The Carte++™ Application Development Process

SRC Computers, LLC
www.srccomputers.com

OVERVIEW
SRC Computers’ reconfigurable servers yield high application performance, low power consumption, greatly reduced space requirements and superior TCO advantages when compared to other servers available today. Carte++ is the development environment enabling application developers to access these benefits on SRC reconfigurable systems. A typical Carte++ application development process is very similar to the process used for multicore microprocessor targets and is greatly simplified when compared to application development for typical “accelerator” systems.

SRC Computers’ reconfigurable systems consist of at least two autonomous peer processors: one microprocessor and one SRC-developed reconfigurable MAP® processor. This peer-based architecture is called the IMPLICIT+EXPLICIT™ Architecture. A microprocessor is very good at running control-flow programs, in which conditional statements dictate an execution path through the program. A MAP processor excels at data-flow oriented programs, in which large amounts of data flow through functional units to generate results. Since applications are usually a mix of these two types of programs, Carte++ gives the developer the freedom to choose the appropriate execution model for performance. By allowing autonomous peer processors to cooperate on application execution, the SRC servers differ greatly from the master processor/accelerator model prevalent in heterogeneous processing today.

SIMPLE 5-STEP PROCESS
Carte++ application development typically advances along a five-step process:

1. Analyze
2. Partition
3. Performance estimation
4. Prototype
5. Final integration

1. ANALYZE
The analysis step involves examining the performance and data characteristics of an application. These characteristics guide a developer selecting an appropriate processor execution model on the heterogeneous SRC reconfigurable system. For example, large amounts of regular data and a high degree of available parallelism indicate portions of an application that will perform well on the MAP processor. Another example is if 90% of the application execution time is contained in 10% of the code, that performance kernel code is an excellent candidate for the MAP processor.

A less obvious set of characteristics center around the algorithms in an application. A microprocessor has a fixed set of functional units that may not be optimal for an algorithm; Carte++ generates the exact full set of functional units an algorithm requires. Sometimes a key algorithm may be mathematically re-written in such a way to take advantage of this Carte++ capability. For example, an algebraic correlation algorithm may be re-written in such a way to avoid the microprocessor computational cost of continuous summation operations. It is clear after experimentation and analysis which parts of an application will execute well on a microprocessor and which will execute well on the MAP processor.
2. PARTITION

The partition step is a natural outcome of the analysis step. The application is partitioned by splitting the appropriate microprocessor and MAP processor code into separate files. The MAP processor code will appear as a function call in a microprocessor thread. However, once the MAP processor execution is started, it is as autonomous as the microprocessor. Processor cooperation, to the extent it is needed, is defined by the programmer.

3. PERFORMANCE ESTIMATION

Application performance estimation on the microprocessor side is a challenging art as on any microprocessor-only system. Operating system requirements; cache, multi-user, and threading effects; resource competition and other run-time characteristics need to be taken into account when predicting microprocessor performance. On the other hand, the MAP processor performance at run-time is deterministic and easily calculated using a spreadsheet, given information provided by the Carte++ MAP processor compiler.

4. PROTOTYPE

A prototyping step is often used to check assumptions, verify performance estimates and ensure the critical elements of the application are functionally correct. To support prototyping, Carte++ provides a “debug mode” that runs compiled code in a threaded emulation environment. Since debug mode runs on any microprocessor-only system, standard Linux debugging tools may be used and SRC hardware is not required. This step usually requires from 1 to 10 days.

5. FINAL INTEGRATION

The final integration step is common to all software development. This is when the developer team integrates and tests all application modules on actual SRC reconfigurable system hardware in its final configuration. Depending upon application complexity and size, this step takes between 10 and 90 days.

SUMMARY

The Carte++ application development process appears very similar to the process familiar to multi-core microprocessor programmers. There are two slight distinctions: first, application partitioning among multiple cores becomes partitioning across heterogeneous processors on the SRC reconfigurable system; and second, performance estimation is easier with the deterministic MAP processor execution. When compared to accelerator system programming, Carte++ is simpler for two reasons: first, all of the compilers, tools and libraries are fully integrated into one complete programming environment and 2) Carte++ allows developers to use the C and C++ software languages instead of C-like or hardware languages. By moving from multi-core microprocessors or accelerators to the SRC reconfigurable system and Carte++, the application developers win.

---

1 SRC Application Implementations on a Saturn 1 Processor, SRC Computers LLC, MKT-054-00, August 3, 2012.