

Games Console Development

Coursework: Parallelism / SIMD on the Cell Broadband Engine

Issued: Friday 19th March 2010

Report submission (at the latest): 4:30pm Friday 7th May 2010, submitted on CD to the School office, E217.

Assessment contribution: 60% of final mark

This is an individual assignment

Introduction

For this coursework, you are to modify a small application that has been written for you, and improve its performance by exploiting parallel hardware features of the Cell Broadband Engine processor. The majority of your optimisations should focus on taking advantage of parallel features of the Cell BE processor, rather than general code optimisation techniques that could be made on any platform. Any relevant optimisations will be considered during marking.

This program runs on the PS3, sending visual data to an X-Server running on a PC. The display window has been split into 4 separate “viewports” as follows:

0. 3D spinning cubes are transformed from object space through to screen space then rendered using filled triangles using the 2D Xlib API. This is a basic demonstration of a software pipeline, including the painter’s algorithm for depth and backface culling.
1. A 2D “map” is loaded from a text file, with randomly chosen start and goal positions. An “agent” is placed at the start position, and then follows an A-Star path to the goal. This is repeated when the “agent” reaches the goal, with a new goal position being randomly selected, and a new path generated.
2. A 3D model of the “Stanford bunny” is loaded and it is transformed through the software pipeline, as per the cubes on viewport 0. The model itself is never displayed, but a wireframe axis-aligned bounding box is displayed, resizing as appropriate as the model is arbitrarily rotated around the axes.
3. Debug information is displayed to give an *idea* of how the application is performing. High resolution timing will provide more accurate information.

Two “frame rates” are calculated and updated every half second, showing the updates of the display per second, the updates of the tick() function per second (where the calculations take place), and the number of vertices processed per second on viewport 2. You should explore different methods of parallelizing the code, and report on the effectiveness of each method, in terms of the frame rate. For example, you can try using vector intrinsic code, performing data decomposition and sending processing units to the SPEs, and you can try pipeline processing. These are just suggestions, and you can explore your own methods of parallelizing the code.

Comments have been used throughout the source code provided, and these should aid understanding, although the lecturer will be able to further explain the code if required. Ask for help if you need it!

Deliverables

Two deliverables are required:

1. The source code of the optimized application in electronic format (CD/DVD), possibly including several versions featuring different optimizations. (There is no need to submit a hard copy of your source code.)
2. A report that lists and explains the optimizations that you have made to the application, and also any ideas that you have not been able to implement. Credit will be given for discussing possible optimizations without necessarily having implemented them, although the allocation of marks will not be as high as for those that have been implemented. The report should be in Word or Adobe pdf format. This report should ideally be split into four sections, matching the marking scheme below:
 - 2.1. In your identification of inefficiencies, you should refer to the theory covered in this module, your general understandings of graphical applications, and include high resolution timings that clearly indicate bottlenecks in the program.
 - 2.2. Provide detail of how these inefficiencies could be improved upon, again based on theory covered and your previous knowledge.
 - 2.3. Taking your theoretical improvements, you should then make the required changes to the application, and provide proof of any gain made (high resolution timings help). It may not be possible for you to implement all the improvements you have suggested in the time given.
 - 2.4. Detail how your improvements have made use of the specific hardware features of the Cell hardware.

These should be submitted to the school office (E217) by the date indicated at the top of this specification.

Plagiarism

You are reminded that plagiarism is a serious disciplinary offence. The deliverables for this assignment should be your own work. If there are any (small) parts of your project that are not your own work, you must identify these parts and acknowledge the original author. You are free to use without acknowledgement any source code provided by the lecturer during the module. This includes all code found in the lecture slides, the lecture notes, the lab sheets and any other code that is posted on the blackboard web site by the lecturer.

Marking Scheme

Identification of inefficiencies in engine	20%
Improvements (theory)	20%
Improvements (implemented)	40%
Exploitation of Cell hardware	20%
Total	100%