final report

Project code: B.BSC.0045

Prepared by: Dr John Henshall

CSIRO Livestock Industries

Date published: June 2009

PUBLISHED BY

Meat & Livestock Australia Limited
Locked Bag 991
NORTH SYDNEY NSW 2059

Computational analysis of genetic data using field programmable gate arrays

Abstract

With the rapid increase in the quantity of molecular genetics data being collected for livestock, analyses using sequential computers are becoming the rate limiting step. Concurrently, the exponential increase in computer speed over the last decades has slowed, with recent improvements due to multi-core processing rather than clock speed alone. This project successfully applied new and emerging computer technologies to capitalise on the investments in molecular genetics made by the Australian livestock industries, with the goal of allowing the efficient incorporation of molecular genetics data into industry genetic evaluations. The focus was on parallelizing problems to run on processors with multiple cores, ranging from quad-core CPUs, graphics processors with hundreds of cores, and user configurable hardware devices with thousands of cores.

Executive Summary

The quantity of data recorded on industry flocks and herds for genetics purposes has increased exponentially over the last decades and, with the advent of dense SNP data, this increase is not slowing. There is a need to improve the speed both of routine analyses and of research analyses. It is noteworthy that in the days of interval mapping based on microsatellites, permutation testing was commonly accepted as the only method able to provide believable measures of significance. Now, permutation testing is less frequently mentioned, as with analyses taking hours per trait, the cost of tens of thousands of permutations is very high. Thus, with increased data has come less rigour in the analyses, due to computational cost. This project was focussed on reducing the limitations due to computational cost.

While the quantity of data has increased, so to has the power of computers. Increases in computer power have been exponential, but in recent years the increase has been due more to multiple cores per CPU than to increases in clock speed. To exploit these increases then requires that computations be run in parallel, on multiple cores at once. Recently, two other hardware platforms for parallel computing have become available to industry. First, tools for implementing scientific computations on general purpose graphics processors (GPGPU) became freely available in the last two years, and these devices can be used as massively parallel numeric processors. Second, field programmable gate arrays (FPGA),which are user configurable computer chips, were until recently too small and too expensive for use in routine computations. Now however, their size and price point is well within the reach of medium sized enterprises.

The intent of this project was to investigate the potential of using emerging computer hardware devices, such as multi-core CPU, GPGPU and FPGA, to accelerate the computations on genetics and genomics data for Australian livestock species.

In this project, algorithms for the analysis of genetics/genomics data on all three of these multi-core hardware platforms were successfully implemented. There are two areas in these results that are directly and immediately applicable. First: applying available optimised routines for linear algebra to existing genetic evaluation systems. A direct result of this project has been AGBU's recognition of the need to investigate the development of BLUP optimised for multi-core processors. Further, it is evident that components of software to estimate SNP effects using linear models can also be greatly improved for only a modest investment. Second: the estimation of genetic merit from dense SNP data. It is difficult to predict which models will ultimately be used for this purpose, and it is likely that different models will be used depending on population structure, costs of recording, position the genetics supply chain, and genetic parameters for the traits of importance. The results from this project keep open the option to use analysis methods that would otherwise be considered too computationally expensive. An example is the estimation of haplotype relationships under a coalescent model. The software developed in the project for this purpose can be used on existing industry SNP datasets.

The ultimate benefit from improving the speed of computations on genetics and genomics data arises from more accurate estimates of genetic merit, i.e. more accurate EBVs. The path to this benefit is through the ability to fit models to the data that, although computationally demanding, better describe the underlying characteristics of the data. The return on industry's investment in genomics will be greater as the information contained in the data will be better integrated with EBVs derived from pedigree and phenotype data. The beneficiaries of the increased accuracy in estimates of genetic merit will be the livestock breeder, and the producer who benefits from the superior genetics.

Contents

1 Background

Over the last 5 years, the quantity of genomics data on Australian livestock populations has gone from the order of a few thousand microsatellites assayed on a few thousand individuals to tens of thousands of single nucleotide polymorphisms (SNP) assayed on tens of thousands of individuals. The impact of this increase on analysis is more than just a proportional increase in the time taken by the computations, as the density of the data allows far more complex questions to be asked of it. The statistical models and analysis methods appropriate for low density microsatellites fail to exploit the information contained in high density SNP data, and are of very limited utility in analysing data from the experimental designs favoured today. A variety of methods for the prediction of genetic merit from dense SNP data have been proposed, and some have been implemented in some industries. As data accumulate the methods most useful for industrial deployment will be identified and validated. However, even with the quantity of SNP data already collected, some methods are computationally infeasible in a reasonable time using standard approaches to computing. Data collection with current SNP platforms (~50K SNP) continues, and the next generation of SNP platforms (~300K SNP) are likely to be in use within a few years. There is no reason to expect that the exponential increase the quantity of genomics data will not continue in the short to medium term.

Advances in computer processing power have also been exponential, as described in the well known Moore's Law. Until recently, these advances were due to more transistors per CPU, running at higher speeds. This path to faster computing is reaching its limits, and in recent years we have seen CPU improvements due largely to increasing the number of cores in a CPU (e.g. Intel Core 2 Duo) rather than increasing the power of a single core CPU. To exploit the multi-core properties of the processor requires that the computations being processed can run in parallel. On an office desk-top computer this is often the case, multiple programs are running, with little interaction between them, and to benefit from multi-core processors requires few or no changes to the software. This is not always the case for scientific computations such as those required to process genomics data. For some algorithms, computations on a single region of the genome might take several hours, and ideally would take into account the results from computations on other regions of the genome. To apply multi-core processors to such problems requires far more intervention on the part of the designers of the software.

2 Project Objectives

The objectives of this project were:

- to determine whether multi-core processors could be applied to the processing of genetics/genomics data.
- to establish which analysis problems could most benefit from a multi-core approach.
- to determine whether the speed advantages in processing justified any additional development costs.
- if feasible, to produce a prototype system for an industry relevant problem.

3 Methods

Three broad areas in processing genetics/genomics data were considered in this project:

1. Allele transmission through pedigrees to estimate allelic probabilities. For other than very small pedigrees or pedigrees with no inbreeding, exact computation of these probabilities is infeasible. The probabilities allow the estimation of the exact relationship between individuals at each location on the genome given the pedigree and the SNP data, which would allow better prediction of genetic merit.
2. Estimation of the relationship between haplotypes using population genetics models such as the coalescent model. These models do not require pedigree data, essentially the relationships between the base animals in the pedigree are estimated given the SNP data. For other than very small datasets, simplified models are required to make these computationally feasible, such as those of Meuwissen and Goddard (2001, 2007). An understanding of haplotype relationships would allow better prediction of genetic merit.
3. Solving the BLUP equations. This requires computations that are very similar to those required in many linear algebra applications, such as repeated matrix multiplication. Faster BLUP computations would allow the fitting of more complex models, in particular models that incorporate the estimates obtained from 2. above.

Three hardware classes were used:

A. Field Programmable Gate Arrays (FPGA). The initial focus of the project, FPGA consist of an array of user-configurable gates on a computer chip, allowing the design and implementation of a custom computer chip for a specific application, in our case, a genetics processor. The design commonly consists of many very simple, specific purpose processors, that can run in parallel. For example, there might be one processor for each animal in the pedigree, or one processor for each SNP on the chromosome. The circuit is loaded onto the chip at run time, so the FPGA can be re-used for multiple problems. FPGA are available on PCI cards that fit in a standard desktop computer. Our analyses were on a Virtex-5 SX-50 card, using the Xilinx ISE Foundation tools. The development tools cost around $3,000 and the PCI card around $1,500.
B. General Purpose Graphics Processor Units (GPGPU). These can contain up to 720 single-precision floating point calculators on a single chip. They have been developed from technology for consumer graphics cards, and are inexpensive and fast. The two main suppliers, Nvidia and AMD have Software Developer Kits allowing programmers to develop custom algorithms for the devices. Our analyses were on a low-cost Nvidia 8600 GT card (currently around $100). The development tools are supplied at no charge by Nvidia. Larger cards aimed at consumer use are approximately $500 per GPGPU.
C. Multi-core CPU. CPUs such as the recently released Intel Corei7 processor are easily

programmed and inexpensive. They have 4 processing cores per chip, each capable of running two independent algorithms, providing 8 virtual cores. Intel has produced Basic Linear Algebra sub-programs (BLAS) library algorithms, including highly optimised double precision matrix multiplication, for Intel multi-core CPU. These are specifically designed for computations such as those in 3. above. We have used a PC with an Intel Corei7 920 CPU. These have a premium of around $250 over current typical desktop CPU for approximately 2.5 times the performance.

4 Results and Discussion

4.1 Simulating allele transmission through pedigrees to estimate identity by descent probabilities.

This is a computationally hard problem, where increasing the number of animals in the pedigree produces an exponential increase in the computing time required. We successfully developed both software and hardware (FPGA) based approaches, building on the earlier work reported in Henshall and Little (2006). Despite the invention of a novel implementation of the genotype elimination algorithm in FPGA, our final conclusion was that this was not a fruitful area of research and unlikely to lead to a major advantage to hardware based implementations. However, an unexpected output from this research was an improved algorithm for computing exact genotype probabilities in small pedigrees using conventional CPU. By traversing the space of possible solutions more efficiently the algorithm is faster, and therefore able to operate on larger datasets in a reasonable time. A manuscript reporting this has been prepared which will be submitted to a peer reviewed journal.

4.2 Estimation of the relationship between haplotypes based on population genetics models.

Our first task was to obtain conventional sequential software to estimate haplotype relationships as in Zöllner and Pritchard (2005). A review of available software failed to identify anything suitable for our needs and we chose to write our own. This had the added advantage of greatly increasing our understanding of likelihood and inference under the coalescent model, which was of great benefit when applying hardware based approaches. A fully operational version of the software was completed in early September 2008, comprising 1600 lines of code (Fortran 95). Since then this has been thoroughly tested and deficiencies highlighted. The main deficiency is speed, performing inference on haplotypes under the coalescent model is well known to be computationally challenging and so this was not unexpected. Despite this, we believe that the software is suitable for estimating haplotype relationships to exploit further the data obtained in the large SNP experimental populations for sheep and cattle. We would like to evaluate the software on one of these datasets.

4.2.1 Implementation on FPGA

The complete haplotype relationship algorithm has many levels, with an inner loop called many times to evaluate the likelihood of sampled relationship trees. It is the inner-most loop that was implemented on FPGA, called from the PC. On the FPGA used in the study we successfully implemented the algorithm for four SNP haplotypes, using 16 bit floating point numbers (3 significant digits), occupying approximately 70% of the FPGA. A similarly priced FPGA, the

Virtex5-FX70T, is predicted to be able to handle 32 bit floating point (7 significant digits) for 4-SNP haplotypes, using the design implemented. A PC uses 64 bit floating point (15 significant figures), but given the nature of stochastic algorithms such as this one, this accuracy may offer no advantage over 32 bits.

The inner loop of the algorithm took 989ns on a single “Core 2” processor core running at 2.67GHz. On the FPGA this operation took just 10ns, running the FPGA at 100MHz. That is a speed multiple improvement of 99 times. The FPGA could, theoretically, be increased in speed to at least 200Mhz. Alternatively, multiple core CPUs could be utilised to also give an increase in speed. It should be noted however, this is the result for this particular experiment. It does not automatically imply that every algorithm can be sped up by a similar amount by utilising FPGA instead of general purpose CPUs. Further, development costs are higher for FPGA. This will be discussed further below.

4.2.2 Implementation on GPGPU

Implementing the inner loop of the algorithm on a GPGPU proved to be relatively straight forward, much more so than designing a circuit for an FPGA. Our naïve implementation of the inner-loop achieved a demonstrated 8 fold increase in speed over the fastest multi-core CPU available. This should increase to approximately 20 fold for the latest dual chip GPGPU boards.

4.2.3 Optimum implementation

Table 1 below compares the effort and cost of the technologies for implementing the inner-loop of the coalescent algorithm on the 3 hardware platforms considered. Although FPGA are likely to have the performance edge per device, the considerable effort to configure (i.e. program) them negates their benefit, unless the problem being addressed is static, requiring many repeated runs with no reconfiguring of the FPGA required.

Table 1. Comparison of three hardware platforms for implementing the inner-most loop of the software to estimate the relationships between haplotypes under a coalescent model.

Technology	Additional software development cost after CPU implementation	Hardware cost	Approx. Speed Increase over single core of Intel Core2	Approx. Speed Increase over quad-core Intel Corei7
CPU	$ 0	$ 2,000	1x	1x
FPGA	$ 100,000	$ 1,500 (V5 SX50)	100x	20x
		$ 15,000 (V5 FX200)	400x*	80x*
GPGPU	$ 2,000	$ 100 (Nvidia GT8600)	5x	1x
		$ 400 (Nvidia GT9800)	40x*	8x*

		$ 900 (Nvidia GTX 295)	100x*	20x*

* Expected speed increase

4.2.4 Implementing the complete system

In the comparison above, the assumption was that only the inner loop of the algorithm runs on the hardware device. For FPGA this is almost a requirement, as it would take a huge effort to design a complete system on a chip. However, there is no such limitation with GPGPU. Currently, only C based computer languages are available for GPGPU. Our haplotype relationship estimation program was written in Fortran, so there was a cost in porting it to C for the GPGPU, but this will not be a cost once Fortran tools become available (Fortran95 compilers are due for release soon).

In Table 2 the comparison between CPU and GPGPU for implementing the whole haplotype relationship estimation program is presented. The complete program was not altered other than re-coding in C, and no optimisations were made to suit the GPGPU. Nevertheless, the GPGPU was as fast as a single core of an Intel Core 2, and an increase of 4 times over the Intel Corei7 multi-core CPU is likely with GTX 295 device with only a marginal increase in development costs. This is before any effort is made to optimise the algorithm for greater performance on the GPGPU. Two of these devices could fit into a computer giving an 8x speed increase.

Table 2. Comparison of two hardware platforms for implementing the complete software for estimating the relationships between haplotypes under a coalescent model.

Technology	Additional Software cost after CPU implementation	Hardware cost	Approx. Speed Increase over single core of Intel Core2	Approx. Speed Increase over quad-core Intel Corei7
CPU	$ 0	$ 2,000	1x	1x
GPGPU	$ 2,000	$ 100 (Nvidia GT8600)	1x	0.2x
		$ 400 (Nvidia GT9800)	8x*	1.6x*
		$ 900 (Nvidia GTX 295)	20x*	4x*

* Expected speed increase

4.3 Solving the BLUP equations.

The key bottleneck for accelerating BLUP is the speed of a matrix multiplication. The required calculation is the repeated multiplication of a constant, double precision floating point matrix with size of around 40 x 40, by a dense matrix of 1.5 million x 40 double precision floating point numbers, producing a matrix of the same size. This requires 0.5 GB of memory. Matrix multiplication is a very common operation in linear algebra computations and highly optimised code is available for most hardware platforms. Consequently, it is not efficient to develop code for this operation from scratch.

4.3.1 Implementation on FPGA

Matrix multiplication code is not readily available, and as code development is much more expensive for FPGA than for other hardware platforms there is little point in pursuing this option.

4.3.2 Implementation on GPGPU

Software libraries for matrix multiplication are provided by the hardware companies. Using these we were able to implement the matrix multiplication component of the BLUP solver in less than a day. The speed improvement was of the same order as can be obtained using optimised code on a multi-core CPU, but there are additional overheads in transferring data to and from the GPGPU card.

4.3.3 Implementation on multi-core CPU

Implementing the matrix multiplication component of the BLUP solver using the Intel BLAS double precision matrix multiplication on a multi-core Intel based computer took only a few days of development time. A relatively small engineering effort of around 1 man-month would be required for full integration into existing systems. This should realise an 80 to 120 times speed increase for the 80% of the time currently devoted to matrix multiplication in the BLUP solver. Given that the remaining 20% will remain unchanged; this will deliver an overall increase of approximately five-fold.

4.4 General Discussion

A number of important developments in hardware design occurred during the course of this project. When the project commenced multi-core CPU were rare other than in high end computers and no tools for exploiting the multiple cores were available. GPGPU were available for scientific programming, but there were no tools so development was expensive, especially as there was no guarantee that code developed for today's GPGPU would run on tomorrow's faster and more powerful GPGPU. FPGA were small, but are becoming large enough for small problems. Now, at the end of the project, multi-core CPU are standard on desktop computers, and tools to exploit the multiple cores are available in several languages (Fortran and C). The GPGPU vendors are promoting GPGPU for scientific computing and have developed BLAS libraries (currently only C, but Fortran due soon). FPGA are large enough for industrial scale problems, but development tools are nowhere near as developed as those for multi-core CPU and GPGPU. Given these developments the following recommendations regarding choice of hardware can be made:

- FPGA. Massively fine grained parallelisation is possible and custom design can map

the algorithm to fully exploit the hardware. However, development costs are very high, and there are considerable overheads in transfer of data between the FPGA and off-card memory. As such, to justify the development costs requires that the computations will be run many times with no changes to the algorithm. That is, routine analyses rather than one-off analyses. Further, the algorithm must map well to the FPGA, requiring many repetitions of a task with few changes to the data. Sampling based algorithms can be very well suited to FPGA, and if the end use of the data analysis justifies the development cost then an FPGA implementation should be considered.

- GPGPU. The benefit of GPGPU over FPGA is in the lower cost of development, assuming that the off-the-shelf tools are used. However, using these tools results in much less flexibility in mapping the algorithm to the hardware, so speed-ups are likely to be significantly less than in an optimal FPGA (or GPGPU) implementation. Sometimes the algorithm might be one that the tools can map well to the hardware, in which case a GPGPU implementation will be very fast.
- Multi-core CPU. These will have far fewer cores than the number processors possible in FPGA and GPGPU, but each core can perform more complex computations. For problems requiring standard linear algebra operations, the highly optimised BLAS tools for multi-core CPU allow cheap, and very fast, implementations.

In the short term, using the tools that are becoming available to fully exploit multi-core CPU is a cost effective way of obtaining a significant speedup in computations. This is particularly the case when the computations required are widely used linear algebra subroutines. For these the hardware companies are investing in producing very highly optimised code that runs an order of magnitude faster than code generated by a regular compiler. Similar code for GPGPU exists, but it achieves its maximum speed for only a restricted set of problems that happen to hit the “sweet spot” of the GPGPU. To fully exploit the power of the GPGPU would require that the algorithm for the analysis of genetics data be re-formulated to map to the GPGPU “sweet spot”, requiring an investment in software development.

5 Success in Achieving Objectives

The objectives of the project have been met. We have demonstrated that all three of the hardware platforms tested can be applied to genetics/genomics data analysis tasks. In some cases the benefit far outweighs the cost. An example is implementing a BLUP solver that exploits multi-core CPU. AGBU are beginning to plan for this. We have also produced software for estimating haplotype relationships under a coalescent model and using these to estimate SNP effects. We believe that the software in its current state is suitable for analysing at least selected regions in the whole genome selection experimental populations currently available. Finally, our research into allele transmission in pedigrees to estimate allelic probabilities has resulted in a new method that can also be applied to existing datasets. Two journal papers will be published based on the research outcomes of this project.

6 Impact on Meat and Livestock Industry

The use of genomics by the Meat and Livestock Industry is accelerating, and it is over the next 5 years that measurable impact should occur. Leading to faster computations on genomic data, this research will allow better use of the data arising from the industry investment in genomics, producing breeding value estimates of greater accuracy in young animals.

7 Conclusions and Recommendations

In this project we successfully implemented algorithms for the analysis of genetics/genomics data on all three of the examined multi-core hardware platforms. There are two areas in these results that are directly applicable.

1. Utilising available optimised BLAS routines for existing genetic evaluation systems. A direct result of this project has been AGBU's recognition of the need to investigate the development of BLUP optimised for multi-core processors. Further, it is evident that components of software to estimate SNP effects using linear models can also be greatly improved for only a modest investment.

2. The estimation of genetic merit from dense SNP data. It is difficult to predict which models will ultimately be used for this purpose, and it is likely that different models will be used depending on population structure, costs of recording, position the genetics supply chain, and genetic parameters for the traits of importance. The results from this project keep open the option to use analysis methods that would otherwise be considered too computationally expensive. An example is the estimation of haplotype relationships under a coalescent model. The software developed in the project for this purpose can be used on existing industry SNP datasets, although further development would be required if it was to be for routine and repeated whole genome data.

It is noteworthy that in the days of interval mapping based on microsatellites, permutation testing was commonly accepted as the only method able to provide believable measures of significance. Now, permutation testing is less frequently mentioned, as with analyses taking hours per trait, the cost of tens of thousands of permutations is very high. With increased data has come less rigour in the analyses, due to computational cost. This alone suggests that a continued investment in accelerating the computations is warranted.

8 Bibliography

Henshall JM and Little BA (2006). Parallel computations on pedigree data through mapping to configurable computing devices. *Genetics Selection Evolution* **38**: 265-279

Meuwissen THE and Goddard ME (2001). Prediction of identity by descent probabilities from marker-haplotypes. *Genetics Selection Evolution* **33:** 605-634

Meuwissen THE and Goddard ME (2007). Multipoint identity-by-descent prediction using dense markers to map quantitative trait loci and estimate effective population size. *Genetics* **176**: 2551-2560

Zöllner S and Pritchard JK (2005) Coalescent-Based Association Mapping and Fine Mapping of Complex Trait Loci. *Genetics* **169**: 1071-1092