. The code running on each individual object can be written either in Java or in assembly. Due to the simplicity of the architecture, the RISC assembly language is relatively straightforward to use to produce optimal code.

Once an application is developed and compiled in Ambric's Eclipse-based IDE, aDesigner, its cycle-accurate simulator is used for initial testing and debugging. Then automatic placement and routing tools map the design onto processors, memories and channels, and download it into the Am2045 device, in a minute or less. Source-level parallel debugging and performance analysis on the actual running system makes application validation and tuning straightforward. Development is (reportedly) actually fun.

MPPA Hardware

Ambric's Am2045 device contains 168 32-bit DSPs, 168 32-bit RISC CPUs and 336 memory banks, which all run at 300 MHz. Each processor and memory executes a leaf object in the programming model. A 32-bit hierarchical configurable interconnect of Ambric channels implements the structure. Its dedicated debug network connects to the host for complete runtime control and visibility.

Am2045 delivers performance of over one trillion operations per second, fifty 16-bit GMACS and over 125 thousand 32-bit MIPS. It has a 4-lane PCI Express host interface, two 32-bit SDRAM controllers, and four 32-bit general-purpose I/O ports for interconnecting multiple chips, and for interfacing with ADCs, DACs and other hardware. Am2045 is implemented as a 130nm standard cell ASIC. Its power dissipation is from 6 to 14 W.

MPPA Application Development

JPEG is at the root of nearly all image and video compression algorithms, so a JPEG encoder is a realistic example of a complete HPEC application. A three phase methodology was used: functional implementation, optimization and validation.

The first phase is to produce a reference code that will run correctly in the development environment of that chip, as a starting point for the fully optimized implementation. Usually this code developed during this phase is abstracted from most features of the target architecture, allowing it to remain simpler, more readable, and more general.

The second phase is to improve speed to meet application requirements. As with hardware designs, the software developer has an opportunity to trade area for speed. Time-consuming blocks can be parallelized onto multiple processors in many ways. Functional parallelism can be used to run different parts of an algorithm on successive processors. Data parallelism can be used to run the same algorithm on independent blocks of data using different processors. The developer may also optimize each object's code, as with conventional DSPs. Simpler target processors combined with less stringent requirements on the code running on each processor makes fully optimized assembly code simpler and not as often needed. This phase may be done in simulation with a testbench and/or on real hardware with live data.

Finally the third phase is to thoroughly validate the application on hardware in real time. Am2045's dedicated debug and visibility facilities are used through aDesigner's runtime debugging and performance analysis tools.

The entire JPEG encoder uses less than 5% of the Am2045 device capacity. It achieves 72 frames per second throughput, vs. the 60 fps target, using a balanced combination of Java and about 300 lines of assembly code.

The JPEG implementation on the Ambric Am2045 MPPA architecture illustrates an HPEC platform and development methodology that can easily be scaled to achieve levels of performance higher than any high-end DSP and comparable to those achieved by many FPGAs and even ASICs.

References

[1] M. Butts, A. M. Jones, P. Wasson, *A Structural Object Programming Model, Architecture, Chip and Tools for Reconfigurable Computing*, IEEE FCCM 2007, pp. 55-64.

[2] Michael Butts, *Synchronization through Communication in a Massively Parallel Processor Array*, IEEE Micro, Sep. 2007.

[3] Edward A. Lee, *The Problem with Threads*. IEEE Computer, 39(5):33-42, May 2006.

[4] Wen-mei Hwu, et. al., *Performance Insights on Executing Non-Graphics Applications on CUDA on the NVIDIA GeForce 8800 GTX*, IEEE Hot Chips 19 Symposium, 2007.

[5] John Nickolls, *GPU Parallel Computing Architecture and CUDA Programming Model*. IEEE Hot Chips 19 Symposium, August 2007.