|  |  |
| --- | --- |
|  | Interim Project Report |
|  |  |
| 2/11/2011 | Final Year Project 2011 |
|  | Author: Andrew Bolster, 11614064  Supervisor: Prof A.Marshall  Moderator: Dr J.McCallister  Research Cluster: Digital Communications |

Interim Project Report

High Speed bit-loading algorithms for Dynamic Spectrum management in ADSL

Table of Contents

[Project Specification 2](#_Toc285149206)

[Project Task Breakdown 2](#_Toc285149207)

[Project Task Detail, Explanation, and Status 3](#_Toc285149208)

[Research 3](#_Toc285149209)

[Project Management 4](#_Toc285149210)

[Framework Design 5](#_Toc285149211)

[Implementation 6](#_Toc285149212)

[Comparison 7](#_Toc285149213)

[Reporting 7](#_Toc285149214)

[Current Project Schedule 9](#_Toc285149215)

[Issues Experienced and Resolutions presented 9](#_Toc285149216)

[Time Management 9](#_Toc285149217)

[Existing Implementation Difficulties 9](#_Toc285149218)

[Bibliography 10](#_Toc285149219)

# Project Specification

Digital subscriber lines need to employ bit-loading techniques to improve their throughput; this is known as “Dynamic Spectrum Management” (DSM). Currently all implementations of DSM use level 1 whereby individual lines use a “water filling” algorithm to adjust the power allocated to each tone in the spectrum, and a central management station adjusts the water filling parameters for each line in the subscriber bundle.

However this approach is sub-optimal whenever the capacity of the overall bundle is considered, hence more recent research has proposed (level 2) more intelligent methods to allocate the power for each bit in each tone, considering both near and far end crosstalk in the bundle. A major problem with the level 2 techniques is their computational complexity which currently renders them practically infeasible, e.g. ISB typically takes > 1 week to compute the tones for a 10-line bundle.

Graphic Processing Units (GPUs) represent a new approach to massively parallel floating-point computations. Moreover the Compute Unified Device Architecture (CUDA) developed by NVidia, represents a framework whereby new highly parallel algorithms can be developed. The main aim of this project is to apply this approach to level 2 DSM algorithms, many of which are highly parallel in their operation.

## Project Task Breakdown

* Standard Objectives
  + Become familiar with DSM techniques for DSL (Lvl1,2,3)
  + Become familiar with the CUDA environment for GPU programming
    - Identify a suitable development platform
  + Investigate current efficient implements of Level 2 DSM
  + Develop a massively parallel, GPU optimized implementation of a Level 2 bit-loading algorithm
* MEng Extensions
  + Analyse the performance of this generated GPU implementation in terms of
    - Execution speed
    - Cost (hardware)
    - Scalability (number of lines supported)
  + Compare the new design with existing implementations
* Additional tasks (not stated in Specification, added in concert with Prof Marshall and Dr McKinley)
  + Following on from the doctoral thesis of Dr A McKinley (McKinley & Marshall) re-implement his ADSL bundle simulation system in Python with support for GPU acceleration as a basis for the project
  + Optionally implement multiple selected Level 1, 2 and 3 algorithms for better comparison
  + Optionally attempt implementation of Genetic Algorithms to tackle optimization issues

# Project Task Detail, Explanation, and Status

## Research

* CUDA technologies and development strategies
  + CUDA, and the entire practice of using Graphics hardware for practical applications, is a relatively new technology[[1]](#footnote-2), and new computational optimizations are being developed and improved every day. Staying on the forefront of optimization papers and using resources developed by the original architects of CUDA (Kirk, 2010) has allowed me to further my understanding of the hardware and software considerations when designing algorithms for GPUs
* DSL DSM Techniques
  + While during the course of my studies I was tangentially taught about DSL systems, I felt a ‘back to square one’ approach was needed, and leveraging almost biblical texts (Starr, 2003) (Starr T. a., 1999) (Golden, 2007) I gained the contextual knowledge required to ensure that my implementations don’t fall out of real-world-scope
  + DSM is a mathematically complex and sometimes obtuse area, often dealing in areas of mathematics I had not even heard about let alone studied previously. Beyond the initial complexities of the area, it is a field of highly active research, meaning regular trips to the EBSCOhost Database to keep on top of related developments in the field; (Bezerra, 2010) (Lindqvist, 2008) (Charles Storry, 2008) (Miroslav Zivkovi ́, 2008) (Lindqvist, Lindqvist, Monteiro, Dortschy, Pelaes, & Klautau, 2010) (and many many more…)
* Scientific Computing with Python
  + While CUDA itself is largely based in C/C++, the rapid development cycle of Python, combined with its inherent ability to have highly optimized in-place C/C++ code allows the combination of the best of both worlds; bare-metal performance, and memory management; with abstract, object oriented design practices.
  + To improve my programming ability in a language that was fairly new to me, I undertook a two-day course on Scientific Computing in Python run by Source STC[[2]](#footnote-3)
* Genetic Programming with Python
  + Genetic Programming is an evolutionary methodology to ‘evolve’ programs that perform a task. Essentially, it is a machine learning technique that once presented with a problem space (or fitness landscape) will attempt many possible algorithms within a given solution space, and after running the algorithms and assessing their fitness, utilizes crossover and mutation to ‘breed’ a new ‘generation’ of algorithms, theoretically more suited to the problem space. This process is repeated until an optimization threshold is reached (depending on the application)
  + Python was ideally suited to this area; (Genetic Programming Meets Python, 2009) but its steep and abstract learning curve leaves GP as an academic curiosity, and is not the focus of this project. While some experimental work has been completed in this area, its low priority means little current focus is applied to it.

## Project Management

* Define project specification
  + This took the form of the task list found in the schedules <REFERENCE TO SCHEDULES>; this was done in concert with Prof Marshall and Dr McKinley to provide a stable but wide base on which to explore and build this project
  + Aside from allowing exploration of the more esoteric areas that this project could go in to (GP), it also limited the scope of the project to maintain focus on the major task. For example, there was discussion of re-implementing that onto a multi-system GPU clustering solution after the generation of a suitable GPU implementation, which would involve expansion into areas of optimization such as inter-system bandwidth, OpenMP / MPI[[3]](#footnote-4). It is clear now that this would have been over-stretching the project scope, and the limiting operation of clearly specifying the project has been very constructive
* Define Document Management System
  + Since a significant portion of this project is based on the work of Dr McKinley, it was decided that Dr McKinley would be able to supervise my documentation in real time using a system called Dropbox[[4]](#footnote-5)
  + Dropbox also ensured automated backup of documentation so there was no single-point-of-failure, so that progress would always be secured against accidents, as well as allowing for remote-work without the need of a laptop or pendrive.
  + As for document generation, notes, citations, and references were continuously added and cross-referenced on a private MediaWiki [[5]](#footnote-6)instance, which allowes for LaTeX[[6]](#footnote-7)-like math formatting, and acts as a project diary and outlining tool. Creation of ‘final’ documents such as this, is planned to be done on Microsoft Word (but may be migrated to pure-LaTeX time permitting)
  + Project Management was accomplished using Microsoft Project, using its task and resource based auto-scheduling to allow maximum efficiency while not over-burdening me, as well as allowing the ‘sectioning off’ of days to allow for work towards the Industrial Project, Exams, and Christmas Holidays. Of course, attractive Gantt Charts are always a good thing…
* Define Source / Issue Management System
  + BitBucket [[7]](#footnote-8)was chosen from a range of potential systems for revision control and issue tracking. The Mercurial revision control system allows for seamless multi-point code merging, and supports branching, enabling ‘experiments’ to be explored without jeopardizing the integrity of the project as a whole.
  + BitBucket’s issue tracking abilities allowed dynamic generation of ‘to-do’ lists, for instance; if there is a flaw in the logic of a function, it can be flagged as a ‘bug’. If the logic could be improved, it can be brought up as an ‘enhancement’, and if the functionality of areas of code can be significantly extended or augmented, it can be issued as a ‘proposal’. This meant that often-effervescent ideas could be instantly recorded, and the progress could be monitored, ensuring proper code coverage.

## Framework Design

* Develop and re-implement bundle / channel model framework
  + Dr McKinley provided his original C (inoperable) implementation of the bundle / channel model simulation framework, as well as some algorithm implementations. Due to C and Pythons significant differences, and ofttimes the obtuse and delicate state of Dr McKinley’s codebase, this resource was not always as useful as it could have been (See Page 9), but it was very informative as to some parts of the architectural development and the low-level computation involved.
  + The completed re-implementation is an object-oriented model of individual cables within a bundle, which allows for abstraction of some of the more computationally complex aspects of the simulation to be ‘hidden’.
* Develop in-framework implementations of selected level 1 and 2 DSM algorithms for later performance evaluation and comparison
  + <Milestone> New Framework returns the same results as the original framework
  + At time of writing, Iterative Water Filling (IWF) (Wei Yu, 2002), a Level 1 Algorithm, and Optimal Spectrum Balancing (OSB) (R. Cendrillon, 2004), a Level 2 Algorithm, have been successfully implemented and tested against historical results from Dr McKinley’s (Now Inoperable) implementation.
  + Additional exploratory work is currently under-way on implementing additional algorithms, but not at the expense of the main thrust of the project
* Specify Algorithm API with respect to base framework
  + Due to the Framework’s Object Oriented Structure, algorithms of any type or level have shared interaction functions with the currently running bundle. This functionality is currently restricted to ‘within’ the framework, but with a small amount of refactoring would allow arbitrary interaction with the bundle from outside the running simulation framework.
  + Whether exposing an explicit external API is an actually useful facet of the project is currently under discussion.
* Specify Benchmarking Schema for algorithm comparison
  + The current framework implementation has built-in timing constructs, and a sample set of input data is being ‘grandfathered’ in from Dr McKinleys previous work as to allow like-for-like algorithmic analysis.
  + The use of Pythons available profiling tools also allow for fine-grained analysis of runtimes and memory operations.

## Implementation

* Develop in-framework implementation of ‘Novel Algorithm’
  + Generate basic (CPU based) algorithm
    - This is currently in the middle stages of development and is currently based on Dr McKinley’s doctoral work, but is currently inoperable
  + Expand implementation to leverage GPU optimizations
    - Not Started (waiting on prerequisite)
  + Benchmark and iteratively optimize using profiling techniques
    - Not Started (waiting on prerequisite)
* Develop in-framework implementation of Genetic Algorithm
  + Generate basic algorithm leveraging CPU/GPU co-processing
    - Not Started
  + Benchmark and iteratively optimize using profiling techniques
    - Not Started (waiting on prerequisite)

## Comparison

* Analysis of implementations wrt speed, practicality and optimality
  + Not Started (waiting on prerequisite)
* Quantify improvement of new implementations
  + Not Started (waiting on prerequisite)
* Compare and contract structural changes
  + Not Started (waiting on prerequisite)

## Reporting

* Prepare interim report
  + <Milestone> Interim Report Submission
  + Well, you’re looking at it now, so I certainly hope its done.
* Introduction
  + Complete (Pending project resolution and redrafting)
* Theory & Background
  + Complete (Pending project resolution and redrafting including new reference documentation and additional graphics)
  + Includes discussion on DSL as a communications interface, introduces the problems that require DSM, explanation of DSM levels and exemplar algorithms, as well as references to ‘new’ algorithms (2008-present)
  + Also includes an introduction to Massively Parallel Programming with GPUs, GPGPU [[8]](#footnote-9)architecture and GPU/CPU co-processing
* Software Reference
  + Largely Complete
  + Includes discussion on the CUDA interface, Object Oriented Programming Structures, relevant Design Patterns used, list and introduction of libraries utilized. (includes reviews where relevant)
* Hardware Discussion
  + Largely Complete
  + Includes discussion on GPU/CPU memory management implications and selection of development and test hardware (test hardware not finalized)
* Design Decisions Made
  + Pending project resolution
  + Consists of dated relevant entries from my design diary and issue tracker, including notes on alternative options and though processes
* Design Heuristics Used
  + Ditto wrt Design Decisions Made
* End-System Interface Details
  + Pending Project Resolution
  + Explicit statement of operation of framework and programming interface for adding / editing algorithms.
* Presentation of Results
  + Pending Project Resolution
  + Not started, but will include qualitative discussions on the architectural differences between the selected competing algorithms and implementations; quantative and graphical comparisons of performance in terms of running time, hardware utilization, optimality, and I’m sure there will be some other metrics that have been generated by then…
* Finalisation of Report
  + <Milestone> Report Submission
  + Not even close
  + Re-drafting, collation of references, bibliography, appendices, etc

# Current Project Schedule

See Appendix A for Current Project Schedule

See Appendix B for Current Project Status and Network Diagram

The general status of the project is that a lot of sections are largely complete but waiting on ‘finishing touches’ and therefore I am not satisfied with them enough to say they are ‘done’

# Issues Experienced and Resolutions presented

## Time Management

I found few real issues; primarily the major issue during a significant section of the project so far has been that of time-management. I chair the EEE SSCC, sit on the CS SSCC, was a panelist for EEP, was starting a new QUB society (QUESTS) and trying to kickstart a community for technologists in Belfast. Combining that with Industrial Project work, Christmas Exam preparation, and interview and assessment centre visits, my initial schedule was completely unrealistic.

I have reduced my commitments to all of these in the New Year and expect to be back on track by early March

## Existing Implementation Difficulties

As has been stated, Dr McKinley’s implementation of the bundle simulation framework is quite obstuse, and as such, it is often difficult to see the larger structure of it. Further, due to a highly delicate and fixed-path-customized implementation, it is currently in-operable and is useful as source code only.

It has been very useful in bridging the cognitive gap between pure mathematical theory, as is found in most journal papers, and practical implementation issues.

This has not been a major hindrance to the project as Dr McKinley has made himself available to me when I have questions.

# Bibliography

Bezerra, J. a. (2010). An Evolutionary Algorithm for Improved Diversity in DSL Spectrum Balancing Solutions. *EURASIP Journal on Advances in Signal Processing* .

Charles Storry, M. a. (2008). Aspects of dynamic spectrum management level 3. *Bell Labs Technical Journal* *, 13* (1), 117-127.

*Genetic Programming Meets Python*. (2009, 08 06). Retrieved from PyEvolve: http://pyevolve.sourceforge.net/wordpress/?p=549

Golden, P. a. (2007). *Implementation and applications of DSL technology.* CRC.

Kirk, D. B.-m. (2010). *Programming Massively Parallel Processors: A Hands-on Approach.* Morgan Kaufmann Publishers Inc.

Lindqvist, N. a. (2008). Impact of crosstalk estimation on the dynamic spectrum management performance. *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE.* IEEE.

Lindqvist, N., Lindqvist, F., Monteiro, M., Dortschy, B., Pelaes, E., & Klautau, A. (2010). Impact of Crosstalk Channel Estimation on the DSM Performance for DSL Networks. *EURASIP Journal on Advances in Signal Processing* , 1-11.

McKinley, A., & Marshall, A. (n.d.). Novel Bit-Loading Alogrithms for Dynamic Spectrum Management in Digital Subscriber Lines.

Miroslav Zivkovi ́, G. K. (2008). Performance of digital subscriber line spectrum optimization algorithms. *Bell Labs Technical Journal* *, 13* (1), 129-146.

R. Cendrillon, M. M. (2004). Optimal multiuser spectrum management for digital subscriber lines. *IEEE International Conference on Communications*, (pp. 1-5).

Starr, T. a. (2003). *DSL advances.* Prentice Hall.

Starr, T. a. (1999). *Understanding digital subscriber line technology.* Prentice Hall.

Wei Yu, G. G. (2002). Distributed multiuser power control for digital subscriber lines. *IEEE Journal on Selected Areas in Communications* .

1. CUDA’s initial SDK release was in 2007 http://developer.nvidia.com/object/cuda\_1\_1\_downloads.html [↑](#footnote-ref-2)
2. ‘Scientific Computing with Python and SciPy’, See [www.source-stc.co.uk](http://www.source-stc.co.uk) for more details. [↑](#footnote-ref-3)
3. MPI:Message Passing Interface, used to provide inter-system communications in clustering applications. [↑](#footnote-ref-4)
4. Dropbox is a file hosting service operated by Dropbox, Inc. which uses cloud computing to enable users to store and share files and folders with others across the Internet using file synchronization. www.dropbox.com [↑](#footnote-ref-5)
5. MediaWiki is a popular free web-based wiki software application. www.mediawiki.org [↑](#footnote-ref-6)
6. LaTeX is a document markup language and document preparation system for the TeX typesetting program. [↑](#footnote-ref-7)
7. BitBucket is a web-based hosting service for projects that use the Mercurial revision control system. [↑](#footnote-ref-8)
8. GPGPU: General Purpose GPU, largely synonymous with GPU’s being used in an application other than Graphics or Gaming [↑](#footnote-ref-9)