CARP

Reporting Period 1: Publishable Summary

Grant Agreement:	287767
Project Acronym:	CARP
Project Name:	Correct and Efficient Accelerator Programming
Instrument:	Small or medium scale focused research project (STREP)
Thematic Priority:	Alternative Paths to Components and Systems
Start Date:	1 December 2011
Duration:	36 months
Number of Pages:	2

Publishable Summary

The aim of the CARP project is to design techniques and tools to improve the programmability of accelerator processors, with a particular emphasis on graphics processing units (GPUs). The project follows three main strands for achieving these improvements:

- Increased programmer *productivity* through higher level programming languages
- Increased runtime efficiency of accelerated applications, both in terms of runtime and energy consumption, through advanced *compilation techniques*
- More reliable accelerated software through novel *formal verification* methods.

The first year of project work has involved a collaborative requirements gathering phase, which is now complete. In addition, the consortium have undertaken novel research into high level languages and domain-specific languages for accelerator programming, advanced compilation techniques, methods for analysing the runtime cost of accelerated applications, and formal verification techniques for GPU kernels. The groundwork for validation of the developed technology has been laid through the specification of a set of validation metrics, a description of the validation environment, and the implementation of baseline algorithms in the eye tracking domain for future benchmarking and comparison. Throughout the year, the consortium have been active in disseminating the project's aims and early results online and through various activities, including presentations, papers, a poster and a press release.

The specific major scientific achievements of the project are as follows:

- A state-of-the-art survey and a requirements document providing detailed requirements for compilation and verification techniques targeting accelerated systems;
- The drafting of a specification for PENCIL, a Platform Neutral Compute Intermediate Language for accelerator programming, and the definition of two domain specific languages for which GPU acceleration is desirable;
- Novel polyhedral compilation techniques geared towards *just in time* translation, and the implementation of these techniques in a new compilation framework geared towards the generation of code for GPUs;
- Two analysis techniques for estimating the runtime of software, with applications to accelerated software: a constraint-based technique for synthesising loop invariants for probabilistic programs, and a method based on hybrid worst case execution time analysis which combines dynamic and static analyses to estimate the average- or worst-case cost of executing a GPU kernel;
- The foundations for techniques for formally analysing GPU kernels written in low-level languages such as OpenCL and CUDA, based on modular program verification, separation logic and symbolic execution, and an early implementation of these techniques as a tool, GPUVerify;
- High performance implementations of key algorithms in eye tracking, optimised for multicore PCs and GPUs, to be used as a baseline for comparison with the CARP technology.