Genet. Sel. Evol. 38 (2006) 265–279

DOI: 10.1051/gse:2006003

Original article

Parallel computations on pedigree data through mapping to configurable computing devices

John M. HENSHALL*, Bryce Alvin LITTLE

FD McMaster Laboratory Chiswick, CSIRO Livestock Industries, Armidale, New South Wales 2350, Australia

(Received 16 August 2005; accepted 18 November 2005)

Abstract – Pedigree data structures have a number of applications in genetics, including the estimation of allelic or haplotype probabilities in humans and agricultural species, and the estimation of breeding values in agricultural species. Sequential algorithms for general purpose CPU-based computers are commonly used, but are inadequate for some tasks on large data sets. We show that pedigree data can be directly represented on Field Programmable Gate Arrays (FPGA), allowing highly efficient massively parallel simulation of the flow of genes. Operating on the whole pedigree in parallel, the transmission of genes can occur for all individuals in a single clock cycle. By using FPGA, the algorithms to estimate inbreeding coefficients and allelic probabilities are shown to operate hundreds to thousands of times faster than the corresponding sequentially based algorithms. Where problems can be largely represented in an integer form, FPGA provide an efficient platform for computations on pedigree data.

FPGA / parallel computations / pedigree data

1. INTRODUCTION

Pedigree data structures combine information from multiple individuals allowing genetic effects to be distinguished from environmental effects. The structure is highly regular: in diploid species each individual has exactly two parents, whose identity may or may not be known. Each individual may have records of genotype, either molecular markers from a segment on the chromosome, or implied genotype through an observed presence or absence of a genetically determined characteristic. Quantitative trait measurements may also exist for some or all individuals.

Through the process of meiosis, at each locus, each (diploid) individual inherits exactly one allele (of two available) from each parent. Since each meiosis

* Corresponding author: john.henshall@csiro.au

event is a binary choice, the set of meioses can be represented as a set of meiosis indicators, each taking the value of zero or one. When coupled with a set of genotypes for individuals without known parents (founders), this set of meiosis indicators completely specifies the genotype of all individuals in the pedigree. The likelihood of a set of meiosis indicators and associated set of genotypes for founders is a function of the allele frequencies in the founders and, in the case of multilocus pedigree data, the number of recombinations observed between linked loci.

Sequential algorithms exist to process pedigree data. If the genotype at a locus or at loci is known for some individuals, maximum likelihood (ML) methods are used to estimate the genotype and identity by descent (IBD) probabilities for the remaining individuals (*e.g.* [3, 16]). These tasks are relevant in human and livestock species in order to estimate disease status (carrier *vs.* non-carrier), to seek associations between marked loci and disease or other phenotypic characteristics, and to reconstruct haplotypes (the set of linked loci on a chromosome). Since exact ML methods require that the likelihood be evaluated for every possible combination of meioses, algorithms have been developed that exploit the structure of the data to speed up the computations (*e.g.* [1, 2, 8, 9, 11]), but even then, exact ML is not currently feasible for large complex pedigrees (those with marriage or inbreeding loops). For these pedigrees, the estimates of the probabilities of interest can be obtained using sampling based algorithms [4–6, 10, 17], the most simple of which is the gene dropping algorithm [12], where alleles are simulated for founders and meiosis indicators are simulated for individuals with parents. Genotype and IBD probabilities can be estimated by accumulating samples that are consistent with the observed data.

There may be thousands of individuals, genes or marked loci in the pedigree, and sequential algorithms typically use loop constructs to iterate through the data. Often the problem is combinatorial, and the computational cost comes from the need to process large numbers of loops within loops. Some of the sequential algorithms could be parallelised by running them on clusters; with essentially each processor in the cluster performing sequential computations for a part of the outermost loop. In this case, the speedup (time taken using the serial implementation divided by the time taken using the parallel implementation) is at best the number of processors used in the parallel implementation.

An alternative to using a cluster of processors to parallelise an algorithm is to use a processor that can internally parallelise the data structure. It is not cost effective to manufacture a processor that can only be used for pedigree data, especially since it is likely that each data set would require a custom

designed processor. However, processors that can be reprogrammed to suit specific analysis tasks are available, and are becoming more powerful. Field Programmable Gate Arrays (FPGA) consist of an array of reprogrammable logic cells that can be configured to represent fine grained data structures. Salwinski and Eisenberg [15] show their use for biological data, using them for *in silico* simulation at the molecular level, where the parallelism is the result of many reactions occurring in the cell simultaneously.

In pedigree data, meiosis events for other than the direct ancestors of an individual cannot affect the individual's genotype. In this paper we show that this allows meioses to be simulated in parallel for layers of individuals in a pedigree, and that the FPGA is an ideal tool for this simulation. We show that algorithms such as gene dropping can be implemented on FPGA, with a new sample for the whole pedigree achievable each clock cycle. We also discuss the implementation of FPGA based parallelisation of more sophisticated sampling algorithms for pedigree data.

2. MATERIALS AND METHODS

2.1. FPGA representation of a pedigree

The essential elements of our FPGA representation of a pedigree appear in Figure 1. Individuals are represented by modules, which are arranged in layers (0:m), with founders (modules for individuals without parents) in layer 0 and descendants in subsequent layers. Each layer represents one generation. Where individuals produce descendants over more than one generation, modules, called holders, are utilised to pass on the alleles while remaining synchronised with the rest of the alleles dropping through the pedigree. Holders have only one parent but may have multiple progeny, and contain only memory to store the two alleles as they pass through the module. Modules other than holders contain data and storage for the results from computations on the individual.

At each clock cycle, alleles for founders are generated. These may be random, fixed or generated in a systematic way (*e.g.* enumerating a set). Meiosis indicators are also generated for each descendant module, and may also be random or systematic. Computations on founder, descendant and holder modules are completed in one clock cycle of the system, during which each descendant receives one allele from each of its parent modules, or both alleles from its single parent module in the case of a holder. The holder modules ensure that all alleles in layer i at cycle t dropped through layer $(i - 1)$ at cycle $(t - 1)$ and

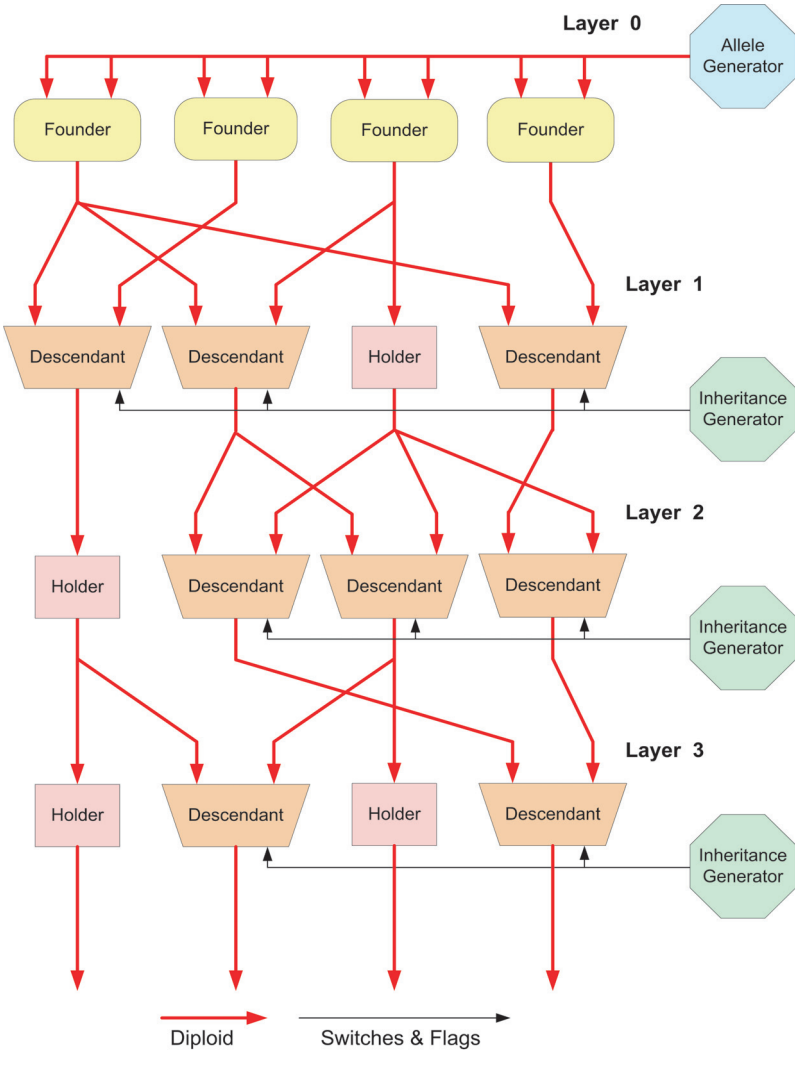

Figure 1. Pipelined FPGA representation of pedigree.

originated in layer 0 at cycle ($t - i$), allowing the system to be pipelined. That is, at any clock cycle, modules in the same layer are processing the same sample, but modules in different layers are not, with as many samples propagating through the pedigree as there are layers. When the alleles drop through layer m the sample is complete. Since this occurs during each clock cycle, a new sample for the whole pedigree is generated each clock cycle.

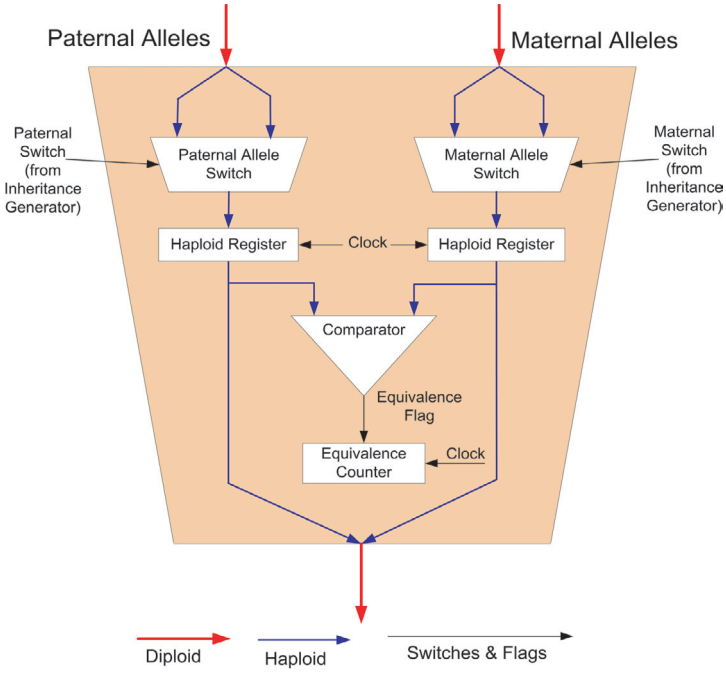

Figure 2. Individual module for estimating inbreeding coefficients.

2.2. Example applications

2.2.1. Estimating inbreeding coefficients

The first application considered is the estimation of inbreeding coefficients. This is a problem for which efficient sequential algorithms exist (*e.g.* [14]), but it serves well to introduce the methods. The coefficient of inbreeding (F) for an individual can be described as the probability that the alleles carried at a random locus are IBD. As such, the gene dropping algorithm estimates F as the proportion of samples for which the alleles inherited by an individual are IBD. For this problem, the structure of a descendant module appears in Figure 2. All that is required is a comparator to check whether the alleles are IBD and a counter to increment when they are. Founder modules are assigned a constant pair of alleles, with only one copy of each allele occurring in layer 0. Meiosis events can be either pseudo random or, if computationally feasible, the complete set of possible meiosis events can be enumerated to produce an exact solution. When programmed on an FPGA, the allele transmission between parents and offspring, the comparison for all individuals and the incrementing of the counters all take place in a single clock cycle. This is regardless of

Table I. Pedigree used for estimation of inbreeding coefficients (F).

ID	Father	Mother	F
1	0	0	0
2	1	0	0
3	0	2	0
4	1	0	0
5	1	2	0.25
6	0	2	0
7	5	0	0
8	0	2	0
9	3	8	0.125
10	3	6	0.125
11	7	4	0.09375
12	3	8	0.125
13	5	2	0.375
14	3	6	0.125
15	11	4	0.296875
16	7	2	0.1875
17	13	4	0.15625
18	13	16	0.34375
19	9	16	0.171875
20	0	16	0
21	3	2	0.25
22	11	0	0
23	13	0	0
24	15	18	0.173828
25	19	18	0.332031
26	5	14	0.1875
27	19	16	0.382813
28	25	18	0.501953
29	7	28	0.277344
30	21	8	0.1875
31	23	2	0.21875
32	15	28	0.160156

the number of the individuals in the pedigree, provided that the FPGA is large enough to store all of the individual modules.

Pedigrees of 32 individuals (Tab. I) and 60 individuals (not shown) were used to compare the FPGA implementation of the gene dropping algorithm with a sequential implementation of the gene dropping algorithm.

2.2.2. Estimating genotype probabilities

The second application considered is the estimation of single locus genotype probabilities where marker information is available on some individuals. Again, the gene dropping algorithm is applied, but founder modules differ from those used to estimate F in that alleles are no longer unique, and are sampled from a distribution. By sampling alleles in the founder layer from the appropriate multinomial distribution, the calculation of the component of the likelihood due to the allele frequencies in the founder layer is avoided, and a test for consistency with the observed data is all that is required. Where inconsistencies between the sampled and observed genotype are identified, the likelihood of the sample is zero, otherwise it is non-zero and the sample can be accumulated. The results can no longer be accumulated in the individual's module because whether the likelihood is zero is not known until alleles have dropped through layer *m*. The storage for results for all individuals is in a terminal layer of allele counters below layer *m*, with holder module chains of appropriate length connecting individual modules to modules in the terminal layer (Fig. 3).

The valid allele signals from all descendants are compared and only if they are all valid is the master valid signal set to on, allowing the count of all alleles in the sample to proceed. Founder modules are identical to descendant modules apart from the source of alleles. On completion, the probability that an individual has a particular genotype is the proportion of samples of non-zero likelihood for which the individual had that particular genotype.

This algorithm was applied to an inbred pedigree of 11 individuals, with four alleles of equal frequency in the founder layer and with genotypes assumed known for four individuals. The pedigree appears in Table II. The algorithm was also applied to a 20 individual pedigree (not shown) that included one additional individual with known genotype.

2.2.3. Multi-locus pedigree data

The final application considered is the estimation of genotype probabilities for multiple linked loci. Here the likelihood is a function of the allele frequency in the founder layer and consistency with observed data as before, but also of the number of recombinations in the sample and the assumed recombination rate. Again the calculation of the component of the likelihood due to the allele frequency in the founder layer is avoided by sampling from the appropriate multinomial distribution. Similarly, the calculation of the component of the likelihood due to recombinations is avoided by sampling recombinations

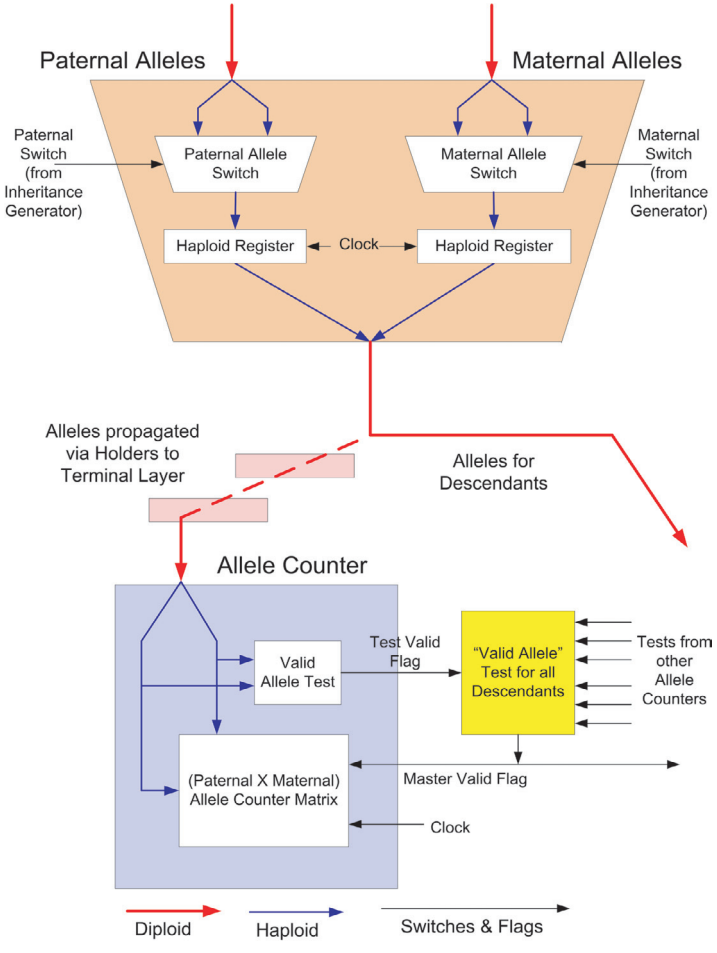

Figure 3. Individual module, chains of holders and module from terminal layer for estimating genotype probabilities.

from the appropriate binomial distribution. Instead of counting alleles as in Figure 3, the inheritance of haplotypes, or sets of alleles at linked loci on one chromosome, are counted.

This algorithm was applied to the pedigree of four individuals in Table III. The assumed recombination rate between the two loci was 0.125; allele frequency assumptions were not required for the founder layer since for these individuals, genotype was known. Speed was compared to a sequential version of the same algorithm.

Table II. Single locus pedigree used for estimation of genotype probabilities.

ID	Father	Mother	Genotype
1	0	0	AB
2	0	0	AD
3	0	0	?
4	2	1	?
5	2	3	?
6	2	4	DD
7	5	4	CD
8	5	3	?
9	6	7	?
10	8	7	?
11	10	9	?

Table III. Multi-locus pedigree used for estimation of genotype probabilities.

ID	Father	Mother	Genotype
1	0	0	A–A B–B
2	0	0	C–C C–C
3	1	2	A–A C–C
4	1	2	B–? C–?

2.3. Hardware and software

A Xilinx Spartan 3 (XC3S400) FPGA operating at 50 MHz was used for the FPGA computations. Software was written in VHDL and Picoblaze™ Assembler. VHDL is a hardware description language that allows any digital electronics circuit to be represented in a text format. The FPGA was configured using VHDL, after pre-processing the pedigree data to identify valid and invalid allele combinations. This was done to minimise the space required for storage of results on the FPGA rather than to speed up the algorithms. Likewise, the VHDL was adjusted to minimise the size of the FPGA space required. A Picoblaze soft processor was configured to provide an interface to the general purpose computer. Picoblaze is the VHDL code for an 8-bit microprocessor. It is available for general use, royalty free from the FPGA manufacturer, Xilinx. The Picoblaze processor is programmed into the FPGA along with a controlling programme. The *allele and inheritance generators* used *cellular automata random number generators* [7], chosen because of their suitability for implementation on an FPGA, as well as their ability to produce high quality pseudo-random numbers.

Table IV. Comparison of FPGA and general purpose CPU: estimating inbreeding coefficients.

Number of individuals in pedigree	Samples per s (3.0 Ghz Pentium)	Samples per s (50 Mhz FPGA)	Speed advantage for FPGA	Samples per s (3.8 GHz Pentium)[a]	Samples per s (390 Mhz FPGA)[ab]	Speed advantage for FPGA[ab]
32	**575 000**	**50 000 000**	**87**	*728 000*	*390 000 000*	*536*
60	**301 000**	**50 000 000**	**166**	*380 000*	*390 000 000*	*1030*
1000	*17 500*	*50 000 000*[a]	*2860*[a]	*22 150*	*390 000 000*	*17 500*

[a] Estimate only by extrapolation shown in italics.
[b] Maximum clock speed estimate is 390 Mhz for Xilinx Virtex 4 devices.

The general purpose CPU used for comparison was a 3 GHz Pentium 4. The parallel algorithms were coded in sequential form in Delphi^{TM}, and run on the PC for a direct comparison.

At the time of writing, a circuit board for connection to a PC, containing the 50 MHz Xilinx Spartan 3 (XC3S400) FPGA and all necessary hardware and software, cost approximately half as much as a 3 GHz Pentium 4 and circuit board. High performance FPGA such as the 390 MHz Xilinx Virtex 4 cost a similar proportion of the cost of a high performance CPU such as a 3.8 GHz Pentium 4. The largest Xilinx Virtex 4 is approximately 25 times larger than the Xilinx Spartan 3 (XC3S400) FPGA and could potentially process pedigrees with 1500 individuals for the inbreeding problem or 500 individuals for estimating genotype probabilities.

3. RESULTS

3.1. Estimating inbreeding coefficients

For the pedigree in Table I and the pedigree of 60 individuals, the comparison of the FPGA implementation with the sequential implementation of the gene dropping algorithm appears in Table IV. The FPGA has a speed improvement over the sequential algorithm of 166 times for the tested processors on the pedigree of 60 individuals. In addition to the sampling speeds observed using the available FPGA and CPU, speeds were estimated for high performance FPGA and CPU that were not available to us, but that are on the market. We estimate that these FPGA could store pedigrees of up to 1500 individuals. For the CPU implementation, the estimate assumed a reduction in the time per sample proportional to the increased CPU speed, and an increase in sample time

Table V. Comparison of FPGA and general purpose CPU: estimating genotype probabilities.

Number of individuals in pedigree	Valid samples per s (3.0 Ghz Pentium)	Valid samples per s (50 Mhz FPGA)	Speed advantage for FPGA	Valid samples per s (3.8 GHz Pentium)[a]	Valid samples per s (390 Mhz FPGA)[ab]	Speed advantage for FPGA[ab]
11	**127**	**6100**	**48**	*161*	*4600*	*295*
20	**8**	**635**	**80**	*10*	*4950*	*495*

[a] Estimate only by extrapolation shown in italics.
[b] Maximum clock speed estimate is 390 Mhz for Virtex 4 devices.

proportional to the increase in the number of individuals. For the FPGA, the estimated number of samples per second is simply the number of clock cycles per second. For a pedigree of 1000 individuals the high performance FPGA is potentially 17 500 times faster than a high performance CPU.

3.2. Estimating genotype probabilities

Again, comparisons and storage could be performed in a single clock cycle so a new sample was produced each clock cycle. The comparison of the FPGA and sequential implementations of gene dropping appear in Table V. The speed advantage of the FPGA over the general purpose CPU would be 295 times if high performance CPU and FPGA were used. This increased to 495 times for the pedigree of 20 individuals with one additional genotyped individual. For these pedigrees, 0.0122% and 0.00127% of samples had non-zero likelihood, respectively.

3.3. Multi-locus pedigree data

Table VI contains the number of samples per second for the FPGA and sequential implementations for the pedigree of four individuals. Even for this small pedigree, the speed advantage would be 322 times with high performance FPGA, and such FPGA could store a larger pedigree, in which case the speed improvement would be greater.

4. DISCUSSION

With increasing amounts of molecular marker data being generated from individuals in human and livestock pedigrees, the estimation of allelic, IBD and

Table VI. Comparison of FPGA and general purpose CPU: estimating multi-locus genotype probabilities.

Number of individuals in pedigree	Valid samples per s (3.0 Ghz Pentium)	Valid samples per s (50 Mhz FPGA)	Speed advantage for FPGA	Valid samples per s (3.8 GHz Pentium)[a]	Valid samples per s (390 Mhz FPGA)[ab]	Speed advantage for FPGA[ab]
4	**119 500**	**6 256 000**	**52**	*151 400*	*48 800 000*	322

[a] Estimate only by extrapolation shown in italics.
[b] Maximum clock speed estimate is 390 Mhz for Virtex 4 devices.

haplotype probabilities is becoming common place, and the analyses are potentially the rate limiting step. In this paper, we have shown that, for the simple algorithm of gene dropping, fine grained parallelisation using FPGA is possible, and it produces considerable speed improvements over equivalent sequential algorithms. This is because, with the FPGA implementation, each individual is allocated what is effectively a dedicated single purpose processor, and the speed improvement is of the order of the number of individuals. The custom configuration of FPGA holds great potential for performance increases for processing highly regular data structures such as pedigrees.

There is an important difference between the estimation of inbreeding coefficients and the other examples presented. With the estimation of inbreeding coefficients, a given number of samples produces estimates of the same level of precision, regardless of the size of the pedigree. So, a parallel implementation, which produces the same number of samples per second regardless of the size of the pedigree, produces estimates of the same precision regardless of the size of the pedigree. This is not the case with the estimation of allelic probabilities. Adding individuals to the pedigree increases the complexity of the problem as well as the size of the problem. For example, using gene dropping on pedigrees with a similar proportion of known genotypes, as the number of individuals increases the proportion of samples that are consistent with the observed data falls, so fewer samples are accumulated. In large datasets, where the complete set of meioses cannot be enumerated, samples of low likelihood may far outnumber samples of moderate or high likelihood. Achieving the speed improvements described here to gene dropping will not be enough to address this problem, even if FPGA of sufficient size are available.

However, more sophisticated sequential algorithms (*e.g.* [1,2,4–6,8,10,17]) than gene dropping [12] have been proposed, and are currently applied to datasets, although for some of these, the size of the analyses that can be attempted is still restricted. These algorithms operate by enumerating the set of

samples of high likelihood or sampling from this set. It is possible that some of these may be implemented on FPGA, if FPGA of sufficient size are available, in which case speed improvements similar to those described here should be achievable.

FPGA are suited to analyses in which the data can be represented in a regular, structured way. Operations such as integer addition, logical *and*, *or*, and *exclusive or*, shift operations, counters and memory storage can be completed in a single clock cycle, allowing pipelining. Operations such as floating point arithmetic can also be completed in a single clock cycle, although the clock speed may need to be lowered. The main problem with floating point operations is the large amount of FPGA circuitry that is required, so algorithms that avoid such operations are favoured. For example, where the likelihood (or something proportional to it) must be calculated, an algorithm such as the Metropolis Hastings [13] algorithm may be preferable to an algorithm such as importance sampling. Although the likelihood is a real number, the storage for results remains the same as for gene dropping algorithms if the Metropolis Hastings algorithm is used, that is, an integer counter for each allele or haplotype for each individual. With importance sampling the likelihood, a floating point number, is added for each allele or haplotype, and the storage requirements are significantly increased.

There is considerable scope to perform computations on pedigree data on the hybrid circuit boards that are now becoming available. These boards contain one or more FPGA closely coupled to one or more general purpose CPU. The design of algorithms that exploit the FPGA for integer and bit operations in parallel while performing floating point operations on the general purpose CPU will be challenging, but if successful, systems based on hybrid boards may be superior to either FPGA or CPU based analysis systems. An FPGA could generate hundreds of millions of samples for a given set of parameters, while on the CPU, a sequential algorithm accumulated the results from previous FPGA runs and determined the optimal set of parameters for the next FPGA run. Some of the hybrid circuit boards are designed to be assembled into arrays, which may help overcome the limitations on the size of pedigree that can be fitted onto a single FPGA. Using an FPGA coupled to a CPU may also allow analyses on pedigrees that are too large to be stored on the FPGA. Subsets of the pedigree may be loaded into the FPGA for processing, in which case the speed advantage of the FPGA could remain constant for pedigrees larger than the maximum that can be stored on the FPGA. If this can be achieved then even the FPGA that are available today may be suitable for large livestock pedigrees, and may offer advantages over sequential algorithms.

In the examples presented, the pedigree and associated genotype data were directly coded in VHDL, and reprogramming of the FPGA was required for each pedigree. That few geneticists currently have skills in programming languages for FPGA, such as VHDL, could slow the adoption of FPGA for analyses on pedigree data, particularly since some knowledge of digital electronics is useful, and learning VHDL is not as easy as learning a new sequential computer language. However, it may be possible to program the basic structure of a general pedigree into an FPGA, and have actual pedigree data loaded at run time by people unskilled in programming FPGA. In addition, it is likely that suites of tools for programming algorithms for pedigree data onto FPGA will be developed. These may have interfaces closer to the sequential programming languages familiar to many geneticists.

It may not be possible to implement on FPGA some of the sequential algorithms currently used for analysis of pedigree data. Small increases in the complexity of the modules cause significant reductions in the number of modules that can be accommodated on the FPGA, and the FPGA available today severely limit the size of the pedigree dataset that can be analysed. However, as with general purpose CPU, the size and speed of FPGA are increasing, and in the future FPGA will be available that allow parallel computations on datasets for which current sequential algorithms cannot produce exact solutions.

5. CONCLUSION

In this paper we have shown that pedigree data can be represented in layers allowing pipelined parallel simulation of the flow of genes. This representation can be implemented on FPGA, and one sample for the whole pedigree can be produced each clock cycle, regardless of the size and complexity of the pedigree, provided that the pedigree can be stored on the FPGA. In the example of gene dropping, this produces a considerable increase in sampling speed over equivalent sequential implementations. Given a sufficiently large FPGA, more complicated algorithms may be implemented, allowing similar speed improvements, increasing the size of dataset on which analysis may be attempted.

REFERENCES

[1] Abecasis G.R., Cherny S.S., Cookson W.O., Cardon L.R., Merlin – rapid analysis of dense genetic maps using sparse gene flow trees, Nat. Genet. 30 (2002) 97–101.

[2] Cottingham R.W. Jr, Idury R.M., Schaffer A.A., Faster sequential genetic linkage computations, Am. J. Hum. Genet. 53 (1993) 252–263.

[3] Elston R.C., Stewart J., A general model for the genetic analysis of pedigree data, Hum. Hered. 21 (1971) 523–542.
[4] Guo S.W., Thompson E.A., A Monte Carlo method for combined segregation and linkage analysis, Am. J. Hum. Genet. 51 (1992) 1111–1126.
[5] Heath S.C., Markov chain Monte Carlo segregation and linkage analysis for oligogenic models, Am. J. Hum. Genet. 61 (1997) 748–760.
[6] Henshall J.M., Tier B., Kerr R.J., Estimating genotypes with independently sampled descent graphs, Genet. Res. 78 (2001) 281–288.
[7] Hortensius P.D., McLeod R.D., Card H.C., Parallel random number generation for VLSI systems using cellular automata, IEEE T. Comput. 38 (1989) 1466–1473.
[8] Kruglyak L., Lander E.S., Complete multipoint sib-pair analysis of qualitative and quantitative traits, Am. J. Hum. Genet. 57 (1995) 439–454.
[9] Lander E.S., Green P., Construction of multilocus genetic linkage maps in humans, Proc. Natl. Acad. Sci. USA 84 (1987) 2363–2367.
[10] Lange K., Matthysse S., Simulation of pedigree genotypes by random walks, Am. J. Hum. Genet. 45 (1989) 959–970.
[11] Lathrop G.M., Lalouel J.M., Julier C., Ott J., Multilocus linkage analysis in humans: detection of linkage and estimation of recombination, Am. J. Hum. Genet. 37 (1985) 482–498.
[12] MacCluer J.W., Vandeberg J.L., Read B., Ryder O.A., Pedigree analysis by computer-simulation, Zoo. Biol. 5 (1986) 147–160.
[13] Metropolis N., Rosebbluth A., Rosenbluth M., Teller A., Teller E., Equations of state calculations by fast computing machines, J. Chem. Phys. 21 (1953) 1087–1092.
[14] Meuwissen T.H.E., Luo Z., Computing inbreeding coefficients in large populations, Genet. Sel. Evol. 24 (1992) 305–313.
[15] Salwinski L., Eisenberg D., *In silico* simulation of biological network dynamics, Nat. Biotechnol. 22 (2004) 1017–1019.
[16] Sobel E., Lange K., Descent graphs in pedigree analysis: applications to haplotyping, location scores, and marker-sharing statistics, Am. J. Hum. Genet. 58 (1996) 1323–1337.
[17] Thompson E.A., Monte Carlo likelihood in genetic mapping, Stat. Sci. 9 (1994) 355–366.