AMD Stream Computing: Software Stack

EXECUTIVE OVERVIEW

Advanced Micro Devices, Inc. (AMD) – a leading global provider of innovative computing solutions – is working with other leading companies and academic institutions worldwide to deliver a complete, accelerated computing ecosystem with software and tools necessary to turn its high performance, low cost, supercomputing vision into reality. AMD’s Stream Computing initiative is ushering processing technologies into the accelerated computing era through integration of CPU, GPU and complete software stack.

AMD Stream Computing is a first step in harnessing the tremendous processing power the GPU (Stream Processor) for high performance, data-parallel computing in a wide range of business, scientific and consumer applications. AMD’s Stream Computing platform provides organizations and individuals the ability to integrate accelerated computing in existing IT Infrastructure, enabling improved decision-making, accelerated work-flows and reduced time-to-discovery.

This paper provides a high-level introduction to AMD’s Stream Computing software stack – providing end-users and developers with a flexible suite of tools to leverage the processing power of AMD Stream Processors.

AMD’s Stream Computing Software Stack includes the following components:

- **Performance Libraries**: ACML and COBRA for optimized domain-specific algorithms
AMD Core Math Library (ACML) includes full implementations of BLAS and LAPACK routines along with FFT, Math transcendental and Random Number Generator routines. ACML release for GPU contains a stream processing backend that permits load balancing of a subset of these routines between the CPU and GPU depending upon the suitability of the task for a particular architecture.

- **Compilers: Brook+ and RapidMind**
  - Brook+ is AMD’s implementation of the open source Brook C/C++ compiler that AMD is enhancing\(^1\) to include new features and back end which targets FireStream™ GPU processors. AMD’s embracing of Brook opens the door to many third party developers creating non-graphics applications to run in the GPU environment.
  - Rapidmind is a complete development environment – C++ compilers and IDEs to improve programmability, performance and portability of 3rd party applications developed for AMD Stream Computing.

- **Lower Level Driver and Programming Language: AMD Compute Abstraction Layer (CAL)**
  - CAL provides access to the various parts of the GPU as needed. Developers are thus able to write directly to the GPU without needing to learn graphics-specific programming languages. CAL provides direct communication to the device.
  - Intermediate language specification provides low-level access to code, increasing the ability to fine-tune device performance.

- **Performance Profiling Tools: GPU ShaderAnalyzer, AMD CodeAnalyst**
  - GPU ShaderAnalyzer performs throughput and flow control analyses on Stream processors generating GUI-based performance data or command line generated reports.
  - AMD CodeAnalyst is a software performance analysis tool which includes system-wide profiling, as well as timer-based and event-based profiling, and sampling and simulation functionality.

---

\(^1\) Release scheduled for Q1 2008.
Performance Libraries

AMD Performance Libraries include a collection of popular software routines designed to help accelerate application development, debugging and optimization. These libraries are optimized for high performance on AMD platforms and hence can provide a quick path to high performance development.

ACML

AMD Core Math Library (ACML) includes implementations of the full BLAS\textsuperscript{2} and LAPACK\textsuperscript{3} routines along with FFT\textsuperscript{4}, Math transcendental and Random Number Generator routines. The latest release of ACML includes a stream processing backend that permits load balancing of a subset of these routines between the CPU and GPU depending upon the suitability of the task for a particular architecture.

Brook Open-Source Data-Parallel ‘C’ Compiler

Brook\textsuperscript{5} provides a data-parallel ‘C’ compiler using extensions to standard ANSI C programming language. Brook started as an open-source project from Stanford University for demonstrating general-purpose data-parallel computations on graphics processors. Brook’s computational model, referred to as streaming, provides two main benefits over traditional sequential programming languages:

- **Data Parallelism**: Brook provides an intuitive mechanism for specifying SIMD (single-instruction multiple-data) operations.
- **Arithmetic Intensity**: Brook’s interface encourages development of efficient algorithms by minimizing global communication and maximizing localized computation on stream processors.

The two key elements in Brook language that make it extremely powerful are:

- **Streams**: A stream is a collection of data elements of the same type which can be operated on in parallel. Streams are denoted using angular brackets 

- **Kernels**: A kernel is a parallel function that operates on every element of input streams. Kernels are specified using the `kernel` keyword.

The following code snippet shows a simple example of a Brook kernel which adds two input streams and stores the results in an output stream. The kernel performs an implicit loop over each element in the output stream.

```c
kernel void sum( float a<>, float b<>, out float c<> )
{
    c = a + b;
}
```

---

\textsuperscript{2} BLAS – Basic Linear Algebra Subroutines – www.netlib.org/blas
\textsuperscript{3} LAPACK – Linear Algebra Package – www.netlib.org/lapack
\textsuperscript{4} FFT – Fast Fourier Transform – www.fftw.org
\textsuperscript{5} BrookGPU - http://graphics.stanford.edu/projects/brookgpu/index.html
The Brook software consists of two important components:

- **BRCC** is a source-to-source meta-compiler which translates Brook programs (.br files) into device-dependent kernels embedded in valid C++ source code. The generated C++ source includes the CPU code and the Stream Processor Device Code that are later linked into the executable.
- **BRT** is a runtime library that executes pre-compiled Kernel routines invoked from the CPU code in the application. Brook includes various runtimes for CPUs and Stream Processors and the execution model can be selected at application runtime. The CPU runtime serves as a good debugging tool when developing Stream kernels.

AMD has enhanced BRCC to produce AMD’s intermediate language (IL). AMD has also enhanced BRT with a backend optimized for AMD Stream Processors using the CAL driver (see below).

### GPU Shader Analyzer

GPU Shader Analyzer (GSA) is a performance profiling tool useful for developing, debugging and profiling GPU kernels using high-level GPU programming languages. GSA provides many advantages including:

- Online kernel compilation to generate the equivalent AMD IL as well as processor-specific ISA\(^6\) assembly.
- Performance characterization in terms of arithmetic, memory and flow control instructions.

The GSA has a Graphical User Interface which is fairly simple to use and requires minimal learning effort. The following snapshot shows an example user provided kernel written in HLSL that has been converted to its AMD IL equivalent.

---

\(^6\) ISA – Instruction Set Architecture
AMD Compute Abstraction Layer

AMD Compute Abstraction Layer (CAL) is a device-driver library that provides a forward compatible interface to AMD’s Stream Processors (Devices). CAL allows software developers to interact with the processing cores at the lowest-level, if needed, for optimized performance, while maintaining forward compatibility. CAL provides the following main functions:

- Device Specific Code Generation
- Device Management
- Resource Management
- Kernel Loading and Execution
- Multi-device support
- Interoperability with 3D Graphics APIs

The CAL SDK includes a small set of ‘C’ routines and data types that allow higher-level software tools to directly interact with and control hardware memory buffers (device-level streams) and GPU programs (device-level kernels). The CAL runtime accepts kernels written in AMD IL and generates optimized code for the target architecture.
The CAL API is ideal for performance-sensitive developers. It also provides access to device specific features.

The following is a list of performance numbers obtained for a few important benchmark programs written directly in CAL on AMD HD 2900 GPUs:

- Peak Single-precision ALU throughput – 512 GFlops
- Peak Cache bandwidth – 200 GB/s
- Matrix Matrix Multiplication Kernel – 200 GFlops

For further information on AMD Stream Computing or to enroll for our early access program, please e-mail us at streamcomputing@amd.com.